Università degli Studi di Torino

PhD in Chemical and Materials Sciences

DOCTORAL THESIS

Development of Algorithms for Molecular Dynamics Simulations and Electronic Transport Properties in the CRYSTAL Code

PhD Student: **Chiara Ribaldone**

Supervisor: **Prof. Silvia Casassa**

UNIVERSITÀ
DI TORINO

Contents

Chapter 1

Introduction

This doctoral thesis encompasses two subjects.

The main topic is the study and the implementation of the Born-Oppenheimer molecular dynamics in the CRYSTAL code, a program for quantum mechanical simulations of materials, whose peculiarity stems from the use of atom-centered basis functions within a linear combination of atomic orbitals to describe the wavefunction of a condensed matter system. The corresponding efficiency in the evaluation of the exact Fock exchange series has led to the implementation of a rich variety of hybrid density functionals, at low computational cost. Indeed, CRYSTAL derives its unique capabilities from the use of a local basis set of (non-orthogonal) atomic orbitals, expressed in terms of linear combinations of Gaussian basis functions. This basis set choice and the extremely efficient algorithms implemented in the code for the analytical calculation of two-electron repulsion integrals make CRYSTAL a primary tool in solid state physics in terms of both accuracy and computational efficiency in large systems. In particular, the use of Gaussian basis functions and a suitable truncation of the exchange series allow a very efficient implementation of hybrid exchange-correlation density functional approximations with respect to codes based on a plane wave basis set.[1] At the same time, ab initio molecular dynamics, based on Hartree-Fock (HF) or density functional theory (DFT), has become established since its early days as an important tool for simulations of an increasingly wider range of problems in geology, condensed matter physics, chemistry and biology.[2, 3, 4] This method combines the quantum mechanical static description given by ab initio theories such as HF and DFT with a classical evolution of the nuclear coordinates in the phase space, thus allowing a dynamic characterization of structural and electronic properties of a condensed matter system. Ab initio molecular dynamics simulations are commonly performed using (semi)local functionals that only depend on the electronic density and its gradients. At the same time, hybrid functionals, that do not depend only on the electronic density, but also on the Kohn-Sham orbitals, introduce non-local terms that have a key role in addressing some of the failures of (semi)local functional, in particular those related to the self-interaction error. Moreover, hybrid functionals are known for their high accuracy in ab initio molecular dynamics simulations, in particular for the improvement in the description of the structural and dynamical properties of liquids[5, 6, 7, 8, 9] and for the accuracy in the computation of free-energy surfaces.[10, 11, 12] Unfortunately, the inclusion of Hartree-Fock exchange operator required in hybrid functionals has a very high computational cost, so that ab initio molecular dynamics simulations with hybrid functionals are not routinely carried out. Although some efforts have been made in the last years to address this problem, using plane waves[13, 14], dual representation of Gaussian and plane wave basis sets,[15] and maximally localized Wannier functions,[16] the high computational costs still remain one of the main issues of ab initio molecular dynamics with hybrid functionals. In this framework, the implementation of the Born-Oppenheimer molecular dynamics in the CRYSTAL code offers the possibility to combine together ab initio molecular dynamics methods with the numerical efficiency in treating non-local exchange and hybrid functionals. The initial and main part of this thesis is therefore devoted to the explanation of the theoretical background underlying molecular dynamics methods and to the description of its implementation in the CRYSTAL code. In particular, Chapter 2 describes how to combine a quantum mechanical description of the electronic part of the system with a classical time evolution of the nuclei, Chapter 3 deals with the initialization of quantities for molecular dynamics simulations, such as the initial nuclear velocities, Chapter 4 outlines the general strategy to obtain classical equations of motion and how to combine them with statistical mechanics to

generate different ensembles, Chapter 5 contains a derivation of the algorithms to integrate the equations of motion for the microcanonical, canonical and isothermal-isobaric ensembles, with a particular attention to the correctness of the statistical ensembles generated by the equations of motion which have been integrated. These first five chapters are main focused on theoretical development, though containing the basis to implement molecular dynamics propagator in a consistent way. Chapter 6 is then devoted to the description of the theoretical derivation and the implementation of some physical quantities that can be derived from the analysis of molecular dynamics trajectories, such as the pair correlation function, the power spectral density (also called vibrational density of states) and the diffusion coefficient. Finally, Chapter 7 reports some of the more significant results obtained by applying the molecular dynamics simulations methods on two test cases, represented by a crystalline ice and a liquid-like cubic periodic system. A comment apart has to be dedicated to Chapter 8, which is aimed at explaining the theoretical foundations and the implementation in the CRYSTAL code of the Fast Inertial Relaxation Engine, a structural optimization algorithm based on molecular dynamics concepts. The interest for this novel method is threefold. Firstly, it does not rely on any approximation on the shape of the potential energy surface, possibly resulting in good convergence behavior regardless the potential energy surface form. Secondly, it does not involve the Hessian approximation, maybe leading to a less computational cost than the commonly used quasi-Newton schemes. Finally, it can be a further approach to test the feasibility of molecular dynamics propagator in the CRYSTAL code, thus supporting the possibility to perform accurate Born-Oppenheimer molecular dynamics simulations.

The second subject of this thesis is related to the calculation of electronic transport properties by means of an ab initio approach in the CRYSTAL code. The electronic ground state wavefunction computed with DFT methods (using approximate functionals to model the exchange-correlation potential) can be used to compute various physical quantities associated to the electronic ground state of a condensed matter system. In particular, properties of solids that are related to the motion of electrons through the material, such as conductivity or thermoelectricity, are of primary interest for the technological development of the so-called thermoelectric devices.[17] Ab initio calculation of these properties can be easily obtained through a post-processing of the ground state wavefunction, which consists in the computation of the derivatives of the band structure with respect to the reciprocal space point coordinates,[18, 19] and in the use of these derivatives in the semiclassical Boltzmann transport equation with relaxation time approximation.[20, 21] In this way, the electrical conductivity, the Seebeck coefficient and the electronic contribution to the thermal conductivity can be computed from ab initio principles. To date, several codes can post-process DFT wavefunctions to evaluate the electron transport properties through the solution of the Boltzmann equation. In particular, the CRYSTAL code combines together the possibility to perform analytical derivatives of the electronic bands with the numerical efficiency in treating non-local exchange and hybrid functionals.[22] However, one of the major drawback of transport properties calculation is the large sampling of the reciprocal space, that is necessary to obtain accurate values of the electronic conductivity and the Seebeck coefficient. This requires a large memory storage in each RAM unit, leading to the impossibility to run transport properties calculations for large systems, even on modern High Performance Computing (HPC) architectures. For this reason, a Massively Parallel Processing (MPP) approach would be very useful in order to address the memory issues and permit transport properties calculations on large atomic systems. The MPP strategy has already been introduced in the CRYSTAL code for the calculation of Kohn-Sham and overlap matrices in reciprocal space and for the diagonalization of the Kohn-Sham matrix, by exploiting the independence of the reciprocal space points for the calculations of the electronic transport properties.[23] This same property of reciprocal space points independence can be used to massively parallelize the computation of the band velocities in the transport properties calculations. Chapter 9 is therefore dedicated to this issue, both revising the calculation of electronic transport properties from a theoretical point of view and presenting some results coming from MPP method implemented on this part of the code.

Finally, Chapter 10 outlines the conclusions and future perspectives of this work. Moreover, a series of appendices, referenced throughout the thesis, can guide the reader through the details of theoretical demonstrations and practical implementation in the code. In particular, Appendix A contains some mathematical demonstrations main related to Chapters 4-5 of the thesis, Appendix B outlines some details (workflows and commented pieces of code) about molecular dynamics implementation in the CRYSTAL code, Appendix C reports the main equations implemented for electronic transport properties

calculations, together with some bugs that are discovered during the implementation of the Massive Parallel version of this part of the code, Appendix D includes the computational details and input files used for molecular dynamics simulations on test cases as described in Chapter 7. Finally, Appendix E contains the manuals with the list of input keywords and options to be used for molecular dynamics simulations and for the structural optimization using Fast Inertial Relaxation Engine method, while Appendix F and G include details of the CRYSTAL code, respectively about some subroutines of interest, which have been reported and commented, and about the information contained in fort.98 unit file.

Chapter 2

Unifying molecular dynamics and electronic structure

"The molecular dynamics method computes phase space trajectories of a collection of atoms which individually obey classical laws of motion."

D. W. Heermann, *Computer Simulation Methods in Theoretical Physics*

The very basic starting point to describe the atomic properties of materials is the studying of the Schrödinger equation for the electronic part of the system, taking into account the electrons-electrons and the nuclei-electrons interactions. The solution of the electronic Schrödinger equation can be performed using different theories, among which the most used are the Hartree-Fock and the Kohn-Sham methods. These theories are based on the parametrization of the nuclei positions, so that each electronic ground state is computed for a particular configuration of the nuclei. Therefore, in this approach the thermal motion of the nuclei is neglected, so that the electronic ground state is computed for an ideal system, frozen in a given nuclei configuration. In this framework, the electronic structure theories computes the electronic properties of a system at a temperature of zero Kelvin. However, the study of the properties of a given system at non zero temperatures is rather important and interesting. The presence of non zero temperature determines a thermal motion of nuclei, that becomes important to describe various properties of condensed matter systems. This chapter has the purpose to construct a theoretical background which can describe the motion of nuclei and electrons in condensed matter systems, introducing some of the most used molecular dynamics methods.

A complete, non-relativistic, description of a system of N atoms whose positions are represented by the set of coordinates $\boldsymbol{q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_i, ..., \boldsymbol{q}_N)$ and with N_e electrons in position and spin states at $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, ..., \boldsymbol{\xi}_\mu, ..., \boldsymbol{\xi}_{N_e})$ is provided by the non-relativistic Schrödinger form of the equation of motion in its coordinate representation

$$i\hbar \frac{\partial}{\partial t} \Psi(\boldsymbol{\xi}, \boldsymbol{q}, t) = \hat{\mathcal{H}}(\boldsymbol{\xi}, \boldsymbol{q})\, \Psi(\boldsymbol{\xi}, \boldsymbol{q}, t) \tag{2.1}$$

where $\boldsymbol{\xi} := (\boldsymbol{r}_e, \sigma)$ is the spatial and spin coordinates of the electrons, and the Hamiltonian $\hat{\mathcal{H}}(\boldsymbol{\xi}, \boldsymbol{q})$ that governs both the electronic and nuclear motion is the electrostatic non-relativistic Hamiltonian form for a system with N nuclei and N_e electrons, which can in general be written as the sum of a nuclear kinetic operator $\hat{T}_n(\boldsymbol{q})$ and an electronic Hamiltonian $\hat{\mathcal{H}}_e(\boldsymbol{\xi}, \boldsymbol{q})$ which depends on the electronic and nuclear coordinates:

$$\hat{\mathcal{H}}(\boldsymbol{\xi}, \boldsymbol{q}) = \hat{T}_n(\boldsymbol{q}) + \hat{\mathcal{H}}_e(\boldsymbol{\xi}, \boldsymbol{q}) \qquad \text{where} \qquad \hat{\mathcal{H}}_e(\boldsymbol{\xi}, \boldsymbol{q}) = \hat{T}_e(\boldsymbol{\xi}) + \hat{v}(\boldsymbol{\xi}, \boldsymbol{q}) \tag{2.2}$$

The explicit form of the electronic Hamiltonian $\hat{\mathcal{H}}_e(\boldsymbol{\xi}, \boldsymbol{q})$ is not specified here, however, it is the entire Hamiltonian of the system with the exception of the kinetic energy operator for the nuclei; i.e., it is the Hamiltonian that governs the fast particle (electronic) motion when the slow (nuclear) particles are at

fixed positions $\boldsymbol{q}$. In the most general case, it includes the electronic kinetic $\hat{T}_e(\boldsymbol{\xi})$ and all inter-particle interaction $\hat{v}(\boldsymbol{\xi},\boldsymbol{q})$ operators (as, for example, the electron-electron, electron-nuclear, nuclear-nuclear and spin-orbit interaction operators). In principle, a quantum mechanical treatment of both the electronic and nuclear dynamics could be employed by solving (2.1). This is impractical, of course, except for the simplest cases, those for which it is adequate to reduce the dimensionality of the problem to only a few degrees of freedom. To handle this problem, two fundamental approximations are introduced, namely, the Born-Oppenheimer approximation and classical mechanical motion of nuclei, which lay at the basis of the most used molecular dynamics models and allow a straightforward implementation in a quantum *ab initio* code. Therefore, the goal of this section is to derive molecular dynamics of classical point particles, that is essentially the classical mechanics rule of motion, starting from the Schrödinger quantum-mechanical wave equation of motion (2.1) for both electrons and nuclei. To this end, the nuclear and electronic contributions to the total wavefunction $\Psi(\boldsymbol{\xi},\boldsymbol{q},t)$ are separated directly such that, ultimately, the classical limit can be imposed for the nuclei only. The so-called Ehrenfest product ansatz is introduced for the wavefunction form in (2.1), which reads as

$$\Psi(\boldsymbol{\xi},\boldsymbol{q},t) = \Psi_e(\boldsymbol{\xi},t)\Psi_n(\boldsymbol{q},t)\exp\left[\frac{i}{\hbar}\int_{t_0}^{t}\tilde{E}_e(t')\,dt'\right] \tag{2.3}$$

where the nuclear $\Psi_n(\boldsymbol{q},t)$ and electronic $\Psi_e(\boldsymbol{\xi},t)$ wavefunctions are time dependent and are separately normalized to unity at every time t with respect to integration over $\boldsymbol{q}$ and $\boldsymbol{\xi}$, respectively

$$\int \Psi_n^*(\boldsymbol{q},t)\Psi_n(\boldsymbol{q},t)\,d\boldsymbol{q} = 1 \qquad \text{and} \qquad \int \Psi_e^*(\boldsymbol{\xi},t)\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi} = 1 \tag{2.4}$$

The arbitrary phase factor

$$\tilde{E}_e(t) = \int\int \Psi_e^*(\boldsymbol{\xi},t)\Psi_n^*(\boldsymbol{q},t)\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\Psi_n(\boldsymbol{q},t)\,d\boldsymbol{\xi}\,d\boldsymbol{q} \tag{2.5}$$

is introduced in (2.3) for convenience, in order to simplify the appearance of the final equations, but since the phase factor $\tilde{E}_e$ does not depend explicitly on $\boldsymbol{\xi}$ or $\boldsymbol{q}$, it could be incorporated into either of the other terms to make equation (2.3) appear as a simple product. It is mentioned in passing that this approximation is called a one-determinant or single-configuration ansatz for the total wavefunction, which at the end must lead to a mean-field description of the coupled dynamics. Note in addition that this product ansatz differs, independently from the issue of phase factor, from the Born product ansatz[24] expressed in terms of *adiabatic* electronic states $\Psi_e(\boldsymbol{\xi};\boldsymbol{q})$, even if only a single electronic state is considered in the Born product.

At this point, the ansatz (2.3) can be substituted into the Schrödinger equation of motion (2.1), using the Hamiltonian form of equation (2.2). Multiplying from the left by $\Psi_n^*(\boldsymbol{q},t)$ and integrating over $\boldsymbol{q}$ yields an effective Schrödinger equation for the fast variables $\boldsymbol{\xi}$ related to the electronic particles:

$$\begin{aligned} i\hbar\frac{\partial\Psi_e(\boldsymbol{q},t)}{\partial t} = &-\frac{\hbar^2}{2}\sum_{\mu=1}^{N_e}\frac{\nabla_\mu^2}{m_\mu}\Psi_e(\boldsymbol{\xi},t) - i\hbar\left\{\int \Psi_n^*(\boldsymbol{q},t)\frac{\partial\Psi_n(\boldsymbol{q},t)}{\partial t}\,d\boldsymbol{q}\right\}\Psi_e(\boldsymbol{\xi},t) + \tilde{E}_e(t)\Psi_e(\boldsymbol{\xi},t) \\ &+\left\{\int \Psi_n^*(\boldsymbol{q},t)\left[-\frac{\hbar^2}{2}\sum_{i=1}^{N}\frac{\nabla_i^2}{m_i} + \hat{v}(\boldsymbol{\xi},\boldsymbol{q})\right]\Psi_n(\boldsymbol{q},t)\,d\boldsymbol{q}\right\}\Psi_e(\boldsymbol{\xi},t) \end{aligned} \tag{2.6}$$

where the electronic and nuclear kinetic operators have been explicitly written. Similarly, multiplying the Schrödinger equation of motion from the left by $\Psi_e^*(\boldsymbol{\xi},t)$ and integrating over the coordinates $\boldsymbol{\xi}$ gives an effective Schrödinger equation for the slow variables $\boldsymbol{q}$ related to the nuclear degrees of freedom:

$$\begin{aligned} i\hbar\frac{\partial\Psi_n(\boldsymbol{q},t)}{\partial t} = &-\frac{\hbar^2}{2}\sum_{i=1}^{N_e}\frac{\nabla_i^2}{m_i}\Psi_n(\boldsymbol{q},t) - i\hbar\left\{\int \Psi_e^*(\boldsymbol{\xi},t)\,\frac{\partial\Psi_e(\boldsymbol{\xi},t)}{\partial t}\,d\boldsymbol{\xi}\right\}\Psi_n(\boldsymbol{q},t) + \tilde{E}_e(t)\Psi_n(\boldsymbol{q},t) \\ &+\left\{\int \Psi_e^*(\boldsymbol{\xi},t)\left[-\frac{\hbar^2}{2}\sum_{\mu=1}^{N_e}\frac{\nabla_\mu^2}{m_\mu} + \hat{v}(\boldsymbol{\xi},\boldsymbol{q})\right]\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi}\right\}\Psi_n(\boldsymbol{q},t) \end{aligned} \tag{2.7}$$

which is the equation of motion that describes the evolution in time of the nuclear wavefunction, in the same way as equation (2.6) rules the evolution in time of the electronic wavefunction. The time derivative term on the right hand side of equation (2.6) and the similar term in equation (2.7) remain to be specified. Note that because of the assumption that $\Psi_e(\boldsymbol{\xi},t)$ and $\Psi_n(\boldsymbol{q},t)$ are normalized, the derivative integrals are pure imaginary; i.e., the derivative factors multiplied by $i\hbar$ are real-valued. Multiplying equation (2.7) from the left by $\Psi_n^*(\boldsymbol{q},t)$ and integrating over $\boldsymbol{q}$ yields

$$\begin{aligned} i\hbar \int \Psi_n^*(\boldsymbol{q},t)\,\frac{\partial \Psi_n(\boldsymbol{q},t)}{\partial t}\,d\boldsymbol{q} + i\hbar \int \Psi_e^*(\boldsymbol{\xi},t)\,\frac{\partial \Psi_e(\boldsymbol{\xi},t)}{\partial t}\,d\boldsymbol{\xi} - \tilde{E}_e(t) \\ = \int\int \Psi_e^*(\boldsymbol{\xi},t)\Psi_n^*(\boldsymbol{q},t)\,\hat{\mathcal{H}}(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\Psi_n(\boldsymbol{q},t)\,d\boldsymbol{\xi}\,d\boldsymbol{q} = E \end{aligned} \tag{2.8}$$

The same equation can be obtained by multiplying equation (2.6) from the left by $\Psi_e^*(\boldsymbol{\xi},t)$ and integrating over the electronic coordinates $\boldsymbol{\xi}$. In order for the total energy E to be conserved, the time derivative of equation (2.8) must be zero. This imposes a constraint on the two derivative factors and the time dependent phase factor $\tilde{E}_e(t)$ in (2.8). The derivative factors can be specified arbitrarily, subject to satisfying equation (2.8). In standard derivations, these factors are chosen so that the resulting effective Schrödinger equations are symmetric in $\boldsymbol{\xi}$ and $\boldsymbol{q}$. In the present situation, however, $\boldsymbol{\xi}$ and $\boldsymbol{q}$ are not equivalent; one is the coordinate related to fast particles and the other to slow ones. It is more convenient here to choose an unsymmetrical definition of the phases.[25] The phase convention is arbitrary, however, so that the present development is entirely equivalent to standard derivations. The derivative phase factors can thus be defined as follows:

$$\begin{aligned} i\hbar \int \Psi_n^*(\boldsymbol{q},t)\,\frac{\partial \Psi_n(\boldsymbol{q},t)}{\partial t}\,d\boldsymbol{q} = E \\ i\hbar \int \Psi_e^*(\boldsymbol{\xi},t)\,\frac{\partial \Psi_e(\boldsymbol{\xi},t)}{\partial t}\,d\boldsymbol{\xi} = \tilde{E}_e(t) \end{aligned} \tag{2.9}$$

On the basis of this assumption, the final effective Schrödinger equations of motion (2.6) for the fast (electronic) and (2.7) for the slow (nuclear) particles become, respectively,

$$i\hbar\frac{\partial \Psi_e(\boldsymbol{\xi},t)}{\partial t} = -\frac{\hbar^2}{2}\sum_{\mu=1}^{N_e}\frac{\nabla_\mu^2}{m_\mu}\Psi_e(\boldsymbol{\xi},t) + \left\{\int \Psi_n^*(\boldsymbol{q},t)\,\hat{v}(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_n(\boldsymbol{q},t)\,d\boldsymbol{q}\right\}\Psi_e(\boldsymbol{\xi},t) \tag{2.10}$$

$$i\hbar\frac{\partial \Psi_n(\boldsymbol{q},t)}{\partial t} = -\frac{\hbar^2}{2}\sum_{i=1}^{N}\frac{\nabla_i^2}{m_i}\Psi_n(\boldsymbol{q},t) + \left\{\int \Psi_e^*(\boldsymbol{\xi},t)\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi}\right\}\Psi_n(\boldsymbol{q},t) \tag{2.11}$$

This set of coupled time-dependent Schrödinger equations defines the basis of the time-dependent self-consistent field (TDSCF) method. Both electrons and nuclei move quantum-mechanically in time-dependent effective potentials, i.e. self-consistently obtained average fields, given by the expressions in the braces. These potentials are obtained from appropriate averages (defined as quantum-mechanical expectation values) over the other class of degrees of freedom by using the nuclear and electronic wavefunctions, respectively. In other words, the fast particles move in the average field of the slow particles, and vice-versa: this is a mean-field theory. Thus, the single-determinant ansatz equation (2.3) produces, as already anticipated, a mean-field description of the coupled nuclear-electronic quantum dynamics. This is the price to pay for the simplest possible separation of electronic and nuclear variables in terms of dynamics. Up to this point, except for the arbitrary definitions of the phase factors, the fast and slow particles are treated identically. The equations would be equally valid if $\boldsymbol{\xi}$ were the slow particles and $\boldsymbol{q}$ the fast ones.

At this stage, the equations which rules the motion of nuclei in standard molecular dynamics can be obtained by approximating the nuclei as classical point particles, i.e. applying the semiclassical approximation to equation (2.11), in the presence of electrons which do move quantum-mechanically in time according to equation (2.10). The first step is to factor the nuclear wavefunction into amplitude and phase terms

$$\Psi_n(\boldsymbol{q},t) = A(\boldsymbol{q},t)\,e^{\frac{i}{\hbar}S(\boldsymbol{q},t)} \tag{2.12}$$

where $A(\boldsymbol{q},t)$ is an amplitude factor and $S(\boldsymbol{q},t)$ is a phase with dimension of an action. The phase $S(\boldsymbol{q},t)$ is in general a complex function, which can be used for describing the system as well as the original nuclear wavefunction $\Psi_n(\boldsymbol{q},t)$. In this treatment, both the amplitude $A(\boldsymbol{q},t)$ and the phase $S(\boldsymbol{q},t)$ are supposed to be real-valued. Substituting the form (2.12) of the nuclear wavefunction into the equation of motion (2.11) and separating real and imaginary terms results in the following two equations, respectively

$$\frac{\partial S(\boldsymbol{q},t)}{\partial t} + \frac{1}{2}\sum_{i=1}^{N}\frac{1}{m_i}[\nabla_i S(\boldsymbol{q},t)]^2 + \int \Psi_e^*(\boldsymbol{\xi},t)\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi} = \frac{\hbar^2}{2}\sum_{i=1}^{N}\frac{1}{m_i}\frac{\nabla_i^2 A(\boldsymbol{q},t)}{A(\boldsymbol{q},t)} \tag{2.13}$$

$$\frac{\partial A(\boldsymbol{q},t)}{\partial t} + \sum_{i=1}^{N}\frac{1}{m_i}\nabla_I A(\boldsymbol{q},t)\nabla_i S(\boldsymbol{q},t) + \frac{1}{2}\sum_{i=1}^{N}\frac{1}{m_i}A(\boldsymbol{q},t)\nabla_i^2 S(\boldsymbol{q},t) = 0 \tag{2.14}$$

Equations (2.13) and (2.14) are entirely equivalent to the original Schrödinger equation (2.11).
The classical limit is obtained by setting $\hbar \to 0$ on the right hand side of equation (2.13), yielding a Hamilton-Jacobi equation

$$\frac{\partial S(\boldsymbol{q},t)}{\partial t} + \mathcal{H}(\boldsymbol{q},\nabla_i S) = 0 \tag{2.15}$$

with the corresponding Hamiltonian

$$\mathcal{H}(\boldsymbol{q},\nabla_i S) = \frac{1}{2}\sum_{i=1}^{N}\frac{1}{m_i}[\nabla_i S(\boldsymbol{q},t)]^2 + \int \Psi_e^*(\boldsymbol{\xi},t)\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi} \tag{2.16}$$

that can be rewritten as a classical Hamiltonian function defined in terms of generalized positions $\boldsymbol{q}$ and their conjugate canonical momenta $\boldsymbol{p}$ in the following way

$$\mathcal{H}(\boldsymbol{q},\nabla_i S) \equiv \mathcal{H}(\boldsymbol{q},\boldsymbol{p}) = \frac{1}{2}\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t)}{m_i} + E_e(\boldsymbol{q}(t)) = T(\boldsymbol{p}(t)) + E_e(\boldsymbol{q}(t)) \tag{2.17}$$

where $T(\boldsymbol{p}(t))$ is the nuclear kinetic energy while $E_e(\boldsymbol{q}(t))$ is the expectation value of the electronic Hamiltonian (2.2), containing the electronic kinetic operator and the inter-particle interaction operators. At the same time, the relation for the amplitude, equation (2.14), may be rewritten after multiplying by $2A(\boldsymbol{q},t)$ from the left as

$$\begin{aligned} &2A(\boldsymbol{q},t)\,\frac{\partial A(\boldsymbol{q},t)}{\partial t} + \sum_{i=1}^{N}\frac{2}{m_i}\,A(\boldsymbol{q},t)[\nabla_i\,A(\boldsymbol{q},t)][\nabla_i\,S(\boldsymbol{q},t)] + \sum_{i=1}^{N}\frac{1}{m_i}\,A^2(\boldsymbol{q},t)\,\nabla_i^2\,S(\boldsymbol{q},t) = 0 \\ &\frac{\partial A^2(\boldsymbol{q},t)}{\partial t} + \sum_{i=1}^{N}\frac{1}{m_i}\Big\{\underbrace{2A(\boldsymbol{q},t)[\nabla_i\,A(\boldsymbol{q},t)][\nabla_i\,S(\boldsymbol{q},t)] + A^2(\boldsymbol{q},t)\,\nabla_i^2\,S(\boldsymbol{q},t)}_{=\,\nabla_i[A^2(\boldsymbol{q},t)\nabla_i S(\boldsymbol{q},t)]}\Big\} = 0 \end{aligned} \tag{2.18}$$

which is easily identified as the continuity equation,

$$\frac{\partial A^2(\boldsymbol{q},t)}{\partial t} + \sum_{i=1}^{N}\frac{1}{m_i}\nabla_i[A^2(\boldsymbol{q},t)\nabla_i S(\boldsymbol{q},t)] = 0 \quad \to \quad \frac{\partial \rho(\boldsymbol{q},t)}{\partial t} + \sum_{i=1}^{N}\nabla_i J_i(\boldsymbol{q},t) = 0 \tag{2.19}$$

where

$$\rho(\boldsymbol{q},t) = A^2(\boldsymbol{q},t) = |\Psi_n(\boldsymbol{q},t)|^2 \qquad \text{and} \qquad J_i(\boldsymbol{q},t) = A^2(\boldsymbol{q},t)\nabla_i S(\boldsymbol{q},t)/m_i \tag{2.20}$$

are the nuclear probability density and the associated current density, respectively. This continuity equation (2.19) is independent of $\hbar$ and ensures locally the conservation of the particle probability density $|\Psi_n(\boldsymbol{q};t)|^2$ of the nuclei in the presence of a flux.
The transformation of expression (2.16) for the classical Hamiltonian into (2.17) has been possible through the definition of the connecting transformation

$$\boldsymbol{p}_i(t) = \nabla_i\,S(\boldsymbol{q},t) \tag{2.21}$$

from which an interesting relation between the nuclei velocities and the nuclear probability and current density can be derived as

$$\boldsymbol{p}_i(t) = m_i\dot{\boldsymbol{q}}_i(t) = \nabla_i\, S(\boldsymbol{q},t) = m_i\,\frac{J_i(\boldsymbol{q},t)}{\rho(\boldsymbol{q},t)} \quad \rightarrow \quad \dot{\boldsymbol{q}}_i(t) = \frac{J_i(\boldsymbol{q},t)}{\rho(\boldsymbol{q},t)} \tag{2.22}$$

Finally, deriving with respect to time the nuclear momenta defined in the previous equations (2.21)-(2.22), the Newtonian equations of motion for the classical nuclei can be obtained

$$\begin{aligned}\frac{d\boldsymbol{p}_i(t)}{dt} = m_i\ddot{\boldsymbol{q}}_i(t) = \frac{d\nabla_i\, S(\boldsymbol{q},t)}{dt} = \nabla_i\left(\frac{\partial\, S(\boldsymbol{q},t)}{\partial t}\right) &\overset{(2.15)}{=} -\nabla_i\,\mathcal{H}(\boldsymbol{q},\nabla_i S) \\ &\overset{(2.17)}{=} -\nabla_i\,[T(\boldsymbol{p}(t)) + E_e(\boldsymbol{q}(t))] = -\nabla_i\, E_e(\boldsymbol{q}(t))\end{aligned} \tag{2.23}$$

and can be rewritten in a more clear way as

$$\frac{d\boldsymbol{p}_i(t)}{dt} = -\nabla_i \int \Psi_e^*(\boldsymbol{\xi},t)\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_e(\boldsymbol{\xi},t)\,d\boldsymbol{\xi} \quad \text{or} \quad m_i\ddot{\boldsymbol{q}}_i(t) = -\nabla_i\, E_e(\boldsymbol{q}(t)) \tag{2.24}$$

Therefore, the calculation leads to conclude that the classical motion of the nuclei in the system is driven by an effective potential $E_e(\boldsymbol{q}(t))$, called the Ehrenfest potential, which is given by the quantum dynamics of the electrons obtained by solving simultaneously the time-dependent electronic Schrödinger equation (2.10). By virtue of its definition, it is clearly seen that this time-local many-body interaction potential due to the explicit time evolution of the quantum electrons stems from averaging the electronic Hamiltonian with respect to the electronic degrees of freedom, $E_e(\boldsymbol{q}(t)) = \langle\Psi_e|\hat{\mathcal{H}}_e|\Psi_e\rangle$. However, the time-dependent equation that describes the time evolution of the electrons, equation (2.10), still contains the full quantum-mechanical nuclear wavefunction $\Psi_n(\boldsymbol{q},t)$ instead of just the classical-mechanical nuclear positions $\boldsymbol{q}(t)$. In this case the classical reduction can be achieved simply by replacing the nuclear density $|\Psi_n(\boldsymbol{q},t)|^2$ in equation (2.10) in the limit $\hbar \to 0$ by a product of delta functions centered at the instantaneous positions $\boldsymbol{q}(t)$ of the classical nuclei. This naive approach yields, e.g. for the position operator,

$$\int \Psi_n^*(\boldsymbol{q},t)\,\boldsymbol{q}_i\,\Psi_n(\boldsymbol{q},t)\;d\boldsymbol{q} \quad \overset{\hbar\to 0}{\longrightarrow} \quad \int \boldsymbol{q}_i \prod_i \delta(\boldsymbol{q}_i - \boldsymbol{q}_i(t))\;d\boldsymbol{q} = \boldsymbol{q}_i(t) \tag{2.25}$$

the required expectation value. This classical limit leads to a time-dependent wavefunction for the electrons

$$i\hbar\,\frac{\partial\Psi_e(\boldsymbol{\xi},t)}{\partial t} = -\frac{\hbar^2}{2}\sum_{\mu=1}^{N_e}\frac{\nabla_\mu^2}{m_\mu}\Psi_e(\boldsymbol{\xi},t) + \hat{v}(\boldsymbol{\xi};\boldsymbol{q}(t))\,\Psi_e(\boldsymbol{\xi},t) = \hat{\mathcal{H}}_e(\boldsymbol{\xi};\boldsymbol{q}(t))\,\Psi_e(\boldsymbol{\xi},\boldsymbol{q}(t),t) \tag{2.26}$$

which evolves self-consistently as the classical nuclei are propagated via equation (2.24). Note that now $\hat{\mathcal{H}}_e$ depends *parametrically* on the classical nuclear positions $\boldsymbol{q}(t)$ at time t through $\hat{v}(\boldsymbol{\xi};\boldsymbol{q}(t))$. This means that feedback between the classical and quantum degrees of freedom is incorporated in both directions, although in a mean-field sense only.

The approach to *ab initio* molecular dynamics that relies on solving Newton equation for the nuclei, equation (2.24), simultaneously with Schrödinger equation for the electrons, equation (2.26), is often called Ehrenfest molecular dynamics, in honor of Paul Ehrenfest who was the first to address the essential question of how Newtonian classical dynamics of point particles can be derived from Schrödinger time-dependent wave equation. In the present case this leads to a hybrid or mixed quantum-classical approach because only the nuclei are forced to behave like classical particles, whereas the electrons are still treated as quantum objects, so that the electronic subsystem evolves explicitly in time according to a time-dependent Schrödinger equation.

Although the approach underlying Ehrenfest molecular dynamics is clearly a mean-field theory concerning the dynamical evolution, transitions between electronic states are included in this scheme. This can be made transparent by expanding the electronic wavefunction $\Psi_e(\boldsymbol{\xi},t)$ in equation (2.3) in a basis of electronic states $\Psi_{e,s}(\boldsymbol{\xi};\boldsymbol{q})$ as

$$\Psi_e(\boldsymbol{\xi},\boldsymbol{q}(t),t) = \sum_{s=0}^{\infty} c_s(t)\,\Psi_{e,s}(\boldsymbol{\xi};\boldsymbol{q}) \tag{2.27}$$

with complex time-dependent coefficients $\{c_s(t)\}$. In this case, the coefficients satisfy the relation

$$\sum_{s=0}^{\infty} |c_s(t)|^2 = 1 \tag{2.28}$$

and they describe explicitly the time evolution of the populations (occupations) of the different states s whereas the necessary interferences between any two such states are included via the off-diagonal terms, $c_k^* c_{s\neq k}$. One possible choice for the basis functions $\{\Psi_{e,k}(\boldsymbol{\xi};\boldsymbol{q})\}$ is the instantaneous adiabatic basis obtained from solving the time-independent electronic Schrödinger equation

$$\hat{\mathcal{H}}_e(\boldsymbol{\xi};\boldsymbol{q})\,\Psi_{e,k}(\boldsymbol{\xi};\boldsymbol{q}) = E_{e,k}(\boldsymbol{q})\,\Psi_{e,k}(\boldsymbol{\xi};\boldsymbol{q}) \tag{2.29}$$

where $\boldsymbol{q}$ are the instantaneous nuclear positions at time t that are determined according to equation (2.24). Here a further simplification is invoked in order to reduce Ehrenfest molecular dynamics to the so called Born-Oppenheimer molecular dynamics. To achieve this, the electronic wavefunction Ψ_e is restricted to be the ground state adiabatic wavefunction $\Psi_{e,0}$ of the electronic Hamiltonian $\hat{H}_e$ at each instant of time according to equation (2.29), which implies $|c_0(t)|^2 \equiv 1$ and thus a single term in the expansion (2.27). This should be a good approximation if the energy difference between $\Psi_{e,0}$ and the first excited state $\Psi_{e,1}$ is large everywhere compared to the thermal energy $k_B T$, roughly speaking. In this limit the nuclei move on a single adiabatic potential energy surface according to equation (2.24), where

$$E_e(\boldsymbol{q}(t)) = \int \Psi_{e,0}^*(\boldsymbol{\xi})\,\hat{\mathcal{H}}_e(\boldsymbol{\xi},\boldsymbol{q})\,\Psi_{e,0}(\boldsymbol{\xi})\,d\boldsymbol{\xi} \equiv E_{e,0}(\boldsymbol{q}) \tag{2.30}$$

This single adiabatic potential energy surface on which the nuclear motion takes place is nothing else than the ground state Born-Oppenheimer potential energy surface that is obtained by solving the time-independent electronic Schrödinger equation (2.29) for $k = 0$ at each nuclear configuration $\boldsymbol{q}$ generated during molecular dynamics. This leads to the identification $E_e(\boldsymbol{q}(t)) = E_{e,0}(\boldsymbol{q})$ and thus, in this limit, the Ehrenfest potential is identical to the ground state Born-Oppenheimer (or clamped nuclei) potential.
The Born-Oppenheimer approach consists in solving the *static* electronic structure problem in each molecular dynamics step, given the set of fixed nuclear positions at that instant of time. Thus, the electronic structure part is reduced to solving a *time-independent* quantum problem, e.g. by solving the *time-independent*, stationary Schrödinger equation, concurrently to propagating the nuclei according to classical mechanics. This implies that the time dependence of the electronic structure is imposed and dictated by its parametric dependence on the classical dynamics of the nuclei which it just follows. The resulting Born-Oppenheimer molecular dynamics method can be written down readily and is defined by the set of equations

$$m_i \ddot{\boldsymbol{q}}_i(t) = -\nabla_i \min_{\Psi_{e,0}} [\langle \Psi_{e,0} | \hat{\mathcal{H}}_e | \Psi_{e,0} \rangle] \qquad i = 1, ..., N \tag{2.31}$$

$$\hat{\mathcal{H}}_e(\boldsymbol{\xi};\boldsymbol{q})\,\Psi_{e,0}(\boldsymbol{\xi};\boldsymbol{q}) = E_{e,0}(\boldsymbol{q})\,\Psi_{e,0}(\boldsymbol{\xi};\boldsymbol{q}) \tag{2.32}$$

for the nuclear equation of motion and the electronic ground state calculation. The minimum of $\langle \hat{\mathcal{H}}_e \rangle$ has to be reached in each step of a Born-Oppenheimer molecular dynamics propagation according to equation (2.31), for instance by diagonalizing the Hamiltonian.
The main issue in molecular dynamics simulation algorithms is the resolution of Newton equations of motion for the nuclei, defined by equation (2.31), that can be rewritten in a more simple way as

$$m_i \ddot{\boldsymbol{q}}_i(t) = \mathbf{F}_i(t) \qquad i = 1, ..., N \tag{2.33}$$

where m_i are the nuclear masses, $\boldsymbol{q}_i(t)$ the positions of the nuclei at time t, N the number of nuclei in the system and $\mathbf{F}_i(t)$ the forces acting on the i-th particle computed as

$$\mathbf{F}_i(t) = -\nabla_i \min_{\Psi_{e,0}} [\langle \Psi_{e,0} | \hat{\mathcal{H}}_e | \Psi_{e,0} \rangle] = -\nabla_i E_{e,0}(\boldsymbol{q}(t)) \qquad i = 1, ..., N \tag{2.34}$$

In Chapter 4, Section 4.2.1, one of the more robust and well-behaved algorithm used for the integration of the equations of motion (2.33) will be derived. As will be explained, the Newton equations of motion (2.33) sample the phase space so as to generate the microcanonical ensemble. However, before performing the integration of the nuclear equations of motion which evolve the coordinates of the nuclei in time, some geometrical constraints have to be introduced, in order to prevent spurious translations or rotations of the system. This issue is treated in the next Section 3.2.

Chapter 3

Initialization of nuclear degrees of freedom

3.1 Initialization of nuclear positions and velocities

The initialization of nuclear positions and velocities is very important in molecular dynamics simulations, since it constitutes the initial conditions by which the dynamic trajectory depends.
The initialization of nuclear positions is essentially based on the geometry and symmetry of the system, i.e. its space group. Sometimes the nuclear positions are taken from experimental data. Otherwise, a structural optimization is performed, so that the equilibrium positions which leads to a minimal energy of the system in the framework of a given theoretical method (e.g. HF or DFT) are computed and taken as starting conditions for the dynamics. In both cases, it is recommended for the initial nuclear positions to be near their equilibrium positions. Indeed, if the initial geometry is too far from equilibrium, instabilities and artifacts can arise during the propagation of coordinates in time along the dynamics trajectory.
In the beginning of a molecular dynamics simulation, it is often the case that only the initial positions of the nuclei are known, but not the velocities. Therefore, the initialization of the nuclear velocities is a little bit more difficult. As molecular dynamics simulations are performed at some finite temperature, it is a good idea to initialize the velocities in a way such that the desired simulation temperature is already present at the beginning. In statistical mechanics, it is often assumed that the velocity distribution of atoms is given by a Maxwell-Boltzmann distribution (which is strictly only the case in idealized gases). It is a reasonable choice to initialize the nuclei velocities according to the Maxwell-Boltzmann equation in the beginning of a dynamics simulation. The goal is to find an initial nuclear velocity distribution in which each degree of freedom possesses a similar amount of energy, such that the equipartition theorem is approximately fulfilled. This can be accomplished by looking for an initial configuration of the nuclear velocities along the three spatial directions that follows a random Gaussian distribution. It is fairly straightforward to generate a list of pseudo-random numbers on a computer. However, these numbers are uniformly distributed. In this case, however, it is necessary to generate a string of numbers with a different probability distribution, i.e. a Gaussian distribution. One of the most famous algorithm which performs this task is the so-called Box-Muller algorithm, introduced by G. P. Box and M. E. Muller.[26] It is a random number sampling method for generating pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a source of uniformly distributed random numbers. In particular, the Box-Muller algorithm can be used to convert two sets of random numbers with uniform distributions into two sets of random numbers with Gaussian distributions. From a computational point of view, the algorithm is more efficient than the inverse transform sampling method. Furthermore, the Box-Muller algorithm does not use any approximation methods. Instead, it makes use of the famous Gaussian integral, as explained in the following.
The Gaussian integral is given by:

$$\int_{-\infty}^{\infty} e^{-x^2/2}\,dx = \sqrt{2\pi} \tag{3.1}$$

The solution of this integral is the funniest part. Trying to integrate the Gaussian function on the right hand side of equation (3.1) using traditional methods, it is found that the integral does not have a

neat solution (i.e. the indefinite integral does not have an algebraic expression in terms of elementary functions such as exponentials, logs or trigonometric functions). However, a little trick to make short work of the integral can be introduces by noting that

$$I^2 = \int_{-\infty}^{\infty} e^{-x^2/2}\,dx \int_{-\infty}^{\infty} e^{-y^2/2}\,dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)/2}\,dx\,dy \tag{3.2}$$

The last two-dimensional integral can be calculated easily using the polar coordinates $x = r\cos\theta$, $y = r\sin\theta$, with area element given by $dx\,dy = J(r,\theta)dr\,d\theta = r\,dr\,d\theta$, where $J(r,\theta)$ is the Jacobian of transformation between (x,y) and (r,θ) coordinates, so that

$$I^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)/2}\,dx\,dy = \int_0^{2\pi}\int_0^{\infty} e^{-r^2/2}\,r\,dr\,d\theta = 2\pi\int_0^{\infty} r\,e^{-r^2/2}\,dr \tag{3.3}$$

A simple substitution $u = r^2/2$, and $du = r\,dr$ can be used to solve this integral, leading to

$$I^2 = 2\pi\int_0^{\infty} r\,e^{-r^2/2}\,dr = 2\pi\int_0^{\infty} e^{-u}\,du = 2\pi \tag{3.4}$$

Finally, renaming the variable y to x in the initial integral expression leads to

$$I^2 = \int_{-\infty}^{\infty} e^{-x^2/2}\,dx \int_{-\infty}^{\infty} e^{-y^2/2}\,dy \quad \rightarrow \quad \left[\int_{-\infty}^{\infty} e^{-x^2/2}\,dx\right]^2 = 2\pi \tag{3.5}$$

Therefore the result (3.1) is finally recovered,

$$\int_{-\infty}^{\infty} e^{-x^2/2}\,dx = \sqrt{2\pi} \tag{3.6}$$

The Box-Muller algorithm is a probabilistic interpretation of the trick to solve the Gaussian integral. Suppose two sets of random numbers (x and y) have to be created, each with a probability density function given by a Gaussian integral ($p(x)$ and $p(y)$). The result above can be used to get expressions for $p(x)$ and $p(y)$, given by

$$p(x) = \frac{1}{\sqrt{2\pi}}\,e^{-x^2/2} \qquad\qquad p(y) = \frac{1}{\sqrt{2\pi}}\,e^{-y^2/2} \tag{3.7}$$

Since x and y are independent sets of numbers, the total probability density function $p(x,y)$ is

$$p(x,y) = p(x)p(y) = \frac{1}{2\pi}\,e^{-(x^2+y^2)/2} \tag{3.8}$$

Letting $R^2 = x^2 + y^2$ gives the total probability density function in radial coordinates $p(R,\Theta)$ equal to

$$p(x,y) \quad \rightarrow \quad p(R,\Theta) = \frac{1}{2\pi}\,e^{-R^2/2} \tag{3.9}$$

Now, suppose two uniformly distributed sets of numbers between zero and one (without including zero) can be generated, and call these two sets u_1 and u_2. First of all, the two sets u_1 and u_2 can be transformed into two new sets, Θ and R, on the base of the expression above:

$$u_1 = \frac{\Theta}{2\pi} \quad \rightarrow \quad \Theta = 2\pi u_1 \tag{3.10}$$

$$u_2 = e^{-R^2/2} \quad \rightarrow \quad R = \sqrt{-2\,\ln u_2} \tag{3.11}$$

The first expression for Θ represents a uniform distribution of numbers over all values from zero to 2π (i.e. over all angles). The second expression for R represents a Gaussian distribution. Then, the polar coordinates can be converted to Cartesian coordinates, so that finally the expressions for x and y variables are obtained to be

$$x = R\cos\Theta = \cos(2\pi u_1)\sqrt{-2\ln u_2} \tag{3.12}$$

$$y = R\sin\Theta = \sin(2\pi u_1)\sqrt{-2\ln u_2} \tag{3.13}$$

Since the variable R has a Gaussian distribution, both x and y have Gaussian distributions. Altogether, the Box-Muller method takes independent standard uniform random variables u_1 and u_2 and produces independent standard normal deviates x and y using the formulas (3.12) and (3.13). It may seem odd that x and y are independent, given that they use the same R and Θ variables. However, not only does the previous algebra shows that this is true,[26] but also computational tests for the distributions independence confirmed this algebraic construction. In Figure 3.1 a graphical interpretation of the Box-Muller algorithm is reported. The Box-Muller method consists in generating a point in a random way on the unit circle. This can be done by choosing Θ uniformly in the interval $[0, 2\pi]$ and then taking the point on the circle to be $(\cos\Theta, \sin\Theta)$. Another way to do this is to choose a point uniformly in the 2×2 square $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ and then rejecting it if it falls outside the unit circle. The first accepted point will be uniformly distributed in the unit disk $x^2 + y^2 \leq 1$, so its angle will be random and uniformly distributed. The final step is to get a point on the unit circle $x^2 + y^2 = 1$ by dividing by the length. From a computational point of view, the Box-Muller algorithm is applied by generating two uniform distributions u_1 and u_2 of pseudo-random numbers, and then applying equation (3.12) of (3.13) to generate a Gaussian distributed variable.

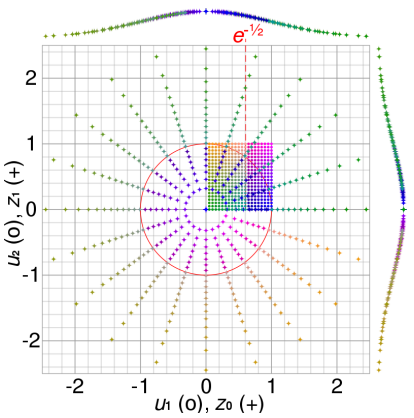

Figure 3.1: Box-Muller graphical interpretation. The coloured points in the unit square (u_1, u_2), drawn as circles, are mapped to a 2D Gaussian $(z_0 \equiv x, z_1 \equiv y)$, drawn as crosses. The plots at the margins are the probability distribution functions of z_0 and z_1. Note that z_0 and z_1 are unbounded; they appear to be in [-2.5,2.5] due to the choice of the illustrated points.

The independence of the variables x and y (which are distributed as independent standard normals thanks to the formulas (3.12) and (3.13)) can be demonstrated following the original proof of Box and Muller.[26] Let u_1, u_2 be independent random variables from the same rectangular density function on the interval $(0, 1)$. Starting from expressions (3.12) and (3.13), and solving these equations with respect to the variables u_1 and u_2, leads to

$$\frac{y}{x} = \tan(2\pi u_1) \quad \rightarrow \quad u_1 = \frac{1}{2\pi}\arctan\left(\frac{y}{x}\right) \tag{3.14}$$

$$x^2 + y^2 = -2\ln u_2 \quad \rightarrow \quad u_2 = e^{-(x^2+y^2)/2} \tag{3.15}$$

The Jacobian matrix is given by

$$\mathbf{J}(x,y) = \frac{\partial(u_1,u_2)}{\partial(x,y)} = \begin{pmatrix} \partial_x u_1 & \partial_y u_1 \\ \partial_x u_2 & \partial_y u_2 \end{pmatrix} = \begin{pmatrix} y/[2\pi(x^2+y^2)] & -x/[2\pi(x^2+y^2)] \\ -x\exp[-(x^2+y^2)/2] & -y\exp[-(x^2+y^2)/2] \end{pmatrix} \tag{3.16}$$

where $\partial_x u_1 = \partial u_1/\partial x$, $\partial_y u_1 = \partial u_1/\partial y$ and so for the partial derivatives related to the variable u_2. The determinant of the Jacobian is therefore

$$\det(\mathbf{J}) = \frac{(-y^2 - x^2)}{2\pi(x^2+y^2)}\, e^{-(x^2+y^2)/2} = -\frac{1}{2\pi}\, e^{-(x^2+y^2)/2} \tag{3.17}$$

It follows that the joint density of the variables x, y is given by the following transformation relation

$$f(x,y) = f(u_1,u_2)\,|\det(\mathbf{J})| = |\det(\mathbf{J})| = \frac{1}{2\pi}\, e^{-(x^2+y^2)/2} = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}\ \frac{1}{\sqrt{2\pi}}\, e^{-y^2/2} \tag{3.18}$$

so that it can be written as the product of two probability distribution,

$$f(x,y) = f(x)f(y) \qquad \text{where} \quad f(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2} \quad \text{and} \quad f(y) = \frac{1}{\sqrt{2\pi}}\, e^{-y^2/2} \tag{3.19}$$

Hence the variables x and y are independent normal deviates, i.e. they are normally distributed with two independent Gaussian distributions.
The above approach is motivated by the following considerations: the probability density of $f(x,y)$ is constant on circles, so $\theta = \arctan(y/x)$ is uniformly distributed $(0, 2\pi)$. Further, the square of the length of the radius vector $R^2 = x^2 + y^2$ has a chi-squared distribution with two degrees of freedom. If u_2 has a rectangular density on $(0,1)$, then $-2\ln(u_2)$ has a chi-squared distribution with two degrees of freedom. Proceeding in the reverse order, the equations (3.12) and (3.13) are recovered.
The Box-Muller algorithm is very useful for the initialization of nuclear velocities in a Molecular Dynamics simulations. These initial nuclear velocities, together with the initial nuclear positions, define the initial conditions for the integration of nuclear equations of motion, and are therefore very important. In the implementation of Box-Muller algorithm, two sets of pseudo-random numbers u_{i1} and u_{i2} are generated,

$$u_{i1}, u_{i2} \in (0,1) \qquad \text{with} \quad i = 1, ..., 3N \tag{3.20}$$

where N is the number of atoms in the system. Then equation (3.12) is applied

$$u_i = R\,\cos\Theta = \sqrt{-2\,\ln u_{i1}}\;\cos(2\pi u_{i2}) \qquad\qquad i = 1, ..., 3N \tag{3.21}$$

in order to obtain $3N$ variables u_i $(i = 1, ..., 3N)$ standard normally distributed (as previously demonstrated), so that each variable u_i respects the relation

$$f(u_i) = \frac{1}{\sqrt{2\pi}}\, e^{-u_i^2/2} \qquad\qquad i = 1, ..., 3N \tag{3.22}$$

and the probability density function of the collection of variables $u = \{u_i\ (i = 1, ..., 3N)\}$ is given by

$$f(u) = \frac{1}{\sqrt{2\pi}}\, e^{-u^2/2} \tag{3.23}$$

Then, the variables u_i $(i = 1, ..., 3N)$ are multiplied by the factor $\sqrt{k_b T_a / m_i}$, where T_a is the initial temperature given by the distribution of the random numbers u_i and m_i is the mass of the i-th atom, so that the velocity components of the i-th atom can be obtained as

$$v_{ij} = u_{3(i-1)+j}\,\sqrt{\frac{k_b T_a}{m_i}} \qquad\qquad i = 1, ..., N \quad \text{and} \quad j = 1, 2, 3 \tag{3.24}$$

where $j = 1, 2, 3$ are the three Cartesian directions x, y, z. Reducing the indexes of the nuclear velocities and masses to one index, in order to obtain a one-to-one more immediate correspondence between v_i, m_i and u_i (so that v_1, v_2 and v_3 will be respectively the x, y and z velocity components of the first atom, and so on until v_{3N} that is the z velocity component of the N-th atom, while $m_1 = m_2 = m_3$ is the mass of the first atom until $m_{3N-2} = m_{3N-1} = m_{3N}$ that is the mass of the N-th atom), lead to the following expression between variable u_i and the nuclear velocities

$$v_i = u_i\,\sqrt{\frac{k_b T_a}{m_i}} \qquad \rightarrow \qquad u_i = v_i\,\sqrt{\frac{m_i}{k_b T_a}} \qquad\qquad i = 1, ..., 3N \tag{3.25}$$

Therefore, the change of variable between the probability density function (3.23) and the probability density function for the collection of the atomic velocities v_j $(j = x, y, z)$ will be ruled by the transformation

$$v_j = u\,\sqrt{\frac{k_b T_a}{m}} \qquad \rightarrow \qquad u = v_j\,\sqrt{\frac{m}{k_b T_a}} \qquad\qquad j = x, y, z \tag{3.26}$$

The conversion factor $\sqrt{k_b T_a / m}$ here introduced can be now justified to give the correct units to the atomic velocities v_i, starting from dimensionless quantities such as the random numbers u_i. However, it has a more profound meaning, as discussed in Appendix A, Section A.1.

Let u_i be a real-valued random variable with probability density function $f(u)$ and let $v_j = h(u)$ $(j = x, y, z)$ with h a strictly increasing continuously differentiable function with inverse $u = g(v_j)$ $(j = x, y, z)$ (see for example the transformation (3.26)), then $v_i = h(u_i)$ will have a continuous distribution too, with a probability density function given by

$$f(v_j) = \frac{f(u(v_j))}{\det[\mathbf{J}(u(v_j))]} = \frac{du}{dv_j} f(u(v_j)) \qquad j = x, y, z \tag{3.27}$$

Applying the change of variable (3.27) to the probability density function (3.23) leads to the following probability density function for the whole collection of each component of the velocities of N atoms with mass m is distributed as a normalized Gaussian

$$f(v_j) = \sqrt{\frac{m}{2\pi k_b T_a}}\, e^{-mv_j^2/(2k_bT_a)} \qquad j = x, y, z \tag{3.28}$$

so that the distribution followed by each atomic velocities in the system can be written as

$$f(v_i) = \sqrt{\frac{m_i}{2\pi k_b T_a}}\, e^{-m_i v_i^2/(2k_bT_a)} \qquad i = 1, ..., 3N \tag{3.29}$$

The distribution in (3.28) represents a normalized Gaussian distribution, and it is related to *each* component of the nuclear velocity. Therefore, written more explicitly, the three distributions describing the collection component-by-component of the atomic velocities $v_j = \{v_{j,k}\}$ with $k = 1, ..., N$ and $j = x, y, z$ are

$$f(v_x) = \sqrt{\frac{m}{2\pi k_b T_a}}\, e^{-mv_x^2/(2k_bT_a)} \tag{3.30}$$

$$f(v_y) = \sqrt{\frac{m}{2\pi k_b T_a}}\, e^{-mv_y^2/(2k_bT_a)} \tag{3.31}$$

$$f(v_z) = \sqrt{\frac{m}{2\pi k_b T_a}}\, e^{-mv_z^2/(2k_bT_a)} \tag{3.32}$$

Multiplying together the three distributions (3.30)-(3.32), a probability density function can be defined, which describes the probability of finding a particle with a speed near a given value $\boldsymbol{v} = (v_x, v_y, v_z)$ in the system formed by a collection of atoms all with mass m, that is

$$F(v_x, v_y, v_z) = f(v_x)f(v_y)f(v_z) = \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} e^{-m(v_x^2+v_y^2+v_z^2)/(2k_bT_a)} \tag{3.33}$$

The probability density function (3.33) is seen to be the product of the distributions of three independent normally distributed variables $v_{x,k}$, $v_{y,k}$ and $v_{z,k}$ $(k = 1, ..., N)$, with variance k_bT_a/m. In particular, the probability of finding a particle with velocity in the infinitesimal element $[dv_x, dv_y, dv_z]$ about velocity $\boldsymbol{v} = (v_x, v_y, v_z)$ can be computed as

$$F(\boldsymbol{v})\, dv_x\, dv_y\, dv_z \equiv F(v_x, v_y, v_z)\, dv_x\, dv_y\, dv_z = \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} e^{-m(v_x^2+v_y^2+v_z^2)/(2k_bT_a)}\, dv_x\, dv_y\, dv_z \tag{3.34}$$

The same form for the three distributions (3.30)-(3.32) for the three spatial components of the atomic velocities is a direct consequence of the rotational symmetry. However, rotational invariance also requires that the full distribution does not depend on the direction of the velocity; it can only depend on the speed (i.e. on the modulus of the velocity vector)

$$v = \|\boldsymbol{v}\| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{3.35}$$

Indeed, since the function $F(v_x, v_y, v_z)$ in (3.34) depends only on the velocity modulus (3.35), it can be written as $\tilde{F}(v)$, a function of the velocity modulus only, that is

$$F(v_x, v_y, v_z)\, dv_x\, dv_y\, dv_z = \tilde{F}(v)\, dv_x\, dv_y\, dv_z = \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} e^{-mv^2/(2k_bT_a)}\, dv_x\, dv_y\, dv_z \tag{3.36}$$

The probability (3.36) is expressed in Cartesian coordinates in the space of the velocity. However, the variable v defined by (3.35) has the form of a typical radial variable, and it is therefore more convenient to rewrite the probability (3.36) using spherical coordinates. The change of variable from Cartesian to spherical coordinates can lead to a probability function $F(v)\,dv$ that depends only on the velocity variable (3.35), and not on the velocity vectors components, thus satisfying the space rotational invariance required. The change of variable is given by

$$d\boldsymbol{v} \equiv dv_x\,dv_y\,dv_z = v^2\,\sin(\theta)\,dv\,d\theta\,d\phi \tag{3.37}$$

where v is the modulus of the velocity, given by (3.35), while θ and ϕ are the polar and the azimuthal angles, respectively, which both determine the direction of the velocity vector in the space. The probability distribution function for the modulus of the nuclear velocities is defined through the integration

$$F(v)\,dv = \int_0^{\pi}\int_0^{2\pi} \tilde{F}(v)\,v^2\,\sin(\theta)\,d\theta\,d\phi\,dv$$

where in the last equivalence the change of variable (3.37) has been introduced. Furthermore, since $F(\boldsymbol{v}) \equiv F(v_x, v_y, v_z)$, given by equation (3.36), depends only on $\|\boldsymbol{v}\| \equiv v$, so that $F(\boldsymbol{v}) = \tilde{F}(v)$, the previous integral reduces to

$$F(v)\,dv = \tilde{F}(v)\,v^2\,dv\int_0^{\pi}\int_0^{2\pi} sin\theta\,d\theta\,d\phi = \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} e^{-mv^2/(2k_bT_a)}\;v^2\,dv\int_0^{\pi}\int_0^{2\pi} sin\theta\,d\theta\,d\phi$$

Integrating with respect to the solid angles $d\Omega = d\theta\,d\phi$ yields an additional factor of 4π,

$$F(v)\,dv = \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} e^{-mv^2/(2k_bT_a)}\;v^2\,dv\underbrace{\int_0^{\pi}\int_0^{2\pi} sin\theta\,d\theta\,d\phi}_{=\,4\pi} = 4\pi\left(\frac{m}{2\pi k_b T_a}\right)^{3/2} v^2\,e^{-mv^2/(2k_bT_a)}\,dv$$

Therefore, the probability (3.36) can be rewritten in spherical coordinates as

$$\tilde{F}(v)\,dv_x\,dv_y\,dv_z = 4\pi\,v^2\,\tilde{F}(v)\,dv = F(v)\,dv \tag{3.38}$$

and it gives the probability to find a particle in the atomic system with a velocity modulus whose value lies in the range $(v, v+dv)$. As a consequence, the probability density function for the modulus of the nuclear velocity vector (associated to a spherical coordinates reference frame) is obtained to be

$$F(v) = 4\pi\left(\frac{m}{2\pi k_b T_a}\right)^{3/2} v^2\,e^{-mv^2/(2k_bT_a)} \tag{3.39}$$

The function (3.39) obtained in this way is the so-called normalized Maxwell-Boltzmann distribution, a continuous probability distribution for the variable v (i.e. for the modulus of the velocity vector), describing particle speeds in idealized gases, where the particles move freely inside a stationary container without interacting with one another, except for very brief collisions in which they exchange energy and momentum with each other or with their thermal environment. Remarkably, the Maxwell distribution also holds in the presence of any interactions. In fact, Maxwell original derivation of the distribution makes no reference to any properties of the gas.
Mathematically, the Maxwell-Boltzmann probability function is the distribution of the positive square root of the sum of squares of three independent random variables $(v_{x,k}, v_{y,k}, v_{z,k})$, with $k = 1, ..., N$, each following a normalized Gaussian (normal) distribution as in equations (3.30)-(3.32), or equivalently, it is the distribution of the Euclidean distance of the three random variables (given by the three velocity components per each atom) from the origin. Thus, it can be stated that, from a mathematical point of view, the Maxwell-Boltzmann distribution is a chi distribution with three degrees of freedom (the components of the velocity vector in Euclidean space) and scale parameters $\lambda = \sqrt{k_bT_a/m}$ measuring speeds in units proportional to the square root of the ratio between the system temperature T_a and the particle mass m. Examples of initial nuclear velocities distributions for two representative systems are

reported and discussed in Appendix A, Section A.1.1.
Note that the probability density function (3.23) is normalized thanks to the identity (3.1),

$$\int_{-\infty}^{\infty} f(u)\,du = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-u^2/2}\,du = 1$$

In the same way, the three probability density functions in (3.28) are also normalized, indeed

$$\int_{-\infty}^{\infty} f(v_j)\,dv_j = \sqrt{\frac{m}{2\pi k_b T_a}} \int_{-\infty}^{\infty} e^{-mv_j^2/(2k_bT_a)}\,dv_j = \sqrt{\frac{m}{2\pi k_b T_a}} \sqrt{\frac{2\pi k_b T_a}{m}} = 1 \qquad\qquad j = x, y, z$$

Finally, note that the distributions and the probability density functions described in this section are all associated to atomic systems with only one atomic species (indeed, only the mass variable m has been used). However, extension to atomic systems with a generic number of atomic species is straightforward, and it has been discussed in Appendix A, Section A.1.1.

3.1.1 Subtraction of total linear momentum

After the nuclear velocities have been initialized, the total linear momentum of the system will probably have some finite value other than zero. As the total linear momentum of the system is (approximately) conserved within a molecular dynamics simulation, this would result in the system drifting away into one direction during the course of the simulation, which is probably not desired. Therefore, the total momentum is explicitly set to zero after the Maxwell-Boltzmann initialization. The equations used and the procedure followed to set the system total linear momentum equal to zero at the beginning of a dynamics simulation are derived and explained in Section 3.2.1. This modification of nuclear velocities might, of course, change the initial temperature. Therefore, a final step is performed, in which all nuclear velocity vectors are multiplied with a factor that is determined such that the initial temperature exactly matches the target value, as described in Section 3.1.2.

3.1.2 Velocity rescaling with respect to target temperature

After the random initialization of the nuclear velocities following the Box-Muller algorithm and the subtraction of the total linear momentum (i.e. the component-by-component subtraction of the non-zero translational velocity to the initial nuclear velocities), the temperature T_a that enters in the Maxwell-Boltzmann distribution (3.39) and which is given by the initial kinetic energy will not necessary be equal to the target temperature T_0 imposed as initial temperature for the molecular dynamics simulation. Therefore, in order to match the initial target temperature, the nuclear velocities have to be rescaled. The calculation of the rescaling factor proceeds as follows. First of all, the kinetic energy correspondent to the initial nuclear velocities is computed

$$\text{Kinetic energy :} \qquad E_k(t_0) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{3} m_i\, v_{ij}^2(t_0) \qquad \text{where} \quad j = x, y, z(\equiv 1, 2, 3) \tag{3.40}$$

and the correspondent temperature T_a is given as a function of the kinetic energy through the formula

$$\text{Temperature :} \qquad T_a = \frac{2}{g k_b}\, E_k(t_0) \tag{3.41}$$

where g are the system degrees of freedom and k_b is the Boltzmann constant. Gathering together the previous two equations, the following relation is obtained

$$T_a = \frac{1}{g k_b} \sum_{i=1}^{N} \sum_{j=1}^{3} m_i\, v_{ij}^2(t_0) \tag{3.42}$$

However, the desired initial velocities $\{\tilde{v}_{ij}\}$ have to respect the condition

$$T_0 = \frac{1}{g k_b} \sum_{i=1}^{N} \sum_{j=1}^{3} m_i\, \tilde{v}_{ij}^2(t_0) \tag{3.43}$$

that can be obtained by multiplying the equation (3.42) by the factor

$$\frac{T_0}{T_a}\,T_a = T_0 = \frac{T_0}{T_a}\frac{1}{gk_b}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ij}^2(t_0) = \frac{1}{gk_b}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\left[\sqrt{\frac{T_0}{T_a}}\, v_{ij}(t_0)\right]^2 = \frac{1}{gk_b}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, \tilde{v}_{ij}^2(t_0)$$

so that the initial velocities $\{\tilde{v}_{ij}\}$ that leads to the correct initial target temperature T_0 can be simply obtained from the initial velocities (3.24) computed through the Box-Muller algorithm as follows

$$\text{Initial temperature scaling :} \qquad v_{ij}(t_0) \leftarrow v_{ij}(t_0)\sqrt{\frac{T_0}{T_a}} = \tilde{v}_{ij}(t_0) \qquad i = 1, ..., N \qquad j = x, y, z \tag{3.44}$$

This operation is performed at the beginning of every molecular dynamics simulation, just after the Box-Muller initialization of the nuclear velocities. The result is a set of initial nuclear velocities $\{v_{ij}\}$ that leads, following the equations (3.43) and (3.44), to a temperature T_0 equal to the target one.
As a consequence of this temperature rescaling performed at the beginning of each molecular dynamics simulation, the multiplicative factor $\sqrt{k_b T_a}$ applied to the standard normal distribution of random numbers $\{u_i\}$ obtained with the Box-Muller algorithm in order to maintain the correct physical dimensions of the nuclear velocities, see equation (3.24), becomes a non essential and fictitious multiplicative factor, which can be ignored in practical implementations (a rigorous demonstration that justifies this sentence is reported in Appendix A, Section A.1.2).

3.2 Translations and rotations removal by initial velocity rescaling

The modeling of condensed matter system is usually performed with the so-called periodic boundary conditions, which enables to simulate an infinite periodic system by means of the replication of a single unit cell. However, the inclusion of geometrical constraints during a simulation affects the statistical mechanics of the sampled microstates.[27] This is mainly because in the presence of such constraints, the kinetic energy of the system cannot be written in a configuration-independent way (unless the constraints are exclusively involved in fully-rigid atom groups, e.g., rigid molecules).
In the absence of stochastic and frictional forces, a few degrees of freedom are not coupled (i.e., do not exchange kinetic energy) with the internal degrees of freedom of the system. These external degrees of freedom correspond to the system rigid-body translation and, for non-fully periodic systems such as molecules or polymers, to the system rigid-body rotation. Because the kinetic energy associated with these external degrees of freedom can take an arbitrary (constant) value determined by the initial atomic velocities, they must be removed from the definition of the system internal quantities computed in (as for example the temperature). Consequently, the number of internal degrees of freedom h is calculated as three times the total number of atoms N in the system, minus the number n_c of geometrical constraint imposed, that is

$$h = 3N - n_c \tag{3.45}$$

The subtraction of constrained degrees of freedom is necessary because geometrical constraints are characterized by a time-independent generalized position associated with a vanishing generalized momentum (i.e., no kinetic energy). A more formal statistical-mechanical justification for the subtraction of the external degrees of freedom in the case of periodic boundary conditions can be found in Ref. [28]. When stochastic and frictional forces are applied, as in Berendsen thermostat, these forces will couple the rigid-body translational and rotational degrees of freedom with the internal ones. In this case all degrees of freedom are considered internal to the system. Thus, equation (3.45) has to be used with $n_c = 0$ in the presence of stochastic and frictional forces, with $n_c = 3$ under periodic boundary conditions, or with values in the range $3 < n_c < 6$ if also the rotation of the system along some directions are removed (as for molecules and polymers, see Section 3.2.2).

3.2.1 Removal of atomic systems translations

Consider a system of N atoms in three dimensions with nuclear initial positions $\{\tilde{\boldsymbol{q}}_i\}$ and initial velocities $\{\tilde{\boldsymbol{v}}_i\}$. In order to remove the translational movement of the system, the nuclear velocities have to be rescaled with respect to the center of mass. The position of the center of mass $\boldsymbol{q}_{cm}$ of a system with N nuclei at positions $\{\tilde{\boldsymbol{q}}_i\}$ is defined as

$$\boldsymbol{q}_{cm} = \frac{\sum_{i=1}^{N} m_i \tilde{\boldsymbol{q}}_i}{\sum_{i=1}^{N} m_i} = \frac{1}{m} \sum_{i=1}^{N} m_i \tilde{\boldsymbol{q}}_i \tag{3.46}$$

where the total mass of the system m has been defined as

$$m = \sum_{i=1}^{N} m_i \tag{3.47}$$

By deriving expression (3.46) with respect to time, the velocity of the center of mass $\boldsymbol{v}_{cm}$ is obtained as

$$\boldsymbol{v}_{cm} = \frac{d\boldsymbol{q}_{cm}}{dt} = \frac{d}{dt}\left(\frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{q}}_i\right) = \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{v}}_i \tag{3.48}$$

Finally, the component-by-component subtraction of the center of mass velocity to each nuclear velocities,

$$\boldsymbol{v}_i \equiv \boldsymbol{v}_{i,t} = \tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm} \qquad i = 1, ..., N \tag{3.49}$$

remove the translation movement of the atomic system with respect to its center of mass. The velocities $\{\boldsymbol{v}_i\}$ defined by equation (3.49) are called internal nuclear velocity coordinates (if only the translations

with respect to the center of mass are removed) and are used to define the kinetic energy and therefore the temperature of the system (see Section 3.3). To understand why the system net linear momentum is set to zero by scaling the nuclear velocities as (3.49), its expression can be computed starting from the translated velocities definition (3.49) as

$$\mathbf{P} = \sum_{i=1}^{N} m_i \boldsymbol{v}_i = \sum_{i=1}^{N} m_i(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) = \sum_{i=1}^{N} m_i \tilde{\boldsymbol{v}}_i - m\boldsymbol{v}_{cm} = m\boldsymbol{v}_{cm} - m\boldsymbol{v}_{cm} = \mathbf{0} \tag{3.50}$$

where for the penultimate equivalence in (3.50) the definition of center of mass velocity (3.48) has been used. Therefore, through equation (3.49) the velocity of the center of mass is set to zero for the system of N nuclei, so that the total linear momentum is set to zero just after the random initialization of the nuclear velocities, before performing the first step of a molecular dynamics simulation. The total linear momentum of the atomic system is a conserved quantity in a molecular dynamics simulation, since it is a first integral of the Hamilton equation of motion (i.e. it is conserved by the Hamiltonian flow). Therefore, the value of zero set for the linear momentum by means of equation (3.49) in the initialization process should theoretically remain equal to zero during all the molecular dynamics run. Thus, the number of degrees of freedom in a simple molecular dynamic run is $g = 3N - 3$, where the subtraction of three degrees of freedom is a consequence of the linear momentum three-components conservation during the system evolution.

3.2.2 Removal of molecular systems rotations

Consider a system of N atoms in a three dimensional space with nuclear initial positions $\{\tilde{\boldsymbol{q}}_i\}$ and initial velocities $\{\tilde{\boldsymbol{v}}_i\}$. Starting from these initial coordinates, a new set of nuclear positions $\{\boldsymbol{q}_{i,t}\} \equiv \{\boldsymbol{q}_i\}$ and velocities $\{\boldsymbol{v}_{i,t}\}$ can be defined through a translation by the center of mass position and velocity, respectively, as previously mentioned in Section 3.2.1, so that

$$\boldsymbol{q}_i \equiv \boldsymbol{q}_{i,t} = \tilde{\boldsymbol{q}}_i - \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{q}}_i = \tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm} \qquad i = 1, ..., N \tag{3.51}$$

$$\boldsymbol{v}_{i,t} = \tilde{\boldsymbol{v}}_i - \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{v}}_i = \tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm} \qquad i = 1, ..., N \tag{3.52}$$

where m is the total mass of the atomic system defined in (3.47) and the positions $\{\boldsymbol{q}_i\}$ defined by equation (3.51) are called internal nuclear position coordinates. The total angular momentum of the collection of N nuclear particles is the sum of the angular momentum of each nucleus

$$\mathbf{L} = \sum_{i=1}^{N} m_i(\boldsymbol{q}_{i,t} \wedge \boldsymbol{v}_{i,t}) \equiv \sum_{i=1}^{N} m_i(\boldsymbol{q}_i \wedge \boldsymbol{v}_{i,t}) \tag{3.53}$$

Inserting the equations (3.51) and (3.52) in the expression of the total angular momentum (3.53), the following relations can be obtained

$$\begin{aligned}
\mathbf{L} &= \sum_{i=1}^{N} m_i[(\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm}) \wedge (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm})] \\
&= \sum_{i=1}^{N} m_i\, \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{v}}_i - \sum_{i=1}^{N} m_i\, \tilde{\boldsymbol{q}}_i \wedge \boldsymbol{v}_{cm} - \sum_{i=1}^{N} m_i\, \boldsymbol{q}_{cm} \wedge \tilde{\boldsymbol{v}}_i + \sum_{i=1}^{N} m_i\, \boldsymbol{q}_{cm} \wedge \boldsymbol{v}_{cm} \\
&= \sum_{i=1}^{N} m_i\, \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{v}}_i - m\, \boldsymbol{q}_{cm} \wedge \boldsymbol{v}_{cm} - \boldsymbol{q}_{cm} \wedge (m\, \boldsymbol{v}_{cm}) + m\, \boldsymbol{q}_{cm} \wedge \boldsymbol{v}_{cm} \\
&= \sum_{i=1}^{N} m_i\, \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{v}}_i - m\, \boldsymbol{q}_{cm} \wedge \boldsymbol{v}_{cm}
\end{aligned} \tag{3.54}$$

The first term of the last expression is the angular momentum of the particles moving relative to the center of mass, while the second term is the angular momentum of the center of mass relative to the origin. The result is that the equation (3.53) can be rewritten as

$$\mathbf{L} = \sum_{i=1}^{N} m_i(\boldsymbol{q}_{i,t} \wedge \boldsymbol{v}_{i,t}) \equiv \sum_{i=1}^{N} m_i(\boldsymbol{q}_i \wedge \boldsymbol{v}_{i,t}) = \sum_{i=1}^{N} m_i\, \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{v}}_i - m\, \boldsymbol{q}_{cm} \wedge \boldsymbol{v}_{cm} \tag{3.55}$$

which identify the angular momentum of the atomic system with respect to its center of mass angular momentum. The angular momentum (3.55) is important since it is related to the angular velocity $\boldsymbol{\omega}$ of the system through the so-called moment of inertia $\mathbf{I}$ by the equation

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega} \tag{3.56}$$

where the angular momentum $\mathbf{L}$ and angular velocity $\boldsymbol{\omega}$ about the center of mass are 3×1 column vectors and the moment of inertia $\mathbf{I}$ is a 3×3 symmetric matrix whose components are defined as

$$I_{ij} = \sum_{k=1}^{N} m_k[\|\boldsymbol{q}_{k,t}\|^2 \delta_{ij} - (r_{k,t})_i (r_{k,t})_j] \qquad \text{with} \quad i,j = x,y,z \tag{3.57}$$

where $(r_{k,t})_i$ are the positions $((r_{k,t})_x, (r_{k,t})_y, (r_{k,t})_z)$ of the k-th atom computed with respect to the center of mass, i.e. the components of the position vector $\boldsymbol{q}_{k,t}$ for the k-th atom, see equation (3.51). Using Cartesian coordinates, $((r_{k,t})_x, (r_{k,t})_y, (r_{k,t})_z) = (x_{k,t}, y_{k,t}, z_{k,t})$, so that the moment of inertia is given by the matrix

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} = \sum_{k=1}^{N} m_k \begin{pmatrix} y_{k,t}^2 + z_{k,t}^2 & -x_{k,t}\, y_{k,t} & -x_{k,t}\, z_{k,t} \\ -y_{k,t}\, x_{k,t} & x_{k,t}^2 + z_{k,t}^2 & -y_{k,t}\, z_{k,t} \\ -z_{k,t}\, x_{k,t} & -z_{k,t}\, y_{k,t} & x_{k,t}^2 + y_{k,t}^2 \end{pmatrix} \qquad \text{with} \quad \boldsymbol{q}_{k,t} = (x_{k,t}, y_{k,t}, z_{k,t})$$

so that equation (3.56) can be written more explicitly as

$$\begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix} = \sum_{k=1}^{N} m_k \begin{pmatrix} y_{k,t}^2 + z_{k,t}^2 & -x_{k,t}\, y_{k,t} & -x_{k,t}\, z_{k,t} \\ -y_{k,t}\, x_{k,t} & x_{k,t}^2 + z_{k,t}^2 & -y_{k,t}\, z_{k,t} \\ -z_{k,t}\, x_{k,t} & -z_{k,t}\, y_{k,t} & x_{k,t}^2 + y_{k,t}^2 \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.58}$$

The angular momentum and the momentum of inertia of the atomic system can then be used to compute the angular velocity $\boldsymbol{\omega}$ of the entire system, with the following equation

$$\boldsymbol{\omega} = \mathbf{I}^{-1} \cdot \mathbf{L} \quad \rightarrow \quad \omega_i = \sum_{j}^{(x,y,z)} {I_{ij}}^{-1}\, L_j \qquad \text{with} \quad i = x,y,z \tag{3.59}$$

where I_{ij}^{-1} are the elements of the inverse matrix $\mathbf{I}^{-1}$, computing by inverting the 3×3 matrix $\mathbf{I}$. At this point, the velocity $\{\boldsymbol{v}_{i,r}\}$ of each nucleus due to the angular motion of the system can be calculated using the relation

$$\boldsymbol{v}_{i,r} = \boldsymbol{\omega} \wedge \boldsymbol{q}_{i,t} \equiv \boldsymbol{\omega} \wedge \boldsymbol{q}_i = (\omega_y\, z_{i,t} - \omega_z\, y_{i,t},\ \omega_z\, x_{i,t} - \omega_x\, z_{i,t},\ \omega_x\, y_{i,t} - \omega_y\, x_{i,t}) \qquad i = 1, ..., N \tag{3.60}$$

where $\boldsymbol{q}_{i,t} = (x_{i,t}, y_{i,t}, z_{i,t})$ are the nuclear positions translated with respect to the center of mass (the subscript t stands for translation) expressed in Cartesian components and the new defined nuclear velocities $\{\boldsymbol{v}_{i,r}\}$ are due to system angular motion which is caused by rotation of the atomic system itself (indeed the subscript r stands for rotation). Finally, subtracting the set of velocities $\{\boldsymbol{v}_{i,r}\}$ due to system angular motion to the set of velocities $\{\boldsymbol{v}_{i,t}\}$ defined in equation (3.52), a new set of velocities is obtained, defined as

$$\boldsymbol{v}_i \equiv \boldsymbol{v}_{i,tr} = \boldsymbol{v}_{i,t} - \boldsymbol{v}_{i,r} = (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{v}_{i,r} = (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{\omega} \wedge \boldsymbol{q}_i \qquad i = 1, ..., N \tag{3.61}$$

where in the last expression the velocities $\{\boldsymbol{v}_{i,t}\}$ are written using equation (3.52) as a function of the initial nuclear velocities $\{\tilde{\boldsymbol{v}}_i\}$ and the center of mass velocity. The velocities $\{\boldsymbol{v}_i\}$ defined by equation

(3.61) are called internal nuclear velocity coordinates and are used to define the kinetic energy and therefore the temperature of the system (see Section 3.3). To understand why the system net linear momentum *and* the net angular momentum are both set to zero by scaling the nuclear velocities as in (3.61), their expression can be computed starting from the translated and rotated velocities definition (3.61) as

$$\begin{aligned}\mathbf{P} &= \sum_{i=1}^{N} m_i \boldsymbol{v}_i = \sum_{i=1}^{N} m_i[(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{\omega} \wedge \boldsymbol{q}_i] = \sum_{i=1}^{N} m_i[(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{\omega} \wedge (\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm})] \\ &= \sum_{i=1}^{N} m_i[(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm})] = \mathbf{0}\end{aligned} \tag{3.62}$$

$$\begin{aligned}\mathbf{L} &= \sum_{i=1}^{N} m_i \, \boldsymbol{q}_i \wedge \boldsymbol{v}_i \equiv \sum_{i=1}^{N} m_i \, \boldsymbol{q}_{i,t} \wedge \boldsymbol{v}_{i,tr} = \sum_{i=1}^{N} m_i \, \boldsymbol{q}_{i,t} \wedge [(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{\omega} \wedge \boldsymbol{q}_i] \\ &= \sum_{i=1}^{N} m_i \, \boldsymbol{q}_{i,t} \wedge [(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - \boldsymbol{\omega} \wedge (\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm})] = \sum_{i=1}^{N} m_i \, \boldsymbol{q}_{i,t} \wedge [(\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm}) - (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm})] = \mathbf{0}\end{aligned} \tag{3.63}$$

where the penultimate equivalence in both (3.62) and (3.63) is proved by using $\boldsymbol{\omega} \wedge (\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm}) = (\tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm})$. Thus, the system net linear and angular momenta of the internal velocities vanish, as expected.
These velocities (3.61) are therefore obtained starting from the random initial nuclear velocities and subtracting from them the velocity of the center of mass of the system (removing translational motions) and the nuclear velocities components due to the angular momentum of the system (removing rotational motions). This task is accomplished just after the random initialization of the velocities in a molecular dynamics simulation. The total angular momentum of the atomic system is a conserved quantity in a molecular dynamics simulation, since it is a first integral of the Hamilton equation of motion (i.e. it is conserved by the Hamiltonian flow). Therefore, the value of zero set for the angular momentum by means of equation (3.61) in the initialization process should theoretically remain equal to zero during all the molecular dynamics run. However, if the system is modeled using periodic boundary conditions, the net angular momentum of the system is not conserved.[29] At the same time, if a system is modeled in a unit cell with periodic boundary conditions, it can be imagined as an infinite system along all the periodic directions, so that the rotations of the system are not allowed along these directions by the periodic boundary conditions themselves.
If the system is an isolated molecule, then the effect of system rotation has to be removed along each one of the three directions in space, so that the final result is the removal of the net angular momentum, and the procedure to follow uses precisely the equations reported and described before. In this case, the net angular momentum is a conserved quantity for Hamilton equation of motion that describes the evolution of the system coordinates (since no periodic boundary conditions are applied in this case). Therefore, equation (3.61) should be used just after the random initialization of the velocity before the first step of the molecular dynamics simulation.
Otherwise, if the atomic system studied is a polymer, the rotation can be performed only along the x axis, since the system is free to rotate only along this axis, while in the other two non-periodic directions y, z the system is prevented from rotating due to the periodic boundary conditions to which the unit cell is constrained. Also in this case the net angular momentum is a conserved quantity for Hamilton equation of motion along the x non periodic direction, so that equation (3.61) should be applied only one time at the beginning of the simulation. The particular case of polymers (1D systems) are treated in Section 3.2.3, where the equations needed to remove the rotation along the periodic x axis are reported. Finally, for the case of slab and crystal systems, the periodic boundary conditions imposed on the unit cell do not allow the systems to rotate along any axis, so that for these cases no conditions for eliminating the angular momentum are introduced. Because the total angular momentum is not conserved due to the presence of periodic boundary conditions,[29] in 2D and 3D systems is not essential to set the initial value of this quantity to zero (i.e. the formula (3.61) for the definition of initial velocities is not used for these two cases). The four different cases are schematized in Table 3.1.

System	Dimensionality	N_{rot}	Rotation axis	Periodic directions	d	n_c
Molecule	0D	3	x, y, z	–	$3N-6$	6
Polymer	1D	1	x	x	$3N-4$	4
Slab	2D	0	–	x, y	$3N-3$	3
Crystal	3D	0	–	x, y, z	$3N-3$	3

Table 3.1: Dimensionality of the system, number of directions N_{rot} along which the system is free to rotate, directions of rotation, periodic directions, number of degrees of freedom d and number of geometrical constraints n_c if the system would not be allowed neither to translate nor to rotate along the possible rotation directions ($d = 3N - 3$ degrees of freedom in a system not allowed to translate along the three space directions, then in a molecule and a polymer a further 3 and 1 degrees of freedom for rotation along the three and one directions, respectively, have to be subtracted).

3.2.3 Removal of polymer systems rotations

Starting from the nuclear positions (3.51) and the velocity (3.52) translated with respect to the center of mass correspondent quantities, the angular momentum along the x direction is computed as

$$L_x = \left[\sum_{i=1}^{N} m_i(\boldsymbol{q}_{i,t} \wedge \boldsymbol{v}_{i,t})\right] \cdot \mathbf{e}_1 = \sum_{i=1}^{N} m_i(y_{i,t}v_{z,i,t} - z_{i,t}v_{y,i,t}) \tag{3.64}$$

where $\mathbf{e}_1 = (1, 0, 0)$ is a basis vector of the three-dimensional space, $\boldsymbol{q}_{i,t} = (x_{i,t}, y_{i,t}, z_{i,t})$ is the position vector in Cartesian coordinates and $\boldsymbol{v}_{i,t} = (v_{x,i,t}, v_{y,i,t}, v_{z,i,t})$ are the velocities components of the i-th nucleus. Then, since the other two angular momentum components are equal to zero ($L_y = L_z = 0$), the matrix expression (3.58) reduces to

$$\begin{pmatrix} L_x \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \omega_x \\ 0 \\ 0 \end{pmatrix} = \sum_{k=1}^{N} m_k \begin{pmatrix} y_{k,t}^2 + z_{k,t}^2 & -x_{k,t}\, y_{k,t} & -x_{k,t}\, z_{k,t} \\ -y_{k,t}\, x_{k,t} & x_{k,t}^2 + z_{k,t}^2 & -y_{k,t}\, z_{k,t} \\ -z_{k,t}\, x_{k,t} & -z_{k,t}\, y_{k,t} & x_{k,t}^2 + y_{k,t}^2 \end{pmatrix} \begin{pmatrix} \omega_x \\ 0 \\ 0 \end{pmatrix} \tag{3.65}$$

where the angular velocity $\boldsymbol{\omega} = (\omega_x, 0, 0)$ has all the components equal to zero except the one along the x direction, so that the simple equation

$$L_x = I_{xx}\omega_x \quad \rightarrow \quad \omega_x = I_{xx}^{-1} L_x \tag{3.66}$$

relates the angular momentum with the angular velocity trough the moment of inertia. Therefore, after the angular momentum component L_x has been computed by means of equation (3.64), the moment of inertia component I_{xx} can be calculated with the formula

$$I_{xx} = \sum_{k=1}^{N} m_k\,(y_{k,t}^2 + z_{k,t}^2) \tag{3.67}$$

and the angular velocity component along the x direction is derived

$$\omega_x = I_{xx}^{-1} L_x = \left[\sum_{k=1}^{N} m_k\,(y_{k,t}^2 + z_{k,t}^2)\right]^{-1} L_x \tag{3.68}$$

At this point, the velocity $\{\boldsymbol{v}_{i,r}\}$ of each nucleus due to the angular motion of the 1D periodic system can be calculated using the relation

$$\boldsymbol{v}_{i,r} = \boldsymbol{\omega} \wedge \boldsymbol{q}_{i,t} = (\omega_x, 0, 0) \wedge \boldsymbol{q}_{i,t} = (0,\ -\omega_x\, z_{i,t},\ \omega_x\, y_{i,t}) \qquad i = 1, ..., N \tag{3.69}$$

where $\boldsymbol{q}_{i,t} = (x_{i,t}, y_{i,t}, z_{i,t})$ are the nuclear positions translated with respect to the center of mass (the subscript t stands for translation) expressed in Cartesian components and the new defined nuclear velocities $\{\boldsymbol{v}_{i,r}\}$ are due to system angular motion which is caused by rotation of the atomic system itself (indeed the subscript r stands for rotation). Finally, subtracting the set of velocities $\{\boldsymbol{v}_{i,r}\}$ due to

system angular motion to the set of velocities $\{\boldsymbol{v}_{i,t}\}$ defined in equation (3.52), a new set of velocities is obtained, defined as

$$\boldsymbol{v}_i \equiv \boldsymbol{v}_{i,tr} = \boldsymbol{v}_{i,t} - \boldsymbol{v}_{i,r} = (v_{x,i,t},\ v_{y,i,t} + \omega_x\, z_{i,t},\ v_{z,i,t} - \omega_x\, y_{i,t}) \qquad i = 1, ..., N \tag{3.70}$$

The new set of velocities $\{\boldsymbol{v}_{i,tr}\} \equiv \{\boldsymbol{v}_i\}$ are called internal nuclear velocity coordinates and are defined so that the 1D periodic system is prevented from translating along the x, y, z directions and from rotating along the x axis.

3.3 Kinetic energy, net linear and angular momentum

The instantaneous internal kinetic energy for a system of N nuclei is classically defined as

$$E_k(t) = \frac{1}{2}\sum_{i=1}^{N} m_i \boldsymbol{v}_i^2 \tag{3.71}$$

where the internal velocities $\boldsymbol{v}_i$ are obtained from the real initial nuclear velocities $\tilde{\boldsymbol{v}}_i$ by excluding any component along the external degrees of freedom. These corrected internal nuclear velocities are derived and discussed in Section 3.2, and can be resumed equal to as

$$\boldsymbol{v}_i = \begin{cases} \tilde{\boldsymbol{v}}_i & n_c = 0 \\ \tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm} & n_c = 3 \\ \tilde{\boldsymbol{v}}_i - \boldsymbol{v}_{cm} - [\boldsymbol{\omega} \wedge (\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_{cm})]_{x,y,z} & n_c = 4, 5, 6 \end{cases} \qquad i = 1, ..., N \tag{3.72}$$

where $\boldsymbol{\omega}$ is the angular velocity of the center of mass for the system, while $\boldsymbol{q}_{cm}$, $\boldsymbol{v}_{cm}$ are the position and the velocity of the system center of mass, computed as

$$\boldsymbol{q}_{cm} = \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{q}}_i \qquad \boldsymbol{v}_{cm} = \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{\boldsymbol{v}}_i \qquad i = 1, ..., N \tag{3.73}$$

where m is the total mass of the atomic system and $\{\tilde{\boldsymbol{q}}_i\}$ and $\{\tilde{\boldsymbol{v}}_i\}$ are the initial nuclear positions and velocities, respectively. In the last condition in (3.72), the velocity components of each particle due to the rotation of the system can be subtracted along one or more of the three spatial directions x, y, z, so that the number of constraints are equal to 3 (for the three translational degrees of freedom removed) plus the number of subtracted velocity components. Application of conditions (3.72) ensures that, as demonstrated in Sections 3.2.1 and 3.2.2, the system net linear momentum is equal to zero, so that

$$\mathbf{P} = \sum_{i=1}^{N} m_i \boldsymbol{v}_i = \mathbf{0} \qquad \text{for} \quad n_c = 3 \text{ or } 4, 5, 6 \qquad \text{(constraint for 0D, 1D, 2D and 3D systems)} \tag{3.74}$$

and, as demonstrated in Section 3.2.2, irrespective of the origin of the coordinate system, the system net angular momentum is also equal to zero,

$$\mathbf{L} = \sum_{i=1}^{N} m_i \boldsymbol{q}_i \wedge \boldsymbol{v}_i = \mathbf{0} \qquad \text{for} \quad n_c = 4, 5, 6 \qquad \text{(constraint for 0D and 1D systems)} \tag{3.75}$$

Thus, depending on the system dimensionality and on the periodic boundary conditions applied on the system, the net linear and angular momenta of the internal velocities vanish, as expected. In the following discussion, it is assumed that all the equation of motions introduced to describe the evolution of nuclear coordinates in the phase space are applied to the internal velocities defined by equation (3.72), while the nuclear coordinates are propagated simultaneously in time using the real velocities. As demonstrated in Chapter 5, Section 5.1, the integration of the simple set of Newtonian equations of motion (2.33) for an atomic system leads, in the limit of infinite sampling, to a trajectory mapping a microcanonical ensemble of microstates. Assuming an infinite numerical precision, this is also what a standard molecular dynamics simulation will reach. The laws of classical mechanics also lead to two additional conserved

quantities, namely, the linear momentum $\mathbf{P}$ of the system, and the angular momentum $\mathbf{L}$ of the system around its center of mass. In simulations under periodic boundary conditions, the two quantities refer to the infinite periodic system. However, in this case, if the linear momentum $\mathbf{P}_{\text{box}}$ of the computational box is also conserved, the corresponding angular momentum $\mathbf{L}_{\text{box}}$ is not. This is because correlated rotational motion in two adjacent boxes exert friction on each other, leading to an exchange of kinetic energy with the other (internal) degrees of freedom of the system.[29] Note that the physical properties of a molecular system are independent of its total linear momentum. However, they depend on the total angular momentum, because the rotation of the system leads to centrifugal forces. For this reason, the total angular momentum should be added to the list of independent variables defining the ensemble sampled. Whenever the value of the total angular momentum is not given, it generally implicitly means that $\mathbf{L} = \mathbf{0}$. Indeed, the use of $\mathbf{L} \neq \mathbf{0}$ in simulations under periodic boundary conditions (overall uniform rotation of the infinite periodic system) is actually impossible, because it would lead to non-periodic centrifugal forces. Finally, it should be specified that the total energy of the atomic system is defined here so as to exclude the kinetic energy contributions corresponding to the overall translation and rotation of the system (so that the total energy is independent on the total linear and angular momenta). Finally, it should be stressed that computer simulations cannot be performed at infinite numerical precision. As a consequence, quantities which are formally time-independent in classical mechanics may still undergo a numerical drift in simulations. In microcanonical simulations, this is typically the case for the total energy, as well as the total linear momentum $\mathbf{P}$ and the total angular momentum $\mathbf{L}$ (if the system is not periodic), or the total linear momentum $\mathbf{P}$ (if periodic boundary conditions are applied).
In the following section, the Hamilton equations of motions are analyzed and a simple algorithm for integrating them is derived.

Chapter 4

Equations of motion

4.1 Hamilton formulation of the nuclear equations of motion

In molecular dynamics simulation the Newton equations of motion for the nuclei are solved for a set of interacting particles

$$m_i \ddot{\boldsymbol{q}}_i(t) = \mathbf{F}_i(t) = -\nabla_i E_{e,0}(\boldsymbol{q}(t)) \qquad i = 1, ..., N \tag{4.1}$$

where m_i are the nuclear masses, $\boldsymbol{q}_i(t)$ the positions of the nuclei at time t, N the number of nuclei in the system and $\mathbf{F}_i(t)$ the forces acting on the i-th particle computed through the Hellmann-Feynman theorem[30] as the analytical derivative of the ground state electronic energy functional. In turn, the ground state electronic energy is computed by minimizing the expectation value of the electronic Hamiltonian, see equations (2.31) and (2.34), reported below

$$\mathbf{F}_i(t) = -\nabla_i \min_{\Psi_{e,0}} [\langle \Psi_{e,0} | \hat{H}_e | \Psi_{e,0} \rangle] = -\nabla_i E_{e,0}(\mathbf{r}(t)) \qquad i = 1, ..., N \tag{4.2}$$

In the specific case of Density Functional Theory, this minimization is carried out using the Levy-Lieb reformulation,[31, 32] that is

$$E_{e,0}(\boldsymbol{q}(t)) = \min_{\tilde{\rho} \to N_e} \left\{ F_{LL}[\tilde{\rho}] + \int v_{ext}(\boldsymbol{\xi})\, \tilde{\rho}(\boldsymbol{\xi})\, d\boldsymbol{\xi} \right\} \tag{4.3}$$

where N_e is the number of electrons in the system, and $F_{LL}[\tilde{\rho}]$ is the universal Levy-Lieb functional,[31, 32] which in the general case is given by

$$F_{LL}[\tilde{\rho}] = \min_{\{|\tilde{\Psi}_{e,k}\rangle\} \to \tilde{\rho}} \left(\sum_k w_k \, \langle \tilde{\Psi}_{e,k} | \hat{T} + \hat{v}_{ee} | \tilde{\Psi}_{e,k} \rangle \right) \qquad \text{with} \quad \sum_k w_k = 1 \quad \text{and} \quad w_k \geq 0 \tag{4.4}$$

In the present treatment, only the case of conservative forces is considered, so that the total energy for the system, expressed as the sum of the kinetic energy E_k and the potential energy (that is the ground state electronic effective potential $E_{e,0}(\boldsymbol{q})$), namely,

$$E_{tot} = E_k + E_{e,0}(\boldsymbol{q}) = \sum_{i=1}^{N} \frac{m_i \, \boldsymbol{v}_i^2}{2} + E_{e,0}(\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + E_{e,0}(\boldsymbol{q}) \tag{4.5}$$

will be conserved. The first term in (4.5) is the kinetic energy, where $\boldsymbol{v}_i$ is the velocity of the i-th nucleus, while the second term is the potential energy. On the contrary, a non-conservative system has to be taken into account when, for example, friction is introduced through a velocity dependent force. This will be discussed, for example, in Section 5.3.

4.1.1 Hamilton equations of motion

Equation (4.1), together with the forces expression in equation (4.2), corresponds to a set of Nd coupled second order ordinary differential equations, where d is the spatial dimension of the system. Given a

set of initial conditions for the positions $(\boldsymbol{q}_1(t_0), ..., \boldsymbol{q}_N(t_0))$ and velocities $(\boldsymbol{v}_1(t_0), ..., \boldsymbol{v}_N(t_0))$, a unique solution can be formally obtained.
It is often convenient to introduce the Hamilton formulation of classical mechanics, in order to obtain a more clear connection to statistical and quantum mechanics. The system is then defined in terms of a set of Nd generalized coordinates q_α and generalized momenta p_α, where d is the spatial dimensionality and Nd is the number of degrees of freedom. For a system with N particles, free to move in a three dimensional space ($d = 3$) without constraints, there are $Nd = 3N$ degrees of freedom. The Hamilton dynamics is derived from the Lagrangian formulation. The Lagrangian $L(q_\alpha, \dot{q}_\alpha, t)$ is a function of the coordinates q_α, their time derivatives $\dot{q}_\alpha$ and (possibly) time. The Hamiltonian is defined to be the Legendre transform of the Lagrangian with respect to the $\dot{q}_\alpha$ variables,

$$\mathcal{H}(q_\alpha, p_\alpha, t) = \sum_{\alpha=1}^{Nd} p_\alpha \dot{q}_\alpha - L(q_\alpha, \dot{q}_\alpha, t) \tag{4.6}$$

where $\dot{q}_\alpha$ is eliminated from the right hand side in favour of p_α by using

$$p_\alpha = \frac{\partial L}{\partial \dot{q}_\alpha} = p_\alpha(q_\beta, \dot{q}_\beta, t) \tag{4.7}$$

and inverting to get $\dot{q}_\alpha = \dot{q}_\alpha(q_\beta, p_\beta, t)$. At the same time, the variation of the Hamiltonian $\mathcal{H}$ can be computed starting from (4.6) as

$$d\mathcal{H} = (dp_\alpha \dot{q}_\alpha + p_\alpha d\dot{q}_\alpha) - \left(\frac{\partial L}{\partial q_\alpha} dq_\alpha + \frac{\partial L}{\partial \dot{q}_\alpha} d\dot{q}_\alpha + \frac{\partial L}{\partial t} dt \right) = dp_\alpha \dot{q}_\alpha - \frac{\partial L}{\partial q_\alpha} dq_\alpha - \frac{\partial L}{\partial t} dt \tag{4.8}$$

The same previous variation can be rewritten as

$$d\mathcal{H} = \frac{\partial \mathcal{H}}{\partial q_\alpha} dq_\alpha + \frac{\partial \mathcal{H}}{\partial p_\alpha} dp_\alpha + \frac{\partial \mathcal{H}}{\partial t} dt \tag{4.9}$$

Equating the terms in (4.8) and (4.9), taking into account the relation (4.7), the famous Hamilton equations of motion are found[1]

$$-\frac{\partial L}{\partial t} = \frac{\partial \mathcal{H}}{\partial t} \tag{4.10}$$

$$\begin{aligned} \dot{q}_\alpha &= \frac{\partial \mathcal{H}}{\partial p_\alpha} \\ \dot{p}_\alpha &= -\frac{\partial \mathcal{H}}{\partial q_\alpha} \end{aligned} \qquad \alpha = 1, ..., Nd \tag{4.11}$$

where $\mathcal{H}$ is the classical Hamiltonian of the system, given by the sum of the kinetic and the potential energies. For a system with N particles in three dimensions, the Hamiltonian form can be written, using Cartesian coordinates, as follows

$$\mathcal{H} = \sum_{i=1}^{N} \frac{m_i \, \boldsymbol{v}_i^2}{2} + E_{e,0}(\boldsymbol{q}) \tag{4.12}$$

where, in this case, $\boldsymbol{q}_i$ are the generalized positions (with $\boldsymbol{q} = \{\boldsymbol{q}_i\}$) and $m_i \boldsymbol{v}_i$ the generalized momenta. Using equations (4.11) it follows that

$$\begin{aligned} \dot{\boldsymbol{q}}_i &= \boldsymbol{v}_i \\ m_i \, \dot{\boldsymbol{v}}_i &= -\frac{\partial}{\partial \boldsymbol{q}_i} E_{e,0}(\boldsymbol{q}) \end{aligned} \qquad i = 1, ..., N \tag{4.13}$$

[1]Note. The components of vectors and covectors will be written in the following with the same index positions (in the subscript), since the underlying metric is an Euclidean metric on the phase space. Remember: in general, in a Cartesian coordinate system on an Euclidean space, the partial derivatives are orthonormal with respect to the Euclidean metric, thus the metric tensor is the Kronecker delta in this coordinate system. Therefore, raising and lowering the indexes of the components of covectors and vectors, respectively, have no cost in an Euclidean metric.

which is equal to the Newton equation of motion (4.1) with the force given by (4.2).
The Hamiltonian form (4.12) can be written in a more simple way as

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{4.14}$$

where $\boldsymbol{p}_i = m_i \boldsymbol{v}_i$ is the generalized momentum of the i-th nucleus and $\phi(\boldsymbol{q})$ is the potential generated by the field effect of the ground state electronic and nuclear degrees of freedom computed at the HF or DFT level, at a fixed nuclear configuration $\boldsymbol{q} \equiv \{\boldsymbol{q}_i\}$. Hamilton equations of motion describe the unique evolution of the positions and momenta subject to a set of initial conditions. More precisely, equations (4.11) specify a trajectory

$$\boldsymbol{x}(t) \equiv (q_1(t), ..., q_d(t), p_1(t), ..., p_d(t)) \equiv (\boldsymbol{q}(t), \boldsymbol{p}(t)) \tag{4.15}$$

in phase space $\mathbb{R}^{2d}$, starting from an initial point $\boldsymbol{x}(t_0)$. For a system of N particles in three dimensions, the phase space is $6N$-dimensional. The constrain dictated by the energy conservation restricts the motion on a $(6N-1)$-dimensional surface in phase space, known as the constant energy hypersurface or simply the constant energy surface. Furthermore, the Hamiltonian in equation (4.12) is invariant when replacing $\boldsymbol{v}_i$ with $-\boldsymbol{v}_i$, which implies that the time evolution is reversible in time.

4.1.2 Symplectiness and canonical transformations

Hamilton equations of motion can also be expressed using a symplectic notation.
Indeed, the time derivative of the phase space vector $\boldsymbol{x}(t) = {}^t(q_1(t), ..., q_{Nd}(t), p_1(t), ..., p_{Nd}(t))$ can be written as a column vector in the following way

$${}^t\dot{\boldsymbol{x}}(t) = \left(\frac{\partial \mathcal{H}}{\partial p_1}, ..., \frac{\partial \mathcal{H}}{\partial p_{Nd}}, -\frac{\partial \mathcal{H}}{\partial q_1}, ..., -\frac{\partial \mathcal{H}}{\partial q_{Nd}}\right) \tag{4.16}$$

and hence the Hamilton equations of motion can be recast as

$$\dot{\boldsymbol{x}} = \mathbf{M}\,\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} \tag{4.17}$$

where $\dot{\boldsymbol{x}}$ and $(\partial\mathcal{H}/\partial\boldsymbol{x})$ are $2Nd\times 1$ column matrices, while $\mathbf{M}$ is a $2Nd\times 2Nd$ skew-symmetric, orthogonal block matrix given by

$$\mathbf{M} = \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0} \end{pmatrix} \tag{4.18}$$

where $\mathbb{0}$ and $\mathbb{1}$ are the $Nd\times Nd$ zero and identity matrices, respectively. Dynamical systems expressible in this form are said to possess a symplectic structure. Equation (4.17) is the so-called symplectic notation for the Hamilton equations of motion. To clearly understand the notation used and the beauty of this matrix notation, see Appendix A, Section A.3.
It is also important to know the behavior of transformation of coordinates in the phase space. First of all, a transformation of variables in phase space can be defined from the system of coordinates $\boldsymbol{x} = (\boldsymbol{q}(t), \boldsymbol{p}(t))$ to the frame of coordinates $\boldsymbol{y} = (\boldsymbol{Q}(\boldsymbol{q}(t), \boldsymbol{p}(t)), \boldsymbol{P}(\boldsymbol{q}(t), \boldsymbol{p}(t)))$ as

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}) \tag{4.19}$$

The initial frame of reference $\boldsymbol{x}$ obeys to equation (4.17). For a restricted canonical transformation, the function $\mathcal{H}(\boldsymbol{x})$ expressed in the new coordinates $\boldsymbol{y}$ corresponds to the Hamiltonian function for the new coordinates $\boldsymbol{y}$, and the Hamilton equations of motion in the $\boldsymbol{y}$ basis have exactly the same form as in (4.17), so that

$$\dot{\boldsymbol{x}} = \mathbf{M}\,\frac{\partial \mathcal{H}(\boldsymbol{x})}{\partial \boldsymbol{x}} \qquad \text{canonical transformation} \longrightarrow \qquad \dot{\boldsymbol{y}} = \mathbf{M}\,\frac{\partial \mathcal{H}(\boldsymbol{y})}{\partial \boldsymbol{y}} \tag{4.20}$$

Taking the time derivative of equation (4.19) for coordinates transformation and using the first relation in (4.20), the following expression is obtained

$$\dot{\boldsymbol{y}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \cdot \frac{\partial \boldsymbol{x}}{\partial t} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\,\mathbf{M}\,\frac{\partial \mathcal{H}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \mathbf{J}\,\mathbf{M}\,\frac{\partial \mathcal{H}(\boldsymbol{x})}{\partial \boldsymbol{x}} \qquad \text{with} \quad \mathbf{J} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \tag{4.21}$$

where the matrix $\mathbf{J}$ is the Jacobian matrix related to coordinates transformation. In components, the previous equation can be rewritten as

$$\dot{y}_\alpha = \sum_{\beta=1}^{2Nd} \frac{\partial y_\alpha}{\partial x_\beta}\,\dot{x}_\beta = \sum_{\beta=1}^{2Nd} J_{\alpha\beta}\,\dot{x}_\beta = \sum_{\beta=1}^{2Nd} J_{\alpha\beta} \sum_{\gamma=1}^{2Nd} M_{\beta\gamma}\,\frac{\partial \mathcal{H}}{\partial x_\gamma} = \sum_{\beta=1}^{2Nd} J_{\alpha\beta} \sum_{\gamma=1}^{2Nd} M_{\beta\gamma} \sum_{\rho=1}^{2Nd} \frac{\partial \mathcal{H}}{\partial y_\rho}\frac{\partial y_\rho}{\partial x_\gamma} \tag{4.22}$$

where the last identity has been computed considering the inverse transformation $\boldsymbol{y} \to \boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y})$, and the relation between gradients in the two reference frame of coordinates. Therefore, the previous equation is given by

$$\dot{y}_\alpha = \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}\sum_{\rho=1}^{2Nd} J_{\alpha\beta}\,M_{\beta\gamma}\,\frac{\partial \mathcal{H}}{\partial y_\rho}\frac{\partial y_\rho}{\partial x_\gamma} = \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}\sum_{\rho=1}^{2Nd} J_{\alpha\beta}\,M_{\beta\gamma}\,\frac{\partial \mathcal{H}}{\partial y_\rho}J_{\rho\gamma} = \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}\sum_{\rho=1}^{2Nd} J_{\alpha\beta}\,M_{\beta\gamma}\,J_{\rho\gamma}\,\frac{\partial \mathcal{H}}{\partial y_\rho}$$

that, written in vector notation, becomes

$$\dot{\boldsymbol{y}} = (\mathbf{J}\,\mathbf{M}\,{}^t\mathbf{J})\,\frac{\partial \mathcal{H}(\boldsymbol{y})}{\partial \boldsymbol{y}} \qquad \text{with} \quad \mathbf{J} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \tag{4.23}$$

This expression for the equations of motion is valid for any set of variables $\boldsymbol{y}$ that are being transformed (independently of time) from the set $\boldsymbol{x}$. Such a transformation is canonical if the equations of motion in the new coordinates have the canonical form (4.20). Comparing the above identity with the last expression in (4.20), it can be stated that the Hamilton equations of motion are left invariant under any transformation whose Jacobian $\mathbf{J}$ satisfies the relation

$$\mathbf{J}\,\mathbf{M}\,{}^t\mathbf{J} = \mathbf{M} \tag{4.24}$$

or, in components,

$$\sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd} \frac{\partial y_\alpha}{\partial x_\beta}\,M_{\beta\gamma}\,\frac{\partial y_\rho}{\partial x_\gamma} = M_{\alpha\rho} \qquad \to \qquad \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd} J_{\alpha\beta}\,M_{\beta\gamma}\,J_{\rho\gamma} = M_{\alpha\rho} \tag{4.25}$$

The Jacobian matrix is said to be symplectic if equation (4.24) holds. A change of variables with a symplectic Jacobian is said to be a canonical transformation.
By means of Hamiltonian formulation, the classical dynamics has been rewritten in terms of first order differential equations in which each point in phase space follows a unique path under time evolution. This time evolution is sometimes called a flow on phase space. The Liouville theorem and its associated equation are important properties of these flows. More details about the canonical transformations are reported in Appendix A, Section A.5.

4.1.3 Liouville theorem

Liouville Theorem. Consider a region in phase space and watch it evolve over time. Then the shape of the region will generically change, but Liouville theorem states that the volume in phase space remains the same.

Proof.
Consider an infinitesimal volume e moving for an infinitesimal time. Starting in a neighbourhood of the point (q_α, p_α) in phase space, with volume

$$v = d\boldsymbol{q}\,d\boldsymbol{p} \equiv dq_1 \cdots dq_{Nd}\,dp_1 \cdots dp_{Nd} \tag{4.26}$$

the evolution in time dt is given by

$$q_\alpha \quad \to \quad q_\alpha + \dot{q_\alpha}dt = q_\alpha + \frac{\partial \mathcal{H}}{\partial p_\alpha}\,dt \equiv \tilde{q}_\alpha \tag{4.27}$$

$$p_\alpha \quad \to \quad p_\alpha + \dot{p_\alpha}dt = p_\alpha - \frac{\partial \mathcal{H}}{\partial q_\alpha}\,dt \equiv \tilde{p}_\alpha \tag{4.28}$$

The new volume in phase space is then computed as

$$\tilde{v} = d\tilde{\boldsymbol{q}}\, d\tilde{\boldsymbol{p}} \equiv d\tilde{q}_1 \cdots d\tilde{q}_{Nd}\, dp_1 \cdots d\tilde{p}_{Nd} = (\det \mathbf{J}) v \tag{4.29}$$

where $\mathbf{J}$ is the Jacobian of the transformation defined by the determinant of the $2Nd \times 2Nd$ matrix

$$\mathbf{J} = \begin{pmatrix} \partial\tilde{\boldsymbol{q}}_\alpha/\partial\boldsymbol{q}_\beta & \partial\tilde{\boldsymbol{q}}_\alpha/\partial\boldsymbol{p}_\beta \\ \partial\tilde{\boldsymbol{p}}_\alpha/\partial\boldsymbol{q}_\beta & \partial\tilde{\boldsymbol{p}}_\alpha/\partial\boldsymbol{p}_\beta \end{pmatrix} \tag{4.30}$$

To prove the theorem, it has to be shown that $\det \mathbf{J} = 1$. First consider a single degree of freedom (i.e. $Nd = 1$), so that

$$\mathbf{J} = \begin{pmatrix} \partial\tilde{q}_\alpha/\partial q_\beta & \partial\tilde{q}_\alpha/\partial p_\beta \\ \partial\tilde{p}_\alpha/\partial q_\beta & \partial\tilde{p}_\alpha/\partial p_\beta \end{pmatrix} \tag{4.31}$$

Then the determinant of the Jacobian is given by

$$\det(\mathbf{J}) = \det \begin{pmatrix} 1 + (\partial^2\mathcal{H}/\partial p \partial q)dt & (\partial^2\mathcal{H}/\partial p^2)dt \\ -(\partial^2\mathcal{H}/\partial q^2)dt & 1 - (\partial^2\mathcal{H}/\partial q \partial p)dt \end{pmatrix} = 1 + O(dt^2) \tag{4.32}$$

which means that

$$\frac{d(\det \mathbf{J})}{dt} = 0 \tag{4.33}$$

so that the volume remains constant for all time. Now, in order to generalize this reasoning to arbitrary d, the determinant can be rewritten in a more general way as

$$\det \mathbf{J} = \det \begin{pmatrix} \delta_{\alpha\beta} + (\partial^2\mathcal{H}/\partial p_\alpha \partial q_\beta)dt & (\partial^2\mathcal{H}/\partial p_\alpha \partial p_\beta)dt \\ -(\partial^2\mathcal{H}/\partial q_\alpha \partial q_\beta)dt & \delta_{\alpha\beta} - (\partial^2\mathcal{H}/\partial q_\alpha \partial p_\beta)dt \end{pmatrix} \tag{4.34}$$

To compute the determinant, the following relation is used

$$\det(\mathbb{1} + \epsilon\, \mathbf{m}) = 1 + \epsilon \operatorname{Tr}(\mathbf{m}) + O(\epsilon^2) \tag{4.35}$$

for any matrix $\mathbf{m}$ and small ϵ. Then the previous determinant can be rewritten as

$$\det \mathbf{J} = 1 + \sum_{\alpha=1}^{Nd} \left(\frac{\partial^2\mathcal{H}}{\partial p_\alpha \partial q_\beta} - \frac{\partial^2\mathcal{H}}{\partial q_\alpha \partial p_\beta} \right) dt + O(dt^2) = 1 + O(dt^2) \tag{4.36}$$

so that the relation (4.33) is recovered also for the general case of a system with d degrees of freedom.

4.1.4 Liouville equation

Consider an ensemble (or collection) of systems with some density function $f(\boldsymbol{p}, \boldsymbol{q}, t)$ (also called distribution function). It can be interesting to taken into account this collection of system because
(i) there is a single system but its exact state is not known very well. Then $f(\boldsymbol{p}, \boldsymbol{q}, t)$ is understood as a probability parameterising the ignorance about the system and

$$\int f(\boldsymbol{p}, \boldsymbol{q}, t) \prod_{\alpha=1}^{Nd} dp_\alpha\, dq_\alpha = 1 \tag{4.37}$$

(ii) there is a large number N of identical, non-interacting systems (e.g. a certain number of atoms in a condensed matter structure) and the only important information to collect is the average behavior. Then the distribution $f(\boldsymbol{p}, \boldsymbol{q}, t)$ satisfies

$$\int f(\boldsymbol{p}, \boldsymbol{q}, t) \prod_{\alpha=1}^{Nd} dp_\alpha\, dq_\alpha = N \tag{4.38}$$

In the latter case, the particles in phase space (i.e. dynamical systems) are neither created nor destroyed, so the number of particles in a given volume is conserved. Since Liouville theorem assures that the volume elements $d\boldsymbol{p}\, d\boldsymbol{q}$ are preserved, the relation $df/dt = 0$ should be respected. This can be written as

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \sum_{\alpha=1}^{Nd} \left[\frac{\partial f}{\partial q_\alpha}\dot{q}_\alpha + \frac{\partial f}{\partial p_\alpha}\dot{p}_\alpha \right] = \frac{\partial f}{\partial t} + \sum_{\alpha=1}^{Nd} \left[\frac{\partial f}{\partial q_\alpha}\frac{\partial \mathcal{H}}{\partial p_\alpha} - \frac{\partial f}{\partial p_\alpha}\frac{\partial \mathcal{H}}{\partial q_\alpha} \right] = 0 \tag{4.39}$$

Rearranging the terms leads to

$$\frac{\partial f}{\partial t} = \sum_{\alpha=1}^{Nd}\left[\frac{\partial f}{\partial p_\alpha}\frac{\partial \mathcal{H}}{\partial q_\alpha} - \frac{\partial f}{\partial q_\alpha}\frac{\partial \mathcal{H}}{\partial p_\alpha}\right] \tag{4.40}$$

which is know as the Liouville equation.
Notice that Liouville theorem holds whether or not the system conserves energy (i.e. whether or not $\partial\mathcal{H}/\partial t = 0$).[2] But the system must be described by a Hamiltonian. For example, systems with dissipation typically head to regions of phase space with $\dot{q}_\alpha = 0$ and so do not preserve phase space volume.

4.1.5 Liouville operator

An important mathematical object which can be defined and it is useful to simplify the notation is the so-called Poisson bracket. Let $f(\boldsymbol{p}, \boldsymbol{q}, t)$ and $g(\boldsymbol{p}, \boldsymbol{q}, t)$ be two functions on phase space, then the Poisson bracket is defined as

$$\{f, g\} \equiv \sum_{\alpha=1}^{Nd}\left[\frac{\partial f}{\partial q_\alpha}\frac{\partial g}{\partial p_\alpha} - \frac{\partial f}{\partial p_\alpha}\frac{\partial g}{\partial q_\alpha}\right] \tag{4.41}$$

The Poisson bracket can be used to define the Liouville operator $i\mathcal{L}$ according to the relation

$$i\mathcal{L}(\cdot) \equiv \{\,\cdot\,, \mathcal{H}\} \tag{4.42}$$

where the imaginary unit i is used in Liouville operator definition as a matter of convention, and has the effect of making the operator $i\mathcal{L}$ Hermitian. More explicitly, the Liouville operator can be written as a differential operator [3]

$$i\mathcal{L} = \sum_{\alpha=1}^{Nd}\left[\frac{\partial}{\partial q_\alpha}\dot{q}_\alpha + \frac{\partial}{\partial p_\alpha}\dot{p}_\alpha\right] \tag{4.44}$$

4.1.6 Liouville operator invariance under canonical transformations

In Section 4.1.2 the change of variables have been classified as canonical transformations if the correspondent Jacobian matrix $\mathbf{J}$ is symplectic. A Jacobian matrix is said to be symplectic if equation (4.24) holds, where $\mathbf{M}$ is the block matrix defined in (4.18). On the base of these definitions, a theorem relating canonical transformations with Poisson brackets, and thus with Liouville operator, can be proved.

Theorem. The Poisson bracket is invariant under canonical transformations. Conversely, any transformation which preserves the Poisson bracket structure so that

$$\{Q_i, Q_j\} = \{P_i, P_j\} = 0 \qquad \text{and} \qquad \{Q_i, P_j\} = \delta_{ij} \tag{4.45}$$

[2] The central idea of Liouville theorem, that volume of phase space is constant, is somewhat reminiscent of quantum mechanics. Indeed, this is the first of several occasions where the ideas of quantum physics creeps into the classical world. Consider a system of particles distributed randomly within a square $\Delta q \Delta p$ in phase space. Liouville theorem implies that if the system evolves in any Hamiltonian manner, the spread of positions of the particles can be cut down only at the cost of increasing the spread of momentum. This is a strongly reminder of Heisenberg uncertainty relation, which is also written as $\Delta q \Delta p =$ constant. While Liouville and Heisenberg seem to be talking the same language, there are very profound differences between them. The distribution in the classical picture reflects the ignorance about details of the system rather than any intrinsic uncertainty. The crucial point is that a system of classical particles is really described by collection of points in phase space rather than a continuous distribution $\rho(q, p, t)$ as modeled above.

[3] Analogously, the Liouville operator can be written using a notation with indexes that permits to easily identify the phase space coordinates associated to each nucleus in the system, that is

$$i\mathcal{L} = \sum_{i=1}^{N}\left[\frac{\partial}{\partial \boldsymbol{q}_i}\dot{\boldsymbol{q}}_i + \frac{\partial}{\partial \boldsymbol{p}_i}\dot{\boldsymbol{p}}_i\right] \tag{4.43}$$

where, considering a three dimensional space, $\boldsymbol{q}_i = (q_{xi}, q_{yi}, q_{zi})$ and $\boldsymbol{p}_i = (p_{xi}, p_{yi}, p_{zi})$ are the three dimensional vectors associated to the generalized coordinates of the i-th nucleus. Therefore, the expression (4.44) for the Liouville operator with $d = 3$ is completely equivalent to the definition (4.43).

is canonical.

Proof.
First of all, the fact that the Poisson bracket is invariant under canonical transformations is demonstrated. Consider two functions $f(x_\beta)$ and $g(x_\beta)$, where $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{p})$. Then,

$$\{f,g\} \equiv \sum_{\alpha=1}^{Nd}\left[\frac{\partial f}{\partial q_\alpha}\frac{\partial g}{\partial p_\alpha} - \frac{\partial f}{\partial p_\alpha}\frac{\partial g}{\partial q_\alpha}\right] = \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}\frac{\partial f}{\partial x_\beta}M_{\beta\gamma}\frac{\partial g}{\partial x_\gamma} \quad \text{with} \quad \{M_{\beta\gamma}\} = \mathbf{M} = \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0} \end{pmatrix} \tag{4.46}$$

where $\mathbf{M} = \{M_{\beta\gamma}\}$ is the $2Nd \times 2Nd$ skew-symmetric, orthogonal block matrix defined in (4.18), with $\mathbb{0}$ and $\mathbb{1}$ the $Nd \times Nd$ zero and identity matrices, respectively. Consider the transformation defined as $\boldsymbol{x} \to \boldsymbol{y}(\boldsymbol{x})$, so that

$$\frac{\partial f}{\partial x_\beta} = \sum_{\sigma=1}^{2Nd}\frac{\partial f}{\partial y_\sigma}J_{\sigma\beta} \quad \text{where} \qquad J_{\sigma\beta} = \frac{\partial y_\sigma}{\partial x_\beta} \tag{4.47}$$

Analogously,

$$\frac{\partial g}{\partial x_\gamma} = \sum_{\rho=1}^{2Nd}\frac{\partial f}{\partial y_\rho}J_{\rho\gamma} = \sum_{\rho=1}^{2Nd}J_{\rho\gamma}\frac{\partial f}{\partial y_\rho} \qquad \text{where} \quad J_{\rho\gamma} = \frac{\partial y_\rho}{\partial x_\gamma} \tag{4.48}$$

and substituting (4.47) and (4.48) in the Poisson bracket (4.46) leads to

$$\{f,g\} = \sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}\left(\sum_{\sigma=1}^{2Nd}\frac{\partial f}{\partial y_\sigma}J_{\sigma\beta}\right)M_{\beta\gamma}\left(\sum_{\rho=1}^{2Nd}J_{\rho\gamma}\frac{\partial f}{\partial y_\rho}\right) = \sum_{\sigma=1}^{2Nd}\sum_{\rho=1}^{2Nd}\frac{\partial f}{\partial y_\sigma}\left(\sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}J_{\sigma\beta}\,M_{\beta\gamma}\,J_{\rho\gamma}\right)\frac{\partial f}{\partial y_\rho} \tag{4.49}$$

Assuming that the transformation is canonical, then equation (4.25) holds, that is

$$\sum_{\beta=1}^{2Nd}\sum_{\gamma=1}^{2Nd}J_{\sigma\beta}\,M_{\beta\gamma}\,J_{\rho\gamma} = M_{\sigma\rho} \tag{4.50}$$

The substitution of this expression in equation (4.49) permits to write the Poisson bracket as

$$\{f,g\} = \sum_{\sigma=1}^{2Nd}\sum_{\rho=1}^{2Nd}\frac{\partial f}{\partial y_\sigma}M_{\sigma\rho}\frac{\partial f}{\partial y_\rho} \tag{4.51}$$

leading to an expression in the frame of reference described with the coordinates $\boldsymbol{y}(\boldsymbol{x})$ equivalent to the original one (4.46) that is instead written using the coordinates $\boldsymbol{x}$. This means that the Poisson brackets can be computed in any coordinates related by a canonical transformation.
Consider now the converse condition. Take into consideration a point in phase space given by (q_α, p_α) and a new set of coordinates defines as $(Q_\alpha(q_\alpha), P_\alpha(p_\alpha))$. A Jacobian matrix element for the transformation is given by

$$J_{\alpha\beta} = \begin{pmatrix} \partial Q_\alpha/\partial q_\beta & \partial Q_\alpha/\partial p_\beta \\ \partial P_\alpha/\partial q_\beta & \partial P_\alpha/\partial p_\beta \end{pmatrix} \tag{4.52}$$

In order to demonstrate that this transformation is canonical if the hypothesis (4.45) are satisfied, the matrix product on the left hand side of equation (4.24) is computed, so that the symplectiness relation

(4.24) can be verified by applying the hypothesis (4.45). In components,

$$
\begin{aligned}
[\mathbf{J}\,\mathbf{M}\,{}^t\mathbf{J}]_{\alpha\beta} &= \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd} J_{\alpha\gamma}\,M_{\gamma\rho}\,({}^t\mathbf{J})_{\rho\beta} = \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd} J_{\alpha\gamma}\,M_{\gamma\rho}\,J_{\beta\rho} \\
&= \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd}\left[\begin{pmatrix} \partial Q_\alpha/\partial q_\gamma & \partial Q_\alpha/\partial p_\gamma \\ \partial P_\alpha/\partial q_\gamma & \partial P_\alpha/\partial p_\gamma \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} \partial Q_\beta/\partial q_\rho & \partial P_\beta/\partial q_\rho \\ \partial Q_\beta/\partial p_\rho & \partial P_\beta/\partial p_\rho \end{pmatrix}\right] \\
&= \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd}\left[\begin{pmatrix} -\partial Q_\alpha/\partial p_\gamma & \partial Q_\alpha/\partial q_\gamma \\ -\partial P_\alpha/\partial p_\gamma & \partial P_\alpha/\partial q_\gamma \end{pmatrix}\begin{pmatrix} \partial Q_\beta/\partial q_\rho & \partial P_\beta/\partial q_\rho \\ \partial Q_\beta/\partial p_\rho & \partial P_\beta/\partial p_\rho \end{pmatrix}\right] \\
&= \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd}\begin{pmatrix} (\partial Q_\alpha/\partial q_\gamma)(\partial Q_\beta/\partial p_\rho) - (\partial Q_\alpha/\partial p_\gamma)(\partial Q_\beta/\partial q_\rho) & (\partial Q_\alpha/\partial q_\gamma)(\partial P_\beta/\partial p_\rho) - (\partial Q_\alpha/\partial p_\gamma)(\partial P_\beta/\partial q_\rho) \\ (\partial P_\alpha/\partial q_\gamma)(\partial Q_\beta/\partial p_\rho) - (\partial P_\alpha/\partial p_\gamma)(\partial Q_\beta/\partial q_\rho) & (\partial P_\alpha/\partial q_\gamma)(\partial P_\beta/\partial p_\rho) - (\partial P_\alpha/\partial p_\gamma)(\partial P_\beta/\partial q_\rho) \end{pmatrix} \\
&= \begin{pmatrix} \{Q_\alpha, Q_\beta\} & \{Q_\alpha, P_\beta\} \\ \{P_\alpha, Q_\beta\} & \{P_\alpha, P_\beta\} \end{pmatrix}
\end{aligned}
$$

Applying the conditions (4.45) in the previous equation leads to

$$
[\mathbf{J}\,\mathbf{M}\,{}^t\mathbf{J}]_{\alpha\beta} = \sum_{\gamma=1}^{Nd}\sum_{\rho=1}^{Nd} J_{\alpha\gamma}\,M_{\gamma\rho}\,J_{\beta\rho} = \begin{pmatrix} \{Q_\alpha, Q_\beta\} & \{Q_\alpha, P_\beta\} \\ \{P_\alpha, Q_\beta\} & \{P_\alpha, P_\beta\} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = M_{\alpha\beta} \tag{4.53}
$$

Taking into consideration the second and the last expression in equation (4.53), the condition (4.25) for the symplectiness of the Jacobian is recovered, demonstrating that the associated change of variables is canonical. Therefore, this derivations allows to conclude that any transformation which preserves the Poisson bracket structure is canonical.

4.1.7 Time dependency in phase space

The time dependence of a function of generalized phase space coordinates and time $A(\boldsymbol{x}(t), t)$, which generically represents a property of the system (such as the distribution function), is formally given by

$$
\frac{d}{dt}\,A(\boldsymbol{x}(t), t) = \frac{\partial A}{\partial t} + \sum_{\alpha=1}^{Nd}\left[\frac{\partial A}{\partial q_\alpha}\,\dot{q}_\alpha + \frac{\partial A}{\partial p_\alpha}\,\dot{p}_\alpha\right] = \frac{\partial A}{\partial t} + \sum_{\alpha=1}^{Nd}\left[\frac{\partial A}{\partial q_\alpha}\frac{\partial \mathcal{H}}{\partial p_\alpha} - \frac{\partial A}{\partial p_\alpha}\frac{\partial \mathcal{H}}{\partial q_\alpha}\right] \tag{4.54}
$$

Using the Liouville operator introduced in Section 4.1.5, equation (4.54) which is valid for any function $A(\boldsymbol{x}(t), t) = A(\boldsymbol{p}(t), \boldsymbol{q}(t), t)$ can be rewritten as

$$
\frac{d}{dt}\,A(\boldsymbol{x}(t), t) = \frac{\partial A}{\partial t} + \{A, \mathcal{H}\} \tag{4.55}
$$

Often in physics there is a particular interest in probability distributions that do not change explicitly in time (i.e. $\partial A/\partial t = 0$). An important class of distributions that do not change in time have the form

$$
A \equiv A(\boldsymbol{p}(t), \boldsymbol{q}(t)) \tag{4.56}
$$

In this case, if the quantity $A(\boldsymbol{x}(t), t)$ does not depend explicitly on time, so that it corresponds to $A(\boldsymbol{x}(t))$, then in equations (4.54) and (4.55) the time partial derivative is equal to zero, so that the total time derivative becomes

$$
\frac{d}{dt}\,A(\boldsymbol{x}(t)) = \sum_{\alpha=1}^{Nd}\left[\frac{\partial A}{\partial q_\alpha}\,\dot{q}_\alpha + \frac{\partial A}{\partial p_\alpha}\,\dot{p}_\alpha\right] = \sum_{\alpha=1}^{Nd}\left[\frac{\partial A}{\partial q_\alpha}\,\frac{\partial \mathcal{H}}{\partial p_\alpha} - \frac{\partial A}{\partial p_\alpha}\,\frac{\partial \mathcal{H}}{\partial q_\alpha}\right] = \{A, \mathcal{H}\} \tag{4.57}
$$

that is

$$
\frac{d}{dt}\,A(\boldsymbol{x}(t)) = \{A, \mathcal{H}\} \tag{4.58}
$$

Comparing this last expression with equation (4.43), it is clearly seen that the time dependence of an arbitrary phase space function $A(\boldsymbol{x}(t))$ can be written in terms of the Liouville operator as

$$\frac{d}{dt}\,A(\boldsymbol{x}(t)) = i\mathcal{L}A(\boldsymbol{x}(t)) \tag{4.59}$$

with the formal solution[4]

$$A(\boldsymbol{x}(t)) = e^{i\mathcal{L}(t-t_0)}A(\boldsymbol{x}(t_0)) \tag{4.60}$$

where the integrations to solve the differential equation have been performed between the initial time t_0 and a generic time t. The operator $\exp(i\mathcal{L}t)$ is known as the classical propagator. By introducing the imaginary unit i into the definition of the Liouville operator the classical propagator $\exp(i\mathcal{L}t)$ resembles the quantum propagator $\exp(-i\hat{\mathcal{H}}t/\hbar)$. Following the result given by equation (4.60), if the identification between $A(\boldsymbol{x}(t))$ and $\boldsymbol{x}(t)$ is made, then the time evolution of the phase space vector $\boldsymbol{x}(t)$ can be written as

$$\boldsymbol{x}(t) = e^{i\mathcal{L}(t-t_0)}\,\boldsymbol{x}(t_0) \tag{4.61}$$

This equation describes the central numerical problem in molecular dynamics simulation, to obtain the time-dependent trajectory $\boldsymbol{x}(t)$ in phase space, given an initial condition $\boldsymbol{x}(t_0)$. Although elegant in its compactness, equation (4.61) amounts to little more than a formal device, since the action of the exponential operator on the phase space vector at time t_0 cannot be evaluated exactly. If this would be possible, then any and every problem in classical mechanics could be solved exactly analytically without developing numerical methods in the first place. However, what equation (4.61) does is to provide a very useful starting point for developing approximate solutions to Hamilton equations, as described in Section 4.2. In order to simplify the notation and generalize previous equations, the Liouville operator can be represented by a vector notation as

$$i\mathcal{L} = \dot{\boldsymbol{\Gamma}}\cdot\frac{\partial}{\partial\boldsymbol{\Gamma}} \qquad \text{with} \qquad \boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t)) \tag{4.62}$$

where the vector $\boldsymbol{\Gamma}$ define a point in the phase space and the dot above the vector variable stands for the time derivative. Using these abbreviate notation, the equations (4.54) and (4.57) can be rewritten as

$$\text{equation (4.54)} \quad \rightarrow \quad \frac{d}{dt}\,A(\boldsymbol{\Gamma}, t) = \frac{\partial A}{\partial t} + \dot{\boldsymbol{\Gamma}}\cdot\frac{\partial A}{\partial\boldsymbol{\Gamma}} \tag{4.63}$$

$$\text{equation (4.57)} \quad \rightarrow \quad \frac{d}{dt}\,A(\boldsymbol{\Gamma}) = \dot{\boldsymbol{\Gamma}}\cdot\frac{\partial A}{\partial\boldsymbol{\Gamma}} \tag{4.64}$$

and the equation (4.61) can be written as

$$\boldsymbol{\Gamma}(t) = e^{i\mathcal{L}(t-t_0)}\,\boldsymbol{\Gamma}(t_0) \tag{4.65}$$

Alternatively, if the phase space coordinates are allowed to propagate P times (i.e. P is the number of molecular dynamics steps in the simulation), and τ is the time step unit, then the previous phase space propagation formula becomes

$$\boldsymbol{\Gamma}(t) = \boldsymbol{\Gamma}(t_0 + P\,\tau) = e^{i\mathcal{L}\,P\,\tau}\,\boldsymbol{\Gamma}(t_0) \tag{4.66}$$

The vector $\boldsymbol{\Gamma}(t)$ is a generalization of the phase space vector $\boldsymbol{x}(t)$. Indeed, it can contain different number of coordinates, depending on the ensemble generated by the equations of motion defined for the system. For the case of standard Newtonian equations of motion, it contains only positions and momenta generalized coordinates. However, there are cases in which the equations of motion are described by more than two variables, as the case of extended system methods that introduces virtual variables for generating the canonical or the isobaric ensembles. In these cases the vector $\boldsymbol{\Gamma}(t)$ contains both the real and the virtual degrees of freedom. Note that all the equations and properties studied and demonstrated for the phase space vector $\boldsymbol{x}(t)$ from Section 4.1.2 up to now are also valid for its generalized phase space vector $\boldsymbol{\Gamma}(t)$.

[4]Exponential mapping from group to algebra permits to solve differential equations with operators in the usual way.

4.1.8 First integrals of Hamilton equations of motion

The Hamiltonian vector field X_H generated by the Hamiltonian H is defined as

$$X_H = \frac{\partial}{\partial t} + \sum_{\alpha=1}^{Nd}\left[\frac{\partial \mathcal{H}}{\partial p_\alpha}\frac{\partial}{\partial q_\alpha} - \frac{\partial \mathcal{H}}{\partial q_\alpha}\frac{\partial}{\partial p_\alpha}\right] = \frac{\partial}{\partial t} + \sum_{\alpha=1}^{Nd}\left[\dot{q}_\alpha \frac{\partial}{\partial q_\alpha} + \dot{p}_\alpha \frac{\partial}{\partial p_\alpha}\right] = \frac{\partial}{\partial t} + i\mathcal{L} \tag{4.67}$$

where $i\mathcal{L}$ is the Liouville operator defined in (4.43). A function $f(\boldsymbol{p}, \boldsymbol{q}, t)$ is said to be an integral function or a first integral of the Hamiltonian vector field X_H if it is constant along each integral curve. The necessary condition for the function $f(\boldsymbol{p}, \boldsymbol{q}, t)$ to be a first integral of the Hamiltonian vector field is therefore

$$X_H(f(\boldsymbol{p}, \boldsymbol{q}, t)) = \frac{df}{dt} = \frac{\partial f}{\partial t} + \sum_{\alpha=1}^{Nd}\left[\frac{\partial \mathcal{H}}{\partial p_\alpha}\frac{\partial f}{\partial q_\alpha} - \frac{\partial \mathcal{H}}{\partial q_\alpha}\frac{\partial f}{\partial p_\alpha}\right] = \frac{\partial f}{\partial t} + i\mathcal{L}f = \frac{\partial f}{\partial t} + \{f, \mathcal{H}\} = 0 \tag{4.68}$$

The Hamiltonian $\mathcal{H}$ itself, if it does not depend on time (i.e. if it is not an explicit function of time but depends on it only through $q_\alpha(t)$ ann $p_\alpha(t)$), is a first integral of X_H. This can be shown taking the time derivative of a time-independent Hamiltonian, following the definition of time derivative given by equation (4.68),

$$\frac{d}{dt}\mathcal{H}(p_\alpha, q_\alpha) = \sum_{\alpha=1}^{Nd}\left[\frac{\partial \mathcal{H}}{\partial p_\alpha}\frac{\partial \mathcal{H}}{\partial q_\alpha} - \frac{\partial \mathcal{H}}{\partial q_\alpha}\frac{\partial \mathcal{H}}{\partial p_\alpha}\right] = 0 \tag{4.69}$$

As a consequence, the total energy for the system described by a time-independent Hamiltonian is a conserved quantity of the system itself. For this reason, if an efficient way for integrating the Hamilton equation of motion (4.11) for a system of N particles is found, the so-called NVE ensemble (where the Number of particles, the Volume and the Energy are conserved) is automatically generated from the integration of the Hamilton equations of motion. Two others first integrals can be derived, namely, the total linear momentum $\boldsymbol{p}_t$ and the total angular momentum $\boldsymbol{L}_t$ of a system with N particles, defined as

$$\boldsymbol{p}_t = \sum_{i=1}^{N} \boldsymbol{p}_i \qquad\qquad \boldsymbol{L}_t = \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \boldsymbol{p}_i \tag{4.70}$$

where $\boldsymbol{p}_i$, $\boldsymbol{q}_i$ are d-dimensional vectors defined in a d-dimensional space, so that the number of degrees of freedom is equal to Nd, as previously defined (for example, in a three-dimensional space $d = 3$ the coordinate vectors are $\boldsymbol{p}_i = (p_{xi}, p_{yi}, p_{zi})$ and $\boldsymbol{q}_i = (q_{xi}, q_{yi}, q_{zi})$). However, the conservation of these two quantities depends on the form of the Hamiltonian of the system, so that the demonstration is not so easy as the proof for total energy conservation, in which the only constraint imposed for the Hamiltonian was the time-independence. A general and elegant demonstration of the conservation of the total linear and angular momentum for a Hamiltonian that is, respectively, translational invariant (translational symmetry) and rotational invariant (rotational symmetry), can be performed by introducing the action and studying its deformation in the presence of an infinitesimal translation and rotation, respectively, and by imposing that the action itself is invariant under these infinitesimal transformations. The use of this argument is an application of the Noether theorem,[33] that links every differentiable symmetry of the action of a physical system (with conservative forces) to a corresponding conservation law.

4.2 Generalized approach for equations of motion integration

The Liouville operator formalism is the starting point to derive symplectic integrators for the classical equations of motion in the phase space.
The time evolution of the phase space point $\boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t))$ from an initial time t_0 to a generic time t is formally written as

$$\boldsymbol{\Gamma}(t) = e^{i\mathcal{L}(t-t_0)}\,\boldsymbol{\Gamma}(t_0) \tag{4.71}$$

Equation (4.71) is a formal solution of Hamilton equations of motion. The exponential operator times the phase space point vector defines the flow of the Hamiltonian system which brings the system phase space

point from the initial state $(\boldsymbol{p}(t_0), \boldsymbol{q}(t_0))$ to the state $(\boldsymbol{p}(t), \boldsymbol{q}(t))$ at a later time t.[5] The transformation (4.71) obeys equation (4.24). Note also that the adjoint of the exponential operator corresponds to the inverse, i.e. $\exp[i\mathcal{L}(t-t_0)]$ is unitary. This implies that the trajectory is exactly time reversible.
In order to propagate in time the phase space variables, the form assumed by the Liouville operator has to be known. It depends on the number of phase space variables, as well as on the form of the Hamilton equations of motion, as stated by equation (4.62), reported hereafter

$$i\mathcal{L} = \dot{\boldsymbol{\Gamma}} \cdot \frac{\partial}{\partial \boldsymbol{\Gamma}} \qquad \text{with} \quad \boldsymbol{\Gamma}(t) \ \text{ point in phase space} \tag{4.73}$$

Therefore, knowing the Hamilton form of the equations of motion which rules the dynamics of the system, the Liouville operator can be computed using equation (4.73), and then used in the time propagator to calculate the evolution of the phase space variables in time, as in (4.71). The number of terms generated by applying the scalar product in the definition (4.73) is equal to the number of set of coordinates introduced to describe the evolution of the system in the phase space. For this reason, the time propagator operator in (4.71) has an exponential form with at least two Liouville operators, corresponding to the set of positions and momenta coordinates with the associated Hamilton equations of motion. Generally speaking, the terms generated by (4.73) to form the general Liouville operator can be labeled as $A, B, C, ...$, so that its generic form will be $i\mathcal{L} = A + B + C + ...$ and equation (4.71) becomes

$$\boldsymbol{\Gamma}(t) = e^{(A+B+C+...)(t-t_0)}\, \boldsymbol{\Gamma}(t_0) \tag{4.74}$$

The Liouville operators in the exponential form generally do not commute with each other. For example, when the positions and momenta coordinates refer to same degree of freedom, the associated Liouville operators do not commute. The non-commutation of the Liouville operators in (4.74) is a problem, because this implies that the whole time propagator operator is not unitary, and thus not time reversible. This problem has been resolved by Trotter[34] and Suzuki[35], with the introduction of an approximate expression for the discrete time propagator that retain both the symplectic and the reversibility property. Indeed, expression (4.74) can be simplified using the Suzuki-Trotter factorization formula, described in Section 4.2.1. Since the minimum number of variables to characterize a physical system of interest are two (the generalized positions and momenta coordinates), the problem of the solution of equation (4.74) will be treated in the following for the specific case of two Liouville operators, A and B, in order to give a simple and useful hint to the solution. The generalization to more than two Liouville operators is straightforward and will be discussed later on in Section 4.2.1.

4.2.1 Suzuki-Trotter factorization scheme

In general, two Liouville operators do not commute with each other. In order to show that two generic Liouville operators, A and B, do not commute, a simple case can be considered, such as a single particle with mass m moving in a one dimensional space. Its phase space is described by two coordinates (x, v) with the Hamiltonian given by

$$\mathcal{H}(p, x) = \frac{p^2}{2m} + \phi(x) \tag{4.75}$$

where $\phi(x)$ is the potential depending on the particle position in the mono-dimensional space. Therefore, the Hamilton equations of motion are

$$\dot{x} = \frac{\partial \mathcal{H}}{\partial p} = \frac{p}{m} = v \qquad\qquad \dot{p} = -\frac{\partial \mathcal{H}}{\partial x} = -\frac{\partial \phi}{\partial x} = F(x) = \frac{a(x)}{m} \tag{4.76}$$

[5] The generic time t in a molecular dynamics simulation is sampled at equal intervals of time, multiple of the so-called time step $\tau = \Delta t$. Starting from the initial time t_0, the expression for a generic time t is $t = t_0 + P\,\tau$, where P is the number of dynamics steps in which the phase space coordinates are allowed to propagate. Therefore, exploiting the discretization of time in a molecular dynamics simulation, if the phase space coordinates are allowed to propagate through P dynamical steps, the previous formula (4.71), that describes the time propagation of the phase space vector in the time interval $[t_0, t]$, has to be applied taking $t = t_0 + P\,\tau$, which gives

$$\boldsymbol{\Gamma}(t) = \boldsymbol{\Gamma}(t_0 + P\,\tau) = e^{i\mathcal{L}\,P\,\tau}\, \boldsymbol{\Gamma}(t_0) \tag{4.72}$$

so that, applying (4.73), the Liouville operator can be written as

$$i\mathcal{L} = v\frac{\partial}{\partial x} + a(x)\frac{\partial}{\partial v} \tag{4.77}$$

and separated in two the operators

$$i\mathcal{L}_x = v\frac{\partial}{\partial x} \qquad\qquad i\mathcal{L}_v = a(x)\frac{\partial}{\partial v} \tag{4.78}$$

The action of $i\mathcal{L}_x i\mathcal{L}_v$ on some arbitrary function $h(x, v)$ is

$$\left[v\frac{\partial}{\partial x}\, a(x)\frac{\partial}{\partial v}\right] h(x,v) = \left[v\,\frac{\partial\, a(x)}{\partial x}\,\frac{\partial}{\partial v} + v\, a(x)\,\frac{\partial^2}{\partial x \partial v}\right] h(x,v) \tag{4.79}$$

while the action of $i\mathcal{L}_v i\mathcal{L}_x$ on the same function is

$$\left[a(x)\frac{\partial}{\partial v}\, v\frac{\partial}{\partial x}\right] h(x,v) = \left[a(x)\frac{\partial}{\partial x} + a(x)\, v\,\frac{\partial^2}{\partial v \partial x}\right] h(x,v) \tag{4.80}$$

Since the function $h(x, v)$ is arbitrary, the commutator results as

$$[i\mathcal{L}_x, i\mathcal{L}_v] = v\,\frac{\partial\, a(x)}{\partial x}\,\frac{\partial}{\partial v} - a(x)\frac{\partial}{\partial x} \tag{4.81}$$

so that, in general, the two operators (4.78) do not commute. For non-commuting operators A and B,

$$e^{(A+B)} \neq e^A\, e^B \tag{4.82}$$

In these cases it is useful to introduce the Suzuki-Trotter identity

$$e^{(A+B)} = \lim_{P\to\infty} [e^{A/P}\, e^{B/P}]^P \tag{4.83}$$

where P is an integer. Using equation (4.24) it is easy to show that the flow defined in equation (4.83) is symplectic, being the product of two successive symplectic transformations. Unfortunately, the propagator defined in (4.83) is not unitary, and therefore the corresponding algorithm is not time reversible. Again the non-unitarity is due to the fact that the two factorized exponential operators are non commuting. This problem is prevented by halving the time step (which corresponds doubling the integer P) and using a symmetric version of the Suzuki-Trotter identity

$$e^{(A+B)} = \lim_{P\to\infty} [e^{B/2P}\, e^{A/2P}\, e^{A/2P}\, e^{B/2P}]^P = \lim_{P\to\infty} [e^{B/2P}\, e^{A/P}\, e^{B/2P}]^P \tag{4.84}$$

The resulting propagator is clearly unitary, and therefore time reversible. Applying this identity to the classical propagator $\exp[i\mathcal{L}(t - t_0)]$, by defining a time step $\tau \equiv \Delta t = (t - t_0)/P$ so that the time t can be discretized and written as a function of an integer P, the resultant expression is given by

$$e^{(A+B)(t-t_0)} = \lim_{\Delta t\to 0 (P\to\infty)} \left[e^{B\Delta t/2}\, e^{A\Delta t}\, e^{B\Delta t/2}\right]^P \tag{4.85}$$

For large but finite integers P, the expression for the propagator $\exp[i\mathcal{L}(t - t_0)]$ can be approximated as

$$e^{(A+B)(t-t_0)} \approx \left[e^{B\Delta t/2}\, e^{A\Delta t}\, e^{B\Delta t/2}\right]^P + \mathcal{O}(P\Delta t^3) \tag{4.86}$$

obtaining a time propagator operator that is clearly unitary, therefore time reversible, and is also correct up to the second order. It can be easily demonstrated that the approximate expression derived from the identity (4.83) generates a time propagator operator which is correct up to the first order in the time step. Thus, requiring that the product of the exponential operators be unitary automatically leads to a more accurate approximations of the true discrete time propagator.
In this way, the time evolution equation (4.74) can be easily solved for one time step (i.e. $P = 1$) by

making the single exponential operator acting sequentially on the phase space coordinates. The integer P can assume the meaning of the total number of steps performed in a molecular dynamics simulation, which generally are very high, justifying the approximation (4.86) for large but finite P, i.e. for a large but finite number of steps. Indeed, during a dynamics the three operators in (4.86) are applied on the phase space point at each step, so that the result of a molecular dynamics simulation with P number of steps are the phase space coordinates given by

$$\mathbf{\Gamma}(t_0 + P\Delta t) \approx \left[e^{B\Delta t/2}\, e^{A\Delta t}\, e^{B\Delta t/2} \right]^P \mathbf{\Gamma}(t_0) \tag{4.87}$$

Note also that a formally equivalent time propagator operator can be obtained exchanging the order of the operators in the previous expression, leading to another way to propagate in time the phase space point, given by

$$e^{(B+A)(t-t_0)} \approx \left[e^{A\Delta t/2}\, e^{B\Delta t}\, e^{A\Delta t/2} \right]^P + \mathcal{O}(P\Delta t^3) \tag{4.88}$$

Generally speaking, if the Liouville operator can be split in n operators, there will be $n!$ number of formally equivalent time propagator operator forms that can be obtained from the original splitting, representing all the possible combination between operators. However, if among the split operators there is a certain number, n_c, of commuting operators, then there will be $(n - n_c)!$ number of possible form of distinguishable time propagators formally equivalent.
Generalization to more than two Liouville operators is trivial. For example, if the whole form of Liouville operator can be split in three operators, A, B and C, the approximated form of the symmetric version of the Suzuki-Trotter decomposition, equation (4.86), becomes

$$e^{(A+B+C)(t-t_0)} \approx \left[e^{C\Delta t/2}\, e^{B\Delta t/2}\, e^{A\Delta t}\, e^{B\Delta t/2}\, e^{C\Delta t/2} \right]^P + \mathcal{O}(P\Delta t^5) \tag{4.89}$$

In this case, iff the three operators do not commute to each other, there are $3! = 6$ possible distinguishable form of the time propagator operator, which can be applied to the system phase space coordinates. Otherwise, if n_c operators among the three commute to each other, then the number of possible distinguishable time propagator operators will be $(3 - n_c)!$ (with $0 \leq n_c \leq 3$).
Further generalizing the splitting procedure to an arbitrary number of Liouville operators, labeled as $L_1, L_2, ..., L_{n-1}, L_n$, that is, if the whole Liouville operator derived from the equations of motion can be written as a sum of n Liouville operators as

$$i\mathcal{L} = L_1 + L_2 + \cdots + L_{n-1} + L_n \tag{4.90}$$

then the Suzuki-Trotter identity,[35, 34] used in the definition of an expression for the time propagator $\exp[i\mathcal{L}(t - t_0)]$ which involves the Liouville operator, given by

$$e^{(L_1+L_2+\cdots+L_n)(t-t_0)} = \lim_{P\to\infty} \left[e^{L_n/2P} \cdots e^{L_2/2P}\, e^{L_1/P}\, e^{L_2/2P} \cdots e^{L_n/2P} \right]^P \tag{4.91}$$

can be written by defining a time step $\tau \equiv \Delta t = (t - t_0)/P$, so that the time t can be discretized and written as a function of an integer P. In this way, taking $t = t_0 + P\,\tau$ leads to the identity

$$e^{(L_1+L_2+\cdots+L_n)P\,\tau} = \lim_{\tau\to 0(P\to\infty)} \left[e^{L_n\tau/2} \cdots e^{L_2\tau/2}\, e^{L_1\tau}\, e^{L_2\tau/2} \cdots e^{L_n\tau/2} \right]^P \tag{4.92}$$

that can be approximated, for large but finite integers P, as

$$e^{(L_1+L_2+\cdots+L_n)P\,\tau} \approx \left[e^{L_n\tau/2} \cdots e^{L_2\tau/2}\, e^{L_1\tau}\, e^{L_2\tau/2} \cdots e^{L_n\tau/2} \right]^P + \mathcal{O}(P\,\tau^{2n-1}) \tag{4.93}$$

Therefore, applying the operators in the parenthesis in (4.93) sequentially on the phase space point at time t_0, for a number of times P equal to the number of steps in a molecular dynamics simulation (which should be large in order to approximate the condition $P \to \infty$), the phase space coordinates evolve in

time according to the general equation (4.72), with the previously introduced approximate expression for the time propagator operator, that is

$$\mathbf{\Gamma}(t) = \mathbf{\Gamma}(t_0 + P\,\tau) \approx \left[e^{L_n \tau/2} \cdots e^{L_2 \tau/2}\, e^{L_1 \tau}\, e^{L_2 \tau/2} \cdots e^{L_n \tau/2} \right]^P \mathbf{\Gamma}(t_0) + \mathcal{O}(P\,\tau^{2n-1}) \tag{4.94}$$

where $\tau \equiv \Delta t$ is the time step in the generalized time propagator Liouville operator formulation.
At each time step, the action of the time propagator operator in equation (4.93) on the phase space coordinates of the previous step is evaluated, so that the system can evolve in time, sampling the phase space according to the ensemble generated by the equations of motion. It is therefore necessary to known the analytical form of the action of the exponential Liouville operators on a phase space point. The action of the Liouville operators on phase space points will be computed analytically and discussed in the following Section 4.2.2.

4.2.2 Action of the Liouville operators

The next step is to evaluate the action of the exponential Liouville operator on the phase space coordinates. The more common forms of Liouville operators are

$$i\mathcal{L} = c\,\frac{\partial}{\partial x} \qquad\qquad i\mathcal{L} = c\,x\,\frac{\partial}{\partial x} \tag{4.95}$$

where c can be a constant or a function, with the essential property to be independent on x.
It can be demonstrated that the action of the Liouville operators in (4.95) on a generic function $f(x)$ is given by

$$\exp\left(c\,\frac{\partial}{\partial x}\right) f(x) = f(x+c) \qquad\qquad \exp\left(c\,x\,\frac{\partial}{\partial x}\right) f(x) = f(xe^c) \tag{4.96}$$

The action of the time propagator generated by the first Liouville operator in (4.95) is first discussed. Consider the time propagator in the form of an exponential operator $\exp(c\,\partial/\partial x)$ acting on an arbitrary function $f(x)$, where c is independent on x. The action of the time propagator operator can be worked out by expanding the exponential in a Taylor series as

$$\exp\left(c\,\frac{\partial}{\partial x}\right) f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} \left(c\,\frac{\partial}{\partial x}\right)^n f(x) = \sum_{n=0}^{\infty} \frac{1}{n!}\, c^n\, \frac{\partial^n}{\partial x^n}\, f(x) = f(x+c) \tag{4.97}$$

The last series in (4.97) is just the Taylor expansion of $f(x+c)$ about $c = 0$, showing that the exponential operator gives rise to a pure translation of the coordinate on which it acts. Therefore, the time propagator generated by the first Liouville operator in (4.95) only translates the phase space coordinate x, as given in the first relation in (4.96).
Now, the action of the time propagator generated by the second Liouville operator in (4.95) can be derived using the previous identity (4.96). First of all, let c be a constant and $g(y)$ and $f(y)$ be two functions of a single variable y. Then,

$$\exp\left(c\,\frac{\partial}{\partial g}\right) f(y) = \exp\left(c\,\frac{\partial}{\partial g}\right) f\{g^{-1}[g(y)]\} = f\{g^{-1}[g(y)+c]\} \tag{4.98}$$

Now consider $c\,x\,\partial/\partial x = c\,\partial/\partial(\ln(x))$, so that the second relation in (4.96) can be easily proven as

$$\begin{aligned} \exp\left(c\,x\,\frac{\partial}{\partial x}\right) f(x) = \exp\left(c\,\frac{\partial}{\partial \ln(x)}\right) f(x) = \exp\left(c\,\frac{\partial}{\partial \ln(x)}\right) f\left(e^{\,ln(x)}\right) \\ = f\left(e^{\,ln(x)+c}\right) = f(xe^c) \end{aligned} \tag{4.99}$$

Therefore, the Suzuki-Trotter factorization, together with the evaluation of the time propagator operator action on the phase space coordinates, allow to generate a symplectic time integrator starting from the Hamiltonian form of the equations of motion. Using the results (4.96), a time reversible and symplectic integration algorithm can now be derived by acting with an Hermitian operator of the form (4.93) onto the phase space point at an initial time t_0, computing updated positions and momenta at a later time t.

4.3 Statistical mechanics and equations of motion

4.3.1 Hamiltonian dynamics

The equations of motion in the phase space are closely related to the statistical mechanics that describes the ensemble sampled during time evolution in the phase space of the particles which follow those equations of motion. Using the symplectic notation introduced in Section 4.1.2, the statistical mechanics implication of the equations of motion can be studied. In the microcanonical ensemble, the partition function in a three dimensional system is defined as

$$Z = \zeta \int d\boldsymbol{p} \int d\boldsymbol{q}\, \delta(\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) - E) \tag{4.100}$$

where the constant factors have been labeled as $\zeta = 1/[N!\ (2\pi\hbar)^{3N}]$, with $\hbar$ the reduced Planck constant, and the shortest notation $d\boldsymbol{p} = d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N$ and $d\boldsymbol{q} = d\boldsymbol{q}_1 \cdots d\boldsymbol{q}_N$ has been adopted for the integrals. The delta function in (4.100) restricts the integration to the hypersurface in the phase space defined by $\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) = E$. The partition function can be rewritten in terms of other phase space coordinates, taking into account that a volume element in the two coordinate sets needs not be the same. The volume element associated with $\boldsymbol{\Gamma}$ is

$$d\boldsymbol{\Gamma} = d\boldsymbol{q}_1 \cdots d\boldsymbol{q}_N\, d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \tag{4.101}$$

and to another set of coordinates $\boldsymbol{\Lambda}$ is

$$d\boldsymbol{\Lambda} = d\boldsymbol{Q}_1 \cdots d\boldsymbol{Q}_N\, d\boldsymbol{P}_1 \cdots d\boldsymbol{P}_N \tag{4.102}$$

These two volume elements are related via the Jacobian matrix of the transformation matrix

$$d\boldsymbol{\Gamma} = |\det(\mathbf{J})|\, d\boldsymbol{\Lambda} \qquad \text{where} \qquad \mathbf{J} = \frac{d\boldsymbol{\Lambda}}{d\boldsymbol{\Gamma}} \tag{4.103}$$

This equation shows that, in general, a coordinate transformation will result in the appearance of a Jacobian in the partition function, so that

$$Z = \zeta \int d\boldsymbol{P} \int d\boldsymbol{Q}\, |\det(\mathbf{J})|\, \delta(\tilde{\mathcal{H}}(\boldsymbol{P}, \boldsymbol{Q}) - E) \tag{4.104}$$

When computing ensemble averages in coordinate systems other than the original Cartesian one, the Jacobian of the transformation may be different from one, and this should be taken into account.
For a transformation that is canonical, i.e. obeys the condition (4.24), the absolute value of the Jacobian is one. To derive this result, the determinant on both sides of the symplectic condition (4.24) is taken, which leads to

$$\det(\mathbf{J}\,\mathbf{M}\,{}^t\mathbf{J}) = \det(\mathbf{M})$$

$$\det(\mathbf{J})\det(\mathbf{M})\det({}^t\mathbf{J}) = \det(\mathbf{J})\det({}^t\mathbf{J})\det(\mathbf{M}) = \det(\mathbf{J})\det(\mathbf{J})\det(\mathbf{M}) = [\det(\mathbf{J})]^2\det(\mathbf{M}) = \det(\mathbf{M})$$

The last identity can be true only if the determinant of the Jacobian matrix respects the condition

$$\det(\mathbf{J}) = \pm 1 \tag{4.105}$$

which implies that for a canonical transformation the absolute value of the Jacobian associated with this transformation must be one. An important point is that the natural time evolution in phase space of a classical system may be considered as a coordinate transformation

$$\boldsymbol{\Gamma}(t_0) \to \boldsymbol{\Gamma}(t) \tag{4.106}$$

One important property of a Hamiltonian system is that the natural time evolution corresponds to a symplectic coordinate transformation. The transformation from $\boldsymbol{\Gamma}(t_0)$ to $\boldsymbol{\Gamma}(t)$ can be considered as a sequence of infinitesimal transformations with time step δt. Suppose to define the evolution of the coordinates during the time interval δt as transformation of coordinates from $\boldsymbol{\Gamma}$ to $\boldsymbol{\Lambda}$, so that

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}(\boldsymbol{\Gamma}) = \boldsymbol{\Gamma}(t + \delta t) = \boldsymbol{\Gamma}(t) + \dot{\boldsymbol{\Gamma}}(t)\, \delta t \tag{4.107}$$

The Jacobian of this transformation is

$$\mathbf{J} = \frac{d\mathbf{\Lambda}}{d\mathbf{\Gamma}} = \mathbb{1} + \delta t \, \frac{\partial}{\partial \mathbf{\Gamma}} \left(\mathbf{M} \, \frac{\partial \mathcal{H}}{\partial \mathbf{\Gamma}} \right) = \mathbb{1} + \delta t \, \mathbf{M} \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \tag{4.108}$$

where the expression (4.17) has been used to write the equations of motion $\dot{\mathbf{\Gamma}}(t)$ in the second equivalence of the previous relation and

$$\left(\frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \right)_{ij} = \frac{\partial^2 \mathcal{H}}{\partial \Gamma_i \, \partial \Gamma_j} \tag{4.109}$$

Taking into account that $\mathbf{M}$ is an antisymmetric matrix (so that ${}^t\mathbf{M} = -\mathbf{M}$), the transpose of the Jacobian matrix (4.108) can be written as

$${}^t\mathbf{J} = \mathbb{1} - \delta t \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \, \mathbf{M} \tag{4.110}$$

Substitution of this expression for the Jacobian into the symplectic condition (4.24) yields, to first order in δt,

$$\begin{aligned} \mathbf{J} \, \mathbf{M} \, {}^t\mathbf{J} &= \left(\mathbb{1} + \delta t \, \mathbf{M} \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \right) \mathbf{M} \left(\mathbb{1} - \delta t \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \, \mathbf{M} \right) \\ &\approx \mathbf{M} + \delta t \, \mathbf{M} \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \, \mathbf{M} - \mathbf{M} \, \delta t \, \frac{\partial^2 \mathcal{H}}{\partial \mathbf{\Gamma} \, \partial \mathbf{\Gamma}} \, \mathbf{M} = \mathbf{M} \end{aligned} \tag{4.111}$$

Hence the symplectic condition holds for the evolution of $\mathbf{\Gamma}$ during an infinitesimal time interval. As the evolution of $\mathbf{\Gamma}$ during a finite interval can be considered as a sequence of canonical transformations of infinitesimal steps, the total time evolution also satisfies the symplectic condition.
The Hamiltonian itself can be viewed as the generator of a canonical transformation acting on all points in phase space. As the Jacobian of a canonical transformation is equal to one, the size of a volume element in phase space does not change during the natural time evolution of a Hamiltonian system. Moreover, the density $f(\boldsymbol{p}(t), \boldsymbol{q}(t))$ around any point in phase space also remains constant during the time evolution. To see this, consider a volume v in phase space bounded by a surface s. During time evolution, the surface moves and so do all points inside the surface. However, a point cannot cross the surface. The reason is simple: if two trajectories in phase space would cross, it would imply that there are two trajectories that start from the same phase space point. But this is impossible, as it would mean that a trajectory starting from this point is not uniquely specified by its initial conditions. Hence, the number of phase space points inside any volume does not change in time. As the volume itself is also constant, this implies that the phase space density (i.e. the number of points per unit volume) is constant. In other words: the phase space density of a Hamiltonian system behaves like an incompressible fluid

$$\frac{df}{dt} = 0 \tag{4.112}$$

While the exact solution of Hamilton equations of motion will satisfy the incompressibility condition, discrete numerical schemes will, in general, violate it. As before, any numerical molecular dynamics algorithm (as for example the velocity Verlet) can be considered as a transformation from $(\boldsymbol{q}(t), \boldsymbol{p}(t))$ to $(\boldsymbol{q}(t + \Delta t), \boldsymbol{p}(t + \Delta t))$. The Jacobian of this transformation can then be computed to check whether it is equal to one (see Section 5.1.2). For all the good algorithms to solve Newton equations of motion, the Jacobian of the transformation from $(\boldsymbol{q}(t), \boldsymbol{p}(t))$ to $(\boldsymbol{q}(t + \Delta t), \boldsymbol{p}(t + \Delta t))$ is equal to one (such algorithms are said to be area preserving). It should be noted that the symplectic condition implies more than just the area preserving properties. Unfortunately, these other consequences do not have such a simple intuitive interpretation. When an algorithm is said to be symplectic, it should be more than area preserving, it should really satisfy the symplectic condition. Fortunately, in many cases the symplectic nature of an algorithm is easy to demonstrate by making use of the fact that any set of classical Hamilton equations of motion satisfies the symplectic condition. An algorithm that can be written as a sequence of exact time evolutions generated by simple Hamiltonians is therefore necessarily symplectic. An example is the Verlet algorithm. As discussed in Section 5.1.2, this algorithm can be viewed as a sequence of exact propagation steps using either the kinetic part of the Hamiltonian or the potential part. Either propagation scheme satisfies the symplectic condition. Hence the Verlet algorithm as a whole is symplectic.

4.3.2 Non Hamiltonian dynamics

The classical Newtonian equations of motion describe the time evolution of a system of N particles in a volume V, at a total energy E. However, often it is more convenient to keep other thermodynamic variables constant, as for example the temperature or the pressure. One way to proceed is to impose the condition of constant temperature or pressure by using an extended Lagrangian, from which the equations of motion are then derived (see Sections 5.3.4 and 5.4.1). While the mechanical consequences of extending the Lagrangian are straightforward, the effects on the statistical mechanics of the system are less obvious. The reason is that, in general, these extended Lagrangians cannot be transformed into a Hamiltonian form. This implies that a connection to statistical mechanics cannot be performed using the methods introduced in Section 4.3. In the following, a general approach for analyzing the extended Lagrangian systems will be presented.

Consider a d dimensional system with N atoms, described by the phase space vector

$$\mathbf{\Gamma} = (q_1, ..., q_{dN}, p_1, ..., p_{dN}, ...) \equiv (\Gamma^1, ..., \Gamma^n) \tag{4.113}$$

that defines n coordinates. In Hamiltonian dynamics, the measure $d\mathbf{\Gamma}$ which enters in the definition of the average is preserved. That is, given a subset of systems with initial conditions in a phase space volume element $d\mathbf{\Gamma}_0$, the trajectories $\mathbf{\Gamma}_t(t; \mathbf{\Gamma}_0)$ obtained by solving Hamilton equations of motion for each initial condition $\mathbf{\Gamma}_0$ will describe a volume element $d\mathbf{\Gamma}_t$ in phase space such that

$$d\mathbf{\Gamma}_0 = d\mathbf{\Gamma}_t \qquad \text{where} \quad d\mathbf{\Gamma}_0 \equiv d\mathbf{\Gamma}(t_0) \quad \text{and} \quad d\mathbf{\Gamma}_t \equiv d\mathbf{\Gamma}(t) \tag{4.114}$$

This is equivalent to the incompressibility of phase space flow, which is a property satisfied by Hamiltonian systems. It is also equivalent to the condition that the Jacobian of the coordinate transformation specified by $\mathbf{\Gamma}_t(t; \mathbf{\Gamma}_0)$ is unity. The implication of an invariant measure is that the average of an observable can be computed with respect to the phase space variables $\mathbf{\Gamma}$ at any time t. One of the signatures of non Hamiltonian flow is that it can have a nonzero phase space compressibility, a property that distinguishes it from the Hamiltonian case, which is always incompressible. Since Hamiltonian flow preserves the measure of phase space, it is usually assumed that this space can be treated mathematically as a Euclidean manifold. In the following, it will be shown that the assumption of a Euclidean phase space manifold cannot be made for non Hamiltonian systems. Thus, in developing statistical mechanical concepts that refer to the geometry of the underlying space for a non Hamiltonian system, one must treat the phase space as a general Riemannian manifold. It will, nevertheless, be shown that key concepts such as invariant measure and continuity, which describe Hamiltonian systems, can be generalized to the non Hamiltonian case by a proper treatment of the geometry of the phase space and that an invariant measure on the phase space manifold can be derived. Following this, an equation of continuity, a generalized Liouville equation for the distribution function of an ensemble of systems evolving according to non Hamiltonian dynamics, will be derived. The resulting equation will be seen to depend explicitly on the phase space metric and thus can be shown to reduce to the ordinary Liouville equation in the Hamiltonian limit.
Consider a non Hamiltonian dynamical system

$$\dot{\Gamma}^\alpha = \lambda^\alpha(\mathbf{\Gamma}, t) \qquad \alpha = 1, ..., n \tag{4.115}$$

for the evolution of the n coordinates $\mathbf{\Gamma} = \Gamma^1, ..., \Gamma^n$ with initial values $\Gamma_0^1, ..., \Gamma_0^n$. The n coordinates describe a point of an n dimensional Riemannian manifold with metric G. If the set of all n-tuples of real numbers is denoted as R^n, then $\mathbf{\Gamma}$ belongs to this set. The solutions $\Gamma_t^1, ..., \Gamma_t^n$ of equations (4.115) provide a coordinate transformation on the manifold from the initial coordinates $\Gamma_0^1, ..., \Gamma_0^n$ to the coordinates at time t, given by

$$\Gamma_t^\alpha = \Gamma_t^\alpha(t; \Gamma_0^1, ..., \Gamma_0^n) \qquad \alpha = 1, ..., n \tag{4.116}$$

The solution leads to a set of n vector functions that depend on time and on the initial phase space coordinates. Since the generator of the coordinate transformations is the single parameter t, equations (4.116) describe a one-parameter family of diffeomorphisms. Equation (4.116) can be viewed as a coordinate transformation from the initial coordinates at time t_0 to the coordinates at time t. Thus, using

equations (4.115) and (4.116) it is possible to determine how the initial phase space volume element $d\mathbf{\Gamma}_0$ transforms under equation (4.116). This is determined by the Jacobian of the coordinate transformation defined by equation (4.116), that is

$$d\mathbf{\Gamma}_t = \det[\mathbf{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)]\, d\mathbf{\Gamma}_0 \equiv \mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)\, d\mathbf{\Gamma}_0 \tag{4.117}$$

where the determinant of the Jacobian can be expressed as, using the well-known relation for the determinant of a matrix,

$$\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) = \det\left[\frac{\partial(\Gamma_t^1,...,\Gamma_t^n)}{\partial(\Gamma_0^1,...,\Gamma_0^n)}\right] = \det(\mathbf{J}) = e^{\mathrm{Tr}(\ln \mathbf{J})} \qquad \text{with} \qquad J_{\alpha\beta} = \frac{\partial \Gamma_t^\alpha}{\partial \Gamma_0^\beta} \tag{4.118}$$

Knowledge of the Jacobian of this transformation is essential for an understanding of how the measure on the manifold transforms in time. An equation of motion for $\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)$ can be derived by computing the time derivative of both sides of equation (4.118), which yields

$$\frac{d\mathcal{J}}{dt} = \mathcal{J}\ \mathrm{Tr}\left(\mathbf{J}^{-1}\,\frac{d\mathbf{J}}{dt}\right) = \mathcal{J}\sum_{\alpha=1}^{n}\sum_{\beta=1}^{n} J^{-1}_{\alpha\beta}\ \frac{dJ_{\beta\alpha}}{dt} \tag{4.119}$$

The matrix elements of $\mathbf{J}^{-1}$ and $d\mathbf{J}/dt$ are given by

$$J^{-1}_{\alpha\beta} = \frac{\partial \Gamma_0^\alpha}{\partial \Gamma_t^\beta} \qquad \text{and} \qquad \frac{dJ_{\beta\alpha}}{dt} = \frac{\partial \dot{\Gamma}_t^\beta}{\partial \Gamma_0^\alpha} \tag{4.120}$$

Substituting these expressions into (4.119), the equation of motion for the determinant of the Jacobian reduces to

$$\begin{aligned}\frac{d\mathcal{J}}{dt} &= \mathcal{J}\sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}\frac{\partial\Gamma_0^\alpha}{\partial\Gamma_t^\beta}\frac{\partial\dot{\Gamma}_t^\beta}{\partial\Gamma_0^\alpha} = \mathcal{J}\sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}\sum_{\gamma=1}^{n}\frac{\partial\Gamma_0^\alpha}{\partial\Gamma_t^\beta}\frac{\partial\dot{\Gamma}_t^\beta}{\partial\Gamma_t^\gamma}\frac{\partial\Gamma_t^\gamma}{\partial\Gamma_0^\alpha} = \mathcal{J}\sum_{\beta=1}^{n}\sum_{\gamma=1}^{n}\frac{\partial\Gamma_t^\gamma}{\partial\Gamma_t^\beta}\frac{\partial\dot{\Gamma}_t^\beta}{\partial\Gamma_t^\gamma} \\ &= \mathcal{J}\sum_{\beta=1}^{n}\sum_{\gamma=1}^{n}\delta_{\beta\gamma}\frac{\partial\dot{\Gamma}_t^\beta}{\partial\Gamma_t^\gamma} = \mathcal{J}\sum_{\alpha=1}^{n}\frac{\partial\dot{\Gamma}_t^\alpha}{\partial\Gamma_t^\alpha} = \mathcal{J}\sum_{\alpha=1}^{n}\frac{\partial\lambda_t^\alpha}{\partial\Gamma_t^\alpha} = \mathcal{J}\,\kappa(\mathbf{\Gamma}_t,t)\end{aligned} \tag{4.121}$$

that is

$$\frac{d\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)}{dt} = \mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)\,\kappa(\mathbf{\Gamma}_t,t) \tag{4.122}$$

where the quantity

$$\kappa(\mathbf{\Gamma}_t,t) \equiv \sum_{\alpha=1}^{n}\frac{\partial\dot{\Gamma}_t^\alpha}{\partial\Gamma_t^\alpha} = \nabla_{\mathbf{\Gamma}_t}\cdot\dot{\mathbf{\Gamma}}_t = \nabla_{\mathbf{\Gamma}_t}\cdot\boldsymbol{\lambda}_t = \sum_{\alpha=1}^{n}\frac{\partial\lambda_t^\alpha}{\partial\Gamma_t^\alpha} \tag{4.123}$$

is known as the phase space compressibility of the dynamical system. Since equation (4.116) represents an identity transformation at $t = 0$, it is clear that equation (4.122) is subject to the obvious initial condition $\mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_0) = 1$. Note that the Jacobian of the inverse transformation

$$\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t) = \det\left[\frac{\partial(\Gamma_0^1,...,\Gamma_0^n)}{\partial(\Gamma_t^1,...,\Gamma_t^n)}\right] \tag{4.124}$$

satisfies the relation

$$\mathcal{J}\,\bar{\mathcal{J}} = 1 \qquad\qquad \forall\ t \tag{4.125}$$

for all time, hence

$$\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\frac{d\mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)}{dt} + \mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\frac{d\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)}{dt} = 0 \tag{4.126}$$

$$\mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\frac{d\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)}{dt} = -\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\frac{d\mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)}{dt} = -\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\kappa(\mathbf{\Gamma}_0,0) \tag{4.127}$$

$$\frac{d\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)}{dt} = -\bar{\mathcal{J}}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)\,\kappa(\mathbf{\Gamma}_0,0) \tag{4.128}$$

where the equation of motion (4.122) has been used. The differential equation (4.128) can be written equivalently as

$$\frac{d\bar{\mathcal{J}}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)}{dt} = -\bar{\mathcal{J}}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)\,\kappa(\mathbf{\Gamma}_t,t) \tag{4.129}$$

Equations (4.122) and (4.129) predict a non unit Jacobian for a compressible system. Therefore, in general, the dynamics that results from solving non Hamiltonian equations of motion is not area preserving. As explained in Appendix 4.3, solving the equations of motion can be considered as a coordinate transformation. If the system is Hamiltonian, any volume element in phase space that is thus transformed may change its shape, but not its volume. In this case, the compressibility (4.123) vanishes, so that $\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) = 1$ for all time, that leads to the familiar conservation of the Euclidean phase space volume element $d\mathbf{\Gamma}$, according to equation (4.117), which indicates that phase space is flat. In contrast, for a non Hamiltonian system, the Jacobian of the transformation associated with the evolution of $\mathbf{\Gamma}(t_0) \equiv \mathbf{\Gamma}_0 \to \mathbf{\Gamma}(t) \equiv \mathbf{\Gamma}_t$ has to be considered, following equation (4.117). The motion in phase space of a Hamiltonian system can be resembled to that of an incompressible liquid, so that in time the volume of the liquid does not change. In contrast, a non Hamiltonian system is compressible. This compressibility, expressed as in (4.123), need not vanish and must be taken into account when considering the generalization of the Liouville theorem to non Hamiltonian systems. As a consequence, the usual phase space measure $d\mathbf{\Gamma}$ is no longer an invariant measure under the dynamical evolution. However, knowledge of the compressibility of a system allows an invariant measure for the manifold to be derived. In fact, the general solution to equation (4.122) is

$$\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) = \exp\left(\int_0^t \kappa(\mathbf{\Gamma}_s,s)\,ds\right) \tag{4.130}$$

Since

$$\kappa(\mathbf{\Gamma},t) = \frac{d\ln[\mathcal{J}(\mathbf{\Gamma};\mathbf{\Gamma}_0)]}{dt} \tag{4.131}$$

is a total time derivative, a variable $w(\mathbf{\Gamma},t)$ related to the compressibility by

$$\dot{w}(\mathbf{\Gamma},t) = \kappa(\mathbf{\Gamma},t) \qquad \to \qquad w(\mathbf{\Gamma},t) = \int \kappa(\mathbf{\Gamma},t)\,dt \tag{4.132}$$

can be introduced as the indefinite time integral of the compressibility. Equation (4.130) can then be written as

$$\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) = \exp[w(\mathbf{\Gamma}_t,t) - w(\mathbf{\Gamma}_0,0)] \tag{4.133}$$

Substituting equation (4.133) into (4.117) and rearranging leads to

$$d\mathbf{\Gamma}_t = \exp[w(\mathbf{\Gamma}_t,t) - w(\mathbf{\Gamma}_0,0)]\,d\mathbf{\Gamma}_0 \qquad \to \qquad \exp[-w(\mathbf{\Gamma}_t,t)]\,d\mathbf{\Gamma}_t = \exp[-w(\mathbf{\Gamma}_0,0)]\,d\mathbf{\Gamma}_0 \tag{4.134}$$

The function $w(\mathbf{\Gamma},t)$ defined in (4.132) can be related to the metric of the manifold as follows. First of all, the metric G on the manifold has tensor components

$$g^{(0)}_{\alpha\beta} = G\left(\frac{\partial}{\partial\Gamma^\alpha_0},\frac{\partial}{\partial\Gamma^\beta_0}\right) \qquad \text{and} \qquad g^{(t)}_{\alpha\beta} = G\left(\frac{\partial}{\partial\Gamma^\alpha_t},\frac{\partial}{\partial\Gamma^\beta_t}\right) \tag{4.135}$$

respectively in the initial coordinate basis $\{\partial/\partial\Gamma^\alpha_0,...,\partial/\partial\Gamma^n_0\}$ and in the corresponding coordinate basis at time t. These representations of G are, in general, different. The determinants of the metric in these two representations are related by the Jacobian as (see Ref. [36])

$$\sqrt{g(\mathbf{\Gamma}_0,0)} = \sqrt{g(\mathbf{\Gamma}_t,t)}\,\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) \qquad\qquad \sqrt{g(\mathbf{\Gamma}_t,t)} = \sqrt{g(\mathbf{\Gamma}_0,0)}\,\bar{\mathcal{J}}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) \tag{4.136}$$

where $\bar{\mathcal{J}}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0) = \mathcal{J}(\mathbf{\Gamma}_0;\mathbf{\Gamma}_t)$ is the inverse of the Jacobian $\mathcal{J}(\mathbf{\Gamma}_t;\mathbf{\Gamma}_0)$.
Since the determinant of the Jacobian and its inverse are not unity, it is clear that the metric determinant, in general, is also not unity. This means that the phase space must be treated as a general Riemannian manifold with arbitrary curvature, and the volume n-form, which determines the volume element in an arbitrary coordinate system, should be expressed as

$$\omega = \sqrt{g}\;d\Gamma^1 \wedge \cdots \wedge d\Gamma^n \tag{4.137}$$

accounting for a nontrivial metric. Moreover, the integral of any function $f : R^n \to R^1$ over the space should be written as

$$\int \omega\, f = \int d\Gamma^1 \wedge \cdots \wedge d\Gamma^n \ \sqrt{g}\ f \tag{4.138}$$

The expression for the volume n-form $\sqrt{g}\ d\Gamma^1 \wedge \cdots \wedge d\Gamma^n$ behaves like a tensor under coordinate transformations, for which there is a Jacobian related to the metric by equation (4.136). The wedge product in the volume n-form transforms according to

$$d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = \mathcal{J}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0)\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n = \exp[w(\mathbf{\Gamma}_t, t) - w(\mathbf{\Gamma}_0, 0)]\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.139}$$

where equation (4.133) has been used. Arranging equation (4.139) so that quantities at time t appear on the left and quantities at time $t = 0$ appear on the right leads to

$$\exp[-w(\mathbf{\Gamma}_t, t)]\, d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = \exp[-w(\mathbf{\Gamma}_0, 0)]\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.140}$$

which shows that

$$\exp[-w(\mathbf{\Gamma})]\, d\Gamma^1 \wedge \cdots \wedge d\Gamma^n \qquad \text{invariant volume} \tag{4.141}$$

is an invariant volume form on the manifold and, by extension,

$$\exp[-w(\mathbf{\Gamma})]\, d\Gamma^1 \cdots d\Gamma^n = \exp[-w(\mathbf{\Gamma})]\, d\mathbf{\Gamma} \qquad \text{invariant measure} \tag{4.142}$$

is an invariant measure. Furthermore, using the first relation in (4.136) between the determinant of the Jacobian and the metric in the first equivalence of (4.139) leads to

$$d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = \mathcal{J}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0)\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n = \frac{\sqrt{g(\mathbf{\Gamma}_0, 0)}}{\sqrt{g(\mathbf{\Gamma}_t, t)}}\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.143}$$

$$\sqrt{g(\mathbf{\Gamma}_t, t)}\ d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = \sqrt{g(\mathbf{\Gamma}_0, 0)}\ d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.144}$$

and comparing (4.140) with (4.144), the metric determinant can be identified as

$$\sqrt{g(\mathbf{\Gamma}_t, t)} = \exp[-w(\mathbf{\Gamma}_t, t)] \qquad \sqrt{g(\mathbf{\Gamma}_0, 0)} = \exp[-w(\mathbf{\Gamma}_0, 0)] \tag{4.145}$$

which satisfies the metric transformation rule (4.136).
Note that for Hamiltonian systems, equation (4.144) reduces to the well-known result

$$\text{Hamiltonian systems :} \qquad d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.146}$$

and the familiar fixed Euclidean geometry of Hamiltonian phase space. The existence of an invariant measure means that the phase space average of some property, expressed in terms of an integral over the manifold, can be related to the time average of the same property over a trajectory generated by the non-Hamiltonian dynamics equations (4.115) under the usual assumption of ergodicity. The above formalism is also true for the case that the compressibility κ does not contain explicit time dependence. In some cases, it may be possible to transform to a set of phase space coordinate where $\sqrt{g(\mathbf{\Gamma}_t, t)}$ is unity or at least a constant. In this case, the phase space can be considered flat. However, transformation to such a coordinate system may not be simple or may not exist for a given non Hamiltonian system. The question of whether such transformations generally exist or not, although an interesting one, will not be addressed in the present work, as the general framework being introduced here is independent of this question. Indeed, the utility of equation (4.140) is that it permits an analysis of any non Hamiltonian system to be carried out in an arbitrary set of coordinates. It should be noted that equations (4.140) and (4.165) (see below) are formulated for a general phase space manifold and are, therefore, valid even if a transformation to a system where $\sqrt{g}$ is unity (or at least constant) does not exist and the phase space is not flat. Indeed, a constant $\sqrt{g}$ is a necessary but not a sufficient condition for a Riemannian space to be a flat space. The more stringent condition is that the Riemann curvature tensor vanish, which will hold in any coordinate system. This condition will be met manifestly if, in a given coordinate system, all of the components G_{ij} of the metric tensor are constant. The complexity of these issues is avoided, however, by using the formalism in Ref. [37, 38].

The existence of an invariant measure also means that there is an underlying fixed manifold with possibly nontrivial curvature, for which the metric is determined by the compressibility. The exponential factor can be viewed as a metric determinant factor $\sqrt{g(\mathbf{\Gamma}_t, t)}$, where $g(\mathbf{\Gamma}_t, t)$ is the determinant of the metric tensor $G(\mathbf{\Gamma}_t, t)$ obtained from $G(\mathbf{\Gamma}_0, 0)$ via the coordinate transformation $\mathbf{\Gamma}_0 \to \mathbf{\Gamma}_t$. $G(\mathbf{\Gamma}_0, 0)$ is the metric tensor describing the geometry of the space. Thus, at any time t, a nonunit metric determinant implies the existence of a nonzero dynamical compressibility. Equation (4.136) implies that the metric satisfies the same differential equation as does $\bar{\mathcal{J}}$, so that

$$\frac{d\sqrt{g(\mathbf{\Gamma}_t, t)}}{dt} = -\sqrt{g(\mathbf{\Gamma}_t, t)}\ \kappa(\mathbf{\Gamma}_t, t) \tag{4.147}$$

Equation (4.147) can be demonstrated by simply taking the time derivative of the second expression in equation (4.136), that is

$$\begin{aligned}
\frac{d\sqrt{g(\mathbf{\Gamma}_t, t)}}{dt} = \frac{d}{dt}\left[\sqrt{g(\mathbf{\Gamma}_0, 0)}\,\bar{\mathcal{J}}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0)\right] &= \frac{d\sqrt{g(\mathbf{\Gamma}_0, 0)}}{dt}\ \bar{\mathcal{J}}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0) + \sqrt{g(\mathbf{\Gamma}_0, 0)}\ \frac{d\bar{\mathcal{J}}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0)}{dt} \\
&\overset{(4.129)}{=} \frac{d\sqrt{g(\mathbf{\Gamma}_0, 0)}}{dt}\ \bar{\mathcal{J}}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0) - \sqrt{g(\mathbf{\Gamma}_0, 0)}\ \bar{\mathcal{J}}(\mathbf{\Gamma}_t; \mathbf{\Gamma}_0)\,\kappa(\mathbf{\Gamma}_t, t) \\
&\overset{(4.136)}{=} \frac{d\sqrt{g(\mathbf{\Gamma}_0, 0)}}{dt}\ \frac{\sqrt{g(\mathbf{\Gamma}_t, t)}}{\sqrt{g(\mathbf{\Gamma}_0, 0)}} - \sqrt{g(\mathbf{\Gamma}_0, 0)}\ \frac{\sqrt{g(\mathbf{\Gamma}_t, t)}}{\sqrt{g(\mathbf{\Gamma}_0, 0)}}\,\kappa(\mathbf{\Gamma}_t, t) \\
&= \sqrt{g(\mathbf{\Gamma}_t, t)}\ \frac{d(\ln\sqrt{g(\mathbf{\Gamma}_0, 0)}\,)}{dt} - \sqrt{g(\mathbf{\Gamma}_t, t)}\,\kappa(\mathbf{\Gamma}_t, t) \\
&\overset{(4.145)}{=} -\sqrt{g(\mathbf{\Gamma}_t, t)}\ \frac{dw(\mathbf{\Gamma}_0, 0)}{dt} - \sqrt{g(\mathbf{\Gamma}_t, t)}\,\kappa(\mathbf{\Gamma}_t, t) = -\sqrt{g(\mathbf{\Gamma}_t, t)}\,\kappa(\mathbf{\Gamma}_t, t)
\end{aligned} \tag{4.148}$$

where the last identity holds because $w(\mathbf{\Gamma}_0, 0)$ does not depend on time, so that $dw(\mathbf{\Gamma}_0, 0)/dt = 0$.

Consider next an arbitrary ensemble described by a distribution function $f : R^{n+1} \to R^1$, i.e. $f \equiv f(\mathbf{\Gamma}, t)$ is a function of the n coordinates and time t. A continuity equation for f can be derived by considering that the number of ensemble members $N(t)$ in a volume Ω of the space is given by

$$N(t) = \int_\Omega \omega f \tag{4.149}$$

The continuity condition is that the rate of change of the number of ensemble members within Ω is balanced by the flux of members through the surface bounding Ω, which is expressed mathematically as

$$-\frac{d}{dt}\int_\Omega \omega\, f = \int_{\partial\Omega} \sigma\, \hat{n}\cdot\boldsymbol{\lambda}\, f = \int_\Omega \mathcal{L}_\lambda(f\omega) \tag{4.150}$$

where σ is the surface $n-1$ form, $\hat{n}$ is the unit normal one-form to the surface, and the surface integral has been converted to a volume integral via a generalization of the divergence theorem to manifolds with nontrivial metrics using the Lie derivative $\mathcal{L}_\lambda$ along the vector $\boldsymbol{\lambda}$. Thus, equation (4.150) can be written as

$$\int_\Omega \left(\frac{\partial}{\partial t} + \mathcal{L}_\lambda\right)(f\omega) = 0 \tag{4.151}$$

Equation (4.151) must hold independently of the choice of Ω and thus implies the local continuity condition

$$\left(\frac{\partial}{\partial t} + \mathcal{L}_\lambda\right)(f\omega) = 0 \tag{4.152}$$

Equation (4.152) represents a continuity equation for f on an arbitrary manifold but makes no reference to a specific choice of coordinate basis. To project equation (4.152) onto a coordinate basis, the Leibniz rule can be applied to the action of the Lie derivative on the product, namely,

$$\mathcal{L}_\lambda(f\omega) = \omega\,\mathcal{L}_\lambda\, f + f\,\mathcal{L}_\lambda\,\omega \tag{4.153}$$

The action of the Lie derivative on the scalar f and on the volume form ω is well known to be

$$\mathcal{L}_\lambda f = \sum_{\alpha=1}^{n} \lambda^\alpha \frac{\partial f}{\partial \Gamma^\alpha} \tag{4.154}$$

$$\begin{aligned}\mathcal{L}_\lambda \varepsilon_{i_1\cdots i_n} &= \sum_{\alpha=1}^{n} \lambda^\alpha \frac{\partial \sqrt{g}}{\partial \Gamma^\alpha} \varepsilon_{i_1\cdots i_n} + \sum_{\alpha=1}^{n} \sqrt{g}\left(\varepsilon_{\alpha\, i_2\cdots i_n} \frac{\partial \lambda^\alpha}{\partial \Gamma^{i_1}} + \cdots + \varepsilon_{i_1\cdots i_{n-1}\alpha} \frac{\partial \lambda^\alpha}{\partial \Gamma^{i_n}}\right) \\ &= \sum_{\alpha=1}^{n} \dot{\Gamma}^\alpha \frac{\partial \sqrt{g}}{\partial \Gamma^\alpha} \varepsilon_{i_1\cdots i_n} + \sum_{\alpha=1}^{n} \sqrt{g} \frac{\partial \lambda^\alpha}{\partial \Gamma^\alpha} \varepsilon_{i_1\cdots i_n}\end{aligned} \tag{4.155}$$

where the component representation of the wedge product is given by $\varepsilon_{i_1\cdots i_n}$, the Levi-Civita tensor, and the last line follows from the properties of the $\varepsilon_{i_1\cdots i_n}$. These results can be used to derive the new form of the Liouville theorem for non Hamiltonian systems. Combining the results of equation (4.152) with equations (4.154) and (4.155) gives the general form for the continuity equation in an arbitrary coordinate basis as

$$\begin{aligned}&\frac{\partial(f\omega)}{\partial t} + \omega\, \mathcal{L}_\lambda f + f\, \mathcal{L}_\lambda \omega = \left[\frac{\partial(f\sqrt{g})}{\partial t} + \sqrt{g} \sum_{\alpha=1}^{n} \lambda^\alpha \frac{\partial f}{\partial \Gamma^\alpha} + f \sum_{\alpha=1}^{n} \dot{\Gamma}^\alpha \frac{\partial \sqrt{g}}{\partial \Gamma^\alpha} + f \sum_{\alpha=1}^{n} \sqrt{g} \frac{\partial \lambda^\alpha}{\partial \Gamma^\alpha}\right] \varepsilon_{i_1\cdots i_n} \\ &= \left[\frac{\partial(f\sqrt{g})}{\partial t} + \sum_{\alpha=1}^{n} \frac{\partial(f\sqrt{g}\,\lambda^\alpha)}{\partial \Gamma^\alpha}\right] d\Gamma^1 \wedge \cdots \wedge d\Gamma^n = 0\end{aligned}$$

Since the volume n-form is not zero, the term in brackets must, therefore, vanish yielding

$$\frac{\partial(f\sqrt{g})}{\partial t} + \sum_{\alpha=1}^{n} \frac{\partial}{\partial \Gamma^\alpha}(f\sqrt{g}\,\lambda^\alpha) = \frac{\partial(f\sqrt{g})}{\partial t} + \nabla_{\mathbf{\Gamma}} \cdot (f\sqrt{g}\,\boldsymbol{\lambda}) = 0 \tag{4.156}$$

$$\frac{\partial(f\sqrt{g})}{\partial t} + \nabla_{\mathbf{\Gamma}} \cdot (f\sqrt{g}\,\dot{\mathbf{\Gamma}}) = 0 \tag{4.157}$$

where

$$\nabla_{\mathbf{\Gamma}} = \left(\frac{\partial}{\partial \Gamma^1}, ..., \frac{\partial}{\partial \Gamma^n}\right) \tag{4.158}$$

Equation (4.156) is a general form of the Liouville equation, valid on a manifold with a nontrivial metric and hence is valid for an ensemble in which the underlying dynamics is compressible. Recall that the metric satisfies equation (4.147), therefore, explicitly writing the derivatives in (4.157) leads to

$$\sqrt{g}\frac{\partial f}{\partial t} + f\frac{\partial \sqrt{g}}{\partial t} + \sqrt{g}\,\dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} f + f\,\dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} \sqrt{g} + f\sqrt{g}\,\nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} = 0 \tag{4.159}$$

$$\sqrt{g}\left[\frac{\partial f}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} f\right] + f\sqrt{g}\,\nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} + f\left[\frac{\partial \sqrt{g}}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} \sqrt{g}\right] = 0 \tag{4.160}$$

$$\sqrt{g}\left[\frac{\partial f}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} f\right] + f\sqrt{g}\,\nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} - f\sqrt{g}\,\nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} = \sqrt{g}\left[\frac{\partial f}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} f\right] = 0 \qquad \forall \sqrt{g} \tag{4.161}$$

$$\rightarrow \qquad \frac{\partial f}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} f = \frac{df}{dt} = 0 \tag{4.162}$$

so that the conservation condition for the distribution function is recovered. Finally, combining this result with equation (4.140) leads to the general statement of Liouville theorem for a non Hamiltonian system,

$$f(\mathbf{\Gamma}_t, t)\, \exp[-w(\mathbf{\Gamma}_t, t)]\, d\Gamma_t^1 \wedge \cdots \wedge d\Gamma_t^n = f(\mathbf{\Gamma}_0, 0)\, \exp[-w(\mathbf{\Gamma}_0, 0)]\, d\Gamma_0^1 \wedge \cdots \wedge d\Gamma_0^n \tag{4.163}$$

where the conventional volume element has been used. Note that for Hamiltonian dynamics, the compressibility vanishes and $\sqrt{g} = 1$. In this case, equation (4.157) reduces to the usual Liouville equation, namely,

$$\text{Hamiltonian systems :} \qquad \frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{\Gamma}} \cdot (f\, \dot{\mathbf{\Gamma}}) \equiv \frac{\partial f}{\partial t} + \nabla_{\mathbf{\Gamma}} \cdot (f\, \dot{\mathbf{\Gamma}}) = 0 \tag{4.164}$$

$$\text{non Hamiltonian systems :} \qquad \frac{\partial (f\sqrt{g})}{\partial t} + \frac{\partial}{\partial \mathbf{\Gamma}} \cdot (f\sqrt{g}\, \dot{\mathbf{\Gamma}}) \equiv \frac{\partial (f\sqrt{g})}{\partial t} + \nabla_{\mathbf{\Gamma}} \cdot (f\sqrt{g}\, \dot{\mathbf{\Gamma}}) = 0 \tag{4.165}$$

where equation (4.165) is a covariant generalization of the standard Liouville equation (4.164). Remember that the covariant form (4.165) was derived from the balance between the rate of decrease of the number of ensemble members in the phase space volume and the flux of members through the boundary surface taking into account the geometry of the space (number conservation). It is, therefore, valid for equilibrium as well as nonequilibrium ensembles generated by Hamiltonian or non-Hamiltonian systems.
Clearly, equation (4.165) reduces to equation (4.164) for Hamiltonian systems where $\sqrt{g(\mathbf{\Gamma}_t, t)} = 1$ and $\nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} = 0$. As alluded to above, if a transformation to a frame where $\nabla_{\mathbf{\Xi}} \cdot \dot{\mathbf{\Xi}} = 0$ exists with $\Xi^k = \Xi^k(\Gamma^1, ..., \Gamma^n)$ then equation (4.165) reduces to (4.164) as can be verified by applying the transformation directly. Thus, equation (4.165) is valid when the phase space is not flat and remains valid in the limit in which it is flat. The important point here is that the phase space distribution $f(\mathbf{\Gamma}_t, t)$ of interest, which gives the probability density in phase space, should be kept separate from the phase space metric $\sqrt{g(\mathbf{\Gamma}_t, t)}$, which ensures that the volume of phase space of a non Hamiltonian system is invariant under time evolution. Indeed, it has to be underlined that equation (4.157) is *different* from the equation which has previously been assumed to be the generalized form of the Liouville equation,

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{\Gamma}} \cdot (f\, \dot{\mathbf{\Gamma}}) = 0 \tag{4.166}$$

Equation (4.157) can be put in this form by defining a new function $\tilde{f} = f\sqrt{g}$. Such an identification is not general, however. The consequences of this identification are explored in the following discussion.
The generalized Liouville equation and the invariant measure will be used to derive an important property satisfied by the Gibbs entropy of a system. Consider, first, a Hamiltonian system for which the Gibbs entropy $S(t)$ is given by

$$S(t) = -k_b \int d\Gamma^1 \cdots d\Gamma^n\, f\, \ln(f) \tag{4.167}$$

where k_b is the Boltzmann constant. Although the standard integral notation has been employed for simplicity, the connection between the volume element and the contraction of the volume form in equation (4.138) (with $\sqrt{g} = 1$) must be kept in mind. By taking the time derivative of both sides of equation (4.167), using the Hamiltonian form of Liouville equation, and performing a few integrations by parts, it can be shown that

$$\frac{dS(t)}{dt} = -k_b\, \frac{d}{dt} \int d\Gamma^1 \cdots d\Gamma^n\, f\, \ln(f) \tag{4.168}$$

Thus, for a Hamiltonian system, the Gibbs entropy, which is a fine-grained quantity, is constant in time. The generalization of the Gibbs entropy for a non Hamiltonian system can be shown to satisfy the same property, when the proper invariant measure is used. Recognizing that the phase space volume element now contains a metric determinant factor, the entropy can be expressed as

$$S(t) = -k_b \int d\Gamma^1 \cdots d\Gamma^n\, \sqrt{g}\, f\, \ln(f) \tag{4.169}$$

where the metric determinant is assumed to contain no explicit time dependence. Again, the standard integral notation is connected to equation (4.138) through a contraction of the volume n-form, but with $\sqrt{g}$ correctly determined by the compressibility. Computing the time derivative of both sides gives

$$\begin{aligned} \frac{dS(t)}{dt} &= -k_b \int d\Gamma^1 \cdots d\Gamma^n\, \sqrt{g}\, [1 + \ln(f)]\, \frac{\partial f}{\partial t} = -k_b \int d\Gamma^1 \cdots d\Gamma^n\, [1 + \ln(f)]\, \frac{\partial (f\sqrt{g})}{\partial t} \\ &= k_b \int d\Gamma^1 \cdots d\Gamma^n\, [1 + \ln(f)]\, \nabla_{\mathbf{\Gamma}} \cdot (f\sqrt{g}\, \dot{\mathbf{\Gamma}}) \end{aligned} \tag{4.170}$$

In general, the ensemble average of any property $A(\mathbf{\Gamma}, t)$ is determined from the invariant measure and the ensemble distribution function $f(\mathbf{\Gamma}, t)$, so that

$$\langle A(t) \rangle = \frac{1}{Z(t)} \int d\mathbf{\Gamma} \sqrt{g(\mathbf{\Gamma}, t)}\, A(\mathbf{\Gamma}, t)\, f(\mathbf{\Gamma}, t) \tag{4.171}$$

where the normalization function Z is the so-called partition function, defined as

$$Z(t) = \int d\mathbf{\Gamma} \sqrt{g(\mathbf{\Gamma}, t)}\, f(\mathbf{\Gamma}, t) \tag{4.172}$$

Consider the case where the λ^α in equation (4.115) do not contain explicit time dependence and there are no external fields present. If, in addition, $\partial\sqrt{g}/\partial t = 0$, then an equilibrium ensemble, for which $\partial f/\partial t = 0$ can be associated with equations (4.115). From equation (4.171), it can be seen that these conditions are compatible with the stationarity of the ensemble average $\langle A \rangle$.
Moreover, as a consequence of the coordinate transformation (4.117), and thank to the relation (4.133), it is always possible to define an infinitesimal weighted phase space volume $dv_{\mathbf{\Gamma}}$ for each point in the phase space, that will retain its extension during the dynamic, that is

$$dv_{\mathbf{\Gamma}} \equiv \exp[-w(\mathbf{\Gamma}(t),\, t)]\, d\mathbf{\Gamma}(t) = \exp[-w(\mathbf{\Gamma}(t_0),\, t_0)]\, d\mathbf{\Gamma}(t_0) \tag{4.173}$$

Equation (4.173) shows that $dv_{\mathbf{\Gamma}}$ is an invariant measure on the phase space, that can be associated to an invariant volume form. A phase space distribution function, $f(\mathbf{\Gamma}(t), t)$, can then be defined so that $f(\mathbf{\Gamma}(t), t)\, dv_{\mathbf{\Gamma}}$ is the fraction of the total number of ensemble members contained in the phase space volume $dv_{\mathbf{\Gamma}}$ at time t. Its associated partition function

$$Z(t) = \int dv_{\mathbf{\Gamma}}\, f(\mathbf{\Gamma}(t), t) \tag{4.174}$$

as a proper normalization factor, represents the total number of accessible microstates in the phase space. In the absence of constraints, $Z(t)$ coincides with the total number of points in a given phase space at a certain instant of time t. As previously mentioned, the continuity condition states that the rate of change of the number of ensemble members within a volume in the phase space is balanced by the flux of members through the surface bounding that volume. Starting from this definition, as previously demonstrated, the use of a generalization of the divergence theorem to manifolds with nontrivial metrics by means of the Lie derivative and the projection of the resultant equation onto a coordinate basis, leads to the formulation of a generalized Liouville equation (4.157), valid on a manifold with a nontrivial metric (and hence valid for an ensemble in which the underlying dynamics has a non-unit compressibility).[37] Furthermore, as demonstrated in (9.98)-(9.104), substituting the time evolution equation of the square root of the metric determinant into the general form of the Liouville equation, leads to the conservation condition

$$\frac{df(\mathbf{\Gamma}(t), t)}{dt} = 0 \tag{4.175}$$

which affirms that, in a ensemble with a distribution in statistical equilibrium, the probability function $f(\mathbf{\Gamma}(t), t)$ is conserved along the trajectory, so that the following equivalences hold true

$$f(\mathbf{\Gamma}(t), t) \equiv f(\mathbf{\Gamma}(t_0), t_0) \equiv f(\mathbf{\Gamma}) \tag{4.176}$$

This means that for a general metric space, the fraction of members of the ensemble in the initial weighted volume element $\exp[-w(\mathbf{\Gamma}(t_0),\, t_0)]\, d\mathbf{\Gamma}(t_0)$ is equal to the fraction of members in any volume element

$$f(\mathbf{\Gamma}(t_0), t_0)\, \exp[-w(\mathbf{\Gamma}(t_0),\, t_0)]\, d\mathbf{\Gamma}(t_0) = f(\mathbf{\Gamma})\, \exp[-w(\mathbf{\Gamma})]\, d\mathbf{\Gamma} = f(\mathbf{\Gamma}) \sqrt{g(\mathbf{\Gamma})}\, d\mathbf{\Gamma} \tag{4.177}$$

It follows that the value of a macroscopic observable A, connected to a microscopic phase space function of the system coordinates $A(\mathbf{\Gamma}, t)$, can be calculated by performing an average over the phase space at any point in time, that is

$$\langle A \rangle = \int d\mathbf{\Gamma}\, f(\mathbf{\Gamma})\, A(\mathbf{\Gamma}) \sqrt{g(\mathbf{\Gamma})} \tag{4.178}$$

This means that a system of an ensemble characterized by a statistical equilibrium distribution is in thermodynamic equilibrium with respect to the set of variables $A(\mathbf{\Gamma}, t)$.
The corresponding partition function, that depends on the macroscopic observable used to define the ensemble, becomes also an equilibrium property, defined by

$$Z \equiv Z(t) = \int d\mathbf{\Gamma}\, f(\mathbf{\Gamma})\, \sqrt{g(\mathbf{\Gamma})} \tag{4.179}$$

where

$$\sqrt{g(\mathbf{\Gamma})} = \exp[-w(\mathbf{\Gamma})] \overset{(4.132)}{=} \exp\left[-\int \kappa(\mathbf{\Gamma})\, dt\right] \qquad \text{with} \qquad \kappa(\mathbf{\Gamma}) \overset{(4.123)}{=} \nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} \tag{4.180}$$

Finally, in the ergodic hypothesis, i.e. when a system during its evolution is able to visit all the allowed configurations, the ensemble average of an observable can be replaced by time averages over the trajectory according to

$$\langle A \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int A(\mathbf{\Gamma}(t))\, dt \tag{4.181}$$

Next, suppose that the set of dynamical equations possesses a set of n_c associated conservation laws or conserved quantities, labeled as

$$\Lambda_k(\mathbf{\Gamma}) = C_k \qquad \text{for} \quad k = 1, ..., n_c \tag{4.182}$$

which satisfy the condition

$$\frac{d\Lambda_k}{dt} = 0 \qquad k = 1, ..., n_c \tag{4.183}$$

Thus, a trajectory generated by equation (4.115) will not sample the entire phase space, but a subspace determined by the intersection of the hypersurfaces $\{\Lambda_k(\mathbf{\Gamma}) = C_k\}$ where C_k is a set of constants. Therefore, the microcanonical distribution function generated by the dynamical system can be constructed from a product of delta functions expressing the conservation laws,

$$f(\mathbf{\Gamma}) = \prod_{k=1}^{n_c} \delta(\Lambda_k(\mathbf{\Gamma}) - C_k) \tag{4.184}$$

with the associated partition function

$$Z = \int d\mathbf{\Gamma}\, \sqrt{g(\mathbf{\Gamma})} \prod_{k=1}^{n_c} \delta(\Lambda_k(\mathbf{\Gamma}) - C_k) \tag{4.185}$$

It is possible to show, by substitution, that equation (4.184) satisfies the generalized Liouville equation (4.165). In fact, equation (4.184) constitutes the complete solution for such a microcanonical ensemble, since all configurations that satisfy the conservation laws have equal probability of being visited by a trajectory. It is important to note that a distribution constructed from a product of δ functions corresponding to an arbitrary subset of the conservation laws, that is

$$f_{red}(\mathbf{\Gamma}) = \prod_{k=1}^{n_c'} \delta(\Lambda_k(\mathbf{\Gamma}) - C_k) \tag{4.186}$$

where $n_c' \leq n_c$, also satisfies the generalized Liouville equation (4.165). If $n_c' < n_c$ this solution will not properly describe the phase space distribution of a system with n_c conservation laws. Therefore, satisfying equation (4.165) is a necessary but not sufficient condition to guarantee that a particular phase space distribution function is generated by a given dynamical system. This illustrates the limitations of relying solely on the Liouville equation to determine the distribution function. Indeed, the true distribution must be consistent with *all* the conservation laws present.

4.3.2.1 The generalized phase space analysis

Given the above discussion, it is now possible to introduce a general procedure for constructing the partition function corresponding to the equilibrium ensemble generated by a non Hamiltonian dynamical system satisfying the above conditions. This procedure can be schematized in the following steps.

1. Determine *all* the conservation laws satisfied by the equations of motion.
   The distribution function $f(\mathbf{\Gamma})$ will then be given by equation (4.184).

2. Using the conservation laws and the equations of motion, identify and eliminate linearly dependent variables/solutions. That is, if $\Gamma^2(t) = c\,\Gamma^1(t)$, where c is a constant, then $\Gamma^2(t)$ must be removed from the formal analysis of the dynamical system. Note, however, that such dependencies form a set of conservation laws and can also be eliminated in steps 4 and 5 below if they are not eliminated in this step. Driven or trivial or uncoupled variables must also be eliminated in the formal analysis. That is, if $\dot{\Gamma}^\alpha = \lambda^\alpha(\Gamma^\alpha)$ and $\dot{\Gamma}^\beta = \lambda^\beta(\Gamma^\alpha, \Gamma^\beta)$ with Γ^α of primary importance, and there only exists conservation laws of the form $\Lambda_k(\Gamma^\alpha) = 0$ and $\Lambda_m(\Gamma^\beta) = 0$, then Γ^β must be eliminated, as the phase space distribution of Γ^α is, in no way, affected by Γ^β. This step will identify variables of lesser physical importance, e.g., center of mass motion, in complex systems. Idealized systems, such as separable systems, need to be treated as special cases.

3. Calculate the phase space compressibility, $\nabla_{\mathbf{\Xi}} \cdot \dot{\mathbf{\Xi}}$ of the remaining dynamical system $\dot{\mathbf{\Xi}} = \tilde{\boldsymbol{\lambda}}(\mathbf{\Xi})$ (having eliminated trivial variables and linear dependent solutions as described in step 2). Using the compressibility, determine the phase space metric $\sqrt{g(\mathbf{\Xi})}$ and generate the invariant volume element $\sqrt{g(\mathbf{\Xi})}\, d\mathbf{\Xi}$.

4. The microcanonical partition function for the non Hamiltonian system can now be constructed using the formula

$$Z(C_1, ..., C_n) = \zeta \int d\mathbf{\Xi}\, \sqrt{g(\mathbf{\Xi})}\; f(\mathbf{\Xi}) = \zeta \int d\mathbf{\Xi}\, \sqrt{g(\mathbf{\Xi})} \prod_{k=1}^{n_c} \delta(\Lambda_k(\mathbf{\Xi}) - C_k) \tag{4.187}$$

   Note that the partition function is invariant under coordinate transformations $\mathbf{\Xi}_0 \to \mathbf{\Xi}_t$ and can therefore be computed in terms of the coordinates at any time t.

5. If equation (4.115) corresponds to a system with an extended phase space, then the partition function (4.187) must be integrated over the extended phase space variables in order to determine the distribution function sampled by the physical variables.

The theoretical results of steps 1 to 5 should always be tested numerically on a set of model problems. Step 5 assumes ergodicity on the $\mathbf{\Xi}$ phase space subject to the conservation laws, and steps 1 to 4 are merely a necessary but not sufficient condition to ensure ergodicity. In particular, steps 1 to 4 do not address more complex issues of ergodicity such as arise when there are large potential energy barriers. However, steps 1 to 4 are sufficient for the problems studied here. It is also important to note that only an exceedingly small number of systems can be shown formally to be ergodic and it is unlikely that ergodicity can be shown analytically for the complex set of systems of interest in condensed matter physics.

4.3.2.2 Other forms of the generalized Liouville equation

Of primary importance is the generalized Liouville equation (4.165). Although a rigorous derivation of this equation was presented in Section 4.3.2, the line of reasoning that leads to this formulation is as follows.

(i) Equation (4.140) implies that there is a measure conservation law that, most generally, involves a nontrivial metric. This suggests that phase space should be carefully treated using the general rules of the geometry of manifolds.

(ii) Based on considerations at item (i), the Liouville equation was rederived using the mathematical techniques of differential geometry.

(iii) The rules of differential geometry require that the phase space distribution function $f(\mathbf{\Gamma}, t)$ be kept separate from the phase space metric $\sqrt{g(\mathbf{\Gamma}, t)}$ and, therefore, leads naturally to equation (4.165) which contains both a metric and a distribution function.

Formalisms existing before the work of M. E. Tuckerman et al.[38] for non Hamiltonian systems are based on a generalized Liouville equation of the form

$$\frac{\partial \tilde{f}}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} \tilde{f} = -\tilde{f}\, \nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} \tag{4.188}$$

which is derived assuming a flat phase space. In equation (4.188), the notation $\tilde{f}$ is used to distinguish it from f in equation (4.165). Clearly, the expression (4.188) is related to the generalized form (4.165) through the identification $\tilde{f} = \sqrt{g}\, f$. Therefore, equation (4.188) is not incorrect, but is incomplete. The more mathematically precise and fundamental form (4.165) has a powerful advantage. It allows one to directly construct the microcanonical partition function in a manner analogous to the Hamiltonian case. Because equation (4.188) is incomplete, its use to extract the partition function is awkward and is not rigorous. What is generally done is to postulate a form for $\tilde{f}$ and show that this form satisfies equation (4.188). Such an ad hoc procedure usually misses important information and often leads to incomplete solutions. Specific failures of this simplistic approach are discussed in detail in Ref [38]. In order to demonstrate this more generally, note that equation (4.188) admits any general solution of the form

$$\tilde{f}(\mathbf{\Gamma}, t) = f_0(\mathbf{\Gamma}, t)\, f_1(\mathbf{\Gamma}, t) \qquad \text{where } f_0(\mathbf{\Gamma}, t) \text{ is any function satisfying} \quad \frac{df_0}{dt} = 0 \tag{4.189}$$

Once this fact is recognized, then it is clear that equation (4.188) reverts to the form (4.165) since, in (4.165), the distribution function $f(\mathbf{\Gamma}, t)$ satisfies $df/dt = 0$. In other words, equation (4.188) alone cannot determine the function $f_0(\mathbf{\Gamma}, t)$ since it simply cancels out. By contrast, the formalism presented in the work of M. E. Tuckerman et al.[38] shows how to determine the phase space metric and how to construct the phase space distribution so that a full and general solution to equation (4.165) is obtained. Other authors have suggested working only in coordinate systems in which $\sqrt{g}$ is constant, in which case, the deficiencies of equation (4.188) are bypassed. However, this transformation may not exist or may simply be nontrivial to employ. The covariant formulation presented in Ref. [38] and described above also permits the use of such coordinate systems, but does not require them.

The remainder of this section will be devoted to applying the above procedure to several commonly used non Hamiltonian dynamical systems and demonstrating the failures of older analyses. In particular, canonical (NVT) and isothermal-isobaric (NPT) extended system methods, and the isokinetic method will be considered. The canonical and isothermal-isobaric ensembles are very useful. Therefore, various extended system molecular dynamics methods have been designed to treat them. However, in formulating these methods, the use of the standard Liouville equation (4.188) has led to incorrect definitions of the ensemble produced by the dynamics. That is, rather than using a generalized microcanonical partition function (see equation (4.187)), a distribution function $\tilde{f}(\mathbf{\Gamma}, t)$ is postulated, shown to satisfy equation (4.188), and the equations of motion pronounced correct. Often, the postulated form is incomplete in the sense of equation (4.186) and does not properly describe the distribution generated by the dynamics, i.e., the equations of motion may generate a distribution that is not of the predicted form. As demonstrated in Section 5.3.4.4, the new phase space picture succeeds in predicting complex distribution functions where these older methods fail. Furthermore, it will be shown during the derivations that the extended system equations of motion for the canonical and isothermal-isobaric ensemble possess relatively simple metrics. It is, therefore, possible to define a simple transformation to a coordinate system where $\sqrt{g(\mathbf{\Gamma}, t)}$ is constant. For this reason, in Section 5.3.1 the isokinetic equations of motion, derived from Gauss principle of least constraint, will be examined. Indeed, they have a nontrivial metric and can be used to analyze one case for which the new non Hamiltonian phase space formalism leads to the correct isokinetic partition function.

In many applications, the correct partition function for a given ensemble (NVT or NPT) can be obtained from the generalized microcanonical partition function (4.187), by carrying out the integration over the unphysical variables that have been introduced to represent the effect of a thermostat or barostat. In order to do this properly, it is essential to identify all the conservation laws. Moreover, it is useful to eliminate from the analysis all those coordinates that are linearly dependent on other variables and

variables that are driven (variables are called driven when they are not coupled through a conservation law and they do not influence the time evolution of the physical variables of interest in the system, even though their own time evolution may depend on these last variables).

Chapter 5

Generation of statistical ensembles

5.1 The microcanonical ensemble

5.1.1 Equations of motion

The equations of motion analyzed here are given by the simple Newton time evolution equations (4.11) for a system in d dimensions, that can be rewritten as

$$\dot{\boldsymbol{q}}_i = \frac{d\boldsymbol{q}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} \qquad i = 1, ..., N \tag{5.1}$$

$$\dot{\boldsymbol{p}}_i = \frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} \tag{5.2}$$

where $\boldsymbol{q}_i$ and $\boldsymbol{q}_i$ are d dimensional vectors. The Hamiltonian is given by the sum of nuclear kinetic and potential energies, that is

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{5.3}$$

where $\boldsymbol{p}_i = m_i \boldsymbol{v}_i$ is the generalized momentum of the i-th nucleus and $\phi(\boldsymbol{q})$ is the potential generated by the field effect of the ground state electronic and nuclear degrees of freedom computed at the HF or DFT level, at a fixed nuclear configuration $\boldsymbol{q} \equiv \{\boldsymbol{q}_i\}$. Substituting the form of the Hamiltonian (5.3) in the Hamiltonian equations of motion (5.1) - (5.2), they become

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.4}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} = \mathbf{F}_i \tag{5.5}$$

The Hamilton equations of motion (5.1)-(5.2) induce some conserved quantities, if particular conditions are satisfied. The study of the constants of motion is essential to correctly describe the ensemble sampled in the phase space by the Hamilton equations of motion, though the definition of the distribution and the partition functions generated by the Hamiltonian flow.

5.1.2 Integration of the equations of motion

The time evolution of the phase space point $\boldsymbol{x}(t)$ is formally given by equation (4.61), or by its generalization (4.65), and it can be written in a compact form thanks to the definition of the Liouville operator $i\mathcal{L}$, given by equation (4.43). Using the general vector notation $\boldsymbol{\Gamma}$, which is defined in this case by $\boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t))$, the time evolution in phase space (4.65) from an initial time t_0 to a generic time t is written as

$$\boldsymbol{\Gamma}(t) = e^{i\mathcal{L}(t-t_0)}\, \boldsymbol{\Gamma}(t_0) \tag{5.6}$$

where the Liouville operator is given by equation (4.62) reported here below

$$i\mathcal{L} = \dot{\boldsymbol{\Gamma}} \cdot \frac{\partial}{\partial \boldsymbol{\Gamma}} \qquad \text{with} \qquad \boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t)) \tag{5.7}$$

where the dot over the phase space vector variable stands for the time derivative. The Newton equations of motion (4.11) can be rewritten in a more familiar way as

$$\dot{\boldsymbol{q}}_i = \frac{d\boldsymbol{q}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} \qquad \dot{\boldsymbol{p}}_i = \frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} \qquad i = 1, ..., N \tag{5.8}$$

Applying equation (5.7), the Liouville operator for the time evolution of the phase space point that follows Newton equations of motion in the Hamilton form (5.8) can be written as

$$\begin{aligned} i\mathcal{L} &= \sum_{i=1}^{N} \frac{d\boldsymbol{q}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \sum_{i=1}^{N} \frac{d\boldsymbol{p}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} = \sum_{i=1}^{N} \dot{\boldsymbol{q}}_i \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \sum_{i=1}^{N} \dot{\boldsymbol{p}}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \\ &= \sum_{i=1}^{N} \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} - \sum_{i=1}^{N} \frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} - \sum_{i=1}^{N} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \end{aligned} \tag{5.9}$$

where the partial derivative of the Hamiltonian has been computed using the Hamiltonian form (5.3). For a system of N point particles in three dimensions ($d = 3N$) the Liouville equation can be reformulated as follows

$$i\mathcal{L} = \sum_{\alpha=1}^{3N} \left[\dot{q}_\alpha \frac{\partial}{\partial q_\alpha} + \dot{v}_\alpha \frac{\partial}{\partial v_\alpha} \right] = \sum_{i=1}^{N} \left[\boldsymbol{v}_i \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \boldsymbol{a}_i \cdot \frac{\partial}{\partial \boldsymbol{v}_i} \right] \tag{5.10}$$

so that the Liouville operator form defined in equation (4.43) is recovered. The explicit summation in the first expression in (5.10) is partially substituted and simplified with a vector notation which groups together three elements q_α or v_α and it allows to quickly identify the components of a given atom, that is to say, $\boldsymbol{v}_i = (v_{xi}, v_{yi}, v_{zi})$, $\boldsymbol{q}_i = (q_{xi}, q_{yi}, q_{zi})$ and $\boldsymbol{a}_i = (a_{xi}, a_{yi}, a_{zi})$, with $i = 1, ..., N$, are respectively the nuclear velocity, position and acceleration vectors of the i-th atom. For further details about the difference in notation between (4.43) and the last espression in (5.10), see Appendix A, Section A.3. The operator (5.10) can be divided into two parts

$$i\mathcal{L} = i\mathcal{L}_q + i\mathcal{L}_v \tag{5.11}$$

where

$$i\mathcal{L}_q = \sum_{i=1}^{N} \boldsymbol{v}_i \cdot \frac{\partial}{\partial \boldsymbol{q}_i} \qquad i\mathcal{L}_v = \sum_{i=1}^{N} \boldsymbol{a}_i \cdot \frac{\partial}{\partial \boldsymbol{v}_i} \tag{5.12}$$

Using this Liouville operator, the equation (4.61) can then be formulated as

$$\boldsymbol{x}(t) = e^{i(\mathcal{L}_q + \mathcal{L}_v)(t - t_0)}\, \boldsymbol{x}(t_0) \tag{5.13}$$

or, generalizing the vector in phase space (but in the present case the following equation is perfectly equivalent to the previous one), it can be formulated as

$$\boldsymbol{\Gamma}(t) = e^{i(\mathcal{L}_q + \mathcal{L}_v)(t - t_0)}\, \boldsymbol{\Gamma}(t_0) \tag{5.14}$$

However, the two operators $i\mathcal{L}_q$ and $i\mathcal{L}_v$ do not commute

$$i\mathcal{L}_q\, i\mathcal{L}_v \neq i\mathcal{L}_v\, i\mathcal{L}_q \tag{5.15}$$

and hence

$$e^{i(\mathcal{L}_q + \mathcal{L}_v)(t - t_0)} \neq e^{i\mathcal{L}_q (t - t_0)} e^{i\mathcal{L}_v (t - t_0)} \tag{5.16}$$

so that equation (5.13) (or (5.14)) is not easily solved. For non-commuting operators, the Suzuki-Trotter decomposition can be used (see Section 4.2.1), which is very helpful to simplify and solve equation (5.13) (or (5.14)). By defining a time step $\Delta t = (t - t_0)/P$ so that the time t can be discretized and written as a function of an integer P, equation (5.14) can be written as

$$\boldsymbol{\Gamma}(t_0 + P\,\Delta t) = e^{i\mathcal{L}\,P\Delta t}\, \boldsymbol{\Gamma}(t_0) \tag{5.17}$$

The application of the identity (4.85) to the present case leads to the following expression for the time propagator

$$e^{i\mathcal{L}\,P\Delta t} = e^{i(\mathcal{L}_q+\mathcal{L}_v)P\Delta t} = \lim_{\Delta t\to 0(P\to\infty)} \left[e^{i\mathcal{L}_v\Delta t/2}\, e^{i\mathcal{L}_q\Delta t}\, e^{i\mathcal{L}_v\Delta t/2} \right]^P \tag{5.18}$$

For large but finite integers P, the expression for the propagator $\exp[i\mathcal{L}(t-t_0)] = \exp[i\mathcal{L}P\Delta t]$ can be approximated applying equation (4.86), that is

$$e^{i\mathcal{L}\,P\Delta t} \approx \left[e^{i\mathcal{L}_v\Delta t/2}\, e^{i\mathcal{L}_q\Delta t}\, e^{i\mathcal{L}_v\Delta t/2} \right]^P + \mathcal{O}(P\Delta t^3) \tag{5.19}$$

thus obtaining a time propagator operator that is clearly unitary, therefore time reversible, and is also correct up to the second order.

Note: A formally identical expression for the time propagator can be obtained by exchanging the order of the Liouville operator in the exponent formulation in (5.18), so that for large but finite integers P, the expression for the propagator can be approximated as

$$e^{i\mathcal{L}\,P\Delta t} = e^{i(\mathcal{L}_v+\mathcal{L}_q)P\Delta t} \approx \left[e^{i\mathcal{L}_q\Delta t/2}\, e^{i\mathcal{L}_v\Delta t}\, e^{i\mathcal{L}_q\Delta t/2} \right]^P + \mathcal{O}(P\Delta t^3) \tag{5.20}$$

The two different form of the time integrators (5.19) and (5.20) lead to two different algorithms for phase space evolution of the simple Newton equations of motion for the nuclei, namely, the velocity Verlet and the position Verlet algorithms, respectively. The first one is the most stable and useful algorithm in molecular dynamics simulations, so that it is discussed and derived in more details in the following. The position Verlet is essentially identical to the velocity Verlet. A shift of a time origin by $\Delta t/2$ of either the position or the velocity Verlet equations would actually make both integrator perfectly equivalent. However, as pointed out in Ref. [37], half time steps are not formally defined, being the right hand side of equations (5.19) and (5.20) an approximation of the discrete time propagator for the full step Δt. Velocity Verlet and position Verlet, therefore, do not generate numerically identical trajectories, although of course they are formally equivalent and the trajectories produced by both of them are similar. It is indeed noticeable that using the same Liouville formalism different long-time integrator schemes can be derived. This is a first hint of the power of the Liouville approach, that represents a unifying treatment for understanding the properties and relationships between stepwise time integrators.

Inserting the approximate expression (5.19) for the time propagator in equation (5.17), the time evolution of the phase space vector for P dynamical steps and with a time step equal to Δt can be written in an approximate way as

$$\mathbf{\Gamma}(t_0 + P\,\Delta t) \approx \left[e^{i\mathcal{L}_v\Delta t/2}\, e^{i\mathcal{L}_q\Delta t}\, e^{i\mathcal{L}_v\Delta t/2} \right]^P \mathbf{\Gamma}(t_0) + \mathcal{O}(P\Delta t^3) \tag{5.21}$$

and considering the time evolution starting from a generic time t (up to a value of time equal to $t+P\Delta t$), the previous equation can be written as

$$\mathbf{\Gamma}(t + P\,\Delta t) \approx \left[e^{i\mathcal{L}_v\Delta t/2}\, e^{i\mathcal{L}_q\Delta t}\, e^{i\mathcal{L}_v\Delta t/2} \right]^P \mathbf{\Gamma}(t) + \mathcal{O}(P\Delta t^3) \tag{5.22}$$

The action of the two propagators (5.12) in (5.22) can be evaluated analytically using the derivation given in Section 4.2.1. As previously demonstrated, the action of the propagator with form $\exp(i\mathcal{L}_q t)$ only translates all position coordinates

$$\boldsymbol{q}_i \;\leftarrow\; \boldsymbol{q}_i + \boldsymbol{v}_i t \qquad\qquad i = 1, ..., N \tag{5.23}$$

and the propagator $\exp(i\mathcal{L}_v t)$ only translates all velocity coordinates

$$\boldsymbol{v}_i \;\leftarrow\; \boldsymbol{v}_i + \boldsymbol{a}_i t \qquad\qquad i = 1, ..., N \tag{5.24}$$

Once the action of the time propagator operators are known, the equation (5.22) can be used in a numerical propagation scheme. To see how this works in practice, consider the i-th particle moving in a

three-dimensional space without constraint, and apply the equation (5.22) for only one dynamical time step, i.e. taking $P = 1$,

$$\mathbf{\Gamma}(t + \Delta t) \approx \left[e^{i\mathcal{L}_v \Delta t/2}\, e^{i\mathcal{L}_q \Delta t}\, e^{i\mathcal{L}_v \Delta t/2} \right] \mathbf{\Gamma}(t) + \mathcal{O}(\Delta t^3) \tag{5.25}$$

The action of the time propagator in equation (5.22) for one dynamical step ($P = 1$) on the phase space coordinates $(\boldsymbol{q}_i, \boldsymbol{v}_i)$ from a generic time t to a value of time equal to $t + \Delta t$ can be computed as [1]

$$\begin{aligned}
\begin{pmatrix} \boldsymbol{q}_i(t+\Delta t) \\ \boldsymbol{v}_i(t+\Delta t) \end{pmatrix} &= e^{i\mathcal{L}_v \Delta t/2}\, e^{i\mathcal{L}_q \Delta t}\, e^{i\mathcal{L}_v \Delta t/2} \begin{pmatrix} \boldsymbol{q}_i(t) \\ \boldsymbol{v}_i(t) \end{pmatrix} = e^{i\mathcal{L}_v \Delta t/2}\, e^{i\mathcal{L}_q \Delta t} \begin{pmatrix} \boldsymbol{q}_i(t) \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] \end{pmatrix} \\
&= e^{i\mathcal{L}_v \Delta t/2} \begin{pmatrix} \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t)] \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{q}_i(t) + \Delta t\, \{\boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)]\} \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t) + \Delta t\, \{\boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)]\}] \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] + \frac{\Delta t}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)]] \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2} \{\boldsymbol{a}_i[\boldsymbol{q}_i(t)] + \boldsymbol{a}_i[\boldsymbol{q}_i(t + \Delta t)]\} \end{pmatrix}
\end{aligned} \tag{5.26}$$

The resultant expression for one dynamical time step ($P = 1$) is

$$\begin{pmatrix} \boldsymbol{q}_i(t+\Delta t) \\ \boldsymbol{v}_i(t+\Delta t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\, \boldsymbol{a}_i[\boldsymbol{q}_i(t)] \\ \boldsymbol{v}_i(t) + \frac{\Delta t}{2} \{\boldsymbol{a}_i[\boldsymbol{q}_i(t)] + \boldsymbol{a}_i[\boldsymbol{q}_i(t + \Delta t)]\} \end{pmatrix} \qquad i = 1, ..., N \tag{5.27}$$

which defines the so-called velocity Verlet algorithm, resumed more simply in Table 5.1.

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{v}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 1. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t + \Delta t/2) \leftarrow \boldsymbol{v}_i(t) + \dfrac{\Delta t}{2m_i}\, \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 2. Propagator $e^{i\mathcal{L}_q \Delta t}$	:	$\boldsymbol{q}_i(t + \Delta t) \leftarrow \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t + \Delta t/2)$
Step 3. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t + \Delta t)\}]$
Step 4. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t + \Delta t) \leftarrow \boldsymbol{v}_i(t + \Delta t/2) + \dfrac{\Delta t}{2m_i}\, \mathbf{F}_i[\{\boldsymbol{q}_i(t + \Delta t)\}]$

Table 5.1: Velocity Verlet integrator algorithm, applied for each nuclear phase space coordinate ($i = 1, ..., N$).

First notice that each of the three transformations defining the velocity Verlet algorithm obeys the symplectic condition (4.24) and has a Jacobian determinant equal to one. The product of the three transformation is also symplectic and, thus, phase volume preserving. Finally, since the discrete time propagator (5.19) is unitary, the algorithm is time reversible.
The above analysis demonstrates how the velocity Verlet algorithm can be obtained via the powerful Suzuki-Trotter factorization scheme. The first step is a velocity translation by a half time step $\Delta t/2$,

[1] In the following, in the numerical propagation schemes the approximate symbol in (5.25) will be substituted with an equal sign. Indeed, the treatment is associated to a numerical propagation scheme, which is intrinsically approximated. For this reason, it has to be keep in mind the fact that the equal signs in the numerical propagation schemes are used and referred to a practical implementation, though be approximate models from a theoretical point of view.

and it is sometimes called a *kick*, since it impulsively changes the velocity of each atomic nucleus without altering its positions. The second step is a position translation by a full time step Δt, and it is often called a *drift* step, because it advances the nuclear positions without changing the velocities. The accelerations are then updated, since they depend on the positions modified in the second step. The last step is another translation of the nuclear velocities by a half time step $\Delta t/2$, with the new accelerations computed in the third step. These are just the steps required by the above operator factorization scheme. The power of this method is due to the fact that the instructions in the computer code can be written directly from the factorization scheme, following the order of the propagator with the appropriate time step, bypassing all the lengthy algebra needed to derive an explicit expression for the finite difference equations.

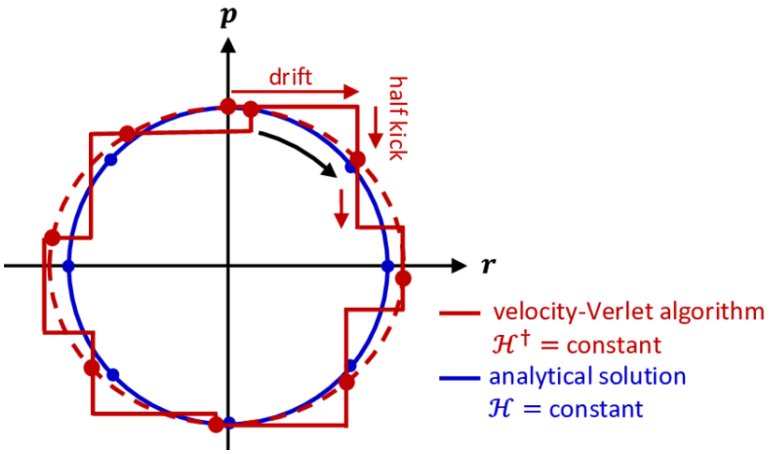

Figure 5.1: Illustration of the phase portrait for the harmonic oscillator. The red line represents the Liouville operator splitting acting between successive red points. In the context of the velocity Verlet algorithm, this implies solving equations in Table 5.1, for a time $(t + \Delta t)$, by the action of (i) a half kick (half integration of the p term at fixed q position), (ii) a drift (integration of the q term at fixed p momentum) and finally (iii) a half kick. Formally, the algorithm presents a long-term conservation of the quantity associated to the pseudo Hamiltonian $\mathcal{H}_p$ (indicated in the figure as $\mathcal{H}^\dagger$). The blue line illustrates the analytical solution which strictly conserves the real Hamiltonian $\mathcal{H}$ of the system. This means that, in the phase portrait, the simulated system remains on an ellipse $\mathcal{H}^\dagger \equiv \mathcal{H}_p$ = constant, which differs from the ellipse $\mathcal{H}$ = constant of the exact solution, with a difference that decreases as the time step Δt used to integrate the equation of motions trough the procedure in Table 5.1 decreases.

The velocity Verlet algorithm has been introduced by W. Swope *et al.*[39] in 1982, as a modification of the so-called position Verlet algorithm[40] (see Appendix A.2). The previously demonstrated procedure of integration in Table 5.1 defining the velocity Verlet algorithm has three main important properties. In particular, the velocity Verlet algorithm is

1. exactly time reversible (symmetric splitting gives exact time reversibility)
2. symplectic (it conserves phase space volume)
3. exactly conserves the pseudo Hamiltonian $\mathcal{H}_p = \mathcal{H} + \mathcal{O}(\Delta t^2)$

All these features lead to a good stability of the integration algorithm, so that the energy related to the pseudo Hamiltonian remains nearly-conserved over a long time.

In order to prove the reversible and symplectic nature of velocity Verlet algorithm, some definitions and notation have to be introduced.
First of all, the discretized equations of motion considered here are the two derived in (5.26) and (5.27), which are here rewritten as

$$\boldsymbol{q}_i(t + \Delta t) = \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\, \boldsymbol{a}_i[\boldsymbol{q}(t)] \qquad i = 1, ..., N \tag{5.28}$$

$$\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\{\boldsymbol{a}_i[\boldsymbol{q}(t)] + \boldsymbol{a}_i[\boldsymbol{q}(t + \Delta t)]\} \tag{5.29}$$

The relation between the acceleration of the nuclei $\boldsymbol{a}_i$ and the forces $\mathbf{F}_i$ $(i = 1, ..., N)$ acting on them permits to write the previous equations (5.28) and (5.29) in the following way:

$$\boldsymbol{q}_i(t + \Delta t) = \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2} \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} \qquad (5.30)$$

$$\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \frac{\Delta t}{2} \left\{ \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t + \Delta t)]}{m_i} \right\} \qquad (5.31)$$

$$i = 1, ..., N$$

In order to simplify the notation further, the dependence on the time step will be substituted with a dependence on an index, so that the time t will be identified with index n and the time $(t + \Delta t)$ with index $(n + 1)$. The time step in the remaining formula will be replaced by $h \equiv \Delta t$, and the dependence on the mass m_i will be included in the forces defined as $\bar{\mathbf{F}}_i = \mathbf{F}_i/m_i$. Applying these changes, equations (5.30) and (5.31) become

$$\boldsymbol{q}_{i,n+1} = \boldsymbol{q}_{i,n} + h\, \boldsymbol{v}_{i,n} + \frac{h^2}{2} \bar{\mathbf{F}}_i(\boldsymbol{q}_n) \qquad (5.32)$$

$$\boldsymbol{v}_{i,n+1} = \boldsymbol{v}_{i,n} + \frac{h}{2} \bar{\mathbf{F}}_i(\boldsymbol{q}_n) + \frac{h}{2} \bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1}) \qquad (5.33)$$

$$i = 1, ..., N$$

Secondly, the following definitions have to be introduced:

Let $\Phi_h : (\boldsymbol{q}_{i,n}, \boldsymbol{v}_{i,n}) \to (\boldsymbol{q}_{i,n+1}, \boldsymbol{v}_{i,n+1})$ be a one-step method for the first order system of differential equations

$$\boldsymbol{v}_i = \dot{\boldsymbol{q}}_i \qquad \dot{\boldsymbol{v}}_i = \mathbf{F}_i(\boldsymbol{q}) \qquad i = 1, ..., N \qquad (5.34)$$

Definition 1. The map Φ_h is called *symmetric* if

$$\Phi_h = \Phi_{-h}^{-1} \qquad (5.35)$$

where Φ_{-h}^{-1} denotes the method $\Phi_{-h}^{-1} : (\boldsymbol{q}_{i,n+1}, \boldsymbol{v}_{i,n+1}) \to (\boldsymbol{q}_{i,n}, \boldsymbol{v}_{i,n})$, with the inversion of the subscripts n and $n + 1$, as a reflection based on $\boldsymbol{q}_{i,n+1/2}$.

Definition 2. The numerical flow Φ_h is called *reversible* if

$$\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i) = (\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{v}}_i) \quad \to \quad \Phi_h(\tilde{\boldsymbol{q}}_i, -\tilde{\boldsymbol{v}}_i) = (\boldsymbol{q}_i, -\boldsymbol{v}_i) \qquad \forall\ \boldsymbol{q}_i, \boldsymbol{v}_i, h \qquad (5.36)$$

Definition 3. The map Φ_h is called *symplectic* if it satisfies

$${}^t J[\Phi_h(\boldsymbol{q}, \boldsymbol{v})] \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0} \end{pmatrix} J[\Phi_h(\boldsymbol{q}, \boldsymbol{v})] = \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0} \end{pmatrix} \qquad \forall\ \boldsymbol{q}, \boldsymbol{v}, h \qquad (5.37)$$

where $J[\Phi_h(\boldsymbol{q}, \boldsymbol{v})]$ denotes the Jacobian of Φ_h, the vectors $\boldsymbol{q} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_N)$ and $\boldsymbol{v} = (\boldsymbol{v}_1, ..., \boldsymbol{v}_N)$ collect all particle positions and velocities and the matrix with non-zero off-diagonal elements is the metric tensor for a Hamiltonian system, where $\mathbb{0}$ and $\mathbb{1}$ are the $N \times N$ zero and identity matrices, respectively. In order to simplify the calculations, the above expression can be written as a function of the coordinates of each atom, so that

$${}^t J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \forall\ \boldsymbol{q}_i, \boldsymbol{v}_i, h \qquad i = 1, ..., N \qquad (5.38)$$

where the metric becomes a simple 2×2 matrix.

Using these definitions, the following theorems can be proved.

Theorem 1. The velocity Verlet algorithm is symmetric.

Proof. The numerical flow Φ_{-h} is given by

$$\Phi_{-h} : (\boldsymbol{q}_{i,n}, \boldsymbol{v}_{i,n}) \to (\boldsymbol{q}_{i,n+1}, \boldsymbol{v}_{i,n+1})$$

where

$$\boldsymbol{q}_{i,n+1} = \boldsymbol{q}_{i,n} - h\,\boldsymbol{v}_{i,n} + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n)$$
$$\boldsymbol{v}_{i,n+1} = \boldsymbol{v}_{i,n} - \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n) - \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1})$$

Inverting the indexes n and $n+1$, the flow

$$\Phi_{-h}^{-1} : (\boldsymbol{q}_{i,n+1}, \boldsymbol{v}_{i,n+1}) \to (\boldsymbol{q}_{i,n}, \boldsymbol{v}_{i,n})$$

can be written explicitly as

$$\boldsymbol{q}_{i,n} = \boldsymbol{q}_{i,n+1} - h\,\boldsymbol{v}_{i,n+1} + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1})$$
$$\boldsymbol{v}_{i,n} = \boldsymbol{v}_{i,n+1} - \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1}) - \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n)$$

and finally,

$$\begin{aligned}\boldsymbol{q}_{i,n+1} &= \boldsymbol{q}_{i,n} + h\,\boldsymbol{v}_{i,n+1} - \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1}) = \boldsymbol{q}_{i,n} + h\left[\boldsymbol{v}_{i,n+1} - \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1})\right] \\ &= \boldsymbol{q}_{i,n} + h\left[\boldsymbol{v}_{i,n} + \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n)\right] = \boldsymbol{q}_{i,n} + h\,\boldsymbol{v}_{i,n} + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n) \\ \boldsymbol{v}_{i,n+1} &= \boldsymbol{v}_{i,n} + \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_{n+1}) + \frac{h}{2}\,\bar{\mathbf{F}}_i(\boldsymbol{q}_n)\end{aligned}$$

From the last expressions it follows that $\Phi_h = \Phi_{-h}^{-1}$ and hence the flow Φ_h generated by the velocity Verlet algorithm has been demonstrated to be symmetric.

Theorem 2. The velocity Verlet algorithm is reversible.

Proof. Recalling the definition of reversibility given through the equation (5.36), the first quantity to compute in order to prove the theorem is

$$\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i) = (\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{v}}_i) = \left(\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}), \quad \boldsymbol{v}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)\right)$$

so that

$$\Phi_h(\tilde{\boldsymbol{q}}_i, -\tilde{\boldsymbol{v}}_i) = \Bigg(\underbrace{\tilde{\boldsymbol{q}}_i - h\,\tilde{\boldsymbol{v}}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\})}_{\alpha}, \quad \underbrace{-\tilde{\boldsymbol{v}}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i\left(\{\tilde{\boldsymbol{q}}_i - h\,\tilde{\boldsymbol{v}}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\})\}\right)}_{\beta}\Bigg)$$

Substituting the expressions for $\tilde{\boldsymbol{q}}_i$ and $\tilde{\boldsymbol{v}}_i$ obtained in the first equation of this proof in the previously defined α and β quantities leads to

$$\begin{aligned}\alpha &= \tilde{\boldsymbol{q}}_i - h\,\tilde{\boldsymbol{v}}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\}) \\ &= \underbrace{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})}_{=\,\tilde{\boldsymbol{q}}_i} - h\,\underbrace{\left[\boldsymbol{v}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\})\right]}_{=\,\tilde{\boldsymbol{v}}_i} + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\}) = \boldsymbol{q}_i\end{aligned}$$

$$\beta = -\tilde{\boldsymbol{v}}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i\Bigg(\underbrace{\{\tilde{\boldsymbol{q}}_i - h\,\tilde{\boldsymbol{v}}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\})\}}_{=\,\alpha\,=\,\boldsymbol{q}_i}\Bigg)$$
$$= -\underbrace{\left[\boldsymbol{v}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\})\right]}_{=\,\tilde{\boldsymbol{v}}_i} + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\tilde{\boldsymbol{q}}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) = -\boldsymbol{v}_i$$

This gives $\Phi_h(\tilde{\boldsymbol{q}}_i, -\tilde{\boldsymbol{v}}_i) = (\boldsymbol{q}_i, -\boldsymbol{v}_i) \quad \forall\, \boldsymbol{q}_i, \boldsymbol{v}_i, h$, therefore the flow Φ_h generated by the velocity Verlet algorithm has been demonstrated to be reversible.

Theorem 3. The velocity Verlet algorithm is symplectic.

Proof. The flow Φ_h can be written in the following way

$$\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i) = (\Phi_1(\boldsymbol{q}_i, \boldsymbol{v}_i), \Phi_2(\boldsymbol{q}_i, \boldsymbol{v}_i))$$
$$\equiv \left(\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}),\ \boldsymbol{v}_i + \frac{h}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)\right)$$

The Jacobian $J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)]$ is the matrix

$$\begin{pmatrix} \nabla_{\boldsymbol{q}_i}\Phi_1(\boldsymbol{q}_i, \boldsymbol{v}_i) & \nabla_{\boldsymbol{v}_i}\Phi_1(\boldsymbol{q}_i, \boldsymbol{v}_i) \\ \nabla_{\boldsymbol{q}_i}\Phi_2(\boldsymbol{q}_i, \boldsymbol{v}_i) & \nabla_{\boldsymbol{v}_i}\Phi_2(\boldsymbol{q}_i, \boldsymbol{v}_i) \end{pmatrix} \equiv \begin{pmatrix} \nabla_{\boldsymbol{q}_i}\Phi_1 & \nabla_{\boldsymbol{v}_i}\Phi_1 \\ \nabla_{\boldsymbol{q}_i}\Phi_2 & \nabla_{\boldsymbol{v}_i}\Phi_2 \end{pmatrix}$$

where in the following it is sometimes dropped for simplicity the dependence of the flow Φ_h on the coordinates $(\boldsymbol{q}_i, \boldsymbol{v}_i)$ in the elements of the Jacobian, as outlined in the above equivalence. The computation of the left hand side of the equation (5.38) leads to

$${}^t J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] = \begin{pmatrix} -\nabla_{\boldsymbol{q}_i}\Phi_2 & \nabla_{\boldsymbol{q}_i}\Phi_1 \\ -\nabla_{\boldsymbol{v}_i}\Phi_2 & \nabla_{\boldsymbol{v}_i}\Phi_1 \end{pmatrix} \begin{pmatrix} \nabla_{\boldsymbol{q}_i}\Phi_1 & \nabla_{\boldsymbol{v}_i}\Phi_1 \\ \nabla_{\boldsymbol{q}_i}\Phi_2 & \nabla_{\boldsymbol{v}_i}\Phi_2 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & \nabla_{\boldsymbol{q}_i}\Phi_1\nabla_{\boldsymbol{v}_i}\Phi_2 - \nabla_{\boldsymbol{v}_i}\Phi_1\nabla_{\boldsymbol{q}_i}\Phi_2 \\ -\nabla_{\boldsymbol{q}_i}\Phi_1\nabla_{\boldsymbol{v}_i}\Phi_2 + \nabla_{\boldsymbol{v}_i}\Phi_1\nabla_{\boldsymbol{q}_i}\Phi_2 & 0 \end{pmatrix}$$

where

$$\nabla_{\boldsymbol{q}_i}\Phi_1 = 1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})$$
$$\nabla_{\boldsymbol{q}_i}\Phi_2 = \frac{h}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)\cdot\left(1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\right)$$
$$\nabla_{\boldsymbol{v}_i}\Phi_1 = h$$
$$\nabla_{\boldsymbol{v}_i}\Phi_2 = 1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)$$

This gives

$$\nabla_{\boldsymbol{q}_i}\Phi_1\nabla_{\boldsymbol{v}_i}\Phi_2 - \nabla_{\boldsymbol{v}_i}\Phi_1\nabla_{\boldsymbol{q}_i}\Phi_2 = \left[1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\right]\left[1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)\right]$$
$$- h\left[\frac{h}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\}) + \frac{h}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i\left(\{\boldsymbol{q}_i + h\,\boldsymbol{v}_i + \frac{h^2}{2}\,\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\}\right)\cdot\left(1 + \frac{h^2}{2}\,\nabla_{\boldsymbol{q}_i}\bar{\mathbf{F}}_i(\{\boldsymbol{q}_i\})\right)\right] = 1$$

so that

$${}^t J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} J[\Phi_h(\boldsymbol{q}_i, \boldsymbol{v}_i)] = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

and the definition (5.38) of a symplectic flow is recovered. Thus, the flow Φ_h generated by the velocity Verlet algorithm has been demonstrated to be symplectic.

5.1.3 Conserved quantities

By definition, the total Hamiltonian (5.3) is a constant of motion for (5.4) - (5.5) (more precisely, the Hamiltonian (5.3) can be defined as an integral of motion, or first integral, since it does not depend on time), so that

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) = C_1 \tag{5.39}$$

Anyway, the fact that the Hamiltonian is a constant of motion can be simply demonstrated by taking the total time derivative of (5.3), as follows

$$\frac{d\mathcal{H}}{dt} = \sum_{i=1}^{N}\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i}\frac{d\boldsymbol{q}_i}{dt} + \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i}\frac{d\boldsymbol{p}_i}{dt}\right) = \sum_{i=1}^{N}\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i}\frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i} - \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i}\frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i}\right) = 0 \tag{5.40}$$

Furthermore, if there are no external forces acting on the system, so that the sum of all interparticle forces is equal to zero, that is

$$\mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.41}$$

then d additional conservation laws must be accounted for when determining the properties of the system, which take the form

$$\mathbf{P} = \mathbf{K} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.42}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions (with d the dimension of the physical space) and $\mathbf{P}$ is the total linear momentum, that is the sum of the linear momentum of all the particles, i.e.

$$\mathbf{P} = \sum_{i=1}^{N} \boldsymbol{p}_i \tag{5.43}$$

Indeed, the total time derivative of the total linear momentum (5.43) is given by

$$\frac{d\mathbf{P}}{dt} = \sum_{i=1}^{N}\frac{d\boldsymbol{p}_i}{dt} = \sum_{i=1}^{N}\left(-\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i}\right) = \sum_{i=1}^{N}\mathbf{F}_i = 0 \qquad \leftrightarrow \qquad \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.44}$$

where the equation of motion (5.5) has been used. The conservation law (5.42) can be written using a single independent variable (note that the d components of the total linear momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.42), so that

$$\|\mathbf{P}\| = P = \|\mathbf{K}\| = K \tag{5.45}$$

Hence, the conservation law in the absence of net forces on the system becomes

$$P = K = C_2 \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.46}$$

Moreover, if the following condition

$$\sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.47}$$

is satisfied by the system, then there are d additional conservation laws satisfied by the Hamilton equations of motion, which take the form

$$\mathbf{L} = \mathbf{K} \qquad \leftrightarrow \qquad \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.48}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions and $\mathbf{L}$ is the total angular momentum of the system defined as the sum of the angular momentum of all the particles, i.e.

$$\mathbf{L} = \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \boldsymbol{p}_i \tag{5.49}$$

Indeed, its total time derivative can be computed as

$$\begin{aligned} \frac{d\mathbf{L}}{dt} &= \sum_{i=1}^{N} \left(\frac{d\boldsymbol{q}_i}{dt} \wedge \boldsymbol{p}_i + \boldsymbol{q}_i \wedge \frac{d\boldsymbol{p}_i}{dt} \right) = \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i}{m_i} \wedge \boldsymbol{p}_i - \boldsymbol{q}_i \wedge \frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \right) \\ &= \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \qquad \leftrightarrow \qquad \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \end{aligned} \tag{5.50}$$

where the equations of motion (5.4) and (5.5) have been used. The conservation law (5.48) can be written using a single independent variable (note that the d components of the total angular momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.48), so that

$$\|\mathbf{L}\| = L = \|\mathbf{K}\| = K \tag{5.51}$$

Hence, the conservation law is given by

$$L = K = C_3 \qquad \leftrightarrow \qquad \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.52}$$

A more general criterion for the existence of these conservation laws is provided by symmetry considerations. If the system is invariant to translation in a particular direction, then the corresponding momentum component is conserved. If the system is invariant to rotation about an axis, then the corresponding angular momentum component is conserved. Thus, the three quantities (5.39), (5.46) and (5.52) are conserved for a completely isolated set of interacting molecules subject to the equations of motion (5.1) - (5.2). In practice, however, a completely isolated system is rarely considered, except for the case of an isolated molecule. Indeed, when systems are enclosed in a spherical box are encounter, then all the three components of total angular momentum about the centre of symmetry will be conserved, but total translational momentum will not be. If the surrounding walls formed a cubical box, none of these quantities would be conserved. This is an important case in practice, since for the practical simulations, periodic boundary conditions are often used, i.e. the system is enclosed in a periodically repeated box. In the case of such periodic boundary conditions, it is easy to see that translational invariance is preserved, and hence total linear momentum is conserved. Several different box geometries can be considered (e.g. cubic, orthorhombic, or truncated octahedron), but none of them can be spherically symmetrical; in fact it is impossible (in Euclidean space) to construct a spherically symmetric periodic system. Hence, total angular momentum is not conserved in most molecular dynamics simulations. As a consequence, if periodic boundary conditions are included in the simulation, then there are only two quantities conserved by the non Hamiltonian flow of the equations of motion (5.1) - (5.2), given by (5.39) and (5.46).

Because the total linear momentum of the system is constant, the equation (5.43) can be integrated over the time variable as

$$\mathbf{G}(t) = \int_{t_0}^{t} \mathbf{P}\, dt' = \int_{t_0}^{t} \sum_{i=1}^{N} \boldsymbol{p}_i\, dt' = \mathbf{P}\,(t - t_0) = \sum_{i=1}^{N} \boldsymbol{p}_i\, t - \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i \tag{5.53}$$

so that the initial value of the quantity is given by

$$\mathbf{G}(0) = -\sum_{i=1}^{N} m_i\, \boldsymbol{q}_i \tag{5.54}$$

The numerical value of $\mathbf{G}$ is therefore associated with the initial position of the center of mass of the system. As demonstrated by J. R. Ray and H. Zhang,[41] the quantity $\mathbf{G}$ is conserved in molecular dynamics simulations when the conservation of total linear momentum holds. This can be proved by a simple time derivative of equation (5.53), that is

$$\begin{aligned}\frac{d\mathbf{G}(t)}{dt} &= \frac{d}{dt}\left(\sum_{i=1}^{N}\boldsymbol{p}_i\,t-\sum_{i=1}^{N}m_i\,\boldsymbol{q}_i\right)=t\cdot\frac{d}{dt}\left(\sum_{i=1}^{N}\boldsymbol{p}_i\right)+\sum_{i=1}^{N}\boldsymbol{p}_i-\frac{d}{dt}\left(\sum_{i=1}^{N}m_i\,\boldsymbol{q}_i\right)\\ &= t\cdot\frac{d\mathbf{P}}{dt}+\sum_{i=1}^{N}\boldsymbol{p}_i-\sum_{i=1}^{N}m_i\,\frac{d\boldsymbol{q}_i}{dt}=t\cdot\frac{d\mathbf{P}}{dt}+\sum_{i=1}^{N}\boldsymbol{p}_i-\sum_{i=1}^{N}m_i\,\frac{\boldsymbol{p}_i}{m_i}\\ &= t\cdot\frac{d\mathbf{P}}{dt}+\sum_{i=1}^{N}\boldsymbol{p}_i-\sum_{i=1}^{N}\boldsymbol{p}_i=t\cdot\frac{d\mathbf{P}}{dt}=0\qquad\leftrightarrow\qquad\sum_{i=1}^{N}\mathbf{F}_i=0\end{aligned}\tag{5.55}$$

where the derivation in (5.44) has been used in the last passage. The constants of the motion represented by $\mathbf{G}$ are associated with Galilean boost (transformation between inertial reference frames that have infinitesimally different velocities) and $\mathbf{G}$ is the generator of infinitesimal boosts like the Hamiltonian $\mathcal{H}$ is the generator of infinitesimal time translations and $\mathbf{P}$, $\mathbf{L}$ are the generators of infinitesimal spatial translations and rotations, respectively. In the following, the constraint that $\mathbf{G}$ is constant will be referred to as the boost constraint. Therefore, due to the additional conserved quantity (5.53), d additional conservation laws must be accounted for when determining the properties of the system, which take the form

$$\mathbf{G}=\tilde{\mathbf{K}}\qquad\leftrightarrow\qquad\mathbf{F}=\sum_{i=1}^{N}\mathbf{F}_i=0\tag{5.56}$$

where $\tilde{\mathbf{K}}$ is an arbitrary constant vector in d dimensions (with d the dimension of the physical space). In order to resume these concepts, in Table 5.2, the constants of motion for the Hamilton equations (5.1) - (5.2) are reported, together with the simulation conditions that are required to preserve their values. Note that the total Hamiltonian (5.39) is always a constant of motion, independently of the simulation conditions.

conservation laws of Hamilton equations (5.1) - (5.2)				
$\mathcal{H}(\boldsymbol{p},\boldsymbol{q})=\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}+\phi(\boldsymbol{q})=C_1$				
$P=K=C_2$	iff	$\mathbf{F}=\sum_{i=1}^{N}\mathbf{F}_i=0$		
$L=K=C_3$	iff	$\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i=0$	and	NO PBC

Table 5.2: Constants of motion for the Hamilton equations of motion (5.1) - (5.2), together with the simulation conditions that are necessary to preserve these quantities.

5.1.4 Jacobi coordinates

It is convenient to introduce a set of coordinates and momenta, referred to as Jacobi coordinates and momenta, for the later discussion of phase space integrals. These coordinates are defined so the first $(N-1)$ Jacobi coordinates are internal coordinates, while the N-th Jacobi coordinate is the center of mass coordinate of the system of particles. For a system of N particles, the Jacobi coordinates and

momenta are defined by

$$\boldsymbol{\rho}_\alpha = \left[\sum_{i=1}^{\alpha} m_i\right]^{-1} \sum_{i=1}^{\alpha} m_i\,\boldsymbol{q}_i - \boldsymbol{q}_{\alpha+1} \qquad \alpha = 1,...,N \qquad \text{with} \quad \boldsymbol{q}_{N+1} = \boldsymbol{0} \tag{5.57}$$

$$\boldsymbol{\pi}_\alpha = \frac{m_{\alpha+1}}{M_1^{\alpha+1}} \sum_{i=1}^{\alpha} \boldsymbol{p}_i - \frac{M_1^\alpha}{M_1^{\alpha+1}}\,\boldsymbol{p}_{\alpha+1} \qquad \alpha = 1,...,N-1 \qquad \text{and} \quad \boldsymbol{\pi}_N = \sum_{i=1}^{N} \boldsymbol{p}_i \tag{5.58}$$

where

$$M_i^j = \sum_{k=i}^{j} m_k \qquad i \le j \quad \text{and} \quad i,j,k = 1,...,N \tag{5.59}$$

Note that the Jacobi coordinates give the position of the center of mass of the previous particles with respect to the next particle. The Jacobi coordinates are orthogonal and are a generalization of the usual coordinates for the center of mass and relative coordinates of a two particle system. The Jacobi coordinates are canonical, so that the Jacobian (i.e. the determinant of the transformation matrix) from the original Cartesian coordinates and momenta to the Jacobi coordinates and momenta is equal to one. Also the Jacobians for the transformation from the Cartesian coordinates alone or the momenta alone are equal to one.
The kinetic energy of the system in Jacobi coordinates is

$$T = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} = \sum_{\alpha=1}^{N-1} \frac{\boldsymbol{\pi}_\alpha^2}{2\mu_\alpha} + \frac{\boldsymbol{\pi}_N^2}{2\mu_N} \tag{5.60}$$

where the reduced masses μ_α are defined by

$$\mu_\alpha = m_{\alpha+1}\,\frac{M_1^\alpha}{M_1^{\alpha+1}} \qquad \alpha = 1,...,N-1 \qquad \text{and} \qquad \mu_N = M_1^N \tag{5.61}$$

Equation (5.60) gives the kinetic energy in terms of the internal kinetic energy plus the kinetic energy of the center of mass and is a well known result in classical mechanics. The reduced masses satisfy the following product rule

$$\prod_{\alpha=1}^{N-1} \mu_\alpha = \frac{1}{\mu_N} \prod_{i=1}^{N} m_i \tag{5.62}$$

where $\mu_N = M_1^N$ is the total mass of the system.[2]

[2]For example, having $N = 3$ particles, the Jacobi coordinates and the reduced masses are

$$\boldsymbol{\rho}_1 = \frac{1}{m_1} m_1\,\boldsymbol{q}_1 - \boldsymbol{q}_2 = \boldsymbol{q}_1 - \boldsymbol{q}_2$$
$$\boldsymbol{\rho}_2 = \frac{1}{m_1+m_2}(m_1\,\boldsymbol{q}_1 + m_2\,\boldsymbol{q}_2) - \boldsymbol{q}_3$$
$$\boldsymbol{\rho}_3 = \frac{1}{m_1+m_2+m_3}(m_1\,\boldsymbol{q}_1 + m_2\,\boldsymbol{q}_2 + m_3\,\boldsymbol{q}_3) - \boldsymbol{q}_4 \overset{\boldsymbol{q}_4=\boldsymbol{0}}{=} \frac{1}{m_1+m_2+m_3}(m_1\,\boldsymbol{q}_1 + m_2\,\boldsymbol{q}_2 + m_3\,\boldsymbol{q}_3)$$

$$\boldsymbol{\pi}_1 = \frac{m_2}{m_1+m_2}\,\boldsymbol{p}_i - \frac{m_1}{m_1+m_2}\,\boldsymbol{p}_2$$
$$\boldsymbol{\pi}_2 = \frac{m_3}{m_1+m_2+m_3}\,\boldsymbol{p}_i - \frac{m_1+m_2}{m_1+m_2+m_3}\,\boldsymbol{p}_3$$
$$\boldsymbol{\pi}_3 = \sum_{i=1}^{3} \boldsymbol{p}_i = \boldsymbol{p}_1 + \boldsymbol{p}_2 + \boldsymbol{p}_3$$

$$\mu_1 = \frac{m_2\,m_1}{m_1+m_2} \qquad \mu_2 = \frac{m_3\,(m_2+m_1)}{m_1+m_2+m_3} \qquad \mu_3 = M_1^3 = m_1+m_2+m_3$$

5.1.5 Statistical mechanical ensemble

The statistical ensemble sampled by the microcanonical equations of motion depends on the form of the laws of motion, together with the associated conserved quantities. In the following, the statistical ensemble sampled by the Hamilton equations of motion (5.1)-(5.2) will be derived, taking into account the presence and the absence of periodic boundary conditions. As a general statement, it is worth mentioning that the microcanonical ensemble, ruled by an Hamiltonian dynamic, posses a fundamental property known as the condition of phase space incompressibility. This can be easily demonstrated by computing the compressibility according to its definition (4.180) as

$$\kappa(\mathbf{\Gamma}) = \nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} = \sum_{i=1}^{N} \left[\frac{\partial \dot{\boldsymbol{q}}_i}{\partial \boldsymbol{q}_i} + \frac{\partial \dot{\boldsymbol{p}}_i}{\partial \boldsymbol{p}_i} \right] = \sum_{i=1}^{N} \left[\frac{\partial}{\partial \boldsymbol{q}_i} \cdot \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} - \frac{\partial}{\partial \boldsymbol{p}_i} \cdot \frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} \right] = \sum_{i=1}^{N} \left[\frac{\partial^2 \mathcal{H}}{\partial \boldsymbol{q}_i \partial \boldsymbol{p}_i} - \frac{\partial^2 \mathcal{H}}{\partial \boldsymbol{p}_i \partial \boldsymbol{q}_i} \right] = 0 \quad (5.63)$$

so that

$$\text{Hamilton equations of motion :} \qquad \kappa(\mathbf{\Gamma}) = 0 \quad \rightarrow \quad \sqrt{g(\mathbf{\Gamma})} = 1 \tag{5.64}$$

Then, the time evolution of the infinitesimal volume element $dv_{\mathbf{\Gamma}}$, ruled by equation (4.173), given in this case by

$$dv_{\mathbf{\Gamma}} = d\mathbf{\Gamma}(t) = d\mathbf{\Gamma}(t_0) \tag{5.65}$$

can be associated to a canonical coordinates transformation.
The partition function (4.185) can be finally written, for the case of Hamilton equations of motion (microcanonical ensemble), as

$$Z = \zeta \int d\mathbf{\Gamma} \prod_{k=1}^{n_c} \delta(\Lambda_k(\mathbf{\Gamma}) - C_k) \tag{5.66}$$

where

$$\Lambda_k(\mathbf{\Gamma}) = C_k \qquad \text{for} \quad k = 1, ..., n_c \tag{5.67}$$

are the set of n_c conservation laws associated to the Hamilton equations of motion and $\Lambda_k(\mathbf{\Gamma})$ are the related conserved quantities which satisfy the condition (4.183).

5.1.5.1 Periodic boundary conditions

As demonstrated in Section 5.1.3, if periodic boundary conditions are used, then three quantities are conserved by the Hamilton equations of motion (5.1) - (5.2), that are given by (5.39), (5.42) and (5.56), reported in the following for completeness

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) = C_1 \equiv E \tag{5.68}$$

$$\mathbf{P} = \sum_{i=1}^{N} \boldsymbol{p}_i = \mathbf{K} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.69}$$

$$\mathbf{G} = \sum_{i=1}^{N} \boldsymbol{p}_i \, t - \sum_{i=1}^{N} m_i \, \boldsymbol{q}_i = t \, \mathbf{P} - \sum_{i=1}^{N} m_i \, \boldsymbol{q}_i = \tilde{\mathbf{K}} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.70}$$

As a consequence, the partition function for microcanonical molecular dynamics when using periodic boundary conditions can be written by means of the equations (5.66) and (5.67) as

$$Z = \zeta \int d\boldsymbol{p} \int d\boldsymbol{q} \; \delta(\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) - E) \, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i \right) \delta\left(\mathbf{G} - t \sum_{i=1}^{N} \boldsymbol{p}_i + \sum_{i=1}^{N} m_i \, \boldsymbol{q}_i \right) \tag{5.71}$$

where $d\boldsymbol{\Gamma} = d\boldsymbol{p}\, d\boldsymbol{q} = d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N\, d\boldsymbol{q}_1 \cdots d\boldsymbol{q}_N$. The boost constraint is straightforwardly handled via the introduction of Jacobi coordinates for the position vectors (see Section 5.1.4), so that the third term in the boost constraint delta function can be written using the definition (5.57) as

$$\boldsymbol{\rho}_N = \left[\sum_{i=1}^{N} m_i\right]^{-1} \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i - \boldsymbol{q}_{N+1} = \left[\sum_{i=1}^{N} m_i\right]^{-1} \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i \quad \rightarrow \quad \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i = \boldsymbol{\rho}_N \sum_{i=1}^{N} m_i = \mu_N \boldsymbol{\rho}_N$$

where μ_N is the total mass of the system. Moreover, the Jacobian of the transformation between Jacobi and Cartesian position coordinates and vice-versa is unitary, so that

$$d\boldsymbol{q}_1 \cdots d\boldsymbol{q}_N = d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_N \quad \rightarrow \quad d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N\, d\boldsymbol{q}_1 \cdots d\boldsymbol{q}_N = d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N\, d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_N \tag{5.72}$$

Therefore, the partition function (5.71) can be rewritten as

$$\begin{aligned} &Z(N, V, E, \mathbf{P}, \mathbf{G}) \\ &= \zeta \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_N\; \delta(\mathcal{H}(\boldsymbol{p}, \boldsymbol{\rho}) - E)\, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \delta\left(\mathbf{G} - t \sum_{i=1}^{N} \boldsymbol{p}_i + \mu_N \boldsymbol{\rho}_N\right) \end{aligned} \tag{5.73}$$

where $\boldsymbol{\rho}_N$ is the N-th Jacobi coordinate, that is center of mass coordinate of the system of particles, and the Hamiltonian $\mathcal{H}(\boldsymbol{p}, \boldsymbol{\rho})$ is now expressed as a function of the Jacobi position coordinates. For a system in which the potential $\phi(\boldsymbol{q}) \rightarrow \phi(\boldsymbol{\rho})$ is only a function of the relative distances between particles (as is considered here), the Hamiltonian is not a function of the center of mass of the particles and so it is independent of one of the Jacobi coordinates. Consequently, the delta function containing the boost constraint can be immediately integrated over by carrying out the integral over the center of mass coordinate $\boldsymbol{\rho}_N$ to yield

$$Z(N, V, E, \mathbf{P}, \mathbf{G}) = \zeta \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\; \delta(\mathcal{H}(\boldsymbol{p}, \boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_{N-1}) - E)\, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \tag{5.74}$$

where $\boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_{N-1}$ are the remaining Jacobi coordinates for the nuclear positions. This step has the effect of reducing the number of spatial integrations by three because the integral over the boost delta function does not change any other parts of the integrand; recall that it has been assumed that the potential energy depends only on the relative coordinates and not the center of mass coordinate. Next, a Laplace transform of the energy ($E \rightarrow t$) is performed on the partition function. In general, the lower bound on the resulting integral over E should be set equal to the appropriate minimum energy E_{min} (corresponding to zero kinetic energy and all particle pairs residing in their lowest energy state), which may be different from zero. A simple coordinate transformation ($s = E - E_{min}$), however, converts the lower bound of the Laplace transform back to zero. In the present case, the energy is represented by the constant E, as written in (5.68). Therefore, the Laplace transform $E \rightarrow t$ or more precisely $s \rightarrow t$ leads to

$$\begin{aligned} \mathfrak{L}[Z(N, V, E, \mathbf{P}, \mathbf{G})] = Z(N, V, t, \mathbf{P}, \mathbf{G}) &= \int_0^\infty e^{-t\,E}\, Z(N, V, E, \mathbf{P}, \mathbf{G})\, dE \\ &= \zeta \int_0^\infty dE \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\; e^{-t\,E}\; \delta(\mathcal{H}(\boldsymbol{p}, \boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_{N-1}) - E)\, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \\ &= \zeta \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\; \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) e^{-t\,\mathcal{H}(\boldsymbol{p}, \boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_{N-1})} \end{aligned}$$

The Hamiltonian $\mathcal{H}$ in the exponential factor is given by equation (5.68), so that the Laplace transform of the partition function can be written explicitly as

$$\begin{aligned} \mathfrak{L}[Z(N, V, E, \mathbf{P}, \mathbf{G})] &= \zeta \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\; \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \exp\left[-t \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{\rho})\right)\right] \\ &= \zeta \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N\, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \exp\left[-t \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right] \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\, \exp[-t\,\phi(\boldsymbol{\rho})] \end{aligned} \tag{5.75}$$

In the following, the integral over the Jacobi position coordinates will be sometimes abbreviated, using the notation

$$I(t, \{\boldsymbol{\rho}\}) \equiv \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1} \, \exp[-t \, \phi(\boldsymbol{\rho})] \tag{5.76}$$

In order to perform the integration over the momenta variables, a Fourier transform of the total momentum ($\mathbf{P} \to \boldsymbol{k}$) can be done,

$$\begin{aligned} \mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]] &= \mathcal{F}[Z(N,V,t,\mathbf{P},\mathbf{G})] \\ &= Z(N,V,t,\boldsymbol{k},\mathbf{G}) = \frac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} d\mathbf{P} \; \exp[-\imath \, \boldsymbol{k} \cdot \mathbf{P}] \, Z(N,V,t,\mathbf{P},\mathbf{G}) \end{aligned} \tag{5.77}$$

where $\imath$ is the imaginary unit. The Fourier transform of equation (5.75) will affect only the integral over the momenta coordinates, so that it can be written as

$$\begin{aligned} Z(N,V,t,\boldsymbol{k},\mathbf{G}) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]] \\ &= \frac{\zeta}{(2\pi)^{d/2}} \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \int_{-\infty}^{\infty} d\mathbf{P} \, \exp[-\imath \, \boldsymbol{k} \cdot \mathbf{P}] \, \delta\left(\mathbf{P} - \sum_{i=1}^{N} \boldsymbol{p}_i\right) \exp\left[-t \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right] I(t, \{\boldsymbol{\rho}\}) \end{aligned}$$

The integral over the variable $\mathbf{P}$ can be solved through the Dirac delta function, resulting in the following expression

$$\begin{aligned} \mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]] &= \frac{\zeta}{(2\pi)^{d/2}} \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \, \exp\left[-\imath \, \boldsymbol{k} \cdot \sum_{i=1}^{N} \boldsymbol{p}_i\right] \exp\left[-t \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right] I(t, \{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{(2\pi)^{d/2}} \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \, \exp\left[-\frac{t}{2} \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i^2}{m_i} + 2 \, \frac{\imath \, \boldsymbol{k} \cdot \boldsymbol{p}_i}{t}\right)\right] I(t, \{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{(2\pi)^{d/2}} \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \, \exp\left[-\frac{t}{2} \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i}{\sqrt{m_i}} + \frac{\imath \, \boldsymbol{k}}{t} \sqrt{m_i}\right)^2\right] \exp\left[-\frac{\boldsymbol{k}^2}{2t} \sum_{i=1}^{N} m_i\right] I(t, \{\boldsymbol{\rho}\}) \end{aligned}$$

so that

$$\begin{aligned} \mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]] &= Z(N,V,t,\boldsymbol{k},\mathbf{G}) \\ &= \frac{\zeta}{(2\pi)^{d/2}} \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t}\right] \int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \, \exp\left[-\frac{t}{2} \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i}{\sqrt{m_i}} + \frac{\imath \, \boldsymbol{k}}{t} \sqrt{m_i}\right)^2\right] I(t, \{\boldsymbol{\rho}\}) \end{aligned} \tag{5.78}$$

The integral that involves the momenta variables is a Gaussian integral, which can be resolved as follows

$$\begin{aligned} &\int d\boldsymbol{p}_1 \cdots d\boldsymbol{p}_N \, \exp\left[-\frac{t}{2} \sum_{i=1}^{N} \left(\frac{\boldsymbol{p}_i}{\sqrt{m_i}} + \frac{\imath \, \boldsymbol{k}}{t} \sqrt{m_i}\right)^2\right] \\ &= \int d\boldsymbol{p}_1 \, \exp\left[-\frac{t}{2} \left(\frac{\boldsymbol{p}_1}{\sqrt{m_1}} + \frac{\imath \, \boldsymbol{k}}{t} \sqrt{m_1}\right)^2\right] \cdots \int d\boldsymbol{p}_N \, \exp\left[-\frac{t}{2} \left(\frac{\boldsymbol{p}_N}{\sqrt{m_N}} + \frac{\imath \, \boldsymbol{k}}{t} \sqrt{m_N}\right)^2\right] \\ &= \left(\frac{2\pi \, m_1}{t}\right)^{d/2} \cdots \left(\frac{2\pi \, m_N}{t}\right)^{d/2} = \left(\frac{2\pi}{t}\right)^{Nd/2} \prod_{i=1}^{N} m_i^{d/2} \end{aligned} \tag{5.79}$$

Indeed, each one of the Gaussian integral in the second line of the previous expression has the solution

$$\begin{aligned}
&\int d\boldsymbol{p}_i \, \exp\left[-\frac{t}{2}\left(\frac{\boldsymbol{p}_i}{\sqrt{m_i}} + \frac{\imath\,\boldsymbol{k}}{t}\sqrt{m_i}\right)^2\right] \\
&= \int dp_{i,j} \, \exp\left[-\frac{t}{2}\left(\frac{p_{i,j}}{\sqrt{m_i}} + \frac{\imath\,k_j}{t}\sqrt{m_i}\right)^2\right] \cdots\cdots \int dp_{i,d} \, \exp\left[-\frac{t}{2}\left(\frac{p_{i,d}}{\sqrt{m_i}} + \frac{\imath\,k_d}{t}\sqrt{m_i}\right)^2\right] \\
&= \int dp_{i,j} \, \exp\left[-\frac{t}{2m_i}\left(p_{i,j} + \frac{\imath\,k_j}{t}\,m_i\right)^2\right] \cdots\cdots \int dp_{i,d} \, \exp\left[-\frac{t}{2m_i}\left(p_{i,d} + \frac{\imath\,k_d}{t}\,m_i\right)^2\right] \\
&= \underbrace{\sqrt{\frac{2\pi\,m_i}{t}} \cdots\cdots \sqrt{\frac{2\pi\,m_i}{t}}}_{d \text{ times}} = \left(\frac{2\pi\,m_i}{t}\right)^{d/2}
\end{aligned} \tag{5.80}$$

with $\boldsymbol{p}_i$ a d-dimensional vector, so that d identifies the dimension of the physical space of the system. Therefore, substituting the result (5.79) in (5.1.5.1), the Fourier transform of the Laplace transform of the partition function can be written as

$$\begin{aligned}
Z(N,V,t,\boldsymbol{k},\mathbf{G}) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]] \\
&= \frac{\zeta}{(2\pi)^{d/2}} \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t}\right]\left(\frac{2\pi}{t}\right)^{Nd/2} \left(\prod_{i=1}^{N} m_i^{d/2}\right) I(t,\{\boldsymbol{\rho}\}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right) \exp\left[-\frac{m\boldsymbol{k}^2}{2t}\right] t^{-Nd/2} \, I(t,\{\boldsymbol{\rho}\})
\end{aligned} \tag{5.81}$$

In order to found the partition function, it is necessary to perform an inverse Fourier transform ($\boldsymbol{k} \to \mathbf{P}$) of (5.81), by applying a Fourier anti-transform, that is the inverse operation of (5.1.5.1), namely

$$\begin{aligned}
\mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]]\} &= \mathcal{F}^{-1}\{\mathcal{F}[Z(N,V,t,\mathbf{P},\mathbf{G})]\} = \mathcal{F}^{-1}\{Z(N,V,t,\boldsymbol{k},\mathbf{G})\} \\
&= Z(N,V,t,\mathbf{P},\mathbf{G}) = \frac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp[\imath\,\boldsymbol{k}\cdot\mathbf{P}]\, Z(N,V,t,\boldsymbol{k},\mathbf{G})
\end{aligned} \tag{5.82}$$

Applying the Fourier anti-transform to (5.81), following the definition (5.1.5.1), leads to

$$\begin{aligned}
&\mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]]\} = \mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})] = Z(N,V,t,\mathbf{P},\mathbf{G}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right) \frac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp[\imath\boldsymbol{k}\cdot\mathbf{P}] \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t}\right] t^{-Nd/2} \, I(t,\{\boldsymbol{\rho}\})
\end{aligned} \tag{5.83}$$

$$\begin{aligned}
&\mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]]\} = \mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})] = Z(N,V,t,\mathbf{P},\mathbf{G}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right) \frac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp[\imath\boldsymbol{k}\cdot\mathbf{P}] \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t}\right] t^{-Nd/2} \, I(t,\{\boldsymbol{\rho}\}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right) \frac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t} + \imath\boldsymbol{k}\cdot\mathbf{P}\right] t^{-Nd/2} \, I(t,\{\boldsymbol{\rho}\})
\end{aligned} \tag{5.84}$$

The integral in the variable $\boldsymbol{k}$ can be resolved as follows

$$\begin{aligned}
&\int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp\left[-\frac{m\boldsymbol{k}^2}{2t} + \imath\boldsymbol{k}\cdot\mathbf{P}\right] = \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp\left[\left(\frac{\imath\sqrt{m}\,\boldsymbol{k}}{\sqrt{2t}} + \frac{\mathbf{P}}{\sqrt{m}}\sqrt{\frac{t}{2}}\right)^2\right] \exp\left(-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right) \\
&= \exp\left(-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right) \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp\left[-\frac{m}{2t}\left(\boldsymbol{k} + \frac{\mathbf{P}}{\sqrt{m}}\sqrt{\frac{t}{2}}\,\frac{\sqrt{2t}}{\imath\sqrt{m}}\right)^2\right] \\
&= \exp\left(-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right) \int_{-\infty}^{\infty} d\boldsymbol{k} \, \exp\left[-\frac{m}{2t}\left(\boldsymbol{k} + \frac{t\,\mathbf{P}}{\imath\,m}\right)^2\right] = \exp\left(-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right)\left(\sqrt{\frac{2\pi\,t}{m}}\right)^d
\end{aligned} \tag{5.85}$$

Substituting the obtained result (5.85) in (5.84), the Laplace transform of the partition function can be written as

$$\begin{aligned}
\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})] &= Z(N,V,t,\mathbf{P},\mathbf{G}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right)\frac{1}{(2\pi)^{d/2}}\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right]\left(\sqrt{\frac{2\pi\,t}{m}}\right)^{d} t^{-Nd/2}\ I(t,\{\boldsymbol{\rho}\}) \\
&= \zeta(2\pi)^{(N-1)d/2}\left(\prod_{i=1}^{N} m_i^{d/2}\right)\frac{1}{(2\pi)^{d/2}}\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right](2\pi)^{d/2}\,\frac{t^{d/2}}{m^{d/2}}\ t^{-Nd/2}\ I(t,\{\boldsymbol{\rho}\}) \\
&= \frac{\zeta(2\pi)^{(N-1)d/2}}{m^{d/2}}\left(\prod_{i=1}^{N} m_i^{d/2}\right)\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right]\ t^{-(N-1)d/2}\ I(t,\{\boldsymbol{\rho}\})
\end{aligned} \tag{5.86}$$

Recalling the definition of $I(t,\{\boldsymbol{\rho}\})$ given in (5.76), the Laplace transform of the partition function can be rewritten as

$$\begin{aligned}
\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})] &= Z(N,V,t,\mathbf{P},\mathbf{G}) \\
&= \frac{\zeta(2\pi)^{(N-1)d/2}}{m^{d/2}}\left(\prod_{i=1}^{N} m_i^{d/2}\right)\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right]\ t^{-(N-1)d/2}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ \exp[-t\,\phi(\boldsymbol{\rho})] \\
&= \Omega\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right]\ t^{-(N-1)d/2}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ \exp[-t\,\phi(\boldsymbol{\rho})]
\end{aligned} \tag{5.87}$$

where Ω includes all the constant factors in (5.87) and it is thus defined by

$$\Omega \equiv \frac{\zeta(2\pi)^{(N-1)d/2}}{m^{d/2}}\left(\prod_{i=1}^{N} m_i^{d/2}\right) \tag{5.88}$$

At this point, the last step is to compute the Laplace anti-transform of the equation (5.87), finally obtaining the partition function

$$\begin{aligned}
\mathfrak{L}^{-1}\{\mathfrak{L}[Z(N,V,E,\mathbf{P},\mathbf{G})]\} &= \mathfrak{L}^{-1}\{Z(N,V,t,\mathbf{P},\mathbf{G})\} = Z(N,V,E,\mathbf{P},\mathbf{G}) \\
&= \frac{1}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} e^{tE}\,Z(N,V,t,\mathbf{P},\mathbf{G})\,dt
\end{aligned} \tag{5.89}$$

Applying the definition (5.89) to the Laplace transform of the partition function (5.87), the following expression can be obtained for the partition function

$$\begin{aligned}
Z(N,V,E,\mathbf{P},\mathbf{G}) &= \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ \exp[tE]\ \exp\left[-\frac{\mathbf{P}^2}{m}\,\frac{t}{2}\right]\ t^{-(N-1)d/2}\ \exp[-t\,\phi(\boldsymbol{\rho})] \\
&= \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{\mathbf{P}^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\
&= \frac{\Omega}{2\pi\,\imath}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} dt\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{\mathbf{P}^2}{2m}-\phi(\boldsymbol{\rho})\right)\right]
\end{aligned} \tag{5.90}$$

5.1.5.2 Periodic boundary conditions: a second derivation

If periodic boundary conditions are used, then three quantities are conserved by the Hamilton equations of motion (5.1) - (5.2), that are given by (5.39), (5.42) and (5.56), reported in the following for completeness

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) = C_1 \equiv E \tag{5.91}$$

$$\mathbf{P} = \sum_{i=1}^{N} \boldsymbol{p}_i = \mathbf{K} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.92}$$

$$\mathbf{G} = \sum_{i=1}^{N} \boldsymbol{p}_i\, t - \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i = t\, \mathbf{P} - \sum_{i=1}^{N} m_i\, \boldsymbol{q}_i = \tilde{\mathbf{K}} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0 \tag{5.93}$$

where the last two quantities are conserved if and only if the total force acting on the system is equal to zero. Related to the initial position of the center of mass, the quantity $\mathbf{G}$ is the generator of infinitesimal Galilean boosts (transformations between inertial reference frames that have infinitesimally different velocities). However, as specified by Ref. [42], the quantity $\mathbf{G}$ becomes a constant of motion because the center of mass of the system is a driven variable (i.e. its dynamics does not effect other variables and it does not contribute to a nontrivial conserved quantity). Therefore, the $\mathbf{G}$ constraint accounts for the center of mass of the system being a driven variable,[42] and can be ignored if the center of mass is discarded as a driven variable of the system, following the procedure outlined in Section 4.3.2 (in particular, see Section 4.3.2.1). In order to follow the analysis following Section 4.3.2.1, the driven variables have to be eliminated from the system. As a consequence, the center of mass position $\mathbf{R}$ must be eliminated in the formal analysis. On the other hand, the magnitude of the center of mass momentum is coupled to the other variables through a conservation law and cannot be eliminated from the analysis. At the same time, the components of the center of mass momentum $\mathbf{P}$ are linearly dependent.[3] Thus, $d-1$ components of the center of mass momentum must be eliminated. Therefore of the d components only one component can be chosen independently, otherwise the variable

$$P = \|\mathbf{P}\| = \left(\sum_{\alpha=1}^{d} P_\alpha^2 \right)^{1/2} \tag{5.95}$$

can be taken as the independent variable. Before proceeding further on, the equation of motion for this new variable can be easily found by taking into account the definition of the conserved quantity, equation (5.92), and rewriting it in terms of the new variable P defined in (5.95), by taking the Euclidean norm on both sides of equation (5.92), so that

$$\|\mathbf{P}\| = P = \|\mathbf{K}\| = K \qquad \rightarrow \qquad \dot{P} = \dot{K} = 0 \tag{5.96}$$

Proceeding with the analysis, the two variables $\mathbf{R}$ and $\mathbf{P}$ can be eliminated by considering the positions and momenta relative to the center of mass of the system, $\boldsymbol{\rho}$ and $\boldsymbol{\pi}$, respectively. Thus, a canonical transformation to a set of relative coordinates and momenta $\{\boldsymbol{p}, \boldsymbol{q}\} \rightarrow \{\boldsymbol{\pi}, P, \boldsymbol{\rho}\}$ has to be introduced, using the Jacobi coordinates as outlined in Section 5.1.4, where the equations of motion for the $\mathbf{P}$ will be written in terms of the single independent variable P. Starting from the equation of motion (5.1) - (5.2), this procedure yields the following transformed equations of motion

$$\dot{\boldsymbol{\rho}}_i = \frac{d\boldsymbol{\rho}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\pi}_i} = \frac{\boldsymbol{\pi}_i}{\mu_i} \qquad i = 1, ..., N-1 \tag{5.97}$$

$$\dot{\boldsymbol{\pi}}_i = \frac{d\boldsymbol{\pi}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{\rho}_i} = -\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} \qquad i = 1, ..., N-1 \tag{5.98}$$

$$\dot{P} = 0 \tag{5.99}$$

where μ_i is the reduced mass of the system defined by equation (5.61) and the equation of motion for the momentum $\mathbf{P}$, written in terms of the single independent variable P as defined in (5.95), has been introduced in equation (5.99), following the result obtained in (5.96). The equations of motion have two conservative laws (because the d center of mass momenta components have been replaced by a single variable P only one conservation law for the momenta is left), namely,

$$\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}) = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} + \frac{P^2}{2m} + \phi(\boldsymbol{\rho}) = C_1 \equiv E \tag{5.100}$$

[3] To see this, consider the components of equation (5.298) in a three dimensional system ($d = 3$),

$$\frac{P_x}{K_x} = \frac{P_y}{K_y} = \frac{P_z}{K_z} = e^{\eta} \tag{5.94}$$

which shows that only one of the components is independent.

$$P = C_2 \tag{5.101}$$

where, in the first conservation law (5.100), the variable μ_i is the reduced mass of the system defined by equation (5.61), and $m \equiv \mu_N$ is the total mass of the system, see again equation (5.61). In order to compute the partition function, the compressibility has to be calculated, as already performed in Section 5.1.5, but using, in this case, the phase space vector $\boldsymbol{\Xi} = (\boldsymbol{\pi}, P, \boldsymbol{\rho})$ together with the definition (4.123), as

$$\kappa(\boldsymbol{\Xi}, t) \equiv \nabla_{\boldsymbol{\Xi}} \cdot \dot{\boldsymbol{\Xi}} = \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\rho}_i} \cdot \dot{\boldsymbol{\rho}}_i + \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} = 0 \tag{5.102}$$

From the compressibility, the metric follows directly

$$\sqrt{g(\boldsymbol{\Xi}, t)} = \exp\left(- \int \kappa(\boldsymbol{\Xi}, t)\, dt \right) = 1 \tag{5.103}$$

The conservation laws (5.100), (5.101) and the metric (5.103) can now be used to construct the microcanonical partition function, as in equation (5.66) (see Section 5.1.5), which contains two delta functions that express the two conservation laws

$$\begin{aligned} Z(N, V, E, C_2) &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - C_1)\ \delta(P - C_2) \\ &\equiv \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - E)\ \delta(P - C_2) \end{aligned} \tag{5.104}$$

where $d\boldsymbol{\pi} = d\boldsymbol{\pi}_1 \cdots d\boldsymbol{\pi}_{N-1}$ and $d\boldsymbol{\rho} = d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}$ are the remaining Jacobi coordinates for the nuclear positions and momenta. Next, a Laplace transform of the energy ($E \to t$) is performed on the partition function. In general, the lower bound on the resulting integral over E should be set equal to the appropriate minimum energy E_{min} (corresponding to zero kinetic energy and all particle pairs residing in their lowest energy state), which may be different from zero. A simple coordinate transformation ($s = E - E_{min}$), however, converts the lower bound of the Laplace transform back to zero. In the present case, the energy is represented by the constant E, as written in (5.68). Therefore, the Laplace transform $E \to t$ or more precisely $s \to t$ leads to

$$\begin{aligned} \mathfrak{L}[Z(N, V, E, C_2)] = Z(N, V, t, C_2) &= \int_0^\infty e^{-t\,E}\, Z(N, V, E, C_2)\, dE \\ &= \zeta \int_0^\infty dE \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ e^{-tE}\ \delta(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - E)\ \delta(P - C_2) \\ &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(P - C_2)\ e^{-t\,\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho})} \end{aligned} \tag{5.105}$$

The Hamiltonian $\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho})$ in the exponential factor is given by equation (5.100), so that the Laplace transform of the partition function can be written explicitly as

$$\begin{aligned} \mathfrak{L}[Z(N, V, E, C_2)] &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(P - C_2)\ \exp\left[-t \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} + \frac{P^2}{2m} + \phi(\boldsymbol{\rho}) \right) \right] \\ &= \zeta \int d\boldsymbol{\pi}\ \exp\left[-t \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} \right) \right] \int dP\ \delta(P - C_2) \exp\left(-t\ \frac{P^2}{2m} \right) \int d\boldsymbol{\rho}\ \exp\left[-t\, \phi(\boldsymbol{\rho}) \right] \end{aligned} \tag{5.106}$$

In the following, the integral over the Jacobi position coordinates will be sometimes abbreviated, using the notation

$$I(t, \{\boldsymbol{\rho}\}) \equiv \int d\boldsymbol{\rho}\ \exp\left[-t\, \phi(\boldsymbol{\rho}) \right] \equiv \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\ \exp[-t\, \phi(\boldsymbol{\rho})] \tag{5.107}$$

so that equation (5.106) becomes

$$\mathfrak{L}[Z(N, V, E, C_2)] = \zeta \int d\boldsymbol{\pi}\ \exp\left[-t \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} \right) \right] \int dP\ \delta(P - C_2) \exp\left(-t\ \frac{P^2}{2m} \right)\ I(t, \{\boldsymbol{\rho}\}) \tag{5.108}$$

The integral that involves the momenta variables is a Gaussian integral, which can be resolved as follows

$$\begin{aligned} I(t,\{\boldsymbol{\pi}\}) &\equiv \int d\boldsymbol{\pi}\ \exp\left[-t\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}\right)\right] = \int d\boldsymbol{\pi}_1\cdots d\boldsymbol{\pi}_{N-1}\ \exp\left(-\frac{t}{2}\sum_{i=1}^{N}\frac{\boldsymbol{\pi}_i^2}{\mu_i}\right) \\ &= \int d\boldsymbol{\pi}_1\ \exp\left(-\frac{t}{2}\frac{\boldsymbol{\pi}_1^2}{\mu_1}\right)\ \cdots\ \int d\boldsymbol{\pi}_{N-1}\ \exp\left(-\frac{t}{2}\frac{\boldsymbol{\pi}_{N-1}^2}{\mu_{N-1}}\right) \\ &= \left(\frac{2\pi\,\mu_1}{t}\right)^{d/2}\ \cdots\ \left(\frac{2\pi\,\mu_{N-1}}{t}\right)^{d/2} = \left(\frac{2\pi}{t}\right)^{(N-1)d/2}\ \prod_{i=1}^{N-1}\mu_i^{d/2} \end{aligned} \tag{5.109}$$

where the fact that the integration over $d\boldsymbol{\pi}$ can be split into the product of $N-1$ Gaussian integrals in the variables $\boldsymbol{\pi}_i$ $(i=1,...,N-1)$, which are represented by a d-dimensional vector, is used, where d identifies the dimension of the physical space of the system, so that $d\boldsymbol{\pi}_i = d\pi_1,...,d\pi_d$ for all $i=1,...,N-1$. Therefore, equation (5.108) can be rewritten as

$$\mathfrak{L}[Z(N,V,E,C_2)] = \zeta\ I(t,\{\boldsymbol{\pi}\})\int dP\ \delta(P-C_2)\ \exp\left(-t\ \frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \tag{5.110}$$

where $I(t,\{\boldsymbol{\pi}\})$ is a constant factor as given by (5.109).
In order to simplify the integration over the momentum P variable, it is convenient to perform a Fourier transform of the total momentum $(C_2 \to k)$ so that

$$\begin{aligned} \mathcal{F}[\mathfrak{L}[Z(N,V,E,C_2)]] &= \mathcal{F}[Z(N,V,t,C_2)] \\ &= Z(N,V,t,k) = \frac{1}{(2\pi)^{1/2}}\ \int_{-\infty}^{\infty} dC_2\ \exp[-\imath\,k\,C_2]\ Z(N,V,t,C_2) \end{aligned} \tag{5.111}$$

where ι is the imaginary unit. The Fourier transform of equation (5.111) will affect only the integral over the momentum coordinate P, so that it can be written as

$$\begin{aligned} Z(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,C_2)]] \\ &= \frac{\zeta}{(2\pi)^{1/2}}\ I(t,\{\boldsymbol{\pi}\})\int_{-\infty}^{\infty} dC_2\int dP\ \exp[-\imath\,k\,C_2]\ \delta(P-C_2)\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{(2\pi)^{1/2}}\ I(t,\{\boldsymbol{\pi}\})\int dP\int_{-\infty}^{\infty} dC_2\ \exp[-\imath\,k\,C_2]\ \delta(P-C_2)\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \end{aligned}$$

The integral over the variable C_2 can be solved through the Dirac delta function, resulting in the following expression

$$\begin{aligned} Z(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,C_2)]] \\ &= \frac{\zeta}{(2\pi)^{1/2}}\ I(t,\{\boldsymbol{\pi}\})\int dP\ \exp[-\imath\,k\,P]\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \end{aligned} \tag{5.112}$$

Substituting expression (5.109) for the Gaussian integral $I(t,\{\boldsymbol{\pi}\})$ in the partition function (5.112) gives

$$\begin{aligned} Z(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,C_2)]] \\ &= \frac{\zeta}{(2\pi)^{1/2}}\left(\frac{2\pi}{t}\right)^{(N-1)d/2}\ \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\int dP\ \exp[-\imath\,k\,P]\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \end{aligned} \tag{5.113}$$

The integral over the variable P can be solved through the following steps

$$\begin{aligned} \int dP\ \exp[-\imath\,k\,P]\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) &= \int dP\ \exp\left[-\frac{t}{2}\left(\frac{P^2}{m}+2\ \frac{\imath\,k\,P}{t}\right)\right] \\ &= \int dP\ \exp\left\{-\frac{t}{2}\left[\left(\frac{P^2}{m}+2\ \frac{\imath\,k\,P}{t}-\frac{k^2}{t^2}\,m\right)+\frac{k^2}{t^2}\,m\right]\right\} \\ &= \int dP\ \exp\left[-\frac{t}{2}\left(\frac{P}{\sqrt{m}}+\frac{\imath\,k}{t}\,\sqrt{m}\right)^2\right]\ \exp\left(-\frac{k^2}{2t}\,m\right) \\ &= \int dP\ \exp\left[-\frac{t}{2m}\left(P+\frac{\imath\,k}{t}\,m\right)^2\right]\ \exp\left(-\frac{k^2}{2t}\,m\right) = \sqrt{\frac{2\pi m}{t}}\ \exp\left(-\frac{k^2}{2t}\,m\right) \end{aligned} \tag{5.114}$$

Therefore, substituting the result (5.114) in (5.113), the Fourier transform of the Laplace transform of the partition function can be written as

$$\begin{aligned} Z(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[Z(N,V,E,P)]] \\ &= \frac{\zeta}{(2\pi)^{1/2}}\left(\frac{2\pi}{t}\right)^{(N-1)d/2} \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\sqrt{\frac{2\pi m}{t}}\ \exp\left(-\frac{k^2}{2t}\,m\right) I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{(2\pi)^{1/2}}(2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\sqrt{2\pi m}\ \exp\left(-\frac{k^2}{2t}\,m\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \\ &= \xi\ \exp\left(-\frac{k^2}{2t}\,m\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \end{aligned} \tag{5.115}$$

where ξ collects all the constant factors, that is

$$\xi = \frac{\zeta}{(2\pi)^{1/2}}(2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\sqrt{2\pi m} \tag{5.116}$$

In order to found the partition function, it is necessary to perform an inverse Fourier transform ($k \to P$) of (5.115), by applying a Fourier anti-transform, that is the inverse operation of (5.111), namely

$$\begin{aligned} \mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[Z(N,V,E,P)]]\} &= \mathcal{F}^{-1}\{\mathcal{F}[Z(N,V,t,P)]\} = \mathcal{F}^{-1}\{Z(N,V,t,k)\} \\ &= Z(N,V,t,P) = \frac{1}{(2\pi)^{1/2}}\int_{-\infty}^{\infty} dk\ \exp[\imath\, kP]\, Z(N,V,t,k) \end{aligned} \tag{5.117}$$

Applying the Fourier anti-transform to (5.115), following the definition (5.117), leads to

$$\begin{aligned} \mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[Z(N,V,E,P)]]\} &= \mathfrak{L}[Z(N,V,E,P)] = Z(N,V,t,P) \\ &= \frac{\xi}{(2\pi)^{1/2}}\int_{-\infty}^{\infty} dk\ \exp[\imath\, kP]\ \exp\left(-\frac{k^2}{2t}\,m\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\xi}{(2\pi)^{1/2}}\int_{-\infty}^{\infty} dk\ \exp\left(-\frac{k^2}{2t}\,m + \imath\, kP\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \end{aligned} \tag{5.118}$$

The integral in the variable k can be resolved as follows

$$\begin{aligned} &\int_{-\infty}^{\infty} dk\ \exp\left(-\frac{mk^2}{2t} + \imath kP\right) = \int_{-\infty}^{\infty} dk\ \exp\left[\left(\frac{\imath\sqrt{m}\ k}{\sqrt{2t}} + \frac{P}{\sqrt{m}}\sqrt{\frac{t}{2}}\right)^2\right]\ \exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right) \\ &= \exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\int_{-\infty}^{\infty} dk\ \exp\left[-\frac{m}{2t}\left(k + \frac{P}{\sqrt{m}}\sqrt{\frac{t}{2}}\,\frac{\sqrt{2t}}{\imath\sqrt{m}}\right)^2\right] \\ &= \exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\int_{-\infty}^{\infty} dk\ \exp\left[-\frac{m}{2t}\left(k + \frac{t\,P}{\imath\, m}\right)^2\right] = \exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\left(\sqrt{\frac{2\pi\, t}{m}}\right) \end{aligned} \tag{5.119}$$

Substituting the obtained result (5.119) in (5.118), the Laplace transform of the partition function can be written as

$$\begin{aligned} \mathfrak{L}[Z(N,V,E,P)] &= Z(N,V,t,P) \\ &= \frac{\xi}{(2\pi)^{1/2}}\int_{-\infty}^{\infty} dk\ \exp\left(-\frac{k^2}{2t}\,m + \imath\, kP\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\xi}{(2\pi)^{1/2}}\left(\sqrt{\frac{2\pi\, t}{m}}\right)\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\ t^{-(N-1)d/2-1/2}\ I(t,\{\boldsymbol{\rho}\}) \\ &= \zeta\,(2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\ t^{-(N-1)d/2}\ I(t,\{\boldsymbol{\rho}\}) \end{aligned} \tag{5.120}$$

where the constant factors has been rewritten, recalling the definition (5.116), as

$$\begin{aligned}\frac{\xi}{(2\pi)^{1/2}}\left(\sqrt{\frac{2\pi}{m}}\right) &= \frac{1}{(2\pi)^{1/2}}\left(\sqrt{\frac{2\pi}{m}}\right)\frac{\zeta}{(2\pi)^{1/2}}(2\pi)^{(N-1)d/2}\ \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\sqrt{2\pi m} \\ &= \zeta\,(2\pi)^{(N-1)d/2}\ \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\end{aligned} \tag{5.121}$$

Recalling the definition of $I(t,\{\boldsymbol{\rho}\})$ given in (5.107), the Laplace transform of the partition function can be rewritten as

$$\begin{aligned}&\mathfrak{L}[Z(N,V,E,P)] = Z(N,V,t,P) \\ &\quad= \zeta\,(2\pi)^{(N-1)d/2}\ \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\,\exp[-t\,\phi(\boldsymbol{\rho})] \\ &\quad= \Omega\ \exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\,\exp[-t\,\phi(\boldsymbol{\rho})]\end{aligned} \tag{5.122}$$

where Ω includes all the constant factors in (5.122) and it is thus defined by

$$\Omega = \zeta\,(2\pi)^{(N-1)d/2}\ \left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right) = \frac{\zeta\,(2\pi)^{(N-1)d/2}}{\mu_N^{d/2}}\ \left(\prod_{i=1}^{N}m_i^{d/2}\right) \tag{5.123}$$

where the relation (5.62) has been used, and $\mu_N = m$ is the total mass of the system. At this point, the last step is to compute the Laplace anti-transform of the equation (5.122), performing the change of variable $t \to E$ and finally obtaining the partition function

$$\begin{aligned}\mathfrak{L}^{-1}\{\mathfrak{L}[Z(N,V,E,P)]\} &= \mathfrak{L}^{-1}\{Z(N,V,t,P)\} = Z(N,V,E,P) \\ &= \frac{1}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}e^{tE}\,Z(N,V,t,P)\,dt\end{aligned} \tag{5.124}$$

Applying the definition (5.124) to the Laplace transform of the partition function (5.122), the following expression can be obtained for the partition function

$$\begin{aligned}&Z(N,V,E,P) = \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ \exp(tE)\,\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\ t^{-(N-1)d/2}\ \exp[-t\,\phi(\boldsymbol{\rho})] \\ &= \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ &= \frac{\Omega}{2\pi\,\imath}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right]\end{aligned} \tag{5.125}$$

The complex integral in the previous equation, also known as the Bromwich integral, has the solution (see Appendix, Section A.6)

$$\begin{aligned}&\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ &\qquad= \frac{2\pi\,\imath}{\Gamma[(N-1)d/2]}\,\Theta\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\,\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)^{(N-1)d/2-1}\end{aligned} \tag{5.126}$$

Substituting this result in (5.125), the partition function can be finally written as

$$\begin{aligned}&Z(N,V,E,P) \\ &\quad= \frac{\Omega}{\Gamma[(N-1)d/2]}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\ \Theta\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\,\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)^{(N-1)d/2-1}\end{aligned} \tag{5.127}$$

where the constant Ω is defined by (5.123).

5.1.6 Calculation of temperature and the mean kinetic energy

For the calculation of the temperature and the mean kinetic energy in a microcanonical ensemble using periodic boundary conditions, the phase space volume has to be defined as

$$\begin{aligned}\Phi(N,V,E,C_2) &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \Theta(C_1 - \mathcal{H}(\boldsymbol{\pi},P,\boldsymbol{\rho}))\ \delta(P-C_2) \\ &\equiv \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \Theta(E - \mathcal{H}(\boldsymbol{\pi},P,\boldsymbol{\rho}))\ \delta(P-C_2)\end{aligned} \tag{5.128}$$

where $d\boldsymbol{\pi} = \boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_{N-1}$ and $d\boldsymbol{\rho} = \boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_{N-1}$ are the Jacobi coordinates for the nuclear positions and momenta, as defined in Section 5.1.4, and the Hamiltonian $\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho})$ is given by equation (5.100). Following the same procedure used in Section 5.1.5.2, a Laplace transform of equation (5.128) can be done, so that

$$\begin{aligned}\mathfrak{L}[\Phi(N,V,E,C_2)] = \Phi(N,V,t,C_2) &= \int_0^\infty e^{-t\,E}\,\Phi(N,V,E,C_2)\,dE \\ &= \zeta \int_0^\infty dE \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ e^{-tE}\ \Theta(E - \mathcal{H}(\boldsymbol{\pi},P,\boldsymbol{\rho}))\ \delta(P-C_2) \\ &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(P-C_2) \left(-\frac{1}{t}\right) e^{-t\,E}\Big|_{E=\mathcal{H}(\boldsymbol{\pi},P,\boldsymbol{\rho})}^{E=\infty} \\ &= \zeta \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(P-C_2)\ \frac{1}{t}\, e^{-t\,\mathcal{H}(\boldsymbol{\pi},P,\boldsymbol{\rho})}\end{aligned} \tag{5.129}$$

The Hamiltonian $\mathcal{H}$ in the previous equation can be substituted with its explicit form (5.100), that is

$$\begin{aligned}\mathfrak{L}[\Phi(N,V,E,C_2)] &= \frac{\zeta}{t} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho}\ \delta(P-C_2)\ \exp\left[-t\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i} + \frac{P^2}{2m} + \phi(\boldsymbol{\rho})\right)\right] \\ &= \frac{\zeta}{t} \int d\boldsymbol{\pi}\ \exp\left[-t\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}\right)\right] \int dP\ \delta(P-C_2)\ \exp\left(-t\,\frac{P^2}{2m}\right) \int d\boldsymbol{\rho}\ \exp\left[-t\,\phi(\boldsymbol{\rho})\right]\end{aligned} \tag{5.130}$$

where m is the total mass of the system. In order to simplify the notation, the integral over the Jacobi position coordinates will be sometimes abbreviated as

$$I(t,\{\boldsymbol{\rho}\}) \equiv \int d\boldsymbol{\rho}\ \exp\left[-t\,\phi(\boldsymbol{\rho})\right] \equiv \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}\ \exp[-t\,\phi(\boldsymbol{\rho})] \tag{5.131}$$

so that equation (5.130) becomes

$$\mathfrak{L}[\Phi(N,V,E,C_2)] = \frac{\zeta}{t} \int d\boldsymbol{\pi}\ \exp\left[-t\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}\right)\right] \int dP\ \delta(P-C_2)\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \tag{5.132}$$

The integral that involves the momenta variables is a Gaussian integral, which can be easily resolved, as demonstrated in (5.109), and it is given by

$$I(t,\{\boldsymbol{\pi}\}) \equiv \int d\boldsymbol{\pi}\ \exp\left[-t\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}\right)\right] = \left(\frac{2\pi}{t}\right)^{(N-1)d/2} \prod_{i=1}^{N-1} \mu_i^{d/2} \tag{5.133}$$

Therefore, equation (5.132) can be rewritten as

$$\mathfrak{L}[\Phi(N,V,E,C_2)] = \frac{\zeta}{t}\ I(t,\{\boldsymbol{\pi}\}) \int dP\ \delta(P-C_2)\ \exp\left(-t\,\frac{P^2}{2m}\right)\ I(t,\{\boldsymbol{\rho}\}) \tag{5.134}$$

In order to simplify the integration over the momentum P variable, it is convenient to perform a Fourier transform of the total momentum ($C_2 \to k$) so that

$$\begin{aligned}\mathcal{F}[\mathfrak{L}[\Phi(N,V,E,C_2)]] &= \mathcal{F}[\Phi(N,V,t,C_2)] \\ &= \Phi(N,V,t,k) = \frac{1}{(2\pi)^{1/2}} \int_{-\infty}^{\infty} dC_2\ \exp[-\imath\,k\,C_2]\,\Phi(N,V,t,C_2)\end{aligned} \tag{5.135}$$

where $\imath$ is the imaginary unit. The Fourier transform of equation (5.135) will affect only the integral over the momentum coordinate P, so that it can be written as

$$\begin{aligned}\Phi(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[\Phi(N,V,E,C_2)]] \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}}\; I(t,\{\boldsymbol{\pi}\}) \int_{-\infty}^{\infty} dC_2 \int dP\;\; \exp[-\imath\, k\, C_2]\;\, \delta(P-C_2)\, \exp\left(-t\,\frac{P^2}{2m}\right)\; I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}}\; I(t,\{\boldsymbol{\pi}\}) \int dP \int_{-\infty}^{\infty} dC_2\;\; \exp[-\imath\, k\, C_2]\;\, \delta(P-C_2)\, \exp\left(-t\,\frac{P^2}{2m}\right)\; I(t,\{\boldsymbol{\rho}\})\end{aligned}$$

The integral over the variable C_2 can be solved through the Dirac delta function, resulting in the following expression

$$\begin{aligned}\Phi(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[\Phi(N,V,E,C_2)]] \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}}\; I(t,\{\boldsymbol{\pi}\}) \int dP\;\; \exp[-\imath\, k\, P]\, \exp\left(-t\,\frac{P^2}{2m}\right)\; I(t,\{\boldsymbol{\rho}\})\end{aligned} \tag{5.136}$$

Substituting expression (5.133) for the Gaussian integral $I(t,\{\boldsymbol{\pi}\})$ in the partition function (5.136) gives

$$\begin{aligned}\Phi(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[\Phi(N,V,E,C_2)]] \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}} \left(\frac{2\pi}{t}\right)^{(N-1)d/2} \left(\prod_{i=1}^{N-1} \mu_i^{d/2}\right) \int dP\;\; \exp[-\imath\, k\, P]\, \exp\left(-t\,\frac{P^2}{2m}\right)\; I(t,\{\boldsymbol{\rho}\})\end{aligned} \tag{5.137}$$

The integral over the variable P can be easily solved, as demonstrated in (5.114), by completing the square in the exponent, leading to the following expression for the phase space volume

$$\begin{aligned}\Phi(N,V,t,k) &= \mathcal{F}[\mathfrak{L}[\Phi(N,V,E,P)]] \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}} \left(\frac{2\pi}{t}\right)^{(N-1)d/2} \left(\prod_{i=1}^{N-1} \mu_i^{d/2}\right) \sqrt{\frac{2\pi m}{t}}\;\, \exp\left(-\frac{k^2}{2t}\, m\right)\; I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\zeta}{t\,(2\pi)^{1/2}} (2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1} \mu_i^{d/2}\right) \sqrt{2\pi m}\;\, \exp\left(-\frac{k^2}{2t}\, m\right)\;\, t^{-(N-1)d/2-1/2}\; I(t,\{\boldsymbol{\rho}\}) \\ &= \xi\;\, \exp\left(-\frac{k^2}{2t}\, m\right)\;\, t^{-(N-1)d/2-3/2}\; I(t,\{\boldsymbol{\rho}\})\end{aligned} \tag{5.138}$$

where ξ collects all the constant factors, that is

$$\xi = \frac{\zeta}{(2\pi)^{1/2}} (2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1} \mu_i^{d/2}\right) \sqrt{2\pi m} = \zeta\sqrt{m}(2\pi)^{(N-1)d/2} \left(\prod_{i=1}^{N-1} \mu_i^{d/2}\right) \tag{5.139}$$

In order to found the partition function, it is necessary to perform an inverse Fourier transform ($k \to P$) of (5.138), by applying a Fourier anti-transform, that is the inverse operation of (5.135), namely

$$\begin{aligned}\mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[\Phi(N,V,E,P)]]\} &= \mathcal{F}^{-1}\{\mathcal{F}[\Phi(N,V,t,P)]\} = \mathcal{F}^{-1}\{\Phi(N,V,t,k)\} \\ &= \Phi(N,V,t,P) = \frac{1}{(2\pi)^{1/2}} \int_{-\infty}^{\infty} dk\;\, \exp[\imath\, kP]\, \Phi(N,V,t,k)\end{aligned} \tag{5.140}$$

Applying the Fourier anti-transform to (5.138), following the definition (5.140), leads to

$$\begin{aligned}\mathcal{F}^{-1}\{\mathcal{F}[\mathfrak{L}[\Phi(N,V,E,P)]]\} &= \mathfrak{L}[\Phi(N,V,E,P)] = \Phi(N,V,t,P) \\ &= \frac{\xi}{(2\pi)^{1/2}} \int_{-\infty}^{\infty} dk\;\, \exp[\imath\, kP]\, \exp\left(-\frac{k^2}{2t}\, m\right)\;\, t^{-(N-1)d/2-3/2}\; I(t,\{\boldsymbol{\rho}\}) \\ &= \frac{\xi}{(2\pi)^{1/2}} \int_{-\infty}^{\infty} dk\;\, \exp\left(-\frac{k^2}{2t}\, m + \imath\, kP\right)\;\, t^{-(N-1)d/2-3/2}\; I(t,\{\boldsymbol{\rho}\})\end{aligned} \tag{5.141}$$

The integral in the variable k can be resolved by completing the square, as in (5.119), leading to

$$\int_{-\infty}^{\infty} dk\, \exp\left(-\frac{mk^2}{2t} + \imath kP\right) = \sqrt{\frac{2\pi\, t}{m}}\;\, \exp\!\left(-\frac{P^2}{m}\,\frac{t}{2}\right) \tag{5.142}$$

Substituting this result in equation (5.141), it becomes

$$\begin{aligned}\mathfrak{L}[\Phi(N,V,E,P)] &= \Phi(N,V,t,P) \\ &= \frac{\xi}{(2\pi)^{1/2}}\sqrt{\frac{2\pi\,t}{m}}\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2-3/2}\;I(t,\{\boldsymbol{\rho}\}) \\ &= \zeta\,(2\pi)^{(N-1)d/2}\;\left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2-1}\;I(t,\{\boldsymbol{\rho}\})\end{aligned} \tag{5.143}$$

where the constant factors has been rewritten, recalling the definition (5.139), as

$$\frac{\xi}{(2\pi)^{1/2}}\sqrt{\frac{2\pi}{m}} = \frac{\xi}{\sqrt{m}} = \zeta\,(2\pi)^{(N-1)d/2}\;\left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right) \tag{5.144}$$

Recalling the definition of $I(t,\{\boldsymbol{\rho}\})$ given in (5.131), the Laplace transform (5.143) can be rewritten as

$$\begin{aligned}&\mathfrak{L}[\Phi(N,V,E,P)] = \Phi(N,V,t,P) \\ &= \zeta\,(2\pi)^{(N-1)d/2}\;\left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right)\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2-1}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\,\exp[-t\,\phi(\boldsymbol{\rho})] \\ &= \Omega\;\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\,t^{-(N-1)d/2-1}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\,\exp[-t\,\phi(\boldsymbol{\rho})]\end{aligned} \tag{5.145}$$

where Ω includes all the constant factors in (5.145) and it is thus defined by

$$\Omega = \zeta\,(2\pi)^{(N-1)d/2}\;\left(\prod_{i=1}^{N-1}\mu_i^{d/2}\right) = \frac{\zeta\,(2\pi)^{(N-1)d/2}}{\mu_N^{d/2}}\;\left(\prod_{i=1}^{N}m_i^{d/2}\right) \tag{5.146}$$

where the relation (5.62) has been used, and $\mu_N = m$ is the total mass of the system. At this point, the last step is to compute the Laplace anti-transform of the equation (5.145), performing the change of variable $t \to E$ in the following way

$$\begin{aligned}\mathfrak{L}^{-1}\{\mathfrak{L}[\Phi(N,V,E,P)]\} &= \mathfrak{L}^{-1}\{\Phi(N,V,t,P)\} = \Phi(N,V,E,P) \\ &= \frac{1}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}e^{tE}\,\Phi(N,V,t,P)\,dt\end{aligned} \tag{5.147}$$

and finally obtaining the expression

$$\begin{aligned}&\Phi(N,V,E,P) = \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\;\exp(tE)\,\exp\left(-\frac{P^2}{m}\,\frac{t}{2}\right)\;t^{-(N-1)d/2-1}\;\exp[-t\,\phi(\boldsymbol{\rho})] \\ &= \frac{\Omega}{2\pi\,\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\;t^{-(N-1)d/2-1}\;\exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ &= \frac{\Omega}{2\pi\,\imath}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\;t^{-(N-1)d/2-1}\;\exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right]\end{aligned} \tag{5.148}$$

The complex integral in the previous equation, also known as the Bromwich integral, has the solution (see Appendix, Section A.6)

$$\begin{aligned}&\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\;t^{-(N-1)d/2-1}\;\exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ &= \int_{\gamma-\imath\infty}^{\gamma+\imath\infty}dt\;t^{-[(N-1)d/2+1]}\;\exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ &= \frac{2\pi\,\imath}{\Gamma[(N-1)d/2+1]}\;\Theta\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\;\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)^{(N-1)d/2}\end{aligned} \tag{5.149}$$

Substituting this result in (5.148), the phase space volume can be finally written as

$$\Phi(N,V,E,P) = \frac{\Omega}{\Gamma[(N-1)d/2+1]} \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1} \; \Theta\left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right) \left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right)^{(N-1)d/2} \tag{5.150}$$

where the constant Ω is defined by (5.146). Recalling the results obtained in Section 5.1.5.2, the corresponding partition function can be written as in equation (5.127), that is reported here below

$$Z(N,V,E,P) = \frac{\Omega}{\Gamma[(N-1)d/2]} \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1} \; \Theta\left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right) \left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right)^{(N-1)d/2-1} \tag{5.151}$$

where the constant Ω is defined by (5.123) and it exactly corresponds to the same constant factor appearing in (5.150), see equation (5.146).
Moreover, the mean value of a physical quantity $A(\boldsymbol{\pi}, P, \boldsymbol{\rho})$, given by

$$\langle A \rangle = \frac{\zeta}{Z} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\rho} \; A(\boldsymbol{\pi}, P, \boldsymbol{\rho}) \; \delta(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - E) \; \delta(P - C_2) \tag{5.152}$$

can be written in its explicit form, following identically the procedure in Section 5.1.5.2 used to found the explicit expression of the partition function, leading to the equation

$$\langle A \rangle = \frac{\Omega}{Z\,\Gamma[(N-1)d/2]} \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1} \; A(\boldsymbol{\pi}, P, \boldsymbol{\rho}) \; \Theta\left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right) \left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right)^{(N-1)d/2-1} \tag{5.153}$$

where the denominator contains the partition function $Z \equiv Z(N,V,E,P)$ in the form (5.151), as given by the definition (5.152).
Considering the definition of entropy given by Ref. [43] as

$$S = k_b \ln(\Phi) \tag{5.154}$$

the temperature of the system can be computed using the relation

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_V \tag{5.155}$$

leading to the expression

$$\frac{1}{T} = k_b \left(\frac{1}{\Phi}\frac{\partial \Phi}{\partial E}\right)_V = k_b \left(\frac{Z}{\Phi}\right)_V \qquad \rightarrow \qquad k_b T = \left(\frac{\Phi}{Z}\right)_V \tag{5.156}$$

Therefore, using the derived expression for the phase space volume (5.150) and the partition function (5.151), the temperature of the system can be expressed by the relation

$$k_b T = \left(\frac{\Phi}{Z}\right)_V = \frac{\Omega}{Z\,\Gamma[(N-1)d/2+1]} \int d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1} \; \Theta\left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right) \left(E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho})\right)^{(N-1)d/2} \tag{5.157}$$

Using the following property of the Gamma function

$$\Gamma(n+1) = n\,\Gamma(n) \qquad \rightarrow \qquad \Gamma\left[\frac{(N-1)d}{2}+1\right] = \frac{(N-1)d}{2} \;\; \Gamma\left[\frac{(N-1)d}{2}\right] \tag{5.158}$$

and comparing expression (5.157) with equation (5.153) gives

$$k_b T = \frac{2}{(N-1)d} \; \langle E - \frac{P^2}{2m} - \phi(\boldsymbol{\rho}) \rangle = \frac{2}{(N-1)d} \; \langle E - \phi(\boldsymbol{\rho}) \rangle - \frac{1}{(N-1)d} \left(\frac{P^2}{m}\right) \tag{5.159}$$

$$k_b\,T = \frac{2}{(N-1)d}\,\langle E_k\rangle - \frac{1}{(N-1)d}\left(\frac{P^2}{m}\right) \tag{5.160}$$

where E_k is the kinetic energy of the system, that is $E_k = E - \phi(\boldsymbol{\rho}) \equiv C_1 - \phi(\boldsymbol{\rho})$. Fixing $P = 0$ at the beginning of the molecular dynamics simulation, the value of $P = \|\mathbf{P}\|$ should be constant and equal to zero during the whole simulation (because the total linear momentum $\mathbf{P}$, and thus the value of P that is its norm, is a conserved quantity for the Hamilton equations of motion), so that the mean kinetic energy of the system at each step can be computed using (5.160) with $P = 0$ as

$$\langle E_k\rangle = \frac{(N-1)d}{2}\,k_b\,T \tag{5.161}$$

5.2 Generation of different ensembles

As previously explained, in conventional molecular dynamics the microcanonical ensemble is generated due to the conservation laws of Hamilton's equations derived from Newton laws of motion. Indeed, the simple integration of the classical equations of motion for such a system leads, in the limit of infinite sampling, to a trajectory mapping a microcanonical ensemble of microstates, as a consequence of the Hamiltonian flow conservation. In this ensemble the number of particles N, the volume V, and the energy E are kept constant, so that it is often called NVE ensemble. However, experiments are usually conducted at constant temperature and/or pressure, which corresponds to a canonical and/or isobaric ensemble. In these ensembles, energy is not conserved but fluctuates due to exchange of energy between the system and the thermal and/or pressure reservoir to which it is coupled, so as to generate the proper Boltzmann distribution. Although the energy fluctuations vanish in the thermodynamic limit, the vast majority of simulations are performed far enough from this limit, so that the fluctuations cannot be neglected.
Performing a molecular dynamics simulation in an other ensemble than microcanonical requires a means to keep at least one intensive quantity constant (on average) during the simulation. This can be done either in a hard or in a soft manner. Applying a hard boundary condition on an intensive macroscopic variable means constraining a corresponding instantaneous observable to its specified macroscopic value at every time step during the simulation. This approach is for example used for the Woodcock simple scaling introduced in Section 5.3.2. It is a kind of method also called constraint approach (see Section 5.2.1 but with the imposition of very strict constraint conditions. In contrast, the use of a soft boundary condition allows for fluctuations in the instantaneous observable, only requiring its average to remain equal to the macroscopic value (on a given timescale). The definition of a soft boundary condition generally also requires the specification of a timescale for which the average observable should match the specified value. Typical methods for applying soft boundary conditions are the penalty function, weak coupling, extended system and stochastic coupling methods. The thermostats (except for the Woodcock simple scaling one) and the barostats introduced in the following are all examples for soft boundary conditions. In particular, the idea underneath the constraint methods and the extended system approach will be described in Sections 5.2.1 and 5.2.2, respectively. The equations related to the constraint method and the extended system approach will be analyzed and discussed in details in Section 5.3, where the approaches to obtain a constant temperature molecular dynamics simulation are described, and in Section 5.4, where the equations of motion that generate constant temperature and pressure ensembles are derived.

5.2.1 The constraint methods

In large systems, the relative amplitude of the fluctuations of quantities such as the kinetic energy or the pressure of a system become very small. Therefore, the suppression of the thermal fluctuation of the kinetic energy and/or the pressure fluctuation of the volume of the system do not affect seriously static and dynamical quantities. The constraint method is based on this matter of fact, suppressing the fluctuation of the quantities that are supposed to be constant in the ensemble (e.g. the temperature or the pressure, respectively in the canonical or isobaric ensemble). Therefore, in the constraint approach the total kinetic energy and/or the pressure of the system are kept to a constant value by imposition of a constraint. A constant temperature condition, for example, is attained by keeping the kinetic energy to a constant value by imposing a constraint on the equations of motion.
The earliest proposal of the simulation at constant temperature is the velocity scaling algorithm by L. V. Woodcock.[44] Later on, the strict conditions imposed by Woodcock has been relaxed and modified, so that new constraint approaches arise. Three different techniques to maintain the temperature of the system fixed to a constant value, based of the constraint approach and frequently used in molecular dynamics simulation codes, will be described in Sections 5.3.1, 5.3.2 and 5.3.3.

5.2.2 The extended system methods

In ordinary molecular dynamics simulations, an isolated system in a fixed simulation unit cell is considered. The total energy and the volume of a physical system are constant under this situation. To

break the conservation law which restricts the behavior of a physical system, and to realize a constant temperature or a constant pressure condition, a physical system is extended to a composite system consisting of a physical system and an external system. Usually, a system surrounded by a heat reservoir at constant temperature or pressure is considered. Therefore, additional degrees of freedom corresponding to an external system in contact to the physical one are introduced. The conservation law still holds in an extended system, but the total energy or the volume of a physical system is allowed to fluctuate. This idea accords well with the schematic image of the canonical or the isobaric ensemble in statistical mechanics. The major difference between the extended system method and a real situation is that in the theoretical framework a very small system for an external system instead of a macroscopic reservoir is taken into account.
The idea of the extended system method was first presented by Andersen in his work on the constant pressure method.[45] A temperature version of the extended system method was proposed by Nosé[46, 47, 48] and later modified by Hoover, and it is therefore called the Nosé-Hoover thermostat. The extended system approach is a very general tool, and it is therefore used across the years extend standard molecular dynamics to various ensembles which match more closely the experimental conditions.

5.3 Constant temperature approaches

A modification of the Newtonian equation scheme with the purpose of generating a thermodynamic ensemble at constant temperature is called a thermostat algorithm. The introduction of a thermostat can be motivated by the following reasons: (i) to match experimental conditions, (ii) to study temperature-dependent processes (such as, for example, phase transitions), (iii) to enhance the efficiency of a conformational search (as in simulated annealing technique); (iv) to avoid steady energy drifts caused by the accumulation of numerical errors during a molecular dynamics simulation.[4]
The use of a thermostat requires the definition of an instantaneous temperature. This temperature will be compared to the reference temperature T_0 of the heat bath to which the system is coupled. Following from the equipartition theorem, the average internal nuclear kinetic energy E_k of a system with N nuclei in a three-dimensional space is related to its macroscopic temperature T through the equation

$$E_k = \langle E_k(t) \rangle = \frac{g}{2} k_b T \tag{5.162}$$

where $E_k(t)$ is the instantaneous nuclear kinetic energy computed at time t (whose definition has been introduced in Section 3.3 by means of internal nuclear velocities), k_b is the Boltzmann constant, and g is essentially equal to the number of degrees of freedom of the physical system, which in generally differs from the number of degrees of freedom d defined as described in Section 3.3. Indeed, the number of degrees of freedom d are mainly related to the geometry constraint of the system, while g is connected the total number of degrees of freedom obtained after the addition or the removal from the physical subsystem of some degrees of freedom associated to the external system. However, obviously the total number of degrees of freedom g has to contain also the geometrical constraint of the system, so it is somehow related with the number of degrees of freedom defined by the integer d. Defining the instantaneous temperature $T(t)$ at any time step as

$$T(t) = \frac{2}{g k_b} E_k(t) = \frac{1}{g k_b} \left[\sum_{i=1}^{N} m_i \, \boldsymbol{v}_i^2(t) \right] \tag{5.163}$$

one ensures that the average temperature $\langle T(t) \rangle$ is identical to the macroscopic temperature T.
Different methods for temperature control are developed and actually well-assessed. They are classified into three main types: i) the constraint method, ii) the stochastic method, and iii) the extended system method. A key factor in distinguishing between these methods is the way in which the thermal contact between a physical system and a heat bath is taken into consideration. In the present work, two of these approaches are considered, namely, the constraint and the extended system methods.
The goal is to define particular equations of motion, different from the simple Newtonian equations used in the NVE ensemble, which allows to generate the so-called canonical (NVT) ensemble, where the number of particles N, the volume of the system V and the temperature T are kept constant. In the constraint methods, the total kinetic energy is kept to a constant value by imposition of a constraint, leading to a suppression of the thermal fluctuation of the instantaneous kinetic energy, and therefore (through equation (5.163)) an absence of temperature fluctuations during the dynamics. This behavior does not allow to correctly sample the canonical ensemble. The second main approach is the extended system method, where an additional degree of freedom representing a heat bath is introduced in the equations of motion, and the total energy of the physical system is allowed to fluctuate by a thermal contact with the heat bath. moreover, the extended technique permits the temperature of the physical system to fluctuate during the dynamics, thus possibly generating the correct canonical ensemble. Furthermore, the extended system approaches have a strong theoretical justification, differently from the majority of the constraint methods. Indeed, these approaches rely on precise equations of motion, and this provides an easy way to analytically compute the partition function associated, thus understanding the ensemble generated by the extended dynamics.
In the following, four main methods which allow to maintain constant temperature during a molecular dynamics simulation are introduced. The first two techniques, namely, the Gaussian thermostat and

[4] A thermostat algorithm (involving explicit reference to a heat-bath temperature T_0) will avoid systematic energy drifts, because if the instantaneous temperature is forced to fluctuate within a limited range around T_0, the energy will also fluctuate within a limited range around its corresponding equilibrium value. To perform long microcanonical simulations (no thermostat), it is also advisable to employ an algorithm that will constrain the energy to its reference value.

the simple velocity rescaling are classified as constrained methods; the third one, namely, the Berendsen thermostat, is mainly a constraint method but with the insertion of concepts coming from the stochastic approach, while the last one, the so-called Nosé-Hoover thermostat, is an extended system algorithm. The first three methods do not properly generate a NVT ensemble, whilst the fourth method, the Nosé-Hoover thermostat, introduces a specific Hamiltonian that leads to equations of motion which permit a good sampling of the canonical ensemble.

5.3.1 Gaussian thermostat

The average of the kinetic energy is usually used as a measure of the temperature in molecular dynamics simulations. The kinetic energy should fluctuate both in the microcanonical and the canonical ensembles, but the relative amplitude of the fluctuations becomes very small in a large system. Therefore, a constant temperature condition can be attained by suppressing thermal fluctuations of the kinetic energy and keeping it to a constant value.
A constant kinetic energy condition can be achieved by imposing a constraint

$$R(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{g}{2} k_b T_0 = 0 \tag{5.164}$$

to the equations of motion, where g is the number of degrees of freedom, k_b is the Boltzmann constant and T_0 is the target temperature, which has to be maintained constant throughout the simulation. The relation (5.164) can be equivalently rewritten as

$$E_k(t) = \frac{1}{2}\sum_{i=1}^{N} m_i\, \boldsymbol{v}_i^2(t) = \frac{g}{2} k_b T_0 \quad \rightarrow \quad \text{using equation (5.163):} \;\; T(t) = T_0 \tag{5.165}$$

so that it is easy to see that, through equation (5.164), the instantaneous kinetic energy is constrained to assume a constant value. Indeed, constraining the instantaneous kinetic energy $E_k(t)$ to be a constant, computed with respect to the target temperature T_0, coincides with imposing a constraint on the instantaneous temperature $T(t)$, by means of equation (5.163), so that the instantaneous temperature becomes a constant equal to the target temperature T_0. This constraint depends on the velocities, thus generating a type of problem classified as non-holonomic constrained case in classical dynamics. The application of non-holonomic constraints in molecular dynamics method has been discussed by Haile *et al.*[49]. There is some ambiguity on how to impose this constraint to a mechanical system. A method proposed by Evans *et al.*[50] is based on Gauss principle of least constraint, which states that the actual constrained motion should occur along a trajectory obtained by normal projection of a force onto a constraint hypersurface. A constraint force which is necessary to restrict the trajectory on a constraint hypersurface is the least in this choice. A non-holonomic constraint is generally expressed as

$$R(\boldsymbol{q},\dot{\boldsymbol{q}},t) = 0 \tag{5.166}$$

Differentiating the previous equation (5.166) with respect to time gives a relation which the acceleration $\ddot{\boldsymbol{q}} = \dot{\boldsymbol{p}}$ should satisfy,

$$\frac{dR}{dt} = \dot{\boldsymbol{q}}\frac{\partial R}{\partial \boldsymbol{q}} + \ddot{\boldsymbol{q}}\frac{\partial R}{\partial \dot{\boldsymbol{q}}} + \frac{\partial R}{\partial t} = 0 \quad \rightarrow \quad \ddot{\boldsymbol{q}}\frac{\partial R}{\partial \dot{\boldsymbol{q}}} = -\dot{\boldsymbol{q}}\frac{\partial R}{\partial \boldsymbol{q}} - \frac{\partial R}{\partial t} \tag{5.167}$$

An unconstrained motion described by

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} = \mathbf{F}_i(\boldsymbol{q}(t)) \qquad i = 1, ..., N \tag{5.168}$$

where $\phi(\boldsymbol{q})$ is the potential associated to the nuclear forces,[5] gives a trajectory which does not stay on the constraint hypersurface defined by equation (5.166). Therefore, a constrained force $\mathbf{F}_{i,c}$ has to be added to the equations of motion to prevent the deviation from the constrained hypersurface

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} + \mathbf{F}_{i,c} \qquad i = 1, ..., N \tag{5.169}$$

[5]Note: the potential $\phi(\boldsymbol{q})$ is associated to the forces acting on the nuclei, due to the electronic and nuclear mean field generated by the HF or DFT electronic ground state computed at constant nuclear positions.

The constraint force $\mathbf{F}_{i,c}$ is minimum when it is chosen perpendicular to the constraint surface or parallel to the gradient $\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \partial R/\partial\dot{\boldsymbol{q}}$ of the surface. In particular, a constraint force from the constant temperature constraint is proportional to the velocity, so that the form of $\mathbf{F}_{i,c}$ can be chosen proportional to the gradient vector given by

$$\boldsymbol{n}_i(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \frac{\partial R}{\partial\dot{\boldsymbol{q}}_i} = m_i\dot{\boldsymbol{q}}_i = \boldsymbol{p}_i \qquad\qquad i = 1,...,N \tag{5.170}$$

Following this reasoning, the simple Newtonian equations of motion (5.8) for the nuclei should be modified in the following way

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \tag{5.171}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i} - \gamma\boldsymbol{p}_i = \mathbf{F}_i(\boldsymbol{q}(t)) - \gamma\boldsymbol{p}_i \qquad i = 1,...,N \tag{5.172}$$

where the last equation of motion can be rewritten as

$$m_\alpha\dot{v}_\alpha(t) = F_\alpha(t) - m_\alpha\gamma_\alpha v_\alpha(t) \qquad\qquad \alpha = 1,...,dN \tag{5.173}$$

which is easily identified as a simple form of the Langevin equation, where d is the dimensionality of the system. At the same time, using vector notation that permits to easily identified the coordinates of a given nucleus, equation (5.173) can be expressed as

$$m_i\dot{\boldsymbol{v}}_i(t) = \mathbf{F}_i(\boldsymbol{q}(t)) - m_i\gamma\boldsymbol{v}_i(t) \qquad\qquad i = 1,...,N \tag{5.174}$$

where $\gamma = \gamma_\alpha$ are constants imposed to be equal for all the particles in the system, that cannot be interpreted as friction coefficients and whose values can be positive or negative: a positive value indicates that heat flows from the system to the heat bath, while a negative value indicates a heat flow in the opposite direction. Mathematically, the coefficient γ of the constrained force term is a Lagrangian undetermined multiplier, whose expression is determined to satisfy the time derivative of the constraint equation (5.164), given by

$$\frac{dR(\boldsymbol{q},\boldsymbol{p})}{dt} = \frac{d}{dt}\left[\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} - \frac{g}{2}k_bT_0\right] = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\cdot\dot{\boldsymbol{p}}_i = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\cdot(\mathbf{F}_i(\boldsymbol{q}(t)) - \gamma\boldsymbol{p}_i) = 0 \tag{5.175}$$

Therefore, the following relation is obtained

$$\gamma = \left[\sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\cdot\mathbf{F}_i(\boldsymbol{q}(t))\right]\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}\right)^{-1} = \frac{1}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i(\boldsymbol{q}(t)) = -\frac{1}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\frac{\partial\phi(\boldsymbol{q}(t))}{\partial\boldsymbol{q}_i} \tag{5.176}$$

so that the equations of motion (5.174) can be rewritten using this expression for the constant γ as

$$m_i\dot{\boldsymbol{v}}_i(t) = \mathbf{F}_i(\boldsymbol{q}(t)) - \frac{m_i}{gk_bT_0}\left[\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i(\boldsymbol{q}(t))\right]\boldsymbol{v}_i(t) \qquad\qquad i = 1,...,N \tag{5.177}$$

Therefore, the equations of motion (5.177) derived above corresponds to the Langevin equations (5.174) with the constraint of constant instantaneous kinetic energy (and temperature) given by (5.165). If the initial value of the total nuclear kinetic energy is set equal to $(g/2)k_bT_0$ at an initial step, equations (5.171) and (5.172) will maintain this value during the whole molecular dynamics simulation.[6]
Carrying out simulations with the constrained equations of motion (5.171) and (5.172), the canonical distribution in the coordinate part of a phase space $\boldsymbol{\Gamma} = (\boldsymbol{p},\boldsymbol{q})$ can be obtained. In the following, this

[6] Otherwise, the same result about the expression for γ can be obtained using the following strategy.
On the equations of motion (5.174) the non-holonomic constraint (5.165) is imposed, so that the instantaneous kinetic energy is constrained to assume a constant value. Constraining the instantaneous kinetic energy $E_k(t)$ to be a constant coincides with imposing a constraint on the instantaneous temperature $T(t)$, by means of equation (5.163), so that $T(t)$ becomes a constant equal to the target temperature T_0. Therefore, the constraint (5.165) can be reformulated as in the

assertion will be proved analytically.
The distribution function $f(\boldsymbol{p},\boldsymbol{q})$ expresses the probability that a phase space point $(\boldsymbol{p},\boldsymbol{q})$ will be occupied by the system. A generalized Liouville equation expresses the probability conservation in the phase space $\boldsymbol{\Gamma}$ and it is given by

$$\frac{\partial f}{\partial t}+\frac{\partial}{\partial \boldsymbol{\Gamma}}\cdot(\dot{\boldsymbol{\Gamma}}f)=0 \tag{5.181}$$

where the first and second terms express a change inside a volume element and a change passing through the surface of a volume element, respectively. The generalized Liouville equation can be rewritten as

$$\frac{\partial f}{\partial t}=-\left(\frac{\partial}{\partial \boldsymbol{\Gamma}}\cdot\dot{\boldsymbol{\Gamma}}\right)f-\dot{\boldsymbol{\Gamma}}\cdot\frac{\partial f}{\partial \boldsymbol{\Gamma}} \tag{5.182}$$

At the same time, the total time derivative of the distribution function $f(\boldsymbol{p},\boldsymbol{q})$ along a phase space trajectory is defined by (see Section 4.1)

$$\frac{df}{dt}=\frac{\partial f}{\partial t}+\dot{\boldsymbol{\Gamma}}\cdot\frac{\partial f}{\partial \boldsymbol{\Gamma}} \tag{5.183}$$

so that, substituting the expression (5.182) for the partial time derivative of the distribution function in previous equation (5.183), the total time derivative of the distribution function can be reformulated as

$$\frac{df}{dt}=-\left(\frac{\partial}{\partial \boldsymbol{\Gamma}}\cdot\dot{\boldsymbol{\Gamma}}\right)f \tag{5.184}$$

In ordinary mechanics, the right-hand side of the above equation is zero, and the famous Liouville theorem $df/dt=0$ holds. This means that the equilibrium distribution function does not change by the time evolution. Otherwise, if the equations of motion are given by (5.171) and (5.172), the derivative (5.184) does not vanish and can be calculated as

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\Gamma}}\cdot\dot{\boldsymbol{\Gamma}}&=\sum_{i=1}^{N}\left(\frac{\partial}{\partial \boldsymbol{p}_i}\cdot\dot{\boldsymbol{p}}_i+\frac{\partial}{\partial \boldsymbol{q}_i}\cdot\dot{\boldsymbol{q}}_i\right)\\&=-\sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{p}_i}\cdot(\gamma\boldsymbol{p}_i)=-\sum_{i=1}^{N}d\gamma-\sum_{i=1}^{N}\boldsymbol{p}_i\cdot\frac{\partial\gamma}{\partial \boldsymbol{p}_i}=-(Nd-1)\gamma\end{aligned} \tag{5.185}$$

where d is the dimensionality of the system and the last part in the above equation has been computed as follows

$$\begin{aligned}\sum_{i=1}^{N}\boldsymbol{p}_i\cdot\frac{\partial\gamma}{\partial \boldsymbol{p}_i}&=\sum_{i=1}^{N}\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}\left\{\left[\sum_{j=1}^{N}\frac{\boldsymbol{p}_j}{m_j}\cdot\mathbf{F}_j(\boldsymbol{q}(t))\right]\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_j^2}{m_j}\right)^{-1}\right\}\\&=-\sum_{i=1}^{N}\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}\left\{\left[\sum_{j=1}^{N}\frac{\boldsymbol{p}_j}{m_j}\cdot\frac{\partial\phi(\boldsymbol{q})}{\partial \boldsymbol{q}_j}\right]\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_j^2}{m_j}\right)^{-1}\right\}\\&=-\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i}{m_i}\cdot\frac{\partial\phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i}\right)\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}\right)^{-1}+2\left(\sum_{j=1}^{N}\frac{\boldsymbol{p}_j}{m_j}\cdot\frac{\partial\phi(\boldsymbol{q})}{\partial \boldsymbol{q}_j}\right)\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}\right)\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_j^2}{m_j}\right)^{-2}\\&=\gamma-2\gamma=-\gamma\end{aligned} \tag{5.186}$$

following

$$\dot{T}(t)=\frac{d}{dt}\left[\frac{1}{gk_b}\sum_{i=1}^{N}m_i\,\boldsymbol{v}_i^2(t)\right]=\frac{2}{gk_b}\sum_{i=1}^{N}m_i\,\boldsymbol{v}_i(t)\cdot\dot{\boldsymbol{v}}_i(t)=0 \tag{5.178}$$

Inserting the Langevin equation of motion (5.174) in the previous expression leads

$$\dot{T}(t)=\frac{2}{gk_b}\left[\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i(\boldsymbol{q}(t))-\gamma\sum_{i=1}^{N}m_i\boldsymbol{v}_i^2(t)\right]=\frac{2}{gk_b}\left[\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i(\boldsymbol{q}(t))-\gamma gk_bT_0\right]=0 \tag{5.179}$$

where the penultimate equivalence is performed using the constraint (5.165). Solving equation (5.179) with respect to γ leads to a particular expression for this constant, given by

$$\gamma=\frac{1}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i(\boldsymbol{q}(t))=-\frac{1}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\frac{\partial\phi(\boldsymbol{q}(t))}{\partial \boldsymbol{q}_i} \tag{5.180}$$

If the conservation of linear momentum is taken explicitly into consideration, equation (5.185) results in $-(Nd-4)\gamma$.
Using the computed expression (5.185), the differential equation (5.184) becomes

$$\frac{df}{dt} = (Nd-1)\gamma f = (Nd-1)\left[\sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t))\right]\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i}\right)^{-1} f \tag{5.187}$$

The expression for γ, given by (5.176) and used in the equation above, can be rewritten in a more suitable way which permits to easily solve the differential equation (5.187), that is

$$\begin{aligned}\gamma &= \left[\sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t))\right]\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i}\right)^{-1} = -\frac{1}{gk_bT_0}\sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial\phi(\boldsymbol{q}(t))}{\partial \boldsymbol{q}_i} \\ &= -\frac{1}{gk_bT_0}\sum_{i=1}^{N} \frac{d\boldsymbol{q}_i}{dt} \cdot \frac{\partial\phi(\boldsymbol{q}(t))}{\partial \boldsymbol{q}_i} = -\frac{1}{gk_bT_0}\left[\frac{d\phi(\boldsymbol{q}(t))}{dt}\right]\end{aligned} \tag{5.188}$$

In this way, substituting this last expression for the coefficient γ in (5.187), the differential equation for the distribution function turns into

$$\frac{df}{dt} = (Nd-1)\gamma f = -\left(\frac{Nd-1}{gk_bT_0}\right)\left(\frac{d\phi(\boldsymbol{q})}{dt}\right)f \qquad \xrightarrow{g=Nd-1} \qquad \frac{df}{dt} = -\frac{1}{k_bT_0}\left(\frac{d\phi(\boldsymbol{q})}{dt}\right)f \tag{5.189}$$

and, as previously said, if the conservation of linear momentum is taken explicitly into consideration, then equation (5.189) results equal to

$$\frac{df}{dt} = (Nd-4)\gamma f = -\left(\frac{Nd-4}{gk_bT_0}\right)\left(\frac{d\phi(\boldsymbol{q})}{dt}\right)f \qquad \xrightarrow{g=Nd-4} \qquad \frac{df}{dt} = -\frac{1}{k_bT_0}\left(\frac{d\phi(\boldsymbol{q})}{dt}\right)f \tag{5.190}$$

Therefore, by choosing a value of degrees of freedom equal to $g = Nd-1$ and $g = Nd-4$ for the cases (5.189) and (5.190), respectively, the simple linear differential equation for the distribution function finally becomes

$$\frac{df}{dt} = -\frac{1}{k_bT_0}\left(\frac{d\phi(\boldsymbol{q})}{dt}\right)f \tag{5.191}$$

The solution of this equation is

$$f(\boldsymbol{p},\boldsymbol{q}) = \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{g}{2}k_bT_0\right)\exp\left[-\left(\frac{\phi(\boldsymbol{q})}{k_bT_0}\right)\right] \tag{5.192}$$

The same result can be demonstrated more rigorously starting from equations of motions written using the extended system method, or using the procedure for non Hamiltonian systems outlined in Section 4.3.2.1 (see Appendix A, Section A.9). The distribution in the coordinate part has the canonical form, whereas that in the momentum space has a Dirac delta function form, because of imposition of a constraint on the kinetic energy. The probability in a volume element along a phase space trajectory is not conserved in the Gaussian constraint method. A conserved quantity is instead

$$f(\boldsymbol{p},\boldsymbol{q})\exp\left[\left(\frac{\phi(\boldsymbol{q})}{k_bT_0}\right)\right] \tag{5.193}$$

Therefore, the probability in a phase space along the trajectory changes in proportion to the entity of the exponential function in (5.192). Equation (5.192) only states that the relative probability of two phase points along a trajectory is expressed in a canonical distribution form. In order to identify $f(\boldsymbol{p},\boldsymbol{q})$ with the equilibrium distribution function in statistical mechanics, the assumption that a trajectory determined by equations (5.171), (5.172) and (5.176) passes through almost all point in an accessible part of a phase space has to be made (ergodicity hypothesis). This assumption is essential for the proof. There are no certainties whether this is true or not in a particular case. However, usually the assumption that the ergodic property is satisfied in a many particle system is underpinned. In this situation, the exact canonical ensemble average is obtained in simulations with the Gaussian constraint method for quantities which are functions of coordinates only.
The integration of the equations of motion (5.171) and (5.172) can be performed using the Liouville approach and the Suzuki-Trotter factorization method, as demonstrated in Appendix A, Section A.10.

5.3.2 Simple velocity rescaling

Velocity rescaling method is actually the first technique proposed by Woodcock[44] (in 1971) to keep the temperature fixed during a simulation, without allowing fluctuations of the temperature itself. This was the first attempt to carry out a simulation in a controlled condition. In this method the instantaneous temperature $T(t)$ of the system is constrained to the reference heat bath value T_0 at every step in the simulation, without allowing for any fluctuations. In this sense, temperature constraining represents a hard boundary condition, in contrast to the soft boundary conditions employed by all other thermostats mentioned in the present treatise.
The simple velocity rescaling thermostat does not properly rely on a modified equation of motion, but uses instead the Newton equations of motion (4.1) which sample the microcanonical ensemble, and then introduced a scaling factor at the end of each simulation step by which the velocities are scaled to return the target temperature. The equations of motion (4.1) generate the following simple Hamilton equations

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} = \boldsymbol{v}_i \qquad i = 1, ..., N \tag{5.194}$$

$$\frac{d\boldsymbol{p}_i}{dt} = \mathbf{F}_i(\boldsymbol{q}(t)) \tag{5.195}$$

The integrator used in this case is basically the same as for the NVE ensemble, namely, the velocity Verlet algorithm

$$\boldsymbol{q}_i(t+\Delta t) = \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t) + \frac{\Delta t^2}{2}\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} \qquad i = 1, ..., N \tag{5.196}$$

$$\boldsymbol{v}_i(t+\Delta t) = \boldsymbol{v}_i(t) + \frac{\Delta t}{2}\left\{\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]}{m_i}\right\} \tag{5.197}$$

with an additional scaling for the velocities at each step,

$$\boldsymbol{v}_{i,\lambda}(t+\Delta t) \equiv \lambda\, \boldsymbol{v}_i(t+\Delta t) \qquad i = 1, ..., N \tag{5.198}$$

introduced in order to satisfy the constraint of constant kinetic energy. Note that all the nuclear velocities are scaled by the same factor λ, which is a constant, different for each time step of the dynamics, but equal (at a give time step) for all the nuclei. The value of the scaling factor is computed at each molecular dynamics step and it is determined from the constraint of constant kinetic energy

$$E_{k,\lambda}(t+\Delta t) \equiv \lambda^2 E_k(t+\Delta t) = E_k(t) = \frac{g}{2}k_b T_0 \tag{5.199}$$

where the scaled kinetic energy is defined by the scaled nuclear velocities and it is fixed equal to a constant in the last equivalence above, so that, writing the explicit expression of the first term on the left hand side of previous equation, and equaling it with the constant kinetic energy value leads to

$$\sum_{i=1}^{N}\frac{m_i}{2}\,\boldsymbol{v}_{i,\lambda}^2(t+\Delta t) = \lambda^2\sum_{i=1}^{N}\frac{m_i}{2}\,\boldsymbol{v}_i^2(t+\Delta t) = \frac{g}{2}k_b T_0 \tag{5.200}$$

Thus the value of the scaling factor λ to be used in (5.198) is given by

$$\lambda = \left\{gk_bT_0\left[\sum_{i=1}^{N} m_i\,\boldsymbol{v}_i^2(t+\Delta t)\right]^{-1}\right\}^{1/2} = \{gk_bT_0[gk_bT(t+\Delta t)]^{-1}\}^{1/2} = \sqrt{\frac{T_0}{T(t+\Delta t)}} \tag{5.201}$$

where the temperature $T(t+\Delta t)$ is computed from the nuclear velocities at time $(t+\Delta t)$, which are obtained through the velocity Verlet algorithm (5.197), in the following way

$$T(t+\Delta t) = \frac{1}{gk_b}\sum_{i=1}^{N} m_i\,\boldsymbol{v}_i^2(t+\Delta t) = \frac{2}{gk_b}\,E_k(t+\Delta t) \tag{5.202}$$

This velocity scaling procedure is very simple and convenient. It is performed outside of the molecular dynamics equations of motion, i.e., after the velocity Verlet algorithm is used, updating nuclear positions and momenta. After this coordinates update, the new temperature associated to the new velocities is computed, and then the velocity rescaling is applied, with the value of the scaling factor λ determined by the new and the target temperatures as in (5.201).
One of the main issue for this method has been outlined by S. C. Harvey *et al.*,[51] who observed that the velocity rescaling performed using standard protocols can systematically change the proportion of total kinetic energy found in motions associated with the various degrees of freedom. Under these conditions, the simulation violates the principle of equipartition of energy, which requires that each degree of freedom has the same mean kinetic energy, affecting both structural and dynamical properties.[51] A particularly pathological form of this problem occurs if one does not periodically remove the net translation of (and rotation about) the center of mass. In this case, almost all of the kinetic energy is converted into these two kinds of motion, producing a system with almost no kinetic energy associated with the internal degrees of freedom, a phenomenon called the *flying ice cube effect*.[51, 52] Because velocity rescaling converts much of the kinetic energy of the system into motion of the center of mass, artifactual behavior can be reduced by periodically removing the translational motion of the center of mass (by subtracting the center of mass velocity $\boldsymbol{v}_{cm}(t+\Delta t)$ to the nuclear velocities) and the rotation about it (by subtracting the rotational velocities $\boldsymbol{v}_{i,r}(t+\Delta t)$ of the nuclei to the nuclear velocities). Unfortunately, this still leaves the simulation subject to violation of energy equipartition, because low-frequency modes can still be excited by repeated velocity reassignments.[51] Furthermore, if only translational motion of the center of mass is removed, the excitation of the zero-frequency rotational mode around the center of mass can cause substantial distortion of the structure, due to the growing centrifugal force. For these reasons, the removal of translational and rotational motions of the entire system can be seen as a desirable, but not sufficient, precaution. Therefore, to avoid pathological cases of equipartition energy principle violation, after that the new velocities $\{\boldsymbol{v}_i(t+\Delta t)\}$ are computed through the velocity Verlet integrator, the system net linear and angular momenta are subtracted in order to remove the net translational and rotational motion (this is performed by removing from the nuclear velocities $\boldsymbol{v}_i(t+\Delta t)$ the linear velocity of the center of mass $\boldsymbol{v}_{cm}(t+\Delta t)$ and the nuclear velocities $\boldsymbol{v}_{i,r}(t+\Delta t)$ induced by the system rotation). Then, the instantaneous kinetic energy $E_k(t+\Delta t)$ corresponding to the new velocities $\{\boldsymbol{v}_i(t+\Delta t)\}$ and its associated temperature $T(t+\Delta t)$ are calculated, so that the new velocities $\{\boldsymbol{v}_i(t+\Delta t)\}$ can be rescaled using the factor (5.201) in order to recover the set of velocities $\{\boldsymbol{v}_{i,\lambda}(t+\Delta t)\}$ associated to the target temperature T_0, that is to say

$$\boldsymbol{v}_i(t+\Delta t) \quad \leftarrow \quad \boldsymbol{v}_{i,\lambda}(t+\Delta t) = \boldsymbol{v}_i(t+\Delta t)\sqrt{\frac{T_0}{T(t+\Delta t)}} \qquad i = 1, ..., N \tag{5.203}$$

This velocity rescaling procedure is outlined in Table 5.3. Therefore, the simple velocity rescaling method consists, at a given time step, in performing the nuclear positions and velocities time propagation with the velocity Verlet algorithm, subtracting the system linear and angular momenta components from the updated nuclear velocities, and then multiplying the new velocities of all the nuclei by the same factor, calculated by constraining the total kinetic energy at each step to be equal to the average kinetic energy E_k at the target temperature. The nuclear velocities are then scaled at the end of each molecular dynamics run by a factor λ defined as

$$\lambda = \sqrt{\frac{T_0}{T(t+\Delta t)}} \tag{5.204}$$

where T_0 is the target temperature and $T(t+\Delta t)$ is the temperature at the end of molecular dynamics run (computed with the new velocities propagated through Verlet algorithm). In this way, a target instantaneous temperature is maintained during the simulation. Since the same factor is used for all the particles, there is neither an effect on constrained bond lengths nor on the center of mass motion. Furthermore, there was also not a consensus how often the scaling procedure should be employed. Some were optimistic and scaled in every time step. Some were skeptical and recommended applying scaling as few times as possible because the trajectory in a phase space becomes discontinuous at the instant of the scaling. The equations used for the integrator implemented in the CRYSTAL code for propagating the phase space point in time with simple velocity rescaling thermostat method is reported in Table 5.3, where it can be seen that the velocities rescaling is applied at each molecular dynamics iteration.

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{v}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 1. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t+\Delta t/2) \leftarrow \boldsymbol{v}_i(t) + \dfrac{\Delta t}{2m_i}\mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 2. Propagator $e^{i\mathcal{L}_q \Delta t}$	:	$\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t) + \Delta t\, \boldsymbol{v}_i(t+\Delta t/2)$
Step 3. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 4. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t+\Delta t) \leftarrow \boldsymbol{v}_i(t+\Delta t/2) + \dfrac{\Delta t}{2m_i}\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 5. Remove COM velocities	:	$\boldsymbol{v}_i(t+\Delta t) \leftarrow \boldsymbol{v}_i(t+\Delta t) - \boldsymbol{v}_{cm}(t+\Delta t) - \boldsymbol{v}_{i,r}(t+\Delta t)$
Step 6. Updating kinetic energy	:	$E_k(t+\Delta t) = \dfrac{1}{2}\sum_{i=1}^{3N} m_i \boldsymbol{v}_i^2(t+\Delta t)$
Step 7. Updating temperature	:	$T(t+\Delta t) = \dfrac{2}{gk_b} E_k(t+\Delta t)$
Step 8. Velocity scaling	:	$\boldsymbol{v}_i(t+\Delta t) \leftarrow \lambda \boldsymbol{v}_i(t+\Delta t)$ with λ given by (5.204)

Table 5.3: Simple velocity scaling thermostat integrator algorithm, applied $\forall\, i = 1, ..., N$.

Another important issue is to understand and compute the form of the phase space distribution generated by the scaling procedure. Indeed, it was not clear for long years since Woodcock article[44] whether this approach can really produce the canonical distribution or not. At first sight, there seems to be a difference between the true canonical trajectory of the system and the one generated by the simple velocity rescaling thermostat. Since no particular equations of motion is associated to the simple velocity scaling technique, this ad hoc temperature control algorithm is difficult to investigate theoretically, and its validity was considered questionable.[46] Nevertheless, it can be demonstrated that the simple scaling scheme is an approximate algorithm to solve the Gaussian constraint method described in Section 5.3.1, leading to the correct canonical distribution in the coordinate space with accuracy of order equal to the time step Δt, if the scaling is carried out at *every* molecular dynamics step.[47] Furthermore, the difference between equations of motion for Woodcock algorithm and the Gaussian thermostat is $O(\Delta t)$. The next paragraph will be dedicated to this demonstration.
Since in the thermodynamic limit the average properties do not depend on the ensemble chosen, even this very simple algorithm can be used to produce useful results.[53] However, for small systems or when the observables of interest are dependent on the fluctuations rather than on the averages, this method cannot be employed. Its usage can also lead to simulation artifacts, as for example the previously mentioned flying ice cube effect, which can be partially compensated and minimized by removing net translation and rotational momentum, but cannot be completely avoided.[51, 52]

Relation between the Gaussian constraint method and velocity scaling algorithm The differential equation which can approximately describes the velocity scaling algorithm is here analyzed and discussed. The velocity scaling algorithm consists of two finite difference equations (in a velocity Verlet form) (5.196) and (5.197), and finally a scaling equation (5.203) for the nuclear velocities. The nuclear velocities at time t are given by (5.197)

$$\boldsymbol{v}_i(t) = \boldsymbol{v}_i(t+\Delta t) - \frac{\Delta t}{2}\left\{ \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]}{m_i} \right\} \tag{5.205}$$

An acceleration is expressed by a difference of the nuclear velocity after the scaling at time $(t+\Delta t)$ with the nuclear velocity before the scaling at time t, expressed by (5.205), so that

$$\frac{d^2\boldsymbol{q}_i}{dt^2} = \frac{d\boldsymbol{v}_i}{dt} = \frac{\lambda \boldsymbol{v}_i(t+\Delta t) - \boldsymbol{v}_i(t)}{\Delta t} = \left(\frac{\lambda - 1}{\Delta t}\right)\boldsymbol{v}_i(t+\Delta t) + \frac{1}{2}\left\{\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]}{m_i}\right\} \tag{5.206}$$

In the final line of the above equation, the force terms are separated into an ordinary force and an additional part which is proportional to $(\lambda - 1)$ and to the nuclear velocity. The scaling factor defined in equation (5.201) can be computed as

$$\begin{aligned}
\lambda &= \left\{\frac{1}{gk_bT_0}\sum_{i=1}^{N} m_i\,\boldsymbol{v}_i^2(t+\Delta t)\right\}^{-1/2} \\
&= \left\{\frac{1}{gk_bT_0}\sum_{i=1}^{N} m_i\left[\boldsymbol{v}_i(t) + \frac{\Delta t}{2}\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\Delta t}{2}\frac{\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]}{m_i}\right]^2\right\}^{-1/2} \\
&= \left\{\frac{1}{gk_bT_0}\sum_{i=1}^{N} m_i\left[\boldsymbol{v}_i^2(t) + \frac{\Delta t}{m_i}\,\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t)] + \frac{\Delta t}{m_i}\,\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)] + O((\Delta t)^2)\right]\right\}^{-1/2} \\
&= \left\{\frac{1}{gk_bT_0}\sum_{i=1}^{N}\left(m_i\boldsymbol{v}_i^2(t) + \Delta t\,\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t)] + \Delta t\,\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]\right) + O((\Delta t)^2)\right\}^{-1/2}
\end{aligned} \tag{5.207}$$

The last term in the parenthesis is only indicated as a quantity of order $(\Delta t)^2$. The kinetic energy at time t is assumed to be $(g/2)k_bT_0$, so that the previous expression becomes

$$\lambda = \left[1 + \frac{\Delta t}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t)] + \frac{\Delta t}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)] + O((\Delta t)^2)\right]^{-1/2} \tag{5.208}$$

At this point, the forces acting on nuclei at time $t + \Delta t$ can be written as a Taylor expansion in the following way

$$\begin{aligned}
\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)] \equiv \mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}] &= \mathbf{F}_i\left[\left\{\boldsymbol{q}_i(t) + \Delta t\left(\boldsymbol{v}_i(t) + \frac{\Delta t}{2}\,\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i}\right)\right\}\right] \\
&= \mathbf{F}_i[\boldsymbol{q}(t)] + \Delta t\left(\boldsymbol{v}_i(t) + \frac{\Delta t}{2}\,\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i}\right)\dot{\mathbf{F}}_i[\boldsymbol{q}(t)] + O((\Delta t)^2) \\
&= \mathbf{F}_i[\boldsymbol{q}(t)] + O(\Delta t)
\end{aligned} \tag{5.209}$$

so that, substituting this expression in (5.208) and comparing the result with the formula of γ coefficient defined in (5.176) for the Gaussian thermostat equations of motion, leads to a scaling factor equal to

$$\lambda = \left[1 + \frac{2\Delta t}{gk_bT_0}\sum_{i=1}^{N}\boldsymbol{v}_i(t)\cdot\mathbf{F}_i[\boldsymbol{q}(t)] + O((\Delta t)^2)\right]^{-1/2} = [1 + 2\gamma\Delta t + O((\Delta t)^2)]^{-1/2} \tag{5.210}$$

Finally, using the binomial series for the expansion of the inverse square root,[7] the previous expression becomes

$$\lambda = [1 + 2\gamma\Delta t + O((\Delta t)^2)]^{-1/2} = 1 - \gamma\Delta t + O((\Delta t)^2) \tag{5.211}$$

Substituting this formula in the equation of motion (5.206), and adopting at the same time the approximation (5.209) for the forces acting on the nuclei at time $(t + \Delta t)$, equation (5.206) can be rewritten as

$$\begin{aligned}
\frac{d^2\boldsymbol{q}_i}{dt^2} = \frac{d\boldsymbol{v}_i}{dt} &= -\gamma\boldsymbol{v}_i(t+\Delta t) + \frac{1}{2}\left\{\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + O((\Delta t)^2)\right\} \\
&= \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} - \gamma\boldsymbol{v}_i(t) - \gamma\frac{\Delta t}{2}\left\{\frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} + \frac{\mathbf{F}_i[\boldsymbol{q}(t+\Delta t)]}{m_i}\right\} + O((\Delta t)^2) \\
&= \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} - \gamma\boldsymbol{v}_i(t) + O(\Delta t)
\end{aligned} \tag{5.212}$$

[7] Binomial series...

where the passage from the first to the second line has been performed using the velocity Verlet discretized equation of motion (5.197) for the nuclear velocities $\boldsymbol{v}_i(t+\Delta t)$. Therefore, the differential equation which characterizes approximately the simple velocity scaling algorithm has finally been found, together with an expression for the undetermined multiplier γ, that is

$$\frac{d^2\boldsymbol{q}_i}{dt^2} = \frac{d\boldsymbol{v}_i}{dt} = \frac{\mathbf{F}_i[\boldsymbol{q}(t)]}{m_i} - \gamma \boldsymbol{v}_i(t) + O(\Delta t) \qquad \text{where} \quad \gamma = \frac{1}{g k_b T_0} \sum_{i=1}^{N} \boldsymbol{v}_i(t) \cdot \mathbf{F}_i[\boldsymbol{q}(t)] \tag{5.213}$$

The leading term in equation (5.213) is equivalent to that in the Gaussian constraint method, see equations (5.171), (5.172) and (5.176). If the scaling is carried out in every time step, and if the unit time step Δt is chosen to a reasonably small value, the simulation with the velocity scaling algorithm gives equal results with those in the Gaussian constraint method, and the distribution function sampled in the coordinate space is the canonical distribution, see equation (5.192). The error in this approximation is of order Δt. It should be noted that the accuracy is one order less than that of the ordinary Verlet algorithm, which is $O((\Delta t)^2)$.

5.3.3 Berendsen thermostat

The method introduced by H. J. C. Berendsen *et al.*[54] is a constant temperature technique based on the modification of a Langevin equation. The term for the local disturbance from random noise in the standard generalized Langevin equation is eliminated, so as to mimic the global energy exchange between a ionic system and an external heat bath, while neglecting local coupling by random noise. Since only the global coupling of the system with a heat bath remains, this technique results in relatively modest temperature controlling, leading to an approximation of the perturbation that would occur in an ideal physical non-equilibrium system. This method is called weak coupling or Berendsen thermostat. In the following, the main equations that describes this temperature controlling technique are outlined.
Consider a system of N point particles (nuclei) in d dimensions described using Cartesian coordinates, where the number of degrees of freedom is equal to Nd, that also corresponds to the number of coordinates in the phase space required to define the physical system. The coupling of this system to a heat bath with fixed reference temperature T_0 can be taken into account by inserting stochastic and friction terms in the equations of motions, yielding a Langevin equation of the form

$$m_\alpha \dot{v}_\alpha(t) = F_\alpha(t) - m_\alpha \gamma_\alpha v_\alpha(t) + R_\alpha(t) \qquad \alpha = 1, ..., Nd \tag{5.214}$$

where F_α is the external systematic force (computed through the Hellman-Feynman theorem for a system of N nuclei), the second term on the right hand side is a viscous friction force with γ_α a positive friction coefficient[8] and the last term R_α is a Gaussian stochastic variable force with zero mean, fixed intensity and zero correlation with the velocities for different degrees of freedom, namely,

$$\langle R_\alpha(t) \rangle = 0 \qquad \langle R_\alpha(t) R_\beta(t+\tau) \rangle = 2 m_\alpha \gamma_\alpha \, k_b \, T_0 \, \delta(\tau) \delta_{\alpha\beta} \qquad \langle v_\alpha(t) R_\alpha(t+\tau) \rangle = 0 \tag{5.216}$$

where the physical assumption about the intensity of the random forces R_α given in the second expression in (5.216) has been made on the basis of the fluctuation-dissipation theorem, that gives the

[8] If the last term $R_\alpha(t)$ in (5.214), which represents a stochastic force, is removed and a single friction coefficient is used for all the nuclei, then γ_α loses its physical meaning of a friction coefficient and is no longer restricted to positive values. A positive value indicates that heat flows from the system to the heat bath. A negative value indicates a heat flowing in the opposite direction. In the case the stochastic force $R_\alpha(t)$ is avoided, equation (5.214) becomes

$$m_\alpha \dot{v}_\alpha(t) = F_\alpha(t) - m_\alpha \gamma_\alpha v_\alpha(t) \qquad \alpha = 1, ..., Nd \tag{5.215}$$

An important thing to note here is that if equation (5.215) was applied to the real nuclear velocities $\{\tilde{\boldsymbol{v}}_i\}$ instead of the internal nuclear velocities $\{\boldsymbol{v}_i\}$, the linear and angular momenta of the system would not be conserved (unless they exactly vanish). Any algorithm relying on the equation of motion(5.215) is smooth (i.e., generates a continuous velocity trajectory) and deterministic. It is also time-reversible if γ_α is antisymmetric with respect to time-reversal operations.
The advantages of deterministic algorithms are that (i) the results can be exactly reproduced (in the absence of numerical errors), and (ii) there are well-defined conserved quantities (constants of the motion). Considering a given microstate, time-reversibility is achieved if the change $dt \to -dt$ (leading in particular to $r_\alpha \to r_\alpha$, $v_\alpha \to -v_\alpha$ and $\dot{v}_\alpha \to \dot{v}_\alpha$) leaves the equation of motion for the coordinates unaltered (while the velocities are reversed). Clearly, this condition is satisfied for equation (5.215) only if the corresponding change for γ_α is $\gamma_\alpha \to -\gamma_\alpha$.

relationship between a fluctuating force on some degree of freedom and the damping coefficient that determines dissipation in this degree of freedom (see Appendix A, Section A.4).
The relationships (5.216) can be physically interpreted as follows: (i) the function $R_\alpha(t)$ has no directional preference; (ii) the function $R_\alpha(t)$ is a Gaussian random function of time with zero mean and variance $2m_\alpha\gamma_\alpha\, k_b\, T_0$ for all degrees of freedom interacting with the heat bath; (iii) the force $R_\alpha(t)$ acting on degree of freedom α is uncorrelated with the force $R_\beta(t)$ which acts on another degree of freedom β; (iv) the instantaneous value of $R_\alpha(t)$ is not affected by its preceding values, i.e., the function $R_\alpha(t)$ is uncorrelated with its time history, that is to say, the force component $R_\alpha(t)$ is uncorrelated with any component $R_\beta(t')$ unless $\alpha = \beta$ and $t' = t$; (v) the random force components $R_\alpha(t)$ ($\alpha = 1, ..., Nd$) are uncorrelated with the nuclear velocities $v_\alpha(t')$ and systematic forces $F_\alpha(t')$ at previous times $t' < t$.
In practice, the random force $R_\alpha(t)$ can be sampled at each molecular dynamic step of a numerical simulation as a random Gaussian variable with zero mean and variance (mean square amplitude) $2m_\alpha\gamma_\alpha\, k_b\, T_0$. The sampling procedure is performed independently for each degree of freedom exposed to the thermal noise. Furthermore, samples for two successive time steps are evaluated independently from each other.
Equation (5.214) corresponds physically to a system of particles that experience viscous friction and are subject to frequent collisions with light particles that form an ideal gas at temperature T_0. Mathematically, the form of (5.214) corresponds to a standard Langevin equation with a viscous friction force and a stochastic external force with the property (5.216). These two forces are intended to represent coupling to an external heat bath and scale the atomic velocities during the numerical simulation to add or remove energy from the system as desired. The damping constants γ_α in the friction and stochastic terms determine the strength of the coupling to the thermal bath. Through the Langevin equation (5.214) the system couples not only globally to a heat bath, but is also locally subjected to random noise. If only a global coupling with minimal local disturbance wants to be imposed on the system, equation (5.214) has to be modified in such a way that only the global coupling remains.
The Langevin equation of motion (5.214) is smooth, non-deterministic and time-irreversible. It can be rewritten as

$$\dot{v}_\alpha(t) = -\gamma_\alpha v_\alpha(t) + \frac{1}{m_\alpha}[F_\alpha(t) + R_\alpha(t)] \qquad \alpha = 1, ..., Nd \tag{5.217}$$

so that it is easily recognizable in the form of a first order linear differential equation with variable coefficients, whose solution is given by

$$v_\alpha(t) = v_\alpha(0)e^{-\gamma_\alpha t} + \int_0^t \frac{1}{m_\alpha}[F_\alpha(t) + R_\alpha(t)]\, e^{-\gamma_\alpha(t-t')}\, dt' \qquad \alpha = 1, ..., Nd \tag{5.218}$$

where the first term is just the solution of the correspondent homogeneous differential equation with an exponential decay due to friction, and the second term gives the extra velocity produced by the external force and the random noise acting on the nuclear particles.
For convenience, in the following the friction constants are chosen to be equal for all the particles: $\gamma_\alpha = \gamma$. This is a matter of choice, different classes of degrees of freedom can in principle be coupled to the bath with different friction constants.
As previously mentioned, the idea behind the Berendsen thermostat is to modify the Langevin equation of motion (5.214) in the sense of removing the local temperature coupling through stochastic collisions (random noise), neglecting stochastic fluctuations on the microscopic timescale, while retaining the global coupling (principle of least local perturbation). In various applications, only the global thermodynamic behavior of the system is of importance. Then, it is computationally effective to eliminate the local random noise $R_\alpha(t)$ in the Langevin equation (5.214), and to characterize the system/heat bath coupling via a time-dependent damping term. This damping term can be introduced on the basis of the requirement that the new equation of motion must yield the same averaged behavior of the system kinetic and total energy for a given target temperature T_0.
In the following, the main procedure necessary to derive the Berendsen equations of motion is outlined. First of all, the time derivative of the nuclear kinetic energy E_k can be written in the form

$$\frac{dE_k}{dt} = \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} m_\alpha [v_\alpha^2(t + \Delta t) - v_\alpha^2(t)] \right\} \tag{5.219}$$

and defining the velocity difference as

$$\Delta v_\alpha = v_\alpha(t + \Delta t) - v_\alpha(t) \tag{5.220}$$

the previous expression for the kinetic energy derivative can be rearranged in the following way

$$\frac{dE_k}{dt} = \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} m_\alpha [2v_\alpha(t)\Delta v_\alpha + \Delta v_\alpha^2] \right\} \tag{5.221}$$

At the same time, according to the Langevin equation (5.214) with the additional constraint imposed on the constants γ_α which are chosen to be equal for all particles, i.e. $\gamma_\alpha = \gamma \ \forall \alpha = 1, ..., Nd$, the change in nuclear velocity over a short time interval can be approximated as

$$\begin{aligned} \Delta v_\alpha = v_\alpha(t+\Delta t) - v_\alpha(t) &= \frac{1}{m_\alpha} \int_t^{t+\Delta t} [F_\alpha(t') - m_\alpha \gamma v_\alpha(t') + R_\alpha(t')] dt' \\ &\approx \frac{1}{m_\alpha} [F_\alpha(t)\Delta t - m_\alpha \gamma v_\alpha(t)\Delta t] + \frac{1}{m_\alpha} \int_t^{t+\Delta t} R_\alpha(t')\, dt' \end{aligned} \tag{5.222}$$

Then, the substitution of equation (5.222) in equation (5.221) can be performed, separately considering the first and the second terms of the summation in equation (5.221). The first term is given by

$$\begin{aligned} \lim_{\Delta t \to 0} \left\{ \frac{1}{\Delta t} \sum_{\alpha=1}^{Nd} m_\alpha v_\alpha(t) \Delta v_\alpha \right\} &\approx \sum_{\alpha=1}^{Nd} [F_\alpha(t) v_\alpha(t) - m_\alpha \gamma v_\alpha^2(t)] + \lim_{\Delta t \to 0} \left\{ \frac{1}{\Delta t} \sum_{\alpha=1}^{Nd} \int_t^{t+\Delta t} v_\alpha(t) R_\alpha(t')\, dt' \right\} \\ &= -\dot{E}_{e,0}(\boldsymbol{\xi}(t)) - 2\gamma E_k(t) + \lim_{\Delta t \to 0} \left\{ \frac{Nd}{\Delta t} \int_t^{t+\Delta t} \langle v_\alpha(t) R_\alpha(t') \rangle \, dt' \right\} \\ &= -\dot{E}_{e,0}(\boldsymbol{\xi}(t)) - 2\gamma E_k(t) \end{aligned} \tag{5.223}$$

where ξ are the electronic degrees of freedom (i.e. space and spin electronic coordinates) and the time derivative of the potential energy has been computed as

$$\dot{E}_{e,0}(\boldsymbol{\xi}(t)) = \frac{dE_{e,0}(\boldsymbol{\xi}(t))}{dt} = \sum_{\alpha=1}^{Nd} \nabla_\alpha E_{e,0}(\boldsymbol{\xi}(t)) \frac{dq_\alpha}{dt} = -\sum_{\alpha=1}^{Nd} F_\alpha(t)\, v_\alpha(t) \tag{5.224}$$

The current kinetic energy $E_k(t)$ in (5.223) has the form

$$E_k(t) = \frac{1}{2} \sum_{\alpha=1}^{Nd} m_\alpha v_\alpha^2(t) \tag{5.225}$$

and the ensemble average, given by

$$\frac{1}{Nd} \sum_{\alpha=1}^{Nd} v_\alpha(t) R_\alpha(t') = \langle v_\alpha(t) R_\alpha(t') \rangle = 0 \tag{5.226}$$

has been used to set equal to zero the argument of the last limit in equation (5.223).
Then, according to equation (5.222), the second term in equation (5.221) becomes

$$\begin{aligned} \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} m_\alpha \Delta v_\alpha^2 \right\} \approx & \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} \frac{1}{m_\alpha} F_\alpha^2(t) \Delta t^2 \right\} + \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} m_\alpha (\gamma v_\alpha(t))^2 \Delta t^2 \right\} \\ & - \lim_{\Delta t \to 0} \left\{ \frac{1}{\Delta t} \sum_{\alpha=1}^{Nd} \gamma F_\alpha(t) v_\alpha(t) \Delta t^2 \right\} + Nd \int_t^{t+\Delta t} \langle \frac{F_\alpha(t) R_\alpha(t')}{m_\alpha} \rangle \, dt' \\ & - \gamma Nd \int_t^{t+\Delta t} \langle v_\alpha(t) R_\alpha(t') \rangle \, dt' \\ & + \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} \sum_{\beta=1}^{Nd} m_\alpha \frac{1}{m_\alpha m_\beta} \int_t^{t+\Delta t} \int_t^{t+\Delta t} R_\alpha(t') R_\beta(t'')\, dt'\, dt'' \right\} \\ = & \lim_{\Delta t \to 0} \left\{ \frac{1}{2\Delta t} \sum_{\alpha=1}^{Nd} \sum_{\beta=1}^{Nd} \frac{1}{m_\beta} \int_t^{t+\Delta t} \int_t^{t+\Delta t} R_\alpha(t') R_\beta(t'')\, dt'\, dt'' \right\} \end{aligned} \tag{5.227}$$

where the first five terms are equal to zero: in the first three the application of the limit $\Delta t \to 0$ makes the terms equal to zero, while in the fourth and fifth terms the average values are both zero since the stochastic forces $R_\alpha(t')$ ($\alpha = 1, ..., Nd$) are all uncorrelated with the nuclear velocities $v_\alpha(t)$ and systematic forces $F_\alpha(t)$ at previous times $t < t'$. Therefore, only the sixth term remains, so that, thanks to the equation (5.222), the second term in (5.221) can be written as

$$\begin{aligned}
\lim_{\Delta t\to 0}\left\{\frac{1}{2\Delta t}\sum_{\alpha=1}^{Nd} m_\alpha \Delta v_\alpha^2\right\} &\approx \lim_{\Delta t\to 0}\left\{\frac{1}{2\Delta t}\sum_{\alpha=1}^{Nd}\sum_{\beta=1}^{Nd}\frac{1}{m_\beta}\int_t^{t+\Delta t}\int_t^{t+\Delta t} R_\alpha(t')R_\beta(t'')\,dt'\,dt''\right\} \\
&= \lim_{\Delta t\to 0}\left\{\frac{Nd}{2\Delta t}\sum_{\beta=1}^{Nd}\frac{1}{m_\beta}\int_t^{t+\Delta t}\int_t^{t+\Delta t} \langle R_\alpha(t')R_\beta(t'')\rangle\,dt'\,dt''\right\} \\
&= \lim_{\Delta t\to 0}\left\{\frac{Nd}{\Delta t}\sum_{\beta=1}^{Nd}\frac{1}{m_\beta}\int_t^{t+\Delta t}\int_t^{t+\Delta t} m_\alpha\gamma\, k_b\, T_0\,\delta(t'-t'')\delta_{\alpha\beta}\,dt'\,dt''\right\} \\
&= \lim_{\Delta t\to 0}\left\{\frac{Nd}{\Delta t}\gamma\, k_b\, T_0\int_t^{t+\Delta t} dt'\right\} = Nd\gamma\, k_b\, T_0\lim_{\Delta t\to 0}\int_t^{t+\Delta t} dt' \\
&= Nd\gamma k_b T_0
\end{aligned} \tag{5.228}$$

Finally, summing up the expressions (5.223) and (5.228), the time derivative of the kinetic energy (5.221) can be approximated as

$$\frac{dE_k}{dt} = \sum_{\alpha=1}^{Nd} F_\alpha(t)v_\alpha(t) + 2\gamma\left[\frac{Nd}{2}\,k_bT_0 - E_k(t)\right] \tag{5.229}$$

where the first term on the right hand side equals minus the time derivative of the potential energy, as demonstrated in (5.224), and the second term is an additional quantity describing the global coupling to the heat bath. In terms of temperature, using the relation $E_k = Ndk_bT/2$, this extra term can be rearranged as

$$\frac{dE_k}{dt} = \sum_{\alpha=1}^{Nd} F_\alpha(t)v_\alpha(t) + \frac{Nd}{2}\,k_b\left(\frac{dT}{dt}\right)_{\text{bath}} \tag{5.230}$$

where the last term has been written as a function of the time rate of temperature changing, which is represented by the first order differential equation

$$\left(\frac{dT}{dt}\right)_{\text{bath}} \equiv \frac{dT(t)}{dt} = 2\gamma[T_0 - T(t)] = \frac{1}{\tau_t}[T_0 - T(t)] \tag{5.231}$$

so that the time constant τ_t of this coupling is equal to $\tau_t = (2\gamma)^{-1}$, leading to a temperature deviation that decays exponentially with a time constant τ_t. In order to recover the meaning of the constant τ_t, equation (5.231) can be rewritten in such a way that its character of a non-homogeneous first order linear ordinary differential equation can be easily recognized and solved as

$$\frac{dT(t)}{dt} = -\frac{1}{\tau_t}[T(t) - T_0] \quad \to \quad T(t) = T_0\, e^{-t/\tau_t} + T_0(1 + e^{-t/\tau_t}) \tag{5.232}$$

Therefore, the quantity τ_t (often called the *rise time* or *relaxation time* of the thermostat) describes the strength of the coupling of the system to a hypothetical heat bath and it gives the coupling constant that represents the characteristic time of equilibration of the system with the thermal bath. The larger the rise time τ_t, the weaker the coupling, i.e., the longer the time that the system employs to achieve a given target temperature T_0 from the current one $T = T(t)$. Equation (5.229) can be rewritten in a form that recovers the expression for the derivative of the Hamiltonian of the system, starting from

$$\dot{E}_k = -\dot{E}_{e,0}(\boldsymbol{\xi}(t)) + Nd\gamma k_b T_0 - 2\gamma E_k(t) \tag{5.233}$$

and rearranging this relationship to obtain

$$\frac{d}{dt}[E_k(t) + E_{e,0}(\boldsymbol{\xi}(t))] = \dot{\mathcal{H}} = Nd\gamma k_b T_0 - 2\gamma E_k(t) = Nd\gamma k_b[T_0 - T(t)] \tag{5.234}$$

where $\mathcal{H}$ is the system Hamiltonian, $T(t)$ is the current system temperature viewed as a function of time and the relation $E_k(t) = Ndk_bT(t)/2$ between the instantaneous kinetic energy and the current temperature has been used to obtain the final expression. Equation (5.234) must be satisfied by the equation of motion (5.214) after imposing $\gamma_\alpha = \gamma \ \forall \alpha = 1, ..., Nd$ and the elimination of the random force term $R_\alpha(t)$. In particular, returning to equation (5.214), it is clear that the global additional temperature coupling of equation (5.231) is accomplished by the equations of motion

$$m_\alpha \dot{v}_\alpha(t) = F_\alpha(t) + m_\alpha \gamma \left[\frac{T_0}{T(t)} - 1 \right] v_\alpha(t) \qquad \alpha = 1, ..., Nd \tag{5.235}$$

without adding local stochastic terms, because the equations (5.235), known as Berendsen equations of motion, satisfy the relation (5.229) and, equivalently, the expression (5.234). Indeed, multiplying the equations (5.235) by the velocity $v_\alpha(t)$ and summing over all the degrees of freedom gives

$$\sum_{\alpha=1}^{Nd} m_\alpha \dot{v}_\alpha(t) v_\alpha(t) = \sum_{\alpha=1}^{Nd} F_\alpha(t) v_\alpha(t) + \gamma \left[\frac{T_0}{T(t)} - 1 \right] \sum_{\alpha=1}^{Nd} m_\alpha v_\alpha^2(t) \tag{5.236}$$

and using the expression in (5.225) for the kinetic energy E_k the previous relation becomes

$$\frac{dE_k}{dt} = \sum_{\alpha=1}^{Nd} m_\alpha \dot{v}_\alpha(t) v_\alpha(t) = \sum_{\alpha=1}^{Nd} F_\alpha(t) v_\alpha(t) + \gamma \left[\frac{T_0}{T(t)} - 1 \right] 2E_k(t) = \sum_{\alpha=1}^{Nd} F_\alpha(t) v_\alpha(t) + 2\gamma \left[\frac{Nd}{2} k_b T_0 - E_k(t) \right]$$

finally recovering equation (5.229), which describes the time derivative of the kinetic energy of the system. Thus, from an energetic point of view, the Berendsen equation (5.235) is equivalent to the original Langevin equation (5.214) in which the condition on the damping constants $\gamma_\alpha = \gamma \ \forall \alpha = 1, ..., Nd$ has been imposed, for applications to multiparticle systems.
Thus, the equations of motion (5.214) describing the system of N particles coupled with a thermal bath has been modified, using the conditions (5.216) for the random force $R_\alpha(t)$ and damping constants $\gamma_\alpha = \gamma \ \forall \alpha = 1, ..., Nd$ equal for all the particles, with a resultant equation of motion given by (5.235), whose discretization (using simple Euler discrete formulation) leads to

$$\frac{v_\alpha(t + \Delta t) - v_\alpha(t)}{\Delta t} = \frac{F_\alpha(t)}{m_\alpha} + \gamma \left[\frac{T_0}{T(t)} - 1 \right] v_\alpha(t) = \frac{F_\alpha(t)}{m_\alpha} + \frac{1}{2\tau_t} \left[\frac{T_0}{T(t)} - 1 \right] v_\alpha(t) \qquad \alpha = 1, ..., Nd$$

where the relation $\gamma = (2\tau_t)^{-1}$ has been used. Therefore, the velocity of the α-th particle at time $(t + \Delta t)$ can be computed with the formula

$$v_\alpha(t + \Delta t) = \frac{F_\alpha(t)}{m_\alpha} \Delta t + v_\alpha(t) \left[1 + \frac{\Delta t}{2\tau_t} \left(\frac{T_0}{T(t)} - 1 \right) \right] \qquad \alpha = 1, ..., Nd \tag{5.237}$$

that represents a proportional scaling of the velocities per time step in the algorithm from v_α to $\lambda_p v_\alpha$ with λ_p given by (up to first order)

$$\lambda_p = 1 + \frac{\Delta t}{2\tau_t} \left[\frac{T_0}{T(t)} - 1 \right] \tag{5.238}$$

Otherwise, another strategy for the updating of the velocity at each time step comes from the discretization of equation (5.231), which rules the time variation of the system temperature, so that

$$\frac{T(t + \Delta t) - T(t)}{\Delta t} = \frac{1}{\tau_t} [T_0 - T(t)] \quad \rightarrow \quad T(t + \Delta t) = T(t) \left[1 + \frac{\Delta t}{\tau_t} \left(\frac{T_0}{T(t)} - 1 \right) \right] = \lambda_t \, T(t) \tag{5.239}$$

In this case, the system temperature at time $(t + \Delta t)$ is scaled, with respect to the same temperature at the previous time step, by a factor λ_t defined as

$$T(t + \Delta t) = \lambda_t \, T(t) \qquad \text{with} \quad \lambda_t = 1 + \frac{\Delta t}{\tau_t} \left[\frac{T_0}{T(t)} - 1 \right] \tag{5.240}$$

so that the correspondent velocities has to be scaled at each time step by the factor λ_b defined as

$$v_\alpha(t+\Delta t) = \lambda_b\, v_\alpha(t) \qquad \text{with} \quad \lambda_b = \left[1 + \frac{\Delta t}{\tau_t}\left(\frac{T_0}{T(t)} - 1\right)\right]^{1/2} \qquad \alpha = 1, ..., Nd \tag{5.241}$$

Despite the usefulness of the Berendsen equation, its numerical integration has never raised much attention. The system is not a Hamiltonian system, so one cannot directly use symplectic integrators, which has been shown to be efficient in a variety of studies. This was the main reason to hamper the development of an efficient numerical integration based on a theoretically clear foundation. Most of the integration algorithms for the Berendsen equations of motion are thus based on heuristic approaches, obtained by a combination of the leapfrog or Verlet method and the velocity scaling, which may give $O(\Delta t)$ accuracy. However, these approaches lack both the time-reversibility feature and a protocol to attain higher accuracy. Practical reasons may also have prevented the development of an efficient integrator for this equations of motion. That is, one often supposes that it is sufficient to have a good temperature controllability of the target physical system and that the accuracy is of second importance. However, there are cases where the temperature control is good but a large numerical error is accumulated. Thus a method to capture the error is necessary to get physically correct results. Fukuda *et al.*[55] propose a time-reversible (symmetric) integrator for the Berendsen thermostat, where the equations of motion are extended so as to have a time invariant function. This device is based on the techniques previously developed for non-Hamiltonian systems. From the time reversibility, the integrator map preserves the reversible feature that the original ordinary differential equation has, thus contributing to the accuracy of the integration. By monitoring the value of the constructed invariant function, numerical integration on the extended space can be done without destroying the original solutions of the ordinary differential equations and will detect the integration errors that cannot be detected by simple temperature monitoring.[55] However, in the basic module of Molecular Dynamics in the CRYSTAL code, the integrator used for the Berendsen thermostat does not rely on a mathematical justification, but it is the same used in standard practical simulations and implemented in many other computational codes. This heuristic algorithm is based on the velocity Verlet scheme with an additional nuclear velocity scaling procedure, as reported in Table 5.4. The integration of the Berendsen equations of motion are thus performed using the velocity Verlet algorithm in the same form reported in Table 5.1 as for the equations of motion characterizing the microcanonical ensemble, with the additional operation of a velocity rescaling (5.241) at the end of the integration algorithm.
Therefore, the Berendsen thermostat takes the Langevin equation, removes the stochastic term, and modifies the frictional dissipative force to yield similar temperature time dependence as with the stochastic term present. In practice, this is implemented as a smoother version of the simple velocity rescaling technique, in which the velocities of all particles are rescaled at the end of each timestep by a factor λ_b given by

$$\lambda_b = \left[1 + \frac{\Delta t}{\tau_t}\left(\frac{T_0}{T(t+\Delta t)} - 1\right)\right]^{1/2} \tag{5.242}$$

where τ_t represents a time damping constant and the instantaneous temperature to be considered is that computed using the velocities $v_\alpha(t+\Delta t)$ resultant from the velocity Verlet algorithm, and are accordingly expressed by $T(t+\Delta t)$.
Since the value of the relaxation time constant τ_t greatly influences the dynamics of the system, a general rule has to be find for the setting of a good value for τ_t constant. First of all, it has to be noted that the value of τ_t determines the ensemble explored by Berendsen equations of motion (5.235). Unfortunately, the exact statistical mechanical ensemble for Berendsen thermostat has been unknown for many years. In some studies employing the thermostat, it is reported that fluctuations of thermodynamic properties are coincident with neither those in the canonical ensemble nor microcanonical ensemble. Generally, ensemble averages are identical in different ensembles (e.g. the microcanonical and the canonical ones). However, fluctuations of thermodynamic properties are different, and fluctuation formulas associated with thermodynamic derivatives such as $c_v(= \partial E/\partial T)$ are also dependent on the statistical ensemble. Therefore, it is clear that the statistical ensemble associated with the Berendsen thermostat is neither the canonical nor microcanonical. As a consequence, there is no conserved quantity with which to control the accuracy of the algorithm.[56] In an interesting study, T. Morishita[57] deeply analyzes for the first time the problem and computes an approximate expression of the equilibrium distribution function

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{v}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 1. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t+\Delta t/2) \leftarrow \boldsymbol{v}_i(t) + \dfrac{\Delta t}{2m_i}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 2. Propagator $e^{i\mathcal{L}_q \Delta t}$	:	$\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t) + \Delta t\,\boldsymbol{v}_i(t+\Delta t/2)$
Step 3. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 4. Propagator $e^{i\mathcal{L}_v \Delta t/2}$	:	$\boldsymbol{v}_i(t+\Delta t) \leftarrow \boldsymbol{v}_i(t+\Delta t/2) + \dfrac{\Delta t}{2m_i}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 5. Updating kinetic energy	:	$E_k(t+\Delta t) = \dfrac{1}{2}\sum_{i=1}^{3N} m_i \boldsymbol{v}_i^2(t+\Delta t)$
Step 6. Updating temperature	:	$T(t+\Delta t) = \dfrac{2}{gk_b}\,E_k(t+\Delta t)$
Step 7. Velocity scaling	:	$\boldsymbol{v}_i(t+\Delta t) \leftarrow \lambda_b \boldsymbol{v}_i(t+\Delta t)$ with λ_b given by (5.242)

Table 5.4: Berendsen thermostat integrator algorithm, applied $\forall\, i = 1, ..., N$.

in the configurational space for the Berendsen thermostat. This distribution function is dependent on the parameter τ_t and the corresponding statistical ensemble can change from the canonical to the microcanonical according to the value of τ_t. It is easily recognized that with large τ_t, in the limit $\tau_t \to \infty$ when energy exchange between the system and the heat bath are suppressed, the coefficient γ in (5.235) approaches zero and equations (5.235) become the standard Newton equations of motion (i.e., the microcanonical ensemble is recovered). On the contrary, energy exchange between a ionic system and a heat bath becomes large with small value of τ_t. In particular, when $\tau_t = \Delta t$ (that theoretically corresponds to the limit $\tau_t \to 0$), the scaling factor λ_b in (5.241) becomes equal to the Woodcock[44] simple velocity rescaling factor (5.204) (see Section 5.3.2). In this situation, the fluctuation of kinetic energy is suppressed entirely and the Berendsen thermostat becomes close to the Gaussian thermostat, which corresponds to the canonical ensemble in the configurational space[57] (see Section 5.3.1). All intermediate situations correspond to the sampling of an unusual weak-coupling ensemble, which is neither canonical nor microcanonical (and it is sometimes called weak ensemble, from the fact that the Berendsen thermostat is considered a weak coupling thermostat).[57] Therefore, the statistical mechanical ensemble for the Berendsen thermostat has intermediate properties between canonical and microcanonical ensembles. An approximate expression for the equilibrium distribution function in the configurational space for the Berendsen thermostat, derived under simple approximations, is presented in Ref. [57].

In practice, τ_t is used as an empirical parameter to adjust the strength of the coupling, whose value should be chosen in an appropriate range. On the one hand, a too large value (loose coupling) may cause a systematic temperature drift and a poor temperature control. Indeed, in the limit $\tau_t \to \infty$, the Berendsen thermostat is inactive leading to the simple Newton equations of motion, which samples a microcanonical ensemble. Thus, the temperature fluctuations will increase with τ_t until they reach the appropriate value for a microcanonical ensemble. However, they will never reach the appropriate value for a canonical ensemble, which are larger. For large values of τ_t, a systematic energy (and thus temperature) drift due to numerical errors may also occur. On the other hand, a too small value (tight coupling) will cause unrealistically low temperature fluctuations. Values of $\tau_t \approx 0.1$ ps are typically used in molecular dynamics simulations of condensed matter systems. For liquid water, a value of $\tau_t = 0.4$ ps is good. In the original paper of Berendsen *et al.*,[54] simulations of water with its long range dipolar

interaction have been performed. Normally in these simulations a cutoff noise and drift are produced (see Figure 5.2, left panel, a).[54] Using a temperature coupling the drift is removed but the average temperature is slightly larger than the reference temperature of the bath.[54] With a time constant $\tau_t = 0.4$ ps the opposing fluctuations in kinetic and potential energy remain similar to those of the uncoupled simulation (see Figure 5.2, left panel, b). When τ_t is reduced below 0.1 ps, the fluctuations in the kinetic energy are reduced at the expense of increasing fluctuation in the total energy (see Figure 5.2, right panel), while the fluctuations in potential energy are not sensitive to the value of the coupling time constant.[54]

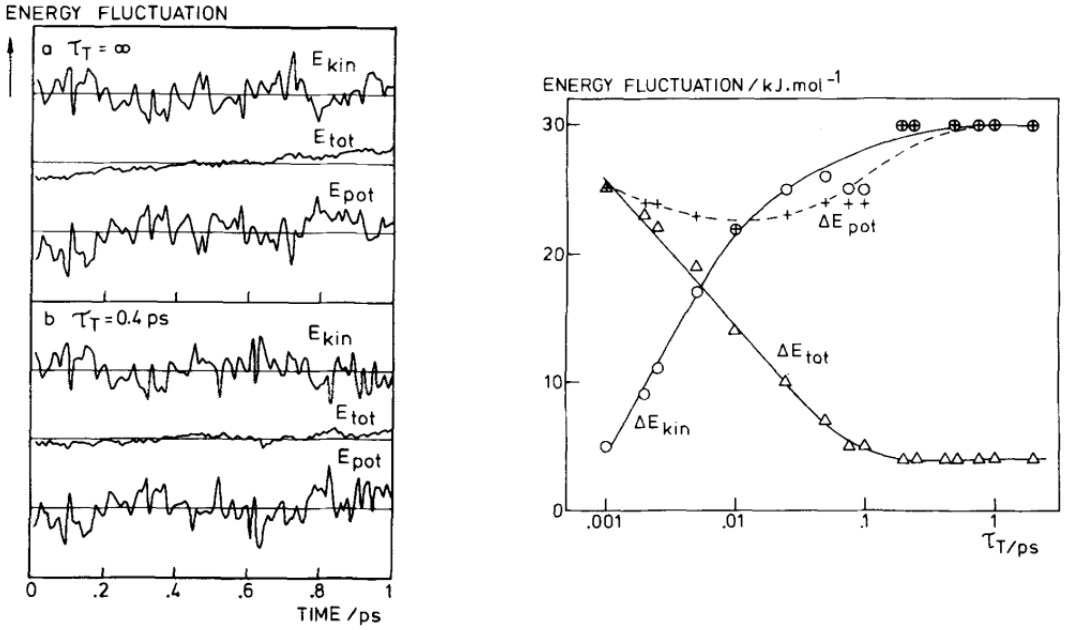

Figure 5.2: Left panel: Fluctuations of kinetic, total, and potential energy in two 1 ps molecular dynamics runs of 216 water molecules, starting from the same initial conditions (T = 300 K). a) isochoric, microcanonical simulation ($\tau_t = \infty$), b) isochoric simulation with weak coupling to constant temperature bath ($\tau_t = 0.4$ ps). The vertical scale respresents 1 kJ/mol per division. Horizontal lines represent averages for each curve. Right panel: Root means square fluctuations in kinetic, total, and potential energies, measured over several 0.1 ps simulations of liquid water. From Ref. [54].

The work of Berendsen *et al.*[54] permits to conclude that (i) although short coupling time constants greatly influence fluctuations, average thermodynamic quantities and static average properties are not disturbed and are not significantly influenced even for time constants as short as 0.01 ps, (ii) fluctuations of global properties, however, are strongly influenced for time constants less than 0.1 ps, so that the intensity of such fluctuations cannot be used to derive thermodynamic properties, and (iii) dynamic properties of individual particles are not significantly altered, although the velocity autocorrelation function shows some deviation for very short time constants. Although these variations may not be significant, they indicate the possibility of a deviation in dynamic properties for time constants as low as 0.01 ps, so that reliable dynamic properties are considered to be derived for coupling time constants above 0.1 ps.

The Berendsen method has two distinct advantages above methods that are based on Lagrangians modified to include certain constraints. First, the coupling can be made as weak as desired to minimize the disturbance of the system, and the strength of the coupling can easily be varied to suit the needs of a given application. Indeed, since the Berendsen thermostat can change the behavior of the system only by controlling parameter τ_t, it can be used in various situations without any changes in the algorithm. Second, the algorithm is numerically stable and truncation errors will not develop undesired deviations that need ad hoc corrections. This is of considerable practical value when conditions are adjusted to new values, as well as for long runs.

However, the Berendsen equations of motion are smooth and deterministic, but time-irreversible. Furthermore, they do not sample the exact canonical ensemble, as demonstrated in Ref. [57], but they produce an ensemble with intermediate properties between the canonical and microcanonical, according

to coupling strength of a heat bath represented by τ_t. Moreover, the Berendsen thermostat suffers from the flying ice cube effect as well as the velocity rescaling procedure (see Section 5.3.2). If motion of the center of mass and global rotational motion are periodically removed from the system, this combination should substantially ameliorate the problems identified by the violation of energy equipartition, even if this issue cannot be completely avoided.[51, 52]
For these reasons, the Berendsen thermostat is not recommended for use in production molecular dynamics runs because it does not strictly conform to the canonical ensemble, but it is considered good to use during equilibration.
The reason why the Berendsen thermostat systematically (for all values of τ_t) underestimates temperature fluctuations (and thus it does not give the correct thermodynamic canonical ensemble) resides in the neglect of the stochastic contribution to these fluctuations on the microscopic timescale. Therefore, eliminating random forces in a Langevin equation fails to produce the canonical (NVT) ensemble. To achieve the goal of sampling the canonical distribution, some fictitious (virtual) degrees of freedom have to be introduced in the equations of motion, allowing to directly derive the canonical distribution function. This methodology, called the extended system method, is explained and derived in the following section.

5.3.4 The Nosé-Hoover thermostat

One of the most used classical mechanic method that generates a constant temperature molecular dynamics simulation is the so-called Nosé-Hoover thermostat, which is one of the first extended system method introduced by S. Nosé,[46, 47, 48] on the wheel of Andersen work on the constant pressure method,[45] where the extended system technique has been first introduced. The temperature version of the extended system method proposed by S. Nosé[46, 47, 48] will be explained in the present section.

A schematic image of the canonical ensemble is a system surrounded by and thermally contacted with a large external reservoir (a heat bath), see Figure 5.3b. In the constant temperature method introduced by Nosé, a degree of freedom s is inserted in the equations of motion, representing the large external system, so that the physical subsystem exchanges energy with this additional degree of freedom. Under this condition, the total energy of the physical system is allowed to fluctuate.
The thermal interactions between the system and the heat bath are expressed by a scaling of the nuclear velocities by a factor s.

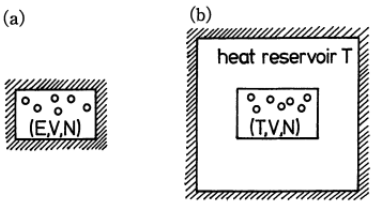

Figure 5.3: Schematic representation for a system (a) in the microcanonical and (b) in the canonical ensembles. The shaded area is a heat insulating wall.

The real velocity $\boldsymbol{v}_i$ of the i-th atom is obtained by multiplying the scaling factor s with a virtual velocity $\tilde{\boldsymbol{v}}_i$, so that

$$\boldsymbol{v}_i = s\tilde{\boldsymbol{v}}_i \tag{5.243}$$

The virtual variables refer to the extended system and will be indicated with a tilde symbols as above, in order to distinguish them from the variables referred to the real physical atomic system. Two additional degrees of freedom are introduced for the description of the external system (heat bath), which stand for momentum $\tilde{p}_s$ and positions $\tilde{s}$ related to the particles representing the heat reservoir. The position coordinates are the same in both frames, but they are defined through virtual and real coordinates to complete the formulation. Obviously, the real variables describe the motion of the nuclei in the atomic system, while the virtual ones are introduced artificially for controlling the temperature. The relation between the two frames is given by a non-canonical transformation,

$$\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i \qquad \boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i/s \qquad s = \tilde{s} \qquad t = \int^t \frac{d\tilde{t}}{s} \tag{5.244}$$

This transformation can be explained in a unified fashion from a basic assumption of the scaling for the infinitesimal time,

$$dt = d\tilde{t}/s \tag{5.245}$$

This relation, in turn, was extracted from the following speculation. Consider the motion of a particle during one simulation time step. The velocity $\boldsymbol{v}$ is defined as a ratio of the difference of the position $\Delta\boldsymbol{q}$ and the time Δt. The temperature of a system is related to the average kinetic energy. A temperature control will be achieved via control of the nuclear velocity. If the time Δt required for the movement $\Delta\boldsymbol{q}$ is lengthened, the velocity is reduced. On the other hand, the motion is accelerated by shortening the time period Δt. Therefore, the temperature control could be achieved by the introduction of a flexible time. The change of the time length is expressed by a scaling of the time variable by a factor s.
With the time scaling (5.245), the velocity scaling in (5.243) can be interpreted as

$$\boldsymbol{v}_i = \frac{d\boldsymbol{q}_i}{dt} = s\frac{d\boldsymbol{q}_i}{d\tilde{t}} = s\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = s\tilde{\boldsymbol{v}}_i \tag{5.246}$$

This type of transformations from virtual to real variables and vice versa will be employed later on to transform the equations of motion from a virtual to a real variables formulation.
The Hamiltonian of the extended system given by the real atomic subsystem and the heat reservoir in terms of the virtual variables is postulated as follows

$$\tilde{\mathcal{H}}(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}, \tilde{p}_s, \tilde{s}) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln \tilde{s} \tag{5.247}$$

The first two terms are the kinetic energy and the potential energy of the physical system of interest. The last two terms correspond to the added degree of freedom, where p_s is a conjugate momentum of s, Q is a parameter with dimension energy·(time)^2 which behaves as a mass for the motion of s and T_0 is the temperature of the heat bath. The parameter g is essentially equal to the number of degree of freedom of the physical system. Its exact value will be chosen to exactly satisfy the canonical distribution in equilibrium. The additional degree of freedom s is introduced to break the conservation of the total energy of the real system, which is inevitably imposed on the molecular dynamics simulations as a result of the use of standard classical mechanics Hamilton equations. The total Hamiltonian (5.247) is still conserved in the whole extended system, but the total energy of the physical system,

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{5.248}$$

can fluctuate and the distribution of the energy will follow the canonical distribution. It is worth noting that a logarithmic dependence of the potential on the variable s, in the form $gk_bT_0 \ln s$, is essential for producing the canonical ensemble. The assumption that the Hamiltonian formalism can be applied to equation (5.247), in the virtual variables frame of reference, is made. The equations of motion are obtained via canonical equations as[46]

$$\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{p}}_i} = \frac{\tilde{\boldsymbol{p}}_i}{m_i s^2} \qquad i = 1, ..., N \tag{5.249}$$

$$\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} = -\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{q}}_i} = -\frac{\partial\phi}{\partial\tilde{\boldsymbol{q}}_i} \qquad i = 1, ..., N \tag{5.250}$$

$$\frac{d\tilde{s}}{d\tilde{t}} = \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{p}_s} = \frac{\tilde{p}_s}{Q} \tag{5.251}$$

$$\frac{d\tilde{p}_s}{d\tilde{t}} = -\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}} = \frac{1}{\tilde{s}}\left[\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\tilde{s}^2} - gk_bT_0\right] \tag{5.252}$$

These equations exhibit a negative feedback mechanism to control the temperature, keeping it around a fixed value. The acceleration of s is proportional to the deviation of the total kinetic energy from its average value $gk_bT_0/2$. This mechanism keeps the kinetic energy around $gk_bT_0/2$. This will be shown more clearly using equations of motion in the real variables frame of reference, see Section 5.3.4.1.
One of the conserved quantity is the Hamiltonian $\tilde{\mathcal{H}}$, that is an integral of motion, because the total time derivative of the virtual Hamiltonian is equal to zero,

$$\frac{d\tilde{\mathcal{H}}}{d\tilde{t}} = \sum_{i=1}^{N}\left(\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{p}}_i}\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{q}}_i}\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}}\right) + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{p}_s}\frac{d\tilde{p}_s}{d\tilde{t}} + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}}\frac{d\tilde{s}}{d\tilde{t}} = 0 \tag{5.253}$$

Other two conserved quantities are the total linear and angular momentum, respectively given by,

$$\sum_{i=1}^{N} \tilde{\boldsymbol{p}}_i \qquad \text{and} \qquad \sum_{i=1}^{N} \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{p}}_i \tag{5.254}$$

The conservation laws for the total linear and angular momenta are derived from equation (5.250) and the properties satisfied by the potential

$$\sum_{i=1}^{N} \frac{\partial \phi}{\partial \tilde{\boldsymbol{q}}_i} = 0 \qquad \sum_{i=1}^{N} \tilde{\boldsymbol{q}}_i \wedge \frac{\partial \phi}{\partial \tilde{\boldsymbol{q}}_i} = 0 \tag{5.255}$$

However, it should be noted here that during the ordinary molecular dynamics simulations with periodic boundary conditions the angular momentum is not conserved.[29] Because of the theoretical linear and angular momenta conservation, the ensembles produced by the molecular dynamics method are slightly different from the usual statistical mechanical ensembles. At the same time, the conservation of the linear momentum in practical simulations generates a different ensemble than the theoretical ones. These deviations are ignored in the following discussion, but they are introduced and discussed later in paragraph A.7.1.

5.3.4.1 Equations of motion in real variables

The equations of motion (5.249)-(5.252) can be transformed from virtual to real variables. The equations of motion in real variables represents the physical system, and they are therefore useful in practical simulations. Therefore, starting from the equations of motion (5.249)-(5.252), variables transformations are applied so that the coordinates change of reference frame, from the virtual $(\tilde{\boldsymbol{p}}_i, \tilde{\boldsymbol{q}}_i, \tilde{p}_s, \tilde{s}, \tilde{t})$ to the real $(\boldsymbol{p}_i, \boldsymbol{q}_i, p_s, s, t)$ set of variables, can be performed

$$(\tilde{\boldsymbol{p}}_i, \tilde{\boldsymbol{q}}_i, \tilde{p}_s, \tilde{s}, \tilde{t}) \quad \rightarrow \quad (\boldsymbol{p}_i, \boldsymbol{q}_i, p_s, s, t) \tag{5.256}$$

The transformations are carried out in stepwise fashion, via the basic relations between the virtual and the real set of variables, which are here reported

$$\boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i / s \qquad \boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i \qquad p_s = \tilde{p}_s / s \qquad s = \tilde{s} \qquad t = \int^{t} \frac{d\tilde{t}}{s} \tag{5.257}$$

Calculations in real variables lead to

$$\frac{d\boldsymbol{q}_i}{dt} = s\frac{d\boldsymbol{q}_i}{d\tilde{t}} = s\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = \frac{\tilde{\boldsymbol{p}}_i}{m_i s} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.258}$$

$$\frac{d\boldsymbol{p}_i}{dt} = s\frac{d}{d\tilde{t}}\left(\frac{\tilde{\boldsymbol{p}}_i}{s}\right) = \frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} - \frac{1}{s}\frac{ds}{d\tilde{t}}\tilde{\boldsymbol{p}}_i = -\frac{\partial \phi}{\partial \tilde{\boldsymbol{q}}_i} - \frac{\tilde{p}_s}{sQ}(s\boldsymbol{p}_i) = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \frac{s p_s}{Q}\boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.259}$$

$$\frac{ds}{dt} = s\frac{ds}{d\tilde{t}} = s\frac{\tilde{p}_s}{Q} = \frac{s^2 p_s}{Q} \tag{5.260}$$

$$\frac{dp_s}{dt} = s\frac{d}{d\tilde{t}}\left(\frac{\tilde{p}_s}{s}\right) = \frac{d\tilde{p}_s}{d\tilde{t}} - \frac{\tilde{p}_s}{s}\frac{ds}{d\tilde{t}} = \frac{1}{s}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right) - \frac{s\, p_s^2}{Q} \tag{5.261}$$

where the equations of motion (5.249)-(5.252) in virtual variables have been used for the derivation. In 1985, one year later than the Nosé article on constant temperature molecular dynamics, W. G. Hoover in its paper on canonical dynamics[58] pointed out that, if a new variable is chosen, with the form

$$\xi = \frac{1}{s}\left(\frac{ds}{dt}\right) = \frac{s\, p_s}{Q} \tag{5.262}$$

then the equations (5.258)-(5.261) can be simplified as

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.263}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \xi\, \boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.264}$$

$$\frac{d(\ln s)}{dt} = \xi \tag{5.265}$$

$$\frac{d\xi}{dt} = \frac{1}{Q}\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right) \tag{5.266}$$

where in particular equation (5.266) has been derived starting from equations (5.261) and (5.260) and manipulating it as follows

$$\frac{dp_s}{dt} = \frac{1}{s}\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right) - \frac{s\, p_s^2}{Q} \quad \rightarrow \quad s\frac{dp_s}{dt} + p_s\left(\frac{s^2\, p_s}{Q}\right) = \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right)$$

$$s\frac{dp_s}{dt} + p_s\frac{ds}{dt} = \frac{d}{dt}(sp_s) = \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right) \quad \rightarrow \quad \frac{1}{Q}\frac{d}{dt}(sp_s) = \frac{d}{dt}\left(\frac{sp_s}{Q}\right) = \frac{d\xi}{dt} = \frac{1}{Q}\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0\right)$$

The form of the equations of motion as reported in (5.263)-(5.266) are known as the Nosé-Hoover thermostat equations of motion.[59] A negative feedback mechanism is more apparent in the Nosé-Hoover thermostat form of the equations of motion (see Figure 5.4a). The first two equations (5.263) and (5.264) have the typical form as those describing the motion of a body with a frictional force. However, the friction coefficient ξ is not a constant and can be positive or negative in this case. The time development of ξ, given by equation (5.266), is driven by the imbalance between the kinetic energy and its average value $(g/2)k_bT_0$. If the kinetic energy is larger than $(g/2)k_bT_0$, the time derivative of ξ is positive ($d\xi/dt > 0$), so that ξ increases and will become positive. The equation with positive values of ξ is equivalent with that of a system with a friction force, see equation (5.264). As a consequence, the velocity of the particle decreases and the kinetic energy also decreases. If the kinetic energy becomes lower than $(g/2)k_bT_0$, the feedback mechanism works in the opposite direction. In this way, the time derivative of ξ becomes negative ($d\xi/dt < 0$), so that ξ decreases. Then, in the region of ξ negative values, the system is heated up again. Following this mechanism on the base of the equations of motion, the kinetic energy fluctuates around its average value $(g/2)k_bT_0$. The time average of a time derivative of a variable will vanish. This guarantees that the average of the kinetic energy coincides with the result of the equipartition theorem $(g/2)k_bT_0$.

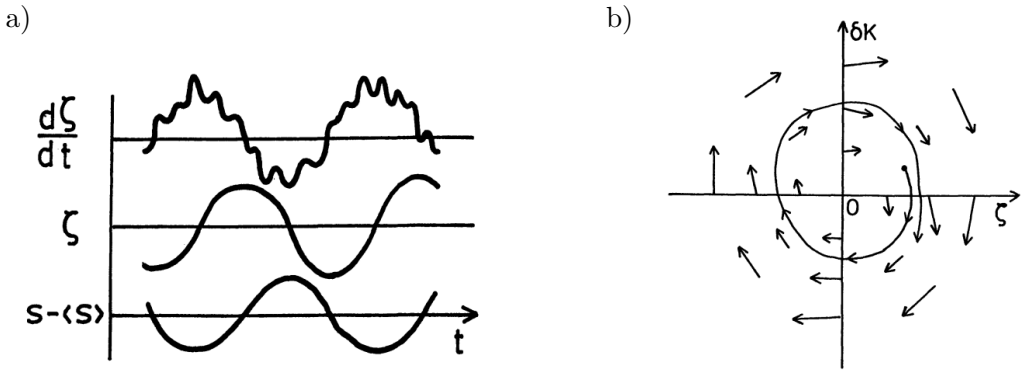

Figure 5.4: a) A schematic time evolution of $(d\xi/dt)$, the kinetic energy ξ, and the heat bath variable s, representing the negative feedback mechanism that works to keep the kinetic energy around its averaged value. b) The movement in a phase space $(\xi, \delta E_k)$. Arrows indicate velocity vectors.

The change of ξ is governed by the deviation of the kinetic energy from its average value. When ξ is positive, the movement of the particles is slowed down, but the motion is accelerated in the negative ξ

region. Consider a motion in a phase space,

$$\boldsymbol{A} = (\xi, \delta E_k) \tag{5.267}$$

where the deviation of the kinetic energy E_k from its average value, $\delta E_k = E_k - \langle E_k \rangle$ is proportional to the time derivative of the variable ξ,

$$\frac{d\xi}{dt} \propto \delta E_k = E_k - \langle E_k \rangle$$

The velocity in this space is

$$\dot{\boldsymbol{A}} = \left(\frac{d\xi}{dt}, \frac{d\left(\delta E_k\right)}{dt} \right) \tag{5.268}$$

and it is depicted in Figure 5.4b. From equation (5.266) follows that

$$\frac{d\xi}{dt} > 0 \quad \text{if} \quad \delta E_k > 0 \qquad \text{and} \qquad \frac{d\xi}{dt} < 0 \quad \text{if} \quad \delta E_k < 0 \tag{5.269}$$

At the same time, from equation (5.264),

$$\frac{d(\delta E_k)}{dt} < 0 \quad \text{if} \quad \xi > 0 \qquad \text{and} \qquad \frac{d(\delta E_k)}{dt} > 0 \quad \text{if} \quad \xi < 0 \tag{5.270}$$

Therefore, the trajectory in this phase space circles around the origin clockwise. The kinetic energy fluctuates and it is kept around a constant.
This reformulation of the extended system method given by Hoover,[58] is based on the above mentioned equations of motion and it is now known as the Nosé-Hoover thermostat.[59] The three equations of motion (5.263), (5.264) and (5.266) form a closed system of equations. The time evolution of the system within a phase space $(\boldsymbol{p}, \boldsymbol{q}, \xi)$ is uniquely determined from these equations. In this sense, equation (5.265) is redundant. However, if also this equation is solved, the conserved quantity correspondent to the virtual Hamiltonian given by (5.247), that generates the equations of motion (5.249)-(5.252) in virtual variables, can be computed. Indeed, this conserved quantity can be written in terms of real variables as

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_s, s) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{(sp_s)^2}{2Q} + gk_bT_0 \ln s \tag{5.271}$$

and using the definition (5.262) of the ξ variable, the previous expression becomes

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, \xi, s) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{Q}{2}\,\xi^2 + gk_bT_0 \ln s \tag{5.272}$$

The resultant form (5.272) is also a conserved quantity for the equations of motion (5.263)-(5.266) in real variables, as demonstrated by the time derivative

$$\begin{aligned} \frac{d\mathcal{H}}{dt} &= \sum_{i=1}^{N} \left(\frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} \cdot \frac{d\boldsymbol{p}_i}{dt} + \frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} \cdot \frac{d\boldsymbol{q}_i}{dt} \right) + \frac{\partial \mathcal{H}}{\partial \xi}\frac{d\xi}{dt} + \frac{\partial \mathcal{H}}{\partial s}\frac{ds}{dt} \\ &= \sum_{i=1}^{N} \left[\frac{\boldsymbol{p}_i}{m_i} \cdot \left(-\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \xi\, \boldsymbol{p}_i \right) + \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\boldsymbol{p}_i}{m_i} \right] + (\xi\, Q)\frac{1}{Q}\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0 \right) + \frac{gk_bT_0}{s}(s\xi) = 0 \end{aligned} \tag{5.273}$$

From equation (5.272) it is clear that the calculation of the conserved quantity in real variables requires the value of the ξ variable. Therefore, the equation of motion (5.265), although redundant, is essential to compute the conserved quantity (5.272). This conservation law can be used as a measure that the simulation is carried out correctly.
The equations of motion (5.258)-(5.261), and as a consequence the equation of motion (5.263)-(5.266), are no longer canonical, since equations (5.259) and (5.261) (so as the correspondent equations (5.264) and (5.266)) have additional force terms with respect to the virtual variables counterpart. Therefore, the quantity (5.271) (and the correspondent (5.272) form) is no longer a Hamiltonian, i.e., the equations of

motion in real variables cannot be derive directly by applying the Hamilton equations of motion starting from the expression (5.271) (or (5.272), equivalently). This is due to the fact that the transformations (5.257) from virtual to real coordinates are not canonical. For example, by computing the Hamilton equations of motion on the form (5.271), the following expressions are recovered:

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.274}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}_i} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} \qquad i = 1, ..., N \tag{5.275}$$

$$\frac{ds}{dt} = \frac{\partial \mathcal{H}}{\partial p_s} = \frac{s^2 p_s}{Q} \tag{5.276}$$

$$\frac{dp_s}{dt} = -\frac{\partial \mathcal{H}}{\partial s} = -\frac{s\, p_s^2}{Q} - \frac{g k_b T_0}{s} \tag{5.277}$$

Since the equations of motion (5.275) and (5.277) do not coincide with the correct forms (5.259) and (5.261), respectively, the conserved quantity (5.271) is non a Hamiltonian (non-Hamiltonian form). The same non-Hamiltonian form can be trivially proved also for the conserved quantity (5.272) written as a function of the variable ξ (instead of the variable p_s): the transformations (5.257) are not canonical, therefore equation (5.272) is no longer a Hamiltonian, and the equations of motion (5.263)-(5.266) cannot be derived from it.

5.3.4.2 Integration of the equations of motion

For the Nosé-Hoover method, a time propagator operator can be defined by decomposition of the Liouville operator, as explained in Section 4.2. By changing the order of these decomposed operators, several time integrators can be thought.[60] For the vector $\boldsymbol{\Gamma}$ in the extended phase space, which is defined by $\boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t), p_\eta(t), \eta(t))$, the time development (4.65) from time t_0 to time $t = t_0 + P\,\tau$, where P is an integer representing the number of molecular dynamics steps the phase space coordinates are allowed to propagate and τ is the simulation time step unit, is written as

$$\boldsymbol{\Gamma}(t) = \boldsymbol{\Gamma}(t_0 + P\,\tau) = e^{i\mathcal{L}\,P\,\tau}\,\boldsymbol{\Gamma}(t_0) \tag{5.278}$$

where the Liouville operator is given by equation (4.62) reported here below for the case of the extended system of interest

$$i\mathcal{L} = \dot{\boldsymbol{\Gamma}} \cdot \frac{\partial}{\partial \boldsymbol{\Gamma}} \qquad \text{with} \quad \boldsymbol{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t), p_\eta(t), \eta(t)) \tag{5.279}$$

where the dot over the phase space vector variable stands for the time derivative.
The conserved quantity corresponding to the Hamiltonian (5.271) expressed in real coordinates (physical system frame of reference) can be rewritten using the transformations (5.291) as

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_\eta, \eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta \tag{5.280}$$

Starting from the formula (5.279), and using the Nosé-Hoover equations of motion (5.309)-(5.312) previously derived, the form of the Liouville operator can be easily derived to be

$$\begin{aligned} i\mathcal{L} &= \sum_{i=1}^{Nd} \frac{d\boldsymbol{q}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \sum_{i=1}^{Nd} \frac{d\boldsymbol{p}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} + \frac{d\eta}{dt}\frac{\partial}{\partial \eta} + \frac{dp_\eta}{dt}\frac{\partial}{\partial p_\eta} \\ &= \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} - \sum_{i=1}^{Nd} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} - \sum_{i=1}^{Nd} \frac{p_\eta}{Q}\,\boldsymbol{p}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} + \frac{p_\eta}{Q}\frac{\partial}{\partial \eta} + \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0 \right) \frac{\partial}{\partial p_\eta} \end{aligned} \tag{5.281}$$

The evolution in time of real system and thermostat variables can be modeled through a velocity Verlet-like reversible scheme of integration derived using the Trotter-Suzuki factorization of the Liouville operator (see Section 4.2.1). Using a five-terms decomposition of the Liouville operator (with the generalized

factorization (4.93) derived in Section 4.2.1), the Liouville operator can be split into the sum of five operators as follows

$$i\mathcal{L} = i\mathcal{L}_1 + i\mathcal{L}_2 + i\mathcal{L}_3 + i\mathcal{L}_4 + i\mathcal{L}_5 \tag{5.282}$$

where

$$i\mathcal{L}_1 = \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} \qquad i\mathcal{L}_2 = -\sum_{i=1}^{Nd} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \tag{5.283}$$

$$i\mathcal{L}_3 = -\sum_{i=1}^{Nd} \frac{p_\eta}{Q} \boldsymbol{p}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \qquad i\mathcal{L}_4 = \frac{p_\eta}{Q} \frac{\partial}{\partial \eta} \qquad i\mathcal{L}_5 = \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0 \right) \frac{\partial}{\partial p_\eta} \tag{5.284}$$

The separated operators $i\mathcal{L}_1$ and $i\mathcal{L}_2$ are the same as in the NVE ensemble (see Section 5.1.2), while the operators $i\mathcal{L}_3$, $i\mathcal{L}_4$ and $i\mathcal{L}_5$ are additional operators which come from the variables associated to the Nosé-Hoover thermostat. This separation has been proposed in Refs. [61] and [62], and the transformations of the variables by these separated operators preserve an extended phase space volume element equal to $\omega = \exp(g\eta)\, d\boldsymbol{\Gamma}$, where $d\boldsymbol{\Gamma} = d\boldsymbol{q}\, d\boldsymbol{p}\, d\eta\, dp_\eta$ (see Refs. [61] and [62] for more details).
From the separation in equations (5.283) and (5.284), several decomposition forms for the time development operator $\exp(i\mathcal{L}\tau)$ can be considered as time integrators. Indeed, different variants of the time integrator form can be obtained by the permutation of the five separated Liouville operators. However, because the operators $i\mathcal{L}_1$ and $i\mathcal{L}_5$ are commutative with each other, the variants number reduces, and it is less than the total number of possible permutations of the five operators. S. Itoh et al.[60] employed and analyzed six different time integrators among all the possible ones, which are extensions of the velocity Verlet or position Verlet algorithms, and they discuss the effects of difference among these integrators on ensemble averages of physical quantities such as temperature and potential energy.
The approximate discrete time propagator can be constructed applying the general Suzuki-Trotter factorization formula (4.93), with $P = 1$ (one molecular dynamics step) and a number of Liouville operators equal to $n = 5$ (see Section 4.2.1), as follows

$$e^{i\mathcal{L}\tau} \approx e^{i\mathcal{L}_5 \tau/2}\, e^{i\mathcal{L}_4 \tau/2}\, e^{i\mathcal{L}_3 \tau/2}\, e^{i\mathcal{L}_2 \tau/2}\, e^{i\mathcal{L}_1 \tau}\, e^{i\mathcal{L}_2 \tau/2}\, e^{i\mathcal{L}_3 \tau/2}\, e^{i\mathcal{L}_4 \tau/2}\, e^{i\mathcal{L}_5 \tau/2} + O(\tau^9) \tag{5.285}$$

The above defined time propagator has to be used in equation (5.278) to compute the time propagation of variables from an initial time t_0 to $t_0 + \tau$ (one time step) in the extended phase space, so that

$$\boldsymbol{\Gamma}(t_0 + \tau) \approx e^{i\mathcal{L}_5 \tau/2}\, e^{i\mathcal{L}_4 \tau/2}\, e^{i\mathcal{L}_3 \tau/2}\, e^{i\mathcal{L}_2 \tau/2}\, e^{i\mathcal{L}_1 \tau}\, e^{i\mathcal{L}_2 \tau/2}\, e^{i\mathcal{L}_3 \tau/2}\, e^{i\mathcal{L}_4 \tau/2}\, e^{i\mathcal{L}_5 \tau/2}\, \boldsymbol{\Gamma}(t_0) + O(\tau^9) \tag{5.286}$$

Obviously, this phase space propagation formula has been defined for a single time step ($P = 1$), starting from an initial time t_0. The number of steps P defined in a molecular dynamics simulation is in generally very high, so that P is a large but finite integer which approximates the condition $P \to \infty$ defined in the exact formulation of Suzuki-Trotter decomposition, equation (4.92). Indeed, in a dynamical simulation the formula (5.286) has to be applied sequentially P times, propagating the phase space point step by step until a final time $t = t_0 + P\,\tau$ (see equation (5.278)), with a discretized time unit of $\tau = \Delta t$.
The action of operators with the form (5.283) and (5.284) on a phase space point has been computed in Section 4.2.2, and it is therefore applied for each operator in the order given by equation (5.286). The resulting algorithm has the structure reported in Table 5.5, for a time step equal to $\tau = \Delta t$.
The form of the time propagator operator in (5.286) follows the form and the structure of the integrator I2 discussed in Ref. [60]. At the same time, by changing the order of the decomposed operators, the algorithms for time development can vary. In this framework, it is worth mentioning that in the CRYSTAL code the propagator (5.286) is the default one, but there is also the possibility to use another kind of slightly different time propagator, with the form reported in equation (A.298), see Appendix A, Section A.12. Even in this case, in order to implement the time evolution algorithm it is necessary to evaluate the action of operators in (A.298) on a phase space point, following the results derived in Section 4.2.2, applied for each operator in the order given by equation (A.299), see Appendix A, Section A.12. The resulting algorithm has the structure reported in Table A.1 of Section A.12 (Appendix A), for a time step equal to $\tau = \Delta t$.

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{p}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}],\ \eta(t),\ p_\eta(t)$
Step 1. Propagator $e^{i\mathcal{L}_5\Delta t/2}$	:	$p_\eta(t+\Delta t/2) \leftarrow p_\eta(t) + \frac{\Delta t}{2}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t)}{m_i} - gk_bT_0\right)$
Step 2. Propagator $e^{i\mathcal{L}_4\Delta t/2}$	:	$\eta(t+\Delta t/2) \leftarrow \eta(t) + \frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)$
Step 3. Propagator $e^{i\mathcal{L}_3\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t)\exp\left[-\frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)\right]$
Step 4. Propagator $e^{i\mathcal{L}_2\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \frac{\Delta t}{2}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 5. Propagator $e^{i\mathcal{L}_1\Delta t}$	:	$\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t) + \frac{\Delta t}{m_i}\,\boldsymbol{p}_i(t+\Delta t/2)$
Step 6. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 7. Propagator $e^{i\mathcal{L}_2\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \frac{\Delta t}{2}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 8. Propagator $e^{i\mathcal{L}_3\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t) \leftarrow \boldsymbol{p}_i(t+\Delta t)\exp\left[-\frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)\right]$
Step 9. Propagator $e^{i\mathcal{L}_4\Delta t/2}$	:	$\eta(t+\Delta t) \leftarrow \eta(t+\Delta t/2) + \frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)$
Step 10. Propagator $e^{i\mathcal{L}_5\Delta t/2}$	:	$p_\eta(t+\Delta t) \leftarrow p_\eta(t+\Delta t/2) + \frac{\Delta t}{2}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t+\Delta t)}{m_i} - gk_bT_0\right)$

Table 5.5: Nosé-Hoover thermostat integrator algorithm (vverlet_nose_I1), applied $\forall\, i = 1, ..., N$.

5.3.4.3 Conserved quantities

The Nosé-Hoover equations of motion (5.258) - (5.261) in the real variable frame of reference, given by

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.287}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{sp_s}{Q}\,\boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.288}$$

$$\frac{ds}{dt} = \frac{s^2\,p_s}{Q} \tag{5.289}$$

$$\frac{dp_s}{dt} = \frac{1}{s}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0\right) - \frac{s\,p_s^2}{Q} \tag{5.290}$$

can be rearranged by defining two new variables η and p_η to be equal to

$$\eta = \ln s \qquad\qquad p_\eta = sp_s \tag{5.291}$$

Moreover, equation (5.290) can be rewritten using the new variable p_η as

$$\frac{dp_s}{dt} = \frac{1}{s}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0\right) - \frac{s\,p_s^2}{Q} \quad \rightarrow \quad s\frac{dp_s}{dt} + p_s\left(\frac{s^2\,p_s}{Q}\right) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0$$

and using the equations of motion (5.289) to rewrite the term in the parenthesis on the left hand side of the previous expression, a new equation of motion can be written for the variable p_η as follows

$$s\frac{dp_s}{dt} + p_s\frac{ds}{dt} = \frac{d}{dt}(sp_s) = \frac{dp_\eta}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0$$

Therefore, the equations of motion (5.288) - (5.290) change slightly under the transformations (5.291), obtaining the following set of equations of motion

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.292}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_\eta}{Q}\,\boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.293}$$

$$\frac{d\eta}{dt} = \frac{p_\eta}{Q} \tag{5.294}$$

$$\frac{dp_\eta}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0 \tag{5.295}$$

where the coordinates $\boldsymbol{q}_i$ and $\boldsymbol{p}_i$ are d dimensional vector (with d the dimension of the physical system), the forces $\mathbf{F}_i = -\partial\phi/\partial\boldsymbol{q}_i$ are derived from an N particle potential $\phi(\boldsymbol{q})$ through the Hellmann-Feynmann theorem, and g is a parameter that has to be determined to generate the canonical distribution in the physical degrees of freedom. In order to determine the statistical mechanical ensemble generated by the equations of motion (5.292) - (5.295), following the analysis outlined in Section 4.3.2.1, the constants of motion have to be found. The simplest one is the total Hamiltonian (5.271), written using the change of variables (5.291), that is

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_\eta, \eta) = \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_\eta^2}{2Q} + gk_bT_0\eta = C_1 \qquad \text{where} \qquad \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{5.296}$$

If there exists a non zero net force acting on the system, then (5.296) is the only one constant of motion preserved by the non Hamiltonian flow of equations (5.292) - (5.295). In general, however, there will be more conserved quantities. For instance, considering a system in the absence of external forces, namely, under the hypothesis

$$\mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.297}$$

then there are d additional conservation laws satisfied by the equations of motion (5.292) - (5.295), which take the form

$$\mathbf{P}\,e^\eta = \mathbf{K} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.298}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions (with d the dimension of the physical space) and $\mathbf{P}$ is the center of mass momentum of the system defined as

$$\mathbf{P} = \sum_{i=1}^{N}\boldsymbol{p}_i \tag{5.299}$$

with the origin taken at the center of mass of the system.
Equation (5.298) can be verified by taking the time derivative explicitly, so that

$$\begin{aligned}\frac{d(\mathbf{P}e^{\eta})}{dt} &= \dot{\mathbf{P}}\, e^{\eta} + \dot{\eta}\,\mathbf{P}\, e^{\eta} = e^{\eta}\,(\dot{\mathbf{P}} + \dot{\eta}\,\mathbf{P}) = e^{\eta}\left(\dot{\mathbf{P}} + \frac{p_{\eta}}{Q}\,\mathbf{P}\right) \\ &= e^{\eta}\left(\sum_{i=1}^{N}\dot{\mathbf{p}}_i + \frac{p_{\eta}}{Q}\,\mathbf{P}\right) = e^{\eta}\left[\sum_{i=1}^{N}\left(-\frac{\partial\phi}{\partial \boldsymbol{q}_i} - \frac{p_{\eta}}{Q}\,\boldsymbol{p}_i\right) + \frac{p_{\eta}}{Q}\,\mathbf{P}\right] \\ &= e^{\eta}\left(\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_{\eta}}{Q}\sum_{i=1}^{N}\boldsymbol{p}_i + \frac{p_{\eta}}{Q}\,\mathbf{P}\right) = e^{\eta}\left(\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_{\eta}}{Q}\,\mathbf{P} + \frac{p_{\eta}}{Q}\,\mathbf{P}\right) = e^{\eta}\sum_{i=1}^{N}\mathbf{F}_i\end{aligned} \tag{5.300}$$

where the equations of motion (5.293) and (5.294) have been used, together with the definition of the total linear momentum (5.299). Therefore, as demonstrated in (5.300), the total time derivative of the quantity (5.298) is equal to zero iff the condition (5.297) is satisfied. The conservation law (5.298) can be written using a single independent variable (note that the d components of the total linear momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.298), so that

$$\|\mathbf{P}\, e^{\eta}\| = P\, e^{\eta} = \|\mathbf{K}\| = K \tag{5.301}$$

Hence, the conservation law in the absence of net forces on the system becomes

$$P e^{\eta} = K = C_2 \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.302}$$

Therefore, the quantity conserved by Nosé-Hoover equations of motion in the absence of external forces, namely when the condition (5.297) holds, is given by (5.302).
Furthermore, if the following condition is satisfied

$$\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.303}$$

then there are d additional conservation laws satisfied by the equations of motion (5.292) - (5.295), which take the form

$$\mathbf{L}\, e^{\eta} = \mathbf{K} \qquad \leftrightarrow \qquad \sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.304}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions and $\mathbf{L}$ is the total angular momentum of the system defined as

$$\mathbf{L} = \sum_{i=1}^{N}\boldsymbol{q}_i \wedge \boldsymbol{p}_i \tag{5.305}$$

with the origin taken at the center of mass of the system. Equation (5.304) can be verified by taking the time derivative explicitly, so that

$$\begin{aligned}\frac{d(\mathbf{L}e^{\eta})}{dt} &= \dot{\mathbf{L}}\, e^{\eta} + \dot{\eta}\,\mathbf{L}\, e^{\eta} = e^{\eta}\,(\dot{\mathbf{L}} + \dot{\eta}\,\mathbf{L}) = e^{\eta}\left(\dot{\mathbf{L}} + \frac{p_{\eta}}{Q}\,\mathbf{L}\right) \\ &= e^{\eta}\left(\sum_{i=1}^{N}\dot{\boldsymbol{q}}_i \wedge \boldsymbol{p}_i + \sum_{i=1}^{N}\boldsymbol{q}_i \wedge \dot{\boldsymbol{p}}_i + \frac{p_{\eta}}{Q}\,\mathbf{L}\right) \\ &= e^{\eta}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i}{m_i} \wedge \boldsymbol{p}_i + \boldsymbol{q}_i \wedge \mathbf{F}_i - \frac{p_{\eta}}{Q}\,\boldsymbol{q}_i \wedge \boldsymbol{p}_i\right) + \frac{p_{\eta}}{Q}\,\mathbf{L}\right] \\ &= e^{\eta}\left(\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i - \frac{p_{\eta}}{Q}\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \boldsymbol{p}_i + \frac{p_{\eta}}{Q}\,\mathbf{L}\right) \\ &= e^{\eta}\left(\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i - \frac{p_{\eta}}{Q}\,\mathbf{L} + \frac{p_{\eta}}{Q}\,\mathbf{L}\right) = e^{\eta}\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i\end{aligned} \tag{5.306}$$

where the equations of motion (5.292), (5.293) and (5.294) have been used, together with the definition of the total angular momentum (5.305). Therefore, as demonstrated in (5.306), the total time derivative of the quantity (5.304) is equal to zero iff the condition (5.303) is satisfied. The conservation law (5.304) can be written using a single independent variable (note that the d components of the total angular momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.304), so that

$$\|\mathbf{L}\, e^{\eta}\| = L\, e^{\eta} = \|\mathbf{K}\| = K \tag{5.307}$$

Hence, the conservation law is given by

$$Le^{\eta} = K = C_3 \qquad \leftrightarrow \qquad \sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.308}$$

Therefore, the quantity conserved by Nosé-Hoover equations of motion when the condition (5.303) holds, is given by (5.308).

As already explained for the microcanonical ensemble in Section 5.1.3, if the system is invariant to translation in a particular direction, then the corresponding momentum component is conserved, and if the system is invariant to rotation about an axis, then the corresponding angular momentum component is conserved. Thus, the three quantities (5.296), (5.302) and (5.308) are conserved for a completely isolated set of interacting molecules subject to the equations of motion (5.292) - (5.295). In practice, however, a completely isolated system is rarely considered, except for the case of an isolated molecule. On the contrary, in the presence of periodic boundary conditions, the total angular momentum is not conserved, so that there are only two constants of motion related to the time evolving equations (5.292) - (5.295), which are given by (5.296) and (5.302). In order to resume these concepts, in Table 5.6, the constants of motion for the Nosé-Hoover equations (5.292) - (5.295) are reported, together with the simulation conditions that are required to preserve their values. Note that the total Hamiltonian (5.296) is always a constant of motion, independently of the simulation conditions.

conservation laws of Nosé - Hoover equations (5.292) - (5.295)				
$\mathcal{H}(\boldsymbol{p},\boldsymbol{q},p_\eta,\eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{p_\eta^2}{2Q} + gk_bT_0\eta = C_1$				
$Pe^{\eta} = K = C_2$	iff	$\mathbf{F} = \sum_{i=1}^{N} \mathbf{F}_i = 0$		
$Le^{\eta} = K = C_3$	iff	$\sum_{i=1}^{N} \boldsymbol{q}_i \wedge \mathbf{F}_i = 0$	and	NO PBC

Table 5.6: Constants of motion for the Nosé-Hoover equations of motion (5.292) - (5.295), together with the simulation conditions that are necessary to preserve these quantities.

5.3.4.4 Statistical mechanical ensemble

The methods of non Hamiltonian dynamics can be applied to the Nosé-Hoover algorithm that has been discussed in Section 5.3.4. This discussion is only intended as an illustration of a systematic technique for predicting the phase space density generated by a particular non Hamiltonian dynamics scheme. Such an analysis is essential when the use of thermostat or barostat is considered in molecular dynamics simulations.

In Section 5.3.4 the Nosé-Hoover algorithm is introduced, and it has been demonstrated that it generates non Hamiltonian dynamics. The equations of motion related to the Nosé-Hoover algorithm that will be

considered in this analysis are given by (5.292) - (5.295), and they are reported here below

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.309}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \frac{p_\eta}{Q}\,\boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.310}$$

$$\frac{d\eta}{dt} = \frac{p_\eta}{Q} \tag{5.311}$$

$$\frac{dp_\eta}{dt} = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0 \tag{5.312}$$

The equations of motion contain two nonphysical variables, η and p_η, giving an overall phase space dimension of $2Nd+2$, where d is the number of spatial dimensions. These new variables can be regarded as a virtual thermostat which controls the fluctuations in the total kinetic energy of the system around its average value given by $gk_bT_0/2$. The parameter Q determines the time scale on which the thermostat evolves and takes the form $Q = gk_bT_0\tau^2$, where τ is a time scale relevant to the physical system. The parameter g will be determined in the following analysis.
The conditions under which the equations of motion (5.309) - (5.312) can generate a canonical distribution in the physical variables will be analyzed by applying the procedure described in Section 4.3.2.1.
As a first step, the conservation laws have to be determined. Assuming that the only conserved quantity is the energy given by (5.296) and here reported

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_\eta, \eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta = \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta = C_1 \tag{5.313}$$

where $\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q})$ is the physical Hamiltonian, then only one conservation law exists. Using the phase space vector $\boldsymbol{\Gamma} = (\boldsymbol{p}, \boldsymbol{q}, p_\eta, \eta)$, the phase space compressibility of this system, given by equation (4.123), can be written as

$$\begin{aligned} \kappa(\boldsymbol{\Gamma}, t) \equiv \nabla_{\boldsymbol{\Gamma}} \cdot \dot{\boldsymbol{\Gamma}} &= \sum_{i=1}^{N} \nabla_{\boldsymbol{q}_i} \cdot \dot{\boldsymbol{q}}_i + \sum_{i=1}^{N} \nabla_{\boldsymbol{p}_i} \cdot \dot{\boldsymbol{p}}_i + \nabla_\eta \cdot \dot{\eta} + \nabla_{p_\eta} \cdot \dot{p}_\eta \\ &= \sum_{i=1}^{N} \nabla_{\boldsymbol{p}_i} \cdot \dot{\boldsymbol{p}}_i = \sum_{i=1}^{N} \nabla_{\boldsymbol{p}_i} \cdot \left(-\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \frac{p_\eta}{Q}\,\boldsymbol{p}_i \right) = -Nd \left(\frac{p_\eta}{Q} \right) = -Nd\,\dot{\eta} \end{aligned} \tag{5.314}$$

where the equations of motion (5.310) and (5.311) have been used, N is the number of atoms and d is the dimensionality of the system. The metric can then be computed using equation (4.180) with the just derived compressibility (5.314) as

$$\begin{aligned} \sqrt{g(\boldsymbol{\Gamma}, t)} &= \exp\left(-\int \kappa(\boldsymbol{\Gamma}, t)\, dt \right) = \exp\left(\int Nd\, \frac{d\eta}{dt}\, dt \right) \\ &= \exp\left(Nd \int \frac{d\eta}{dt}\, dt \right) = \exp(Nd\,\eta) \end{aligned} \tag{5.315}$$

The microcanonical partition function at a given temperature T_0 can be constructed using the metric (5.315) and the energy conservation condition, by means of (4.185), so that

$$Z(N, V, C_1) = \zeta \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\eta \int dp_\eta\, \exp(Nd\,\eta)\; \delta\left(\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1 \right) \tag{5.316}$$

where the microcanonical partition function depends parametrically on the temperature T_0 of the system. The distribution function in the physical subspace can now be obtained by integrating over η and p_η. In equation (5.316), the integration over η and p_η can be performed analytically. Because of the delta function, integration over the variable η gives as condition

$$\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta = C_1 \tag{5.317}$$

Since the argument of the second δ function in equation (5.316) has only one zero as a function of the variable η, a convenient equivalence relation can be employed to resolve the integral, that is

$$\delta[f(\eta)] = \frac{\delta(\eta - \eta_0)}{f'(\eta_0)} \tag{5.318}$$

where η_0 is the zero of function $f(\eta)$ in the argument of the Dirac function, and $f'(\eta_0)$ is the derivative of the function $f(\eta)$ evaluated in the point η_0, which has to be read, inside the integral, as the function $f'(\eta)$, so that the expressions for these objects are

$$f(\eta) = \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1 \qquad f'(\eta) = g k_b T_0 \tag{5.319}$$

$$\eta_0 = \frac{1}{g k_b T_0}\left(C_1 - \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) - \frac{p_\eta^2}{2Q}\right) \qquad f'(\eta_0) = g k_b T_0 \tag{5.320}$$

Using the relation (5.318), together with equation (5.320), in the partition function (5.316), leads to

$$\begin{aligned} Z(N,V,C_1) &= \zeta \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\eta \int dp_\eta \, \exp(Nd\,\eta)\, \frac{\delta(\eta-\eta_0)}{g k_b T_0} = \frac{\zeta}{g k_b T_0} \int d\boldsymbol{q} \int d\boldsymbol{p} \int dp_\eta \, \exp(Nd\,\eta_0) \\ &= \frac{\zeta}{g k_b T_0} \int d\boldsymbol{q} \int d\boldsymbol{p} \int dp_\eta \, \exp\left[\frac{Nd}{g k_b T_0}\left(C_1 - \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) - \frac{p_\eta^2}{2Q}\right)\right] \\ &= \underbrace{\frac{\zeta}{g k_b T_0} \exp\left(\frac{Nd\,C_1}{g k_b T_0}\right) \int dp_\eta \, \exp\left(-\frac{Nd}{g k_b T_0}\frac{p_\eta^2}{2Q}\right)}_{=\ \text{constant}\ \xi} \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{Nd}{g k_b T_0}\, \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right) \end{aligned}$$

The integration over the variable p_η yields a constant prefactor that has no physical importance and has been included in the constant ξ. The analytical expression for this constant prefactor can be computed considering that the integral with respect to the variable p_η in the constant ξ is a Gaussian integral and it can be solved easily,[9] so that the partition function can be written as

$$\begin{aligned} Z(N,V,C_1) &= \frac{\zeta}{g k_b T_0} \exp\left(\frac{Nd\,C_1}{g k_b T_0}\right) \sqrt{\frac{2\pi Q}{Nd}\, g k_b T_0} \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{Nd}{g k_b T_0}\, \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right) \\ &= \underbrace{\frac{\zeta}{\sqrt{g k_b T_0}} \sqrt{\frac{2\pi Q}{Nd}}\, \exp\left(\frac{Nd\,C_1}{g k_b T_0}\right)}_{=\ \text{constant}\ \xi} \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{Nd}{g k_b T_0}\, \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right) \end{aligned}$$

Finally, the partition function has been demonstrated to have the form

$$Z(N,V,C_1) = \xi \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{Nd}{g k_b T_0}\, \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right) \tag{5.322}$$

where the constant factor ξ has the expression

$$\xi = \frac{\zeta}{\sqrt{g k_b T_0}} \sqrt{\frac{2\pi Q}{Nd}}\, \exp\left(\frac{Nd\,C_1}{g k_b T_0}\right) \tag{5.323}$$

Equation (5.322) is the canonical distribution function (modulo constant prefactors), provided that $g = Nd$.
Indeed, if $g = Nd$ then the partition function (5.322) becomes

$$g = Nd \quad \rightarrow \quad Z(N,V,C_1) = \xi \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right) \qquad \text{with} \quad \xi = \frac{\zeta\sqrt{2\pi Q}}{Nd\sqrt{k_b T_0}}\, \exp\left(\frac{C_1}{k_b T_0}\right)$$

[9] A Gaussian integral has the following solution :

$$I(a) = \int_{-\infty}^{\infty} e^{-a\,x^2}\, dx = \sqrt{\frac{\pi}{a}} \tag{5.321}$$

A similar result has been derived in Appendix A.11.1 for the Nosé-Hoover equations in terms of its real variables in a three dimensional space ($d = 3$). It is worth noting that for simple non Hamiltonian systems such as the Nosé-Hoover algorithm, it is possible to eliminate the metric factor by reformulating the equations of motion in terms of a variable $\tilde{\eta} = \exp(Nd\eta)$. In this case, the equations of motion will be incompressible, and the metric factor (5.315) will be constant. Of course, such an elementary transformation may not exist for other non Hamiltonian systems.

It is worth noting how the incomplete version of the Liouville equation, given by (4.188), can lead to an incorrect prediction of the phase space sampled by equations of motion (5.309) - (5.312). Indeed, for the Nosé-Hoover equations (5.309) - (5.312), the Liouville relation (4.188) takes the form

$$\frac{\partial \tilde{f}}{\partial t} + \dot{\mathbf{\Gamma}} \cdot \nabla_{\mathbf{\Gamma}} \tilde{f} = \frac{d\tilde{f}}{dt} = -\tilde{f}\, \nabla_{\mathbf{\Gamma}} \cdot \dot{\mathbf{\Gamma}} = Nd \left(\frac{p_\eta}{Q} \right) \tilde{f} \tag{5.324}$$

where, in the last expression, the result obtained in (5.314) has been used. The previous equation can be rewritten as

$$\frac{d \ln(\tilde{f})}{dt} = Nd \left(\frac{p_\eta}{Q} \right) \tag{5.325}$$

Note, however, that the quantity

$$\tilde{\mathcal{H}}(\boldsymbol{p}, \boldsymbol{q}, p_\eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{p_\eta^2}{2Q} \equiv \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_\eta^2}{2Q} \tag{5.326}$$

satisfies

$$\begin{aligned} \frac{d\tilde{H}}{dt} &= \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \frac{d\boldsymbol{p}_i}{dt} + \sum_{i=1}^{N} \frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \frac{d\boldsymbol{q}_i}{dt} + \frac{p_\eta}{Q} \frac{dp_\eta}{dt} \\ &= \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \left(-\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} - \frac{p_\eta}{Q} \boldsymbol{p}_i \right) + \sum_{i=1}^{N} \frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \frac{\boldsymbol{p}_i}{m_i} + \frac{p_\eta}{Q} \left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - g k_b T_0 \right) = -g k_b T_0 \frac{p_\eta}{Q} \end{aligned} \tag{5.327}$$

where the equations of motion (5.309), (5.310) and (5.312) have been used. Therefore, comparing the two time derivatives (5.325) and (5.327), and setting $g = Nd$, then the conclusion would be

$$\frac{d \ln(\tilde{f})}{dt} = -\beta \frac{d\tilde{H}}{dt} \tag{5.328}$$

$$\ln(\tilde{f}) = -\beta \tilde{H} + \text{ const} \tag{5.329}$$

$$\tilde{f} \propto \exp(-\beta\, \tilde{H}) = \exp\left[-\beta \left(\frac{p_\eta^2}{2Q} + \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) \right) \right] \tag{5.330}$$

where $\beta = 1/(k_b T_0)$. Using these equations, a distribution function of the form (5.330) was generated by the dynamics under all circumstances. However, it has been made clear that if more than one conservation law exists, linear dependent solutions are present and/or driven variables remain unidentified, the above procedure will fail to predict the correct distribution. Again, satisfying the generalized Liouville equation is a necessary but not sufficient condition to guarantee that a given distribution function is, in fact, generated by a dynamical system.

The previous derivation demonstrates that the Nosé-Hoover equations are capable of generating a canonical distribution in the physical subsystem variables when the Hamiltonian $\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_\eta, \eta)$, given by (5.313), is the *only* conserved quantity. Note that the basic assumption used is that there is only a single conservation law, namely the conservation of the Hamiltonian (5.313).
However, as demonstrated and discussed in Section 5.3.4.3, for the case of simulations with periodic boundary conditions, also the quantity (5.302) related to the total linear momentum is conserved, while in the absence of periodic boundary conditions, both the quantities (5.302) and (5.308), where the last one is related to the total angular momentum, are conserved (see Table 5.6). The conservation of these additional quantities will affect the phase space distribution. Therefore, a separate treatment of these two cases has to be performed, to study the form of the partition function with these additional conservation laws. The case of periodic boundary conditions, with the conserved quantities (5.296) and (5.302), will be first analyzed in Section 5.3.4.5, since it is the most important case in practical simulations.

5.3.4.5 Statistical mechanical ensemble under periodic boundary conditions

In this section, the form of the partition function describing a system without external forces (i.e. in which the condition (5.297) holds) and modeled using periodic boundary conditions is analyzed. In this case, two constants of motion have to be taken into account (see Table 5.6), namely, the total Hamiltonian (5.296) and the quantity (5.302) related to the total linear momentum.
To continue the analysis following Section 4.3.2.1, the driven variables have to be eliminated from the system. The center of mass position $\mathbf{R}$ is a driven variable (its dynamics does not effect other variables, and it does not contribute to a nontrivial conserved quantity) and must also be eliminated in the formal analysis. On the other hand, the magnitude of the center of mass momentum is coupled to the other variables through a conservation law and cannot be eliminated from the analysis. At the same time, the components of the center of mass momentum $\mathbf{P}$ are linearly dependent.[10] Thus, $d-1$ components of the center of mass momentum must be eliminated. Therefore of the d components only one component can be chosen independently, otherwise the variable

$$P = \|\mathbf{P}\| = \left(\sum_{\alpha=1}^{d} P_\alpha^2 \right)^{1/2} \tag{5.332}$$

can be taken as the independent variable. Before proceeding further on, the equation of motion for this new variable can be easily found by taking into account the definition of the conserved quantity, equation (5.298), and rewriting it in terms of the new variable P defined in (5.332), by taking the Euclidean norm on both sides of equation (5.298), so that

$$\|\mathbf{P}\, e^{\eta}\| = P\, e^{\eta} = \|\mathbf{K}\| = K \tag{5.333}$$

Therefore, the norm of the vector $\mathbf{P}$ defined in (5.332) can be written as

$$P = e^{-\eta}\, K \tag{5.334}$$

Taking the time derivative on both member of the previous equivalence, and using the equation of motion (5.311) leads to

$$\dot{P} = -\dot{\eta}\, e^{-\eta}\, K = -\dot{\eta}\, P = -\frac{p_\eta}{Q}\, P \tag{5.335}$$

that is the equation of motion which rules the evolution in time of the norm of the total linear momentum of the system. Proceeding with the analysis, the two variables $\mathbf{R}$ and $\mathbf{P}$ can be eliminated by considering the positions and momenta relative to the center of mass of the system, $\boldsymbol{\rho}$ and $\boldsymbol{\pi}$, respectively. Thus, a canonical transformation to a set of relative coordinates and momenta $\{\boldsymbol{\pi}, \mathbf{P}, \boldsymbol{\rho}, \mathbf{R}\}$ (Jacobi coordinates) has to be introduced, in which $\mathbf{R}$ and $\mathbf{P}$ are explicit variables, and the equations of motion for the $\mathbf{P}$, written in terms of the single independent variable P. Starting from the equation of motion (5.309) - (5.312), this procedure yields the following transformed equations of motion

$$\frac{d\boldsymbol{\rho}_i}{dt} = \frac{\boldsymbol{\pi}_i}{\mu_i} \qquad i = 1, ..., N-1 \tag{5.336}$$

$$\frac{d\boldsymbol{\pi}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \frac{p_\eta}{Q}\, \boldsymbol{\pi}_i \qquad i = 1, ..., N-1 \tag{5.337}$$

$$\frac{dP}{dt} = -\frac{p_\eta}{Q}\, P \tag{5.338}$$

$$\frac{d\eta}{dt} = \frac{p_\eta}{Q} \tag{5.339}$$

$$\frac{dp_\eta}{dt} = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{\mu_i} + \frac{P^2}{M} - g k_b T_0 \tag{5.340}$$

[10] To see this, consider the components of equation (5.298) in a three dimensional system ($d = 3$),

$$\frac{P_x}{K_x} = \frac{P_y}{K_y} = \frac{P_z}{K_z} = e^{\eta} \tag{5.331}$$

which shows that only one of the components is independent.

where μ_i is the reduced mass of the system,[11] and the equation of motion for the momentum $\mathbf{P}$, written in terms of the single independent variable P as defined in (5.332), has been introduced in equation (5.338), following the result obtained in (5.335). The equations of motion have two conservative laws (because the d center of mass momenta components have been replaced by a single variable P only one conservation law for the momenta is left), namely,

$$\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}, p_\eta, \eta) = \mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta = C_1 \tag{5.343}$$

$$P\, e^\eta = C_2 \tag{5.344}$$

where in the first conservation law (5.343), the Hamiltonian on the left hand side is given by

$$\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} + \frac{P^2}{2M} + \phi(\boldsymbol{\rho}) \tag{5.345}$$

In order to compute the partition function, the compressibility has to be calculated, using the phase space vector $\boldsymbol{\Xi} = (\boldsymbol{\pi}, P, \boldsymbol{\rho}, p_\eta, \eta)$ and the definition (4.123), as

$$\begin{aligned}
\kappa(\boldsymbol{\Xi}, t) \equiv \nabla_{\boldsymbol{\Xi}} \cdot \dot{\boldsymbol{\Xi}} &= \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\rho}_i} \cdot \dot{\boldsymbol{\rho}}_i + \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} + \nabla_\eta \cdot \dot{\eta} + \nabla_{p_\eta} \cdot \dot{p}_\eta \\
&= \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} = \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \left(-\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \frac{p_\eta}{Q} \boldsymbol{\pi}_i \right) - \frac{p_\eta}{Q} \\
&= -(N-1)d \left(\frac{p_\eta}{Q} \right) - \frac{p_\eta}{Q} = -(N-1)d\, \dot{\eta} - \dot{\eta} = -[(N-1)d + 1]\, \dot{\eta}
\end{aligned} \tag{5.346}$$

From the compressibility, the metric follows directly

$$\begin{aligned}
\sqrt{g(\boldsymbol{\Xi}, t)} &= \exp\left(-\int \kappa(\boldsymbol{\Xi}, t)\, dt \right) = \exp\left(\int [(N-1)d + 1]\ \frac{d\eta}{dt}\, dt \right) \\
&= \exp\left\{ [(N-1)d + 1] \int \frac{d\eta}{dt}\, dt \right\} = \exp\{ [(N-1)d + 1]\, \eta \}
\end{aligned} \tag{5.347}$$

The conservation laws and the metric can now be used to construct the microcanonical partition function, which contains two delta functions that express the two conservation laws

$$\begin{aligned}
&Z(N, V, C_1, C_2) \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta \int dp_\eta\ e^{[(N-1)d+1]\,\eta}\ \delta\left(\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1 \right) \delta(e^\eta\, P - C_2)
\end{aligned} \tag{5.348}$$

where $d\boldsymbol{\rho} = d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}$ and $d\boldsymbol{\pi} = d\boldsymbol{\pi}_1 \cdots d\boldsymbol{\pi}_{N-1}$ and the Hamiltonian $\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho})$ is given by equation (5.345). The distribution function in the physical subspace is obtained by integrating over η and p_η. Since the argument of the second δ function in the above equation (5.348) has only one zero as a function of the variable η, a convenient equivalence relation can be employed to resolve the integral, given by equation (5.318), with

$$f(\eta) = e^\eta\, P - C_2 \qquad\qquad f'(\eta) = e^\eta\, P \tag{5.349}$$

[11] The canonical transformation to a set of relative coordinates and momenta $\{\boldsymbol{\pi}, \mathbf{P}, \boldsymbol{\rho}, \mathbf{R}\}$ leads to a different expression in the kinetic part of the equations of motion and of the Hamiltonian. By introducing the center of mass momenta,

$$\boldsymbol{\alpha}_i = \boldsymbol{p}_i - \mathbf{P}/N \tag{5.341}$$

the kinetic energy term becomes a sum of the relative kinetic energy and the center of mass kinetic energy as follows

$$\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} = \sum_{i=1}^{N} \frac{\boldsymbol{\alpha}_i^2}{2m_i} + \frac{\mathbf{P}^2}{2M} \qquad \text{where} \quad M = \sum_{i=1}^{N} m_i \tag{5.342}$$

$$\eta_0 = \ln\left(\frac{C_2}{P}\right) \qquad\qquad f'(\eta_0) = C_2 \tag{5.350}$$

where η_0 is the zero of function $f(\eta)$ and $f'(\eta_0)$ is the derivative of the function $f(\eta)$ evaluated in the point η_0, which has to be read, inside the integral, as the function $f'(\eta)$. Therefore, using the relation (5.318) in (5.348), together with equations (5.349) and (5.350), the partition function becomes

$$\begin{aligned}
&Z(N,V,C_1,C_2) \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta \int dp_\eta \; e^{[(N-1)d+1]\,\eta} \; \delta\left(\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1\right) \frac{\delta(\eta-\eta_0)}{e^\eta\, P} \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int dp_\eta \; e^{[(N-1)d+1]\,\eta_0} \; \delta\left(\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta_0 - C_1\right) \frac{1}{e^{\eta_0}\, P}
\end{aligned}$$

The substitution of the variable η_0 with its explicit expression, given by the first equation in (5.350), yields the following form for the partition function

$$\begin{aligned}
&Z(N,V,C_1,C_2) \\
&= \frac{\zeta}{C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int dp_\eta \left(\frac{C_2}{P}\right)^{(N-1)d+1} \delta\left(\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \ln\left(\frac{C_2}{P}\right) - C_1\right)
\end{aligned} \tag{5.351}$$

The remaining delta function has an argument that can be viewed as a function of the variable p_η, so that

$$f(p_\eta) = \mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \ln\left(\frac{C_2}{P}\right) - C_1 \qquad\qquad f'(p_\eta) = \frac{p_\eta}{Q} \tag{5.352}$$

The previous function $f(p_\eta)$ has only one zero $p_{n,0}$ with respect to the variable p_η, that is

$$p_{n,0} = \left\{2Q\left[C_1 - \mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) - g k_b T_0 \ln\left(\frac{C_2}{P}\right)\right]\right\}^{1/2} \tag{5.353}$$

$$f'(p_{n,0}) = \left\{\frac{2}{Q}\left[C_1 - \mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) - g k_b T_0 \ln\left(\frac{C_2}{P}\right)\right]\right\}^{1/2} \tag{5.354}$$

Using again the relation (5.318), the integration over the variable p_η results in the following expression for the partition function

$$\begin{aligned}
&Z(N,V,C_1,C_2) \\
&= \frac{\zeta}{C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int dp_\eta \left(\frac{C_2}{P}\right)^{(N-1)d+1} \left[Q\, \frac{\delta(p_\eta - p_{n,0})}{p_\eta}\right] \\
&= \frac{\zeta\, Q}{C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{C_2}{P}\right)^{(N-1)d+1} \left[\frac{1}{p_{n,0}}\right]
\end{aligned}$$

The substitution of the explicit expression for $p_{n,0}$ given by equation (5.353) in the previous integrand leads the following form for the partition function

$$\begin{aligned}
&Z(N,V,C_1,C_2) \\
&= \frac{\zeta\sqrt{Q}}{\sqrt{2}\, C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{C_2}{P}\right)^{(N-1)d+1} \left[C_1 - \mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) - g k_b T_0 \ln\left(\frac{C_2}{P}\right)\right]^{-1/2}
\end{aligned} \tag{5.355}$$

Clearly, this is not the partition function for a canonical ensemble. This problem was first pointed out by Cho and coworkers.[63] Moreover, the phase space distribution function given in equation (5.355) may contain forbidden regions or islands and sharp boundaries corresponding to negative or zero argument of the square root. Only in the case that $C_2 = 0$ the conventional Nosé-Hoover equations of motion can generate a canonical distribution. Indeed, if $P\, e^\eta = C_2 = 0$ then the partition function (5.348) becomes

$$\begin{aligned}
&Z(N,V,C_1,0) \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta \int dp_\eta \; e^{[(N-1)d+1]\,\eta} \; \delta\left(\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1\right) \delta(e^\eta\, P - 0)
\end{aligned} \tag{5.356}$$

The second delta function in the previous equation imposes that $P = 0$. Using the relation (5.318) with

$$f(P) = e^{\eta}\, P \qquad\qquad f(P_0) = 0 \;\;\leftrightarrow\;\; P_0 = 0 \qquad\qquad f'(P) = f'(P_0) = e^{\eta} \tag{5.357}$$

Integration over the variable P then yields

$$\begin{aligned} Z(N,V,C_1,0) &= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int d\eta \int dp_\eta \; e^{[(N-1)d+1]\,\eta}\, e^{-\eta}\; \delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1\right) \\ &= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int d\eta \int dp_\eta \; e^{(N-1)d\,\eta}\; \delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} + g k_b T_0 \eta - C_1\right) \end{aligned} \tag{5.358}$$

where $\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) \equiv \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho},0)$, that is $\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})$ is the Hamiltonian in equation (5.345) with $P = 0$. The remaining delta function can be rewritten using the relation (5.318) with

$$\eta_0 = -\frac{1}{g k_b T_0}\left[\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} - C_1\right] \qquad\qquad \text{and} \qquad\qquad f'(\eta) = f'(\eta_0) = g k_b T_0 \tag{5.359}$$

so that the partition function (5.358) can be rewritten as

$$Z(N,V,C_1,0) = \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int d\eta \int dp_\eta \; e^{(N-1)d\,\eta}\; \frac{\delta(\eta-\eta_0)}{g k_b T_0} = \frac{\zeta}{g k_b T_0} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dp_\eta \; e^{(N-1)d\,\eta_0}$$

Then, the substitution of the explicit expression for the variable η_0 in the previous integrand, with η_0 given by the first equation in (5.359), leads to a partition function of the form

$$Z(N,V,C_1,0) = \frac{\zeta}{g k_b T_0} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dp_\eta \;\; \exp\!\left\{-\frac{(N-1)d}{g k_b T_0}\;\left[\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) + \frac{p_\eta^2}{2Q} - C_1\right]\right\} \tag{5.360}$$

$$= \underbrace{\frac{\zeta}{g k_b T_0} \exp\!\left[\frac{(N-1)d}{g k_b T_0}\; C_1\right] \int dp_\eta \exp\!\left(-\frac{(N-1)d}{g k_b T_0}\;\frac{p_\eta^2}{2Q}\right)}_{=\text{ constant } \xi} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \;\; \exp\!\left[-\frac{(N-1)d}{g k_b T_0}\; \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})\right]$$

where all the constant factors are collected by defining the constant ξ and the Hamiltonian $\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) \equiv \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho},0)$, that is $\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})$ is the Hamiltonian in equation (5.345) with $P = 0$, written as

$$\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho}) \equiv \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho},0) = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\mu_i} + \phi(\boldsymbol{\rho}) \tag{5.361}$$

The integral with respect to the variable p_η is a Gaussian integral and it can be easily solved (see equation (5.321) in the footnote 9), so that

$$\begin{aligned} Z(N,V,C_1,0) &= \frac{\zeta}{g k_b T_0} \exp\!\left[\frac{(N-1)d}{g k_b T_0}\; C_1\right] \sqrt{\frac{2\pi Q g k_b T_0}{(N-1)d}} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \;\; \exp\!\left[-\frac{(N-1)d}{g k_b T_0}\; \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})\right] \\ &= \underbrace{\frac{\zeta\sqrt{2\pi Q}}{\sqrt{g k_b T_0 (N-1)d}} \;\exp\!\left[\frac{(N-1)d}{g k_b T_0}\; C_1\right]}_{=\text{ constant } \xi} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \;\; \exp\!\left[-\frac{(N-1)d}{g k_b T_0}\; \mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})\right] \end{aligned} \tag{5.362}$$

Clearly, imposing $g = (N-1)d$ in the partition function (5.362), it becomes

$$\begin{aligned} g = (N-1)d \qquad \rightarrow \qquad & Z(N,V,C_1,0) = \xi \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \;\; \exp\!\left(-\frac{\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})}{k_b T_0}\right) \equiv Z_{(N-1)VT} \\ & \text{with} \qquad \xi = \frac{\zeta\sqrt{2\pi Q}}{(N-1)d\sqrt{k_b T_0}} \;\exp\!\left(\frac{C_1}{k_b T_0}\right) \end{aligned} \tag{5.363}$$

so that the correct canonical partition function is recovered. Therefore, for the special choice $C_2 = 0$, corresponding to the choice $P(0) = 0$, the previous analysis leads to an (N−1)VT ensemble distribution. The same results have been obtained by K. Cho et al.,[63] taking into account the periodic boundary conditions applied in the simulation, see Appendix A.7, Section A.7.1. In practice, most conventional Nosé-Hoover simulations are performed with a fixed center of mass and therefore obey the condition $P = 0$.

5.3.4.6 Uniqueness of Nosé-Hoover equations of motion

An important implication of the Nosé equations is that in the Hamiltonian (5.272) a logarithmic term $\ln s$ is required to have the correct scaling of time. Any other scheme that does not have such a logarithmic term will fail to describe the canonical ensemble correctly. An important result obtained by Hoover[58] is that the equations of motion (5.263)-(5.266) are unique, in the sense that other relaxation equations of similar form cannot lead to a canonical distribution.
Because the variables $\boldsymbol{q}$, $\boldsymbol{p}$ and ξ used in (5.263)-(5.266) are independent, the components of the flow of probability density $f(\boldsymbol{p},\boldsymbol{q},\xi)$ in the phase space can be easily computed. The equations (5.263)-(5.266) governing the motion in this space are not Hamiltonian. Therefore the derivatives $\partial\dot{\boldsymbol{q}}_i/\partial\boldsymbol{q}_i$ and $\partial\dot{\boldsymbol{p}}_i/\partial\boldsymbol{p}_i$ $(i = 1, ..., N)$ do not generally sum to zero. Thus the analog of Liouville equation, expressing the conservative flow of probability with time, including flow in the ξ direction, is given by equation (5.181) with $\boldsymbol{\Gamma} = (\boldsymbol{p},\boldsymbol{q},\xi)$, that can be written more explicitly as

$$\frac{\partial f}{\partial t} + \sum_{i=1}^{N}\left(\dot{\boldsymbol{q}}_i\,\frac{\partial f}{\partial \boldsymbol{q}_i} + \dot{\boldsymbol{p}}_i\,\frac{\partial f}{\partial \boldsymbol{p}_i}\right) + \dot{\xi}\,\frac{\partial f}{\partial \xi} + f\left[\sum_{i=1}^{N}\left(\frac{\partial \dot{\boldsymbol{q}}_i}{\partial \boldsymbol{q}_i} + \frac{\partial \dot{\boldsymbol{p}}_i}{\partial \boldsymbol{p}_i}\right) + \frac{\partial \dot{\xi}}{\partial \xi}\right] = 0 \tag{5.364}$$

Consider now the density function $f(\boldsymbol{p},\boldsymbol{q},\xi)$ derived in equation (A.293) in the real system frame of reference, which enters the canonical partition function as

$$Z = \frac{\zeta}{gk_bT_0}\int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\frac{Nd}{gk_bT_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E\right)\right] \tag{5.365}$$

where $\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})$ is given by equation (A.295) and the virtual variable $\tilde{p}_s$ can be transformed in real frame of reference through $\tilde{p}_s = sp_s$. Using the relation (5.262), so that $\tilde{p}_s = Q\xi$, the previous equation can be rewritten as

$$\begin{aligned} Z &= \frac{\zeta Q}{gk_bT_0}\int d\boldsymbol{p}\int d\boldsymbol{q}\int d\xi\ \exp\left[-\frac{Nd}{gk_bT_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{Q\xi^2}{2} - E\right)\right] \\ &= \frac{\zeta Q}{gk_bT_0}\exp\left(\frac{Nd}{gk_bT_0}\,E\right)\int d\boldsymbol{p}\int d\boldsymbol{q}\int d\xi\ \exp\left[-\frac{Nd}{gk_bT_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{Q\xi^2}{2}\right)\right] \end{aligned} \tag{5.366}$$

The partition function can be shortly written as

$$Z = \int d\boldsymbol{p}\int d\boldsymbol{q}\int d\xi\ f(\boldsymbol{p},\boldsymbol{q},\xi) \tag{5.367}$$

where the distribution function, taking $g = Nd$ in (5.366) and written explicitly as in (A.295) the form of the Hamiltonian $\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})$, is given by

$$f(\boldsymbol{p},\boldsymbol{q},\xi) = C\exp\left[-\frac{1}{k_bT_0}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{Q\xi^2}{2}\right)\right] \tag{5.368}$$

where C is a constant that collects all the constant terms before the integrals of the last expression in (5.366). Only four derivative terms in (5.364) are non vanishing, and they can be computed from the density function (5.368) as follows

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{q}_i} &= -\frac{1}{k_bT_0}\frac{\partial\phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i}\,f(\boldsymbol{p},\boldsymbol{q},\xi) & \frac{\partial f}{\partial \boldsymbol{p}_i} &= -\frac{1}{k_bT_0}\frac{\boldsymbol{p}_i}{m_i}\,f(\boldsymbol{p},\boldsymbol{q},\xi) \\ \frac{\partial f}{\partial \xi} &= -\frac{1}{k_bT_0}Q\,\xi\,f(\boldsymbol{p},\boldsymbol{q},\xi) & \frac{\partial \dot{\boldsymbol{p}}_i}{\partial \boldsymbol{p}_i} &= -3\xi \end{aligned} \tag{5.369}$$

where for the last equality the equation of motion (5.264) has been used. Plugging these derivatives in (5.364) and using the equations of motion (5.263)-(5.266) in the real frame of reference, the Liouville

formula given by (5.364) can be rewritten as

$$-\frac{1}{k_bT_0}\left[\sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i}f(\boldsymbol{p},\boldsymbol{q},\xi)+\sum_{i=1}^{N}\left(-\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i}-\xi\,\boldsymbol{p}_i\right)\frac{\boldsymbol{p}_i}{m_i}f(\boldsymbol{p},\boldsymbol{q},\xi)+\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}-gk_bT_0\right)\xi\,f(\boldsymbol{p},\boldsymbol{q},\xi)\right.$$
$$\left.+k_bT_0\,f(\boldsymbol{p},\boldsymbol{q},\xi)\sum_{i=1}^{N}(3\xi)\right]=-\frac{1}{k_bT_0}\left[\sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i}f(\boldsymbol{p},\boldsymbol{q},\xi)-\sum_{i=1}^{N}\frac{\boldsymbol{p}_i}{m_i}\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i}f(\boldsymbol{p},\boldsymbol{q},\xi)\right.$$
$$\left.-\xi\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}f(\boldsymbol{p},\boldsymbol{q},\xi)+\xi\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}f(\boldsymbol{p},\boldsymbol{q},\xi)-gk_bT_0\,\xi\,f(\boldsymbol{p},\boldsymbol{q},\xi)+Ndk_bT_0\,\xi\,f(\boldsymbol{p},\boldsymbol{q},\xi)\right]\overset{g=Nd}{=}0$$

So that these terms sum to zero, provided that the coefficient g in the equation of motion (5.266) for the friction coefficient ξ is chosen equal to the number of independent degrees of freedom in the set ($\boldsymbol{p}$, $\boldsymbol{q}$), namely, equal to $g = Nd$ as in (5.368).[12] Thus the canonical distribution (5.368) is a steady equilibrium solution of the flow equation (5.364) and satisfies the equations of motion (5.263)-(5.266). It is important to note that the phase space distribution (5.368) can be used to derive the equation of motion for the friction coefficient ξ. To see this, note that the canonical distribution (5.368) satisfies (5.364) if, and only if, ξ follows the relaxation equation (5.266). Thus Nosé canonical equations of motion are unique. Other relaxation equations, such as Berendsen equation of motion, cannot lead to the canonical distribution (5.368).

5.3.4.7 Dynamical properties

The correct canonical ensemble averages for thermodynamic quantities can be obtained using the Nosé-Hoover constant temperature method. Because the equations of motion are solved numerically, the dynamic properties of the system can be studied. This is a feature of the molecular dynamics method. Therefore, it is a natural question whether the correct dynamical behaviors are also obtainable or not at constant temperature. If the simulations are carried out in an appropriate condition, the answer is affirmative in most cases.
The relaxation from a non-equilibrium state to equilibrium in constant temperature simulations does not correspond to any realistic process in the experiments, it is instead a process introduced artificially to control the temperature. The speed of the relaxation is determined by the value of mass Q associated to the thermostat degrees of freedom in the extended system method. The response is quick with a small mass, while the system relaxes slowly with a large mass. Also a large amplitude slowly decaying continuous oscillation of the heat bath variable s is often observed. In a certain condition, the coupling between the variable s and a physical system is not so strong. It takes quite a long time for the damping of this oscillation. All the behaviors mentioned above are artifacts introduced by the extended system method. The rate of collision with hypothetical particles or the random forces acting on a particle in the stochastic constant temperature method also affect the relaxation. Therefore, the correct dynamical behavior in non-equilibrium state cannot be obtained by modified simulation methods.
In this section, several problems related to dynamical behaviors in the constant temperature methods are considered. In Section 5.3.4.7, the response speed of a thermostat in equilibrium is revised. This is necessary for determining an appropriate mass value associated to the heat bath variable. Indeed, a good choice of this quantity is important to realize the condition of thermal equilibrium between a physical system and a heat bath.

Choice of values for mass parameters In order to derive a set of equations of motion that includes the heat bath variable in the extended system formulation, an artificial kinetic energy term for the external system is introduced. The mass parameter appearing in this term controls the speed of response of a heat bath. It is of great importance to choose an appropriate value for this parameter in dynamics simulations. S. Nosé[47] pointed out that the mass parameter should not be too small. In the study of the lattice vibration in a system containing impurities, it is known that a localized isolated mode appears when the mass of the impurities is smaller than that of host atoms. A similar situation occurs

[12] In the usual molecular dynamics simulation, with periodic boundaries, the center of mass and its velocity are fixed so that this number of degrees of freedom is $d(N-1)$ for a d-dimensional N-body system.

in simulation of a system when Q is small. The heat bath variable becomes an isolated mode and it continues an oscillation independently. The behavior resembles that of a harmonic oscillator. The distribution of the total kinetic energy driven by this oscillation deviates significantly from the Gaussian distribution as shown schematically in Figure 5.5(a). The distribution of a harmonic oscillator has two peaks at turning points: these correspond to two peaks at high and low temperature in the kinetic energy distribution. The variable s oscillates independently from other degrees of freedom. Therefore, the system does not reach an equilibrium state during the simulation, and the ergodic hypothesis is not satisfied.

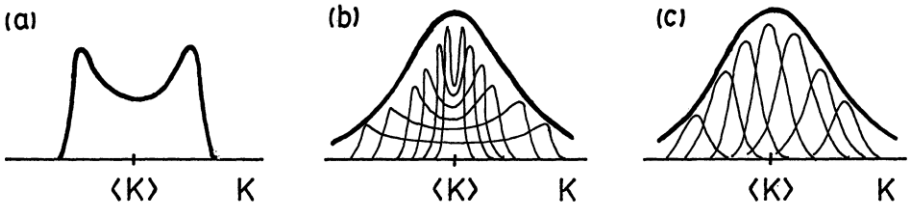

Figure 5.5: Change in the distribution of the total kinetic energy with different values of the mass parameter Q; (a) very small, (b) small and (c) large values. Thin lines indicate the distribution during a short period.

With a larger mass Q, the behavior changes. Typical examples of the time evolution of the temperature (kinetic energy) are depicted in Figure 5.6. The results are obtained by S. Nosé[47] with a 108 particle Lennard-Jones system (for more details about simulation parameters see Ref. [47]). With a smaller mass ($Q = 1.0$, see Figure 5.6, upper panel), the change of the kinetic energy is still mostly driven by the fast oscillation due to the heat bath variable s. The amplitude of the oscillation is due to a coupling between the heat bath and the physical system. In this case, the accumulation of the distribution of harmonic oscillators with various amplitudes forms a Gaussian distribution with a single peak at an average value (Figure 5.5(b)).
At larger mass (see Figure 5.6, lower panel), fast thermal fluctuations of the kinetic energy around a slow systematic change due to the heat bath variable oscillation is observed. A distribution of Gaussian form, but narrower than the Maxwell distribution at temperature T_0 is expected in a short period. The situation is similar to that in the microcanonical ensemble, because the exchange of the heat is slow. The centers of mass of the distribution oscillate synchronizing the oscillation of the variable s. An envelope of these distribution recovers the Maxwell distribution form (see 5.5(c)). Using a large value for the mass Q, a large number of time steps are necessary to get reliable average values from dynamics trajectory.

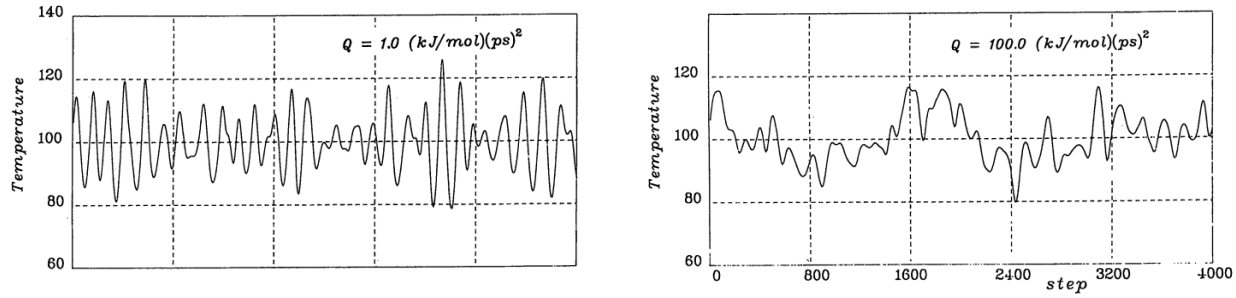

Figure 5.6: Comparison of the temperature fluctuations with different Q values, for a 108 Lennard-Jones system at $T = 100$ K. Values of $Q = 1.0$ (kJ/mol)ps^2 (left panel) and $Q = 100.0$ (kJ/mol)ps^2 (right panel) are considered.

On the base of this behavior, an intermediate value for the mass Q related to the thermostat is the most appropriate one. To obtain a criterion for deciding the entity of intermediate mass Q value for a given system, a relation between the number of atoms or some other quantity related to the system and the value of thermostat mass Q has to be established. In this sense, it is very helpful to analyze the period of oscillation of the variable s with the linearization of equation (5.266) which is reported below

$$\frac{d\xi}{dt} = \frac{1}{Q}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0\right) \tag{5.370}$$

where the variable ξ is related to s through relation (5.262) and a virtual frame of reference can be used by means of transformations (5.257), so that using the equivalences

$$\xi = \frac{1}{s}\left(\frac{ds}{dt}\right) = \frac{s\, p_s}{Q} \qquad\qquad s = \tilde{s} \qquad\qquad \boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i/s \tag{5.371}$$

the equation of motion (5.370) becomes

$$Q\,\frac{d}{dt}\left(\frac{1}{s}\,\frac{ds}{dt}\right) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, s^2} - g k_b T_0 \tag{5.372}$$

Consider a fluctuation δs of the variable s around an average value $\langle s \rangle$, that is

$$s = \langle s \rangle + \delta s \tag{5.373}$$

At a small mass limit, the change of s is much faster than that of the particles, and the change of the momentum in a virtual frame can thus be ignored. The constant temperature condition is mainly maintained by the motion of s, namely,

$$\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i \langle s \rangle^2} = g k_b T_0 \tag{5.374}$$

By manipulating equation (5.372) in the following way,

$$Q\,\frac{d}{dt}\left(\frac{1}{s}\,\frac{ds}{dt}\right) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, s^2} - g k_b T_0 \tag{5.375}$$

$$-\frac{1}{s^2}\left(\frac{ds}{dt}\right)^2 + \frac{1}{s}\,\frac{d^2 s}{dt^2} = \frac{1}{Q}\left(\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, s^2} - g k_b T_0\right) \tag{5.376}$$

and by linearizing it as follows

$$Q\left[-\frac{1}{(\langle s \rangle + \delta s)^2}\left(\frac{d(\langle s \rangle + \delta s)}{dt}\right)^2 + \frac{1}{\langle s \rangle + \delta s}\,\frac{d^2(\langle s \rangle + \delta s)}{dt^2}\right] = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\,(\langle s \rangle + \delta s)^2} - g k_b T_0 \tag{5.377}$$

$$Q\left[-\frac{1}{\langle s \rangle^2}\left(1 - \frac{2\delta s}{\langle s \rangle}\right)\left(\frac{d\delta s}{dt}\right)^2 + \frac{1}{\langle s \rangle}\left(1 - \frac{\delta s}{\langle s \rangle}\right)\left(\frac{d^2 \delta s}{dt^2}\right)\right] = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, \langle s \rangle^2}\left(1 - \frac{2\delta s}{\langle s \rangle}\right) - g k_b T_0 \tag{5.378}$$

By neglecting the first two terms that contain the first derivative squared, and by neglecting the second term in the second parenthesis on the left hand side, the previous equation can be rewritten as

$$\frac{Q}{\langle s \rangle}\left(\frac{d^2 \delta s}{dt^2}\right) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, \langle s \rangle^2}\left(1 - \frac{2\delta s}{\langle s \rangle}\right) - g k_b T_0 \tag{5.379}$$

Finally, using the expression (5.374) for the last term of the previous equation, it becomes

$$\frac{Q}{\langle s \rangle}\left(\frac{d^2 \delta s}{dt^2}\right) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\, \langle s \rangle^2}\left(1 - \frac{2\delta s}{\langle s \rangle}\right) - \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i \langle s \rangle^2} = -\frac{2}{\langle s \rangle}\left(\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i \langle s \rangle^2}\right)\delta s = -\frac{2 g k_b T_0}{\langle s \rangle}\,\delta s \tag{5.380}$$

leading to the equation for the harmonic oscillator

$$\frac{d^2 \delta s}{dt^2} = -\frac{2 g k_b T_0}{Q}\,\delta s = -\omega^2\,\delta s \tag{5.381}$$

with frequency

$$\omega = \sqrt{\frac{2 g k_b T_0}{Q}} \tag{5.382}$$

and period τ equal to

$$\tau = \frac{2\pi}{\omega} = 2\pi \sqrt{\frac{Q}{2gk_bT_0}} \tag{5.383}$$

Therefore, the value of Q can be deduced from the previous expression and computed using the formula

$$Q = 2gk_bT_0 \left(\frac{\tau}{2\pi}\right)^2 \tag{5.384}$$

Using these results, a general criterion for an appropriate choice of the heat bath mass Q value can be given. It should be an intermediate value, since a small Q does not guarantee the equilibration in a whole system, and a large Q is instead inefficient. The most economical choice is to agree the oscillation of the variable s with a typical oscillation period of a physical system. A maximum coupling is generally expected at a resonant condition. The characteristic time τ_0 of a system does not depend much on a system size. An appropriate choice for Q will be calculated from the dependence of the heat bath mass on the system size and temperature as derived in (5.384), that is, with a direct proportionality on the number of particles through g and on the temperature. Therefore, in principle any finite (positive) Q value is guaranteed to generate a canonical ensemble. In practice, however, the heat bath mass value must be chosen carefully. Values of Q too large lead to poor temperature control, in this case the canonical ensemble can only be obtained after very long simulation time. In the limit $Q \to \infty$, the NVE ensemble is recovered. Too small value of Q will result in high-frequency temperature oscillation, which might be off-resonance with the characteristic vibrational frequencies (phonon frequencies) of the real system and effectively decouple from the real system. Finally, when aiming at a precise measurement of dynamical properties, such as diffusion or vibrations, the use of a larger value of Q or the running of a NVE ensemble simulation should be considered, in order to avoid artifacts of the thermostat.

5.3.5 Nosé-Hoover chains

In the previous section, the dynamics and equations of motion derived from an extended Hamiltonian have been demonstrated to give canonically distributed positions and momenta. It is important here to underline that the proof does not guarantee that the system is ergodic and that the correct limiting distribution will be generated, on the contrary, it is the proof itself which involves the assumption that the dynamics itself is ergodic or, in other words, that the trajectory average can be taken into the phase space average. Therefore the method works extremely well in large (ergodic) systems. However, it has been shown for example in Ref. [58] that for small or stiff systems the dynamics is not ergodic and the correct distributions are not generated.
More rigorously, in Ref. [64] a possible reason for the nonergodicity of Nosé original formulation is discussed, and a simple modification based on this analysis is presented and tested on model problems. The method proposed in Ref. [64] succeeds precisely where the original formulation fails, with the advantage to preserves the simplicity of the original approach.
Other methods can also be used to obtain canonical distributions on stiff systems,[45, 65, 66, 67] some of which are variations and generalizations of the original Nosé-Hoover approach.[65, 66, 67] In the general method of Kusnezov et al.,[65] the choice of functions and parameters is somewhat arbitrary and the algorithm is difficult to apply in complex situations such as constant pressure simulations. The method of Andersen,[45] which involves stochastic collisions (at intervals some or all of the velocities are resampled according to the Boltzmann distribution) will also give the canonical ensemble, even in the most trivial systems.[45] However, this method has the disadvantage that it is not a continuous dynamics with well defined conserved quantities. Hence the same trajectory cannot easily be reproduced from the same initial conditions.
Moreover, as demonstrated in Section 5.3.4.5, if more than one conservation law is obeyed by the system, than the standard Nosé-Hoover equations of motion generate a partition function with the form (5.355), which is clearly not equal to a canonical partition function.
The reason for the failure of the Nosé-Hoover equations when more than one conservation law is obeyed by the system is that the equations of motion do not contain a sufficient number of variables in the extended phase space to offset the restrictions placed on the accessible phase space caused by multiple conservation laws. Each conservation law restricts the accessible phase space by one dimension. In order to counterbalance this effect, more phase space dimensions must be introduced, which can be accomplished by introducing additional variables. The question is now how should these variables be added so as to give the correct distribution in the physical phase space. The answer can be gleaned from the fact that the momentum variable $p_\eta \equiv p_{\eta_1}$ in the Nosé-Hoover equations must have a Maxwell-Boltzmann distribution, just as the physical momenta do. In order to ensure that such a distribution is generated, $p_\eta \equiv p_{\eta_1}$ itself can be coupled to a Nosé-Hoover-type thermostat, which will bring in a new set of variables, η_2 and p_{η_2}. But once this is done, the problem arises that p_{η_2} must also have a Maxwell-Boltzmann distribution, which requires introducing a thermostat for this variable. This reasoning can be continued ad infinitum, but the procedure must terminate at some point. If it is terminated after the addition of M new thermostat variable pairs η_μ and p_{η_μ} with $\mu = 1, ..., M$, then a finite set of new equations of motion can be defined for these variables.
Following the reasoning of Ref. [64], it can be observed that the distribution in the partition function (5.360) has a Gaussian dependence on the particle momenta $\boldsymbol{\pi}$, as well as on the thermostat momenta p_η. While the Gaussian fluctuations of $\boldsymbol{\pi}$ are driven with a thermostat, there is nothing to drive the fluctuations of p_η. Although the positions $\boldsymbol{\rho}$ and momenta $\boldsymbol{\pi}$ are of primary interest, if the Nosé-Hoover equations are ergodic, the system covers the whole phase space, which includes the space of the thermostat velocities. The fluctuations of the thermostat variables, which clearly occur in ergodic systems, may even be important in driving the system to fill phase space (the dynamical equations are, of course, coupled). This suggests thermostating p_η and, by analogy, the thermostat of p_η plus its thermostat, and so on, to form a chain.

5.3.5.1 Equations of motion in real variables

According to the theoretical framework described at the beginning of this section, the dynamics of the Nosé-Hoover chain method can be expressed taking the Nosé-Hoover equations of motion (5.309)-(5.312)

and including additional thermostat momenta degrees of freedom, as follows

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.385}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \frac{p_{\eta_1}}{Q_1}\boldsymbol{p}_i \qquad i = 1, ..., N \tag{5.386}$$

$$\frac{d\eta_\mu}{dt} = \frac{p_{\eta_\mu}}{Q_\mu} \qquad \mu = 1, ..., M \tag{5.387}$$

$$\frac{dp_{\eta_1}}{dt} = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0 - p_{\eta_1}\frac{p_{\eta_2}}{Q_2} \tag{5.388}$$

$$\frac{dp_{\eta_\mu}}{dt} = \frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - k_bT_0 - p_{\eta_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} \qquad \mu = 2, ..., M-1 \tag{5.389}$$

$$\frac{dp_{\eta_M}}{dt} = \frac{p_{\eta_{M-1}}^2}{Q_{M-1}} - k_bT_0 \tag{5.390}$$

where M thermostats have been included. Note that even in large systems, the addition of the extra thermostats is relatively inexpensive as they form a simple one dimensional chain. Only the first thermostat interacts with the physical system of N particles. The method presented above increases the size of the phase space and thus helps make the system ergodic.
Equations (5.385)-(5.390) are known as the Nosé-Hoover chain equations. These equations ensure that the first $M-1$ thermostat momenta $p_{\eta_1}, ..., p_{\eta_{M-1}}$ have the correct Maxwell-Boltzmann distribution. Note that for $M = 1$ the equations of motion reduce to the simpler Nosé-Hoover formulation. However, unlike the Nosé-Hoover equations, which are essentially Hamiltonian equations in noncanonical variables, the Nosé-Hoover chain equations have no underlying Hamiltonian structure, meaning no canonical variables exist that transform equations (5.385)-(5.390) into a Hamiltonian system.

5.3.5.2 Integration of the equations of motion

Numerical integrators for non-Hamiltonian systems such as the Nosé-Hoover chain equations can be derived using the Liouville operator formalism developed in Section 4.3.2. However, certain subtleties arise due to the generalized Liouville theorem and, therefore, the subject merits some discussion. For a Hamiltonian system, any numerical integration algorithm must preserve the symplectic property, in which case, it will also conserve the phase space volume. For non-Hamiltonian systems, there is no clear analog of the symplectic property. Nevertheless, the existence of a generalized Liouville theorem, equation (4.163), provides a minimal requirement that numerical solvers for non-Hamiltonian systems should satisfy, specifically, the preservation of the measure $\sqrt{g(\boldsymbol{\Gamma})}\, d\boldsymbol{\Gamma}$. Integrators that fail to obey the generalized Liouville theorem cannot be guaranteed to generate correct distributions. Therefore, in devising numerical solvers for non-Hamiltonian systems, care must be taken to ensure that they are measure-preserving. Despite the fact that the equations (5.385)-(5.390) are non-Hamiltonian, they can be expressed as an operator equation just as in the Hamiltonian case. Indeed, a general non-Hamiltonian system

$$\dot{\boldsymbol{\Gamma}} = \xi(\boldsymbol{\Gamma}) \tag{5.391}$$

can be always expressed as

$$\dot{\boldsymbol{\Gamma}} = i\mathcal{L}\,\boldsymbol{\Gamma} \qquad \text{where} \qquad i\mathcal{L} = \xi(\boldsymbol{\Gamma}) \cdot \nabla_{\boldsymbol{\Gamma}} \tag{5.392}$$

Note that only systems with no explicit time dependence are considered here, although the Liouville operator formalism can be extended to systems with explicit time dependence.[68]
The Liouville operator corresponding to equations (5.385)-(5.390) can be written as

$$i\mathcal{L} = i\mathcal{L}_{\text{NHC}} + i\mathcal{L}_1 + i\mathcal{L}_2 \tag{5.393}$$

The three Liouville operators are given by

$$i\mathcal{L}_1 = \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} \qquad i\mathcal{L}_2 = -\sum_{i=1}^{Nd} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \tag{5.394}$$

$$
\begin{aligned}
i\mathcal{L}_{\text{NHC}} &= -\sum_{i=1}^{N}\frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}+\sum_{\mu=1}^{M}\frac{p_{\eta_\mu}}{Q_\mu}\,\frac{\partial}{\partial \eta_\mu}+\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}-gk_bT_0-p_{\eta_1}\,\frac{p_{\eta_2}}{Q_2}\right)\frac{\partial}{\partial p_{\eta_1}}\\
&\quad+\sum_{\mu=2}^{M-1}\left(\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}}-k_bT_0-p_{\eta_\mu}\,\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\right)\frac{\partial}{\partial p_{\eta_\mu}}+\left(\frac{p_{\eta_{M-1}}^2}{Q_{M-1}}-k_bT_0\right)\frac{\partial}{\partial p_{\eta_M}}\\
&=-\sum_{i=1}^{N}\frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}+\sum_{\mu=1}^{M}\frac{p_{\eta_\mu}}{Q_\mu}\,\frac{\partial}{\partial \eta_\mu}-\sum_{\mu=1}^{M-1}p_{\eta_\mu}\,\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\,\frac{\partial}{\partial p_{\eta_\mu}}\\
&\quad+\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}-gk_bT_0\right)\frac{\partial}{\partial p_{\eta_1}}+\sum_{\mu=2}^{M-1}\left(\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}}-k_bT_0\right)\frac{\partial}{\partial p_{\eta_\mu}}+\left(\frac{p_{\eta_{M-1}}^2}{Q_{M-1}}-k_bT_0\right)\frac{\partial}{\partial p_{\eta_M}}\\
&=-\sum_{i=1}^{N}\frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}+\sum_{\mu=1}^{M}\frac{p_{\eta_\mu}}{Q_\mu}\,\frac{\partial}{\partial \eta_\mu}-\sum_{\mu=1}^{M-1}p_{\eta_\mu}\,\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\,\frac{\partial}{\partial p_{\eta_\mu}}+G_1\,\frac{\partial}{\partial p_{\eta_1}}+\sum_{\mu=2}^{M-1}G_\mu\,\frac{\partial}{\partial p_{\eta_\mu}}+G_M\,\frac{\partial}{\partial p_{\eta_M}}
\end{aligned}
$$

$$
i\mathcal{L}_{\text{NHC}}=-\sum_{i=1}^{N}\frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\cdot\frac{\partial}{\partial \boldsymbol{p}_i}+\sum_{\mu=1}^{M}\frac{p_{\eta_\mu}}{Q_\mu}\,\frac{\partial}{\partial \eta_\mu}+\sum_{\mu=1}^{M-1}\left(G_\mu-p_{\eta_\mu}\,\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\right)\frac{\partial}{\partial p_{\eta_\mu}}+G_M\,\frac{\partial}{\partial p_{\eta_M}} \tag{5.395}
$$

where

$$
G_1=\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}-gk_bT_0 \qquad\qquad G_M=\frac{p_{\eta_{M-1}}^2}{Q_{M-1}}-k_bT_0 \tag{5.396}
$$

$$
G_\mu=\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}}-k_bT_0 \qquad\qquad \mu=2,...,M-1 \tag{5.397}
$$

represent the thermostat forces. Note that the sum $i\mathcal{L}_1+i\mathcal{L}_2$ in equation (5.393), with the two operators defined as in (5.394), constitutes a purely Hamiltonian subsystem. The evolution of the full phase space vector $\boldsymbol{\Gamma}(t)=[\boldsymbol{p}(t),\boldsymbol{q}(t),\boldsymbol{p}_\eta(t),\boldsymbol{\eta}(t)]$ is given by the usual relation

$$
\boldsymbol{\Gamma}(t+\tau)=e^{i\mathcal{L}\tau}\,\boldsymbol{\Gamma}(t) \tag{5.398}
$$

As done in the Hamiltonian case, the Trotter theorem can be employed to factorize the propagator $\exp(i\mathcal{L}\tau)$ for a single timestep τ. Consider a particular factorization of the form

$$
e^{i\mathcal{L}\tau}\approx e^{i\mathcal{L}_{\text{NHC}}\,\tau/2}\,e^{i\mathcal{L}_2\,\tau/2}\,e^{i\mathcal{L}_1\,\tau}\,e^{i\mathcal{L}_2\,\tau/2}\,e^{i\mathcal{L}_{\text{NHC}}\,\tau/2}+O(\tau^3) \tag{5.399}
$$

Note that the three operators in the middle are identical to those defined in the case of microcanonical ensemble (see Section 5.1.2 with timestep $\Delta t=\tau$). Indeed, this factorization, on its own, would generate the velocity Verlet algorithm. However, in equation (5.399), it is sandwiched between the thermostat propagators. This type of separation between the Hamiltonian and non-Hamiltonian parts of the Liouville operator is intuitively appealing and it allows for easy implementation of the numerical integrator for the Nosé-Hoover chains equations of motion.

The operator $i\mathcal{L}_{\text{NHC}}$ contains many terms, so that the operator $\exp(i\mathcal{L}_{\text{NHC}}\tau/2)$ still needs to be separated further. Experience has shown, however, that a simple factorization of the operator based on the separate terms in (5.395) is insufficient to achieve a robust integration scheme. The reason is that the thermostat forces in equations (5.396) and (5.397) vary rapidly, thereby limiting the timestep. To alleviate this problem, the RESPA methodology can be applied to this part of the propagator. Once again, experience shows that several hundred RESPA steps are needed to resolve the thermostat part of the propagator accurately, so RESPA alone cannot easily handle the rapidly varying thermostat forces. Consider, however, employing a higher-order (than τ^3) factorization together with RESPA to the operator $\exp(i\mathcal{L}_{\text{NHC}}\tau/2)$. A judiciously chosen algorithm could improve the accuracy of RESPA without adding significantly to the computational overhead. Fortunately, high order methods suitable for these purposes exist. One scheme in particular, due to Suzuki,[69, 70] and Yoshida,[71] has proved particularly useful for the Nosé-Hoover chains equations.

5.3.5.3 Conserved quantities

In order to analyze the distribution of the physical phase space generated by equations (5.385)-(5.390), first of all the conservation laws must be identified. It can be proved that the following quantity is conserved

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q},\boldsymbol{p}_\eta,\boldsymbol{\eta}) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \sum_{\mu=1}^{M}\frac{p_{\eta_\mu}^2}{2Q_\mu} + gk_bT_0\eta_1 + k_bT_0\sum_{\mu=2}^{M}\eta_\mu = C_1 \tag{5.400}$$

where $\boldsymbol{p}_\eta = (p_{\eta_1}, p_{\eta_2}, ..., p_{\eta_M})$ and $\boldsymbol{\eta} = (\eta_1, \eta_2, ..., \eta_M)$. The conservation of (5.400) can be easily proved by computing the total time derivative of the conserved Hamiltonian (5.400), that is

$$\begin{aligned}
\frac{d\mathcal{H}(\boldsymbol{p},\boldsymbol{q},\boldsymbol{p}_\eta,\boldsymbol{\eta})}{dt} &= \sum_{i=1}^{N}\left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i}\cdot\frac{d\boldsymbol{p}_i}{dt} + \frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i}\cdot\frac{d\boldsymbol{q}_i}{dt}\right) + \sum_{\mu=1}^{M}\frac{\partial\mathcal{H}}{\partial p_{\eta_\mu}}\frac{dp_{\eta_\mu}}{dt} + \frac{\partial\mathcal{H}}{\partial\eta_1}\frac{d\eta_1}{dt} + \sum_{\mu=2}^{M}\frac{\partial\mathcal{H}}{\partial\eta_\mu}\frac{d\eta_\mu}{dt} \\
&= \sum_{i=1}^{N}\left[\frac{\boldsymbol{p}_i}{m_i}\cdot\left(-\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_{\eta_1}}{Q_1}\boldsymbol{p}_i\right) + \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\frac{\boldsymbol{p}_i}{m_i}\right] + \frac{p_{\eta_1}}{Q_1}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} - gk_bT_0 - p_{\eta_1}\frac{p_{\eta_2}}{Q_2}\right) \\
&+ \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu}\left(\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - k_bT_0 - p_{\eta_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\right) + \frac{p_{\eta_M}}{Q_M}\left(\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} - k_bT_0\right) + gk_bT_0\frac{p_{\eta_1}}{Q_1} + k_bT_0\sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu}
\end{aligned} \tag{5.401}$$

which can be rewritten as

$$\begin{aligned}
\frac{d\mathcal{H}(\boldsymbol{p},\boldsymbol{q},\boldsymbol{p}_\eta,\boldsymbol{\eta})}{dt} &= \sum_{i=1}^{N}\left[-\frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\frac{\boldsymbol{p}_i}{m_i} - \frac{p_{\eta_1}}{Q_1}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\frac{\boldsymbol{p}_i}{m_i} + \frac{p_{\eta_1}}{Q_1}\frac{\boldsymbol{p}_i^2}{m_i}\right] - gk_bT_0\frac{p_{\eta_1}}{Q_1} - \frac{p_{\eta_1}^2}{Q_1}\frac{p_{\eta_2}}{Q_2} \\
&+ \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu}\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - k_bT_0\sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu} - \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} + \frac{p_{\eta_M}}{Q_M}\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} - k_bT_0\frac{p_{\eta_M}}{Q_M} + gk_bT_0\frac{p_{\eta_1}}{Q_1} + k_bT_0\sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu} \\
&= -\frac{p_{\eta_1}^2}{Q_1}\frac{p_{\eta_2}}{Q_2} + \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu}\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - k_bT_0\sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu} - \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} + \frac{p_{\eta_M}}{Q_M}\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} + k_bT_0\sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu} \\
&= -\frac{p_{\eta_1}^2}{Q_1}\frac{p_{\eta_2}}{Q_2} + \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu}\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} + \frac{p_{\eta_M}}{Q_M}\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} \\
&= \sum_{\mu=2}^{M-1}\frac{p_{\eta_\mu}}{Q_\mu}\frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - \sum_{\mu=1}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} + \frac{p_{\eta_M}}{Q_M}\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} \\
&= \sum_{\mu=1}^{M-2}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\frac{p_{\eta_\mu}^2}{Q_\mu} - \sum_{\mu=1}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} + \frac{p_{\eta_M}}{Q_M}\frac{p_{\eta_{M-1}}^2}{Q_{M-1}} = \sum_{\mu=1}^{M-1}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}}\frac{p_{\eta_\mu}^2}{Q_\mu} - \sum_{\mu=1}^{M-1}\frac{p_{\eta_\mu}^2}{Q_\mu}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} = 0
\end{aligned}$$

If there exists a non zero net force acting on the system, then (5.400) is the only one constant of motion preserved by the non Hamiltonian flow of equations (5.385) - (5.390). However, considering a system in the absence of external forces, namely, under the hypothesis

$$\mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.402}$$

then there are d additional conservation laws satisfied by the equations of motion (5.385) - (5.390), which take the form

$$\mathbf{P}\,e^{\eta_1} = \mathbf{K} \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.403}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions (with d the dimension of the physical space) and $\mathbf{P}$ is the center of mass momentum of the system defined as in (5.299). Equation (5.403) can be

verified by taking the time derivative explicitly, so that

$$\begin{aligned}\frac{d(\mathbf{P}e^{\eta_1})}{dt} &= \dot{\mathbf{P}}\,e^{\eta_1} + \dot{\eta_1}\,\mathbf{P}\,e^{\eta_1} = e^{\eta_1}\left(\dot{\mathbf{P}} + \dot{\eta_1}\,\mathbf{P}\right) = e^{\eta_1}\left(\dot{\mathbf{P}} + \frac{p_{\eta_1}}{Q_1}\,\mathbf{P}\right) \\ &= e^{\eta_1}\left(\sum_{i=1}^{N}\dot{\mathbf{p}}_i + \frac{p_{\eta_1}}{Q_1}\,\mathbf{P}\right) = e^{\eta_1}\left[\sum_{i=1}^{N}\left(-\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\right) + \frac{p_{\eta_1}}{Q_1}\,\mathbf{P}\right] \\ &= e^{\eta_1}\left(\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_{\eta_1}}{Q_1}\sum_{i=1}^{N}\boldsymbol{p}_i + \frac{p_{\eta_1}}{Q_1}\,\mathbf{P}\right) = e^{\eta_1}\left(\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_{\eta_1}}{Q_1}\,\mathbf{P} + \frac{p_{\eta_1}}{Q_1}\,\mathbf{P}\right) = e^{\eta_1}\sum_{i=1}^{N}\mathbf{F}_i\end{aligned} \tag{5.404}$$

where the equations of motion (5.386) and (5.387) have been used, together with the definition of the total linear momentum (5.299). Therefore, as demonstrated in (5.404), the total time derivative of the quantity (5.403) is equal to zero iff the condition (5.402) is satisfied. The conservation law (5.403) can be written using a single independent variable, because the d components of the total linear momentum are linearly dependent. Taking the Euclidean norm on both sides of equation (5.403) leads to

$$\|\mathbf{P}\,e^{\eta_1}\| = P\,e^{\eta_1} = \|\mathbf{K}\| = K \qquad \text{where} \qquad P = \|\mathbf{P}\| = \left(\sum_{\alpha=1}^{d} P_\alpha^2\right)^{1/2} \tag{5.405}$$

Hence, the conservation law in the absence of net forces on the system becomes

$$Pe^{\eta_1} = K = C_2 \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.406}$$

conservation laws of Nosé - Hoover chains equations of motion (5.385) - (5.390)
$\mathcal{H}(\boldsymbol{p},\boldsymbol{q},\boldsymbol{p}_\eta,\boldsymbol{\eta}) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \sum_{\mu=1}^{M}\frac{p_{\eta_\mu}^2}{2Q_\mu} + gk_bT_0\eta_1 + k_bT_0\sum_{\mu=2}^{M}\eta_\mu = C_1$
$Pe^{\eta_1} = K = C_2 \qquad\qquad \text{iff} \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0$

Table 5.7: Constants of motion for the Nosé-Hoover chains equations of motion (5.385) - (5.390), together with the simulation conditions that are necessary to preserve these quantities.

5.3.5.4 Statistical mechanical ensemble

First of all, the partition function generated if the only conserved quantity is given by (5.400) is derived. In this case, using the phase space vector $\boldsymbol{\Gamma} = (\boldsymbol{p},\boldsymbol{q},\boldsymbol{p}_\eta,\boldsymbol{\eta})$, the phase space compressibility of this system, given by equation (4.123), can be written as

$$\begin{aligned}\kappa(\boldsymbol{\Gamma},t) \equiv \nabla_{\boldsymbol{\Gamma}}\cdot\dot{\boldsymbol{\Gamma}} &= \sum_{i=1}^{N}\nabla_{\boldsymbol{q}_i}\cdot\dot{\boldsymbol{q}}_i + \sum_{i=1}^{N}\nabla_{\boldsymbol{p}_i}\cdot\dot{\boldsymbol{p}}_i + \sum_{\mu=1}^{M}\nabla_{\eta_\mu}\cdot\dot{\eta}_\mu + \sum_{\mu=1}^{M}\nabla_{p_{\eta_\mu}}\cdot\dot{p}_{\eta_\mu} \\ &= \sum_{i=1}^{N}\nabla_{\boldsymbol{p}_i}\cdot\dot{\boldsymbol{p}}_i + \nabla_{p_{\eta_1}}\cdot\dot{p}_{\eta_1} + \sum_{\mu=2}^{M-1}\nabla_{p_{\eta_\mu}}\cdot\dot{p}_{\eta_\mu} + \nabla_{p_{\eta_M}}\cdot\dot{p}_{\eta_M} \\ &= \sum_{i=1}^{N}\nabla_{\boldsymbol{p}_i}\cdot\left(-\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_{\eta_1}}{Q_1}\,\boldsymbol{p}_i\right) - \frac{p_{\eta_2}}{Q_2} - \sum_{\mu=2}^{M-1}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} \\ &= -Nd\,\frac{p_{\eta_1}}{Q_1} - \sum_{\mu=1}^{M-1}\frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} = -Nd\,\frac{p_{\eta_1}}{Q_1} - \sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu} = -Nd\,\dot{\eta}_1 - \sum_{\mu=2}^{M}\frac{p_{\eta_\mu}}{Q_\mu} \equiv -Nd\,\dot{\eta}_1 - \dot{\eta}_c\end{aligned} \tag{5.407}$$

where the equations of motion (5.385) - (5.390) have been used, N is the number of atoms, d is the dimensionality of the system and a new variable η_c has been defined as

$$\eta_c = \sum_{\mu=2}^{M} \eta_\mu \tag{5.408}$$

Among the M thermostats chain (M chains), only η_1 and the thermostat center, given by (5.408), are independently coupled to the dynamics. The remaining chain variables are driven. Therefore, the M chains add only two additional degrees of freedom to the system. The metric can then be computed using the definition (4.180) with the compressibility (5.407) as

$$\sqrt{g(\boldsymbol{\Gamma}, t)} = \exp\left(- \int \kappa(\boldsymbol{\Gamma}, t)\, dt \right) = \exp\left(\int (Nd\, \dot{\eta}_1 + \dot{\eta}_c)\, dt \right) = \exp(Nd\, \eta_1 + \eta_c) \tag{5.409}$$

The microcanonical partition function at a given temperature T_0 can be constructed using the metric (5.409), the conserved quantity (5.400) and the energy conservation condition, by means of equation (4.185), so that

$$\begin{aligned} &Z(N, V, C_1) \\ &= \zeta \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\boldsymbol{\eta} \int d\boldsymbol{p}_\eta \, \exp(Nd\, \eta_1 + \eta_c)\, \delta\left(\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} + g k_b T_0 \eta_1 + k_b T_0 \eta_c - C_1 \right) \end{aligned} \tag{5.410}$$

where the microcanonical partition function depends parametrically on the temperature T_0 of the system, and the Hamiltonian $\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q})$ is given by

$$\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{5.411}$$

First of all, an integration over the variable η_1 can be performed. Since the argument of the second delta function in equation (5.410) has only one zero as a function of the variable η_1, a convenient equivalence relation can be employed to resolve the integral, that is

$$\delta[f(\eta_1)] = \frac{\delta(\eta_1 - \bar{\eta}_1)}{f'(\bar{\eta}_1)} \tag{5.412}$$

where $\bar{\eta}_1$ is the zero of function $f(\eta_1)$ in the argument of the Dirac function, and $f'(\bar{\eta}_1)$ is the derivative of the function $f(\eta_1)$ evaluated in the point $\bar{\eta}_1$, which has to be read, inside the integral, as the function $f'(\eta_1)$, so that the expressions for these objects are

$$f(\eta_1) = \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} + g k_b T_0 \eta_1 + k_b T_0 \eta_c - C_1 \qquad f'(\eta_1) = g k_b T_0 \tag{5.413}$$

$$\bar{\eta}_1 = \frac{1}{g k_b T_0} \left(C_1 - \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) - \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} - k_b T_0 \eta_c \right) \qquad f'(\bar{\eta}_1) = g k_b T_0 \tag{5.414}$$

Using the relation (5.412), together with equation (5.414), in the partition function (5.410), leads to

$$\begin{aligned} Z(N, V, C_1) &= \zeta \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \int d\eta_1 \, \exp(Nd\, \eta_1 + \eta_c)\, \frac{\delta(\eta_1 - \bar{\eta}_1)}{g k_b T_0} \\ &= \frac{\zeta}{g k_b T_0} \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \, \exp(Nd\, \bar{\eta}_1 + \eta_c) \\ &= \frac{\zeta}{g k_b T_0} \int d\boldsymbol{q} \int d\boldsymbol{p} \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \, \exp\left[\frac{Nd}{g k_b T_0} \left(C_1 - \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) - \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} - k_b T_0 \eta_c \right) + \eta_c \right] \end{aligned}$$

$$= \xi \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{Nd}{g k_b T_0} \, \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) \right) \tag{5.415}$$

where the constant ξ which groups together the integration over the variable $\boldsymbol{p}_\eta$ and $\eta_2 \cdots \eta_M$ (that represent constant prefactors with no physical importance) is given by

$$\xi = \frac{\zeta}{g k_b T_0} \exp\left(\frac{Nd\, C_1}{g k_b T_0} \right) \int d\boldsymbol{p}_\eta \, \exp\left(-\frac{Nd}{g k_b T_0} \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} \right) \int d\eta_2 \cdots d\eta_M \exp\left[\sum_{\mu=2}^{M} \eta_\mu \left(1 - \frac{Nd}{g} \right) \right] \tag{5.416}$$

The analytical expression for this constant prefactor can be computed considering that it involves a simple exponential integral in the variables $\eta_2 \cdots \eta_M$ and a Gaussian integral with respect to the variable $\boldsymbol{p}_\eta$ which can be both solved easily, so that it can be rewritten as

$$\xi = \frac{\zeta}{g k_b T_0} \exp\left(\frac{Nd\, C_1}{g k_b T_0} \right) \prod_{\mu=1}^{M} \left(\frac{2\pi Q_\mu}{Nd} \, g k_b T_0 \right)^{1/2} \int d\eta_2 \cdots d\eta_M \exp\left[\sum_{\mu=2}^{M} \eta_\mu \left(1 - \frac{Nd}{g} \right) \right] \tag{5.417}$$

Equation (5.415) is the canonical distribution function (modulo constant prefactors), provided that $g = Nd$. Indeed, if $g = Nd$ then the partition function (5.415) becomes

$$g = Nd \quad \rightarrow \quad Z(N, V, C_1) = \xi \int d\boldsymbol{q} \int d\boldsymbol{p} \, \exp\left(-\frac{\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right) \tag{5.418}$$

with the constant prefactor given by

$$g = Nd \quad \rightarrow \quad \xi = \frac{\zeta}{Nd\, k_b T_0} \exp\left(\frac{C_1}{k_b T_0} \right) \prod_{\mu=1}^{M} (2\pi \, k_b T_0 \, Q_\mu)^{1/2} \int d\eta_2 \cdots d\eta_M \tag{5.419}$$

5.3.5.5 Statistical mechanical ensemble under periodic boundary conditions

In this section, the form of the partition function describing a system without external forces (i.e. in which the condition (5.402) holds) and modeled using periodic boundary conditions is analyzed, following the procedure previously introduced for the simple Nosé-Hoover equations of motion (see Section 5.3.4.5). In this case, two constants of motion have to be taken into account (see Table 5.7), namely, the total Hamiltonian (5.400) and the quantity (5.406) related to the total linear momentum.
To continue the analysis following Section 4.3.2.1, as already outlined in Section 5.3.4.5 in the case of Nosé-Hoover equations of motion ($M = 1$ thermostat), also in the Nosé-Hoover chains the momenta variables in $\boldsymbol{P}$ are linearly dependent and the positions in $\boldsymbol{R}$ are driven variables. Taking the time derivative on both member of the equation (5.406), and using the equation of motion (5.387) leads to

$$\dot{P} = -\dot{\eta}_1 \, e^{-\eta_1} \, K = -\dot{\eta}_1 \, P = -\frac{p_{\eta_1}}{Q_1} \, P \tag{5.420}$$

that is the equation of motion which rules the evolution in time of the norm of the total linear momentum of the system. Proceeding with the analysis, the two variables $\mathbf{R}$ and $\mathbf{P}$ can be eliminated by considering the positions and momenta relative to the center of mass of the system, $\boldsymbol{\rho}$ and $\boldsymbol{\pi}$, respectively. Therefore, the equations of motion (5.385)-(5.390) can be rewritten following the same reasoning as in Section 5.3.4.5, leading to the new set of equations

$$\frac{d\boldsymbol{\rho}_i}{dt} = \frac{\boldsymbol{\pi}_i}{\bar{m}_i} \qquad i = 1, ..., N-1 \tag{5.421}$$

$$\frac{d\boldsymbol{\pi}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \frac{p_{\eta_1}}{Q_1} \, \boldsymbol{\pi}_i \qquad i = 1, ..., N-1 \tag{5.422}$$

$$\frac{dP}{dt} = -\frac{p_{\eta_1}}{Q_1} \, P \tag{5.423}$$

$$\frac{d\eta_\mu}{dt} = \frac{p_{\eta_\mu}}{Q_\mu} \qquad \mu = 1, ..., M \tag{5.424}$$

$$\frac{dp_{\eta_1}}{dt} = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{\bar{m}_i} - g k_b T_0 - p_{\eta_1} \frac{p_{\eta_2}}{Q_2} \tag{5.425}$$

$$\frac{dp_{\eta_\mu}}{dt} = \frac{p_{\eta_{\mu-1}}^2}{Q_{\mu-1}} - k_b T_0 - p_{\eta_\mu} \frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} \qquad \mu = 2, ..., M-1 \tag{5.426}$$

$$\frac{dp_{\eta_M}}{dt} = \frac{p_{\eta_{M-1}}^2}{Q_{M-1}} - k_b T_0 \tag{5.427}$$

where $\bar{m}_i$ is the total mass of the system, and the equation of motion for the momentum $\mathbf{P}$, written in terms of the single independent variable P as defined in (5.405), has been introduced in equation (5.423), following the result obtained in (5.420). The equations of motion have two conservative laws (because the d center of mass momenta components have been replaced by a single variable P only one conservation law for the momenta is left), namely,

$$\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}, \boldsymbol{p}_\eta, \boldsymbol{\eta}) = \mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) + \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} + g k_b T_0 \eta_1 + k_b T_0 \sum_{\mu=2}^{M} \eta_\mu = C_1 \tag{5.428}$$

$$P\, e^{\eta_1} = C_2 \tag{5.429}$$

where in the first conservation law (5.428), the Hamiltonian on the left hand side is given by

$$\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{2\bar{m}_i} + \frac{P^2}{2M} + \phi(\boldsymbol{\rho}) \tag{5.430}$$

In order to compute the partition function, the compressibility has to be calculated, using the phase space vector $\boldsymbol{\Xi} = (\boldsymbol{\pi}, P, \boldsymbol{\rho}, \boldsymbol{p}_\eta, \boldsymbol{\eta})$, the equations of motion (5.421)-(5.427) and the definition (4.123), as

$$\begin{aligned}
\kappa(\boldsymbol{\Xi}, t) \equiv \nabla_{\boldsymbol{\Xi}} \cdot \dot{\boldsymbol{\Xi}} &= \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\rho}_i} \cdot \dot{\boldsymbol{\rho}}_i + \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} + \sum_{\mu=1}^{M} \nabla_{\eta_\mu} \cdot \dot{\eta}_\mu + \sum_{\mu=1}^{M} \nabla_{p_{\eta_\mu}} \cdot \dot{p}_{\eta_\mu} \\
&= \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} + \sum_{\mu=1}^{M} \nabla_{p_{\eta_\mu}} \cdot \dot{p}_{\eta_\mu} = \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \dot{\boldsymbol{\pi}}_i + \nabla_P \cdot \dot{P} + \nabla_{p_{\eta_1}} \cdot \dot{p}_{\eta_1} + \sum_{\mu=2}^{M-1} \nabla_{p_{\eta_\mu}} \cdot \dot{p}_{\eta_\mu} \\
&= \sum_{i=1}^{N-1} \nabla_{\boldsymbol{\pi}_i} \cdot \left(-\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \frac{p_{\eta_1}}{Q_1} \boldsymbol{\pi}_i \right) - \frac{p_{\eta_1}}{Q_1} - \frac{p_{\eta_2}}{Q_2} - \sum_{\mu=2}^{M-1} \nabla_{p_{\eta_\mu}} \cdot \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{\bar{m}_i} - g k_b T_0 - p_{\eta_1} \frac{p_{\eta_2}}{Q_2} \right) \\
&= -(N-1) d \frac{p_{\eta_1}}{Q_1} - \frac{p_{\eta_1}}{Q_1} - \frac{p_{\eta_2}}{Q_2} - \sum_{\mu=2}^{M-1} \frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} = -[(N-1)d + 1] \frac{p_{\eta_1}}{Q_1} - \sum_{\mu=1}^{M-1} \frac{p_{\eta_{\mu+1}}}{Q_{\mu+1}} \\
&= -(Nd - d + 1) \frac{p_{\eta_1}}{Q_1} - \sum_{\mu=2}^{M} \frac{p_{\eta_\mu}}{Q_\mu} = -(Nd - d + 1)\, \dot{\eta}_1 - \dot{\eta}_c
\end{aligned} \tag{5.431}$$

where η_c is the variable defined as in equation (5.408). From the compressibility (5.431), the metric follows directly

$$\begin{aligned}
\sqrt{g(\boldsymbol{\Xi}, t)} &= \exp\left(-\int \kappa(\boldsymbol{\Xi}, t)\, dt \right) = \exp\left(\int [(Nd - d + 1)\, \dot{\eta}_1 + \dot{\eta}_c]\, dt \right) \\
&= \exp\left\{ (Nd - d + 1) \int \frac{d\eta_1}{dt}\, dt + \int \frac{d\eta_c}{dt}\, dt \right\} = \exp\{(Nd - d + 1)\eta_1 + \eta_c\}
\end{aligned} \tag{5.432}$$

The conservation laws and the metric can now be used to construct the microcanonical partition function, which contains two delta functions that express the two conservation laws

$$\begin{aligned}
&Z(N, V, C_1, C_2) \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\eta} \int d\boldsymbol{p}_\eta\; e^{(Nd-d+1)\eta_1}\, e^{\eta_c}\, \delta\Big(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}, \boldsymbol{p}_\eta, \boldsymbol{\eta}) - C_1 \Big)\, \delta(e^{\eta_1} P - C_2)
\end{aligned} \tag{5.433}$$

where $d\boldsymbol{\rho} = d\boldsymbol{\rho}_1 \cdots d\boldsymbol{\rho}_{N-1}$ and $d\boldsymbol{\pi} = d\boldsymbol{\pi}_1 \cdots d\boldsymbol{\pi}_{N-1}$, while $d\boldsymbol{\eta} = d\eta_1 \cdots d\eta_M$ and $d\boldsymbol{p}_\eta = dp_{\eta_1} \cdots dp_{\eta_M}$ and the Hamiltonian is given by equation (5.428). Since the argument of the second delta function in the above equation (5.433) has only one zero as a function of the variable η_1, a convenient equivalence relation can be employed to resolve the integral, given by equation (5.412), with

$$f(\eta_1) = e^{\eta_1}\, P - C_2 \qquad\qquad f'(\eta_1) = e^{\eta_1}\, P \tag{5.434}$$

$$\bar{\eta}_1 = \ln\!\left(\frac{C_2}{P}\right) \qquad\qquad f'(\bar{\eta}_1) = C_2 \tag{5.435}$$

where $\bar{\eta}_1$ is the zero of function $f(\eta_1)$ and $f'(\bar{\eta}_1)$ is the derivative of the function $f(\eta_1)$ evaluated in the point $\bar{\eta}_1$, which has to be read, inside the integral, as the function $f'(\eta_1)$. Therefore, using the relation (5.412) in (5.433), together with equations (5.434) and (5.435), the partition function becomes

$$\begin{aligned}
&Z(N, V, C_1, C_2) \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{\eta} \int d\boldsymbol{p}_\eta \; e^{(Nd-d+1)\eta_1}\, e^{\eta_c}\; \delta\!\left(\mathcal{H}(\boldsymbol{\pi}, P, \boldsymbol{\rho}, \boldsymbol{p}_\eta, \boldsymbol{\eta}) - C_1\right) \frac{\delta(\eta_1 - \bar{\eta}_1)}{e^{\eta_1}\, P} \\
&= \zeta \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \; e^{(N-1)d\,\bar{\eta}_1}\, \frac{e^{\eta_c}}{P}\; \delta\!\left(\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) + \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} + g k_b T_0\, \bar{\eta}_1 + k_b T_0\, \eta_c - C_1\right)
\end{aligned}$$

The substitution of the variable $\bar{\eta}_1$ with its explicit expression, given by the first equation in (5.435), yields the following form for the partition function

$$Z(N, V, C_1, C_2) = \frac{\zeta}{C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \left(\frac{C_2}{P}\right)^{(N-1)d+1} e^{\eta_c}\; \delta[f(\eta_c)] \tag{5.436}$$

where the expression inside the delta in the previous equation can be viewed as a function of the variable η_c and it is given by

$$f(\eta_c) = \mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) + \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} + g k_b T_0\, \ln\!\left(\frac{C_2}{P}\right) + k_b T_0\, \eta_c - C_1 \qquad\qquad f'(\eta_c) = k_b T_0 \tag{5.437}$$

The previous function $f(\eta_c)$ has only one zero $\bar{\eta}_c$ with respect to the variable η_c, that is

$$\bar{\eta}_c = \frac{1}{k_b T_0}\left(C_1 - \mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} - g k_b T_0\, \ln\!\left(\frac{C_2}{P}\right)\right) \qquad\qquad f'(\bar{\eta}_c) = k_b T_0 \tag{5.438}$$

Using again the relation (5.412), the integration over the variable η_c results in the following expression for the partition function

$$\begin{aligned}
&Z(N, V, C_1, C_2) \\
&= \frac{\zeta}{C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\eta_2 \cdots d\eta_M \int d\boldsymbol{p}_\eta \left(\frac{C_2}{P}\right)^{(N-1)d+1} e^{\eta_c}\; \frac{\delta(\eta_c - \bar{\eta}_c)}{k_b T_0} \\
&= \frac{\zeta}{k_b T_0 C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{p}_\eta\, e^{\bar{\eta}_c} \left(\frac{C_2}{P}\right)^{(N-1)d+1} \\
&= \frac{\zeta}{k_b T_0 C_2} \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \int d\boldsymbol{p}_\eta \left(\frac{C_2}{P}\right)^{(N-1)d+1} \exp\!\left\{\frac{1}{k_b T_0}\left[C_1 - \mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho}) - \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu} - g k_b T_0\, \ln\!\left(\frac{C_2}{P}\right)\right]\right\} \\
&= \frac{\zeta}{k_b T_0 C_2} \exp\!\left(\frac{C_1}{k_b T_0}\right) \int d\boldsymbol{p}_\eta \exp\!\left(-\frac{1}{k_b T_0} \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu}\right) \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{C_2}{P}\right)^{(N-1)d+1-g} \exp\!\left[-\frac{\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho})}{k_b T_0}\right] \\
&= \frac{\zeta}{k_b T_0 C_2} \exp\!\left(\frac{C_1}{k_b T_0}\right) \int d\boldsymbol{p}_\eta \exp\!\left(-\frac{1}{k_b T_0} \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu}\right) \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{P}{C_2}\right)^{g-(N-1)d-1} \exp\!\left[-\frac{\mathcal{H}_0(\boldsymbol{\pi}, P, \boldsymbol{\rho})}{k_b T_0}\right]
\end{aligned}$$

Collecting all the prefactors in a constant ξ, the partition function can be written as

$$Z(N,V,C_1,C_2) = \xi \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{P}{C_2}\right)^{g-(N-1)d-1} \exp\left[-\frac{\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho})}{k_b T_0}\right] \tag{5.439}$$

with the constant is given by

$$\begin{aligned} \xi &= \frac{\zeta}{k_b T_0 C_2} \exp\left(\frac{C_1}{k_b T_0}\right) \int d\boldsymbol{p}_\eta \exp\left(-\frac{1}{k_b T_0} \sum_{\mu=1}^{M} \frac{p_{\eta_\mu}^2}{2Q_\mu}\right) \\ &= \frac{\zeta}{k_b T_0 C_2} \exp\left(\frac{C_1}{k_b T_0}\right) \prod_{\mu=1}^{M} \left(2\pi\, k_b T_0\, Q_\mu\right)^{1/2} \end{aligned} \tag{5.440}$$

where the integral over the variables $\boldsymbol{p}_\eta = (p_{\eta_1}, ..., p_{\eta_M})$ is the product of simple Gaussian integrals and it has been solved using the formula (5.321).
In particular, the constant ξ given by equation (5.440) is independent from the variable g. Instead, if $g = Nd$, the partition function (5.439) becomes

$$g = Nd \quad \rightarrow \quad Z(N,V,C_1,C_2) = \xi \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \left(\frac{P}{C_2}\right)^{d-1} \exp\left[-\frac{\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho})}{k_b T_0}\right] \tag{5.441}$$

while if $g = (N-1)d + 1$ then the partition function (5.439) can be written as

$$g = (N-1)d + 1 \quad \rightarrow \quad Z(N,V,C_1,C_2) = \xi \int d\boldsymbol{\rho} \int d\boldsymbol{\pi} \int dP \exp\left[-\frac{\mathcal{H}_0(\boldsymbol{\pi},P,\boldsymbol{\rho})}{k_b T_0}\right] \tag{5.442}$$

with the constant ξ given by equation (5.440) in both cases.

5.4 Constant temperature and pressure approaches

The coupling of a physical system with a thermal and a pressure reservoir can be regarded as a thermodynamic constraint. The extended system approach in molecular dynamics simulation, pioneered by Andersen in his 1980 seminal paper for the constant pressure case,[45] treats the thermodynamic constraints in a dynamic way, where the baths are represented solely by a few additional degrees of freedom. In this context, an extended Hamiltonian is introduced, in which the physical and the extra degrees of freedom are treated at the same level, and the equations of motion are derived by standard methods.[72] These equations are then cast in a non-canonical form by means of a coordinate transformation in order to sample the properties of the physical system from the appropriate statistical ensemble by numerical integration over uniform time separations.[47, 73] Within this general scheme, several different equations of motion have been proposed, all leading to an ensemble equilibrium probability density for the physical system, after integration over the extra variables.[47, 58, 74, 75, 76, 77, 78, 79] In fact, the form of the kinetic term for the extended variables is not fixed and there is freedom for different choices that give rise to different dynamic trajectories, which equally sample in a correct way the phase space of the desired ensemble.[80, 81]
In general the extended system procedure requires the introduction of velocity dependent forces, which are difficult to integrate numerically in the context of a Verlet scheme without spoiling the time reversal properties of the algorithm. The Suzuki-Trotter formalism[69, 82, 83] provides for an elegant solution to these problems. Indeed, it has been shown that an appropriate breakup of the propagator for the Newtonian dynamics of an N particle atomic system originates the well known velocity Verlet algorithm, providing at the same time a theoretical foundation for such a simple and commonly used integration scheme (see Section 4.2.1). Moreover, this approach leads to a procedure that allows one to derive in a systematic way multiple time step strategies in molecular simulations, highly suitable for systems with strong separations in timescales.
Following these general strategies, in the framework of extended system methods (see Section 5.2.2), in Section 5.4.1 a set of equations of motion introduced by Ferrario[78] are described, leading to a constant temperature and pressure molecular dynamics simulation (sampling of the NPT ensemble in phase space).

5.4.1 Ferrario thermostat and barostat

In this section a particular set of equations of motion, previously introduced by M. Ferrario,[78] is described, which combines constant pressure with constant temperature thermodynamic constraints, in the framework of extended system methods. The set of equations of motion fulfills the requirement that, following Andersen,[45] reference can be made to a Hamiltonian system via a non-canonical transformation. In Ref. [78], the equations of motion proposed by Andersen[45] for constant pressure molecular dynamics simulation were combined with the Nosé-Hoover approach for constant temperature.[46, 47, 58] In particular, by introducing first an extended Lagrangian in virtual coordinates, a system of Hamiltonian first-order equations of motion was derived for the canonical virtual coordinates and momenta and rearranged in terms of real variables associated to the physical system of interest by means of a non-canonical transformation. Therefore, these equations are derived by a Lagrangian, or in Hamilton canonical form, only in a virtual system of coordinates and become non-canonical when expressed in term of the real variables of the physical system. The particular form of these equations is chosen in such a way that the microcanonical equilibrium distribution produced by a constant-energy trajectory of the extended system reduces to the desired distribution when the positions and momenta of the system of interest are considered, that is after integration over the extra reservoir variables. Indeed, it can be proved that the molecular dynamics trajectory generated by these equations of motion samples the isobaric-isothermal (NPT) ensemble.
The extended system Lagrangian, introduced by Andersen,[45] was designed to produce a constant pressure, constant enthalpy simulation by adding one extra variable, representing the MD cell volume, which is therefore free to fluctuate, and the coupling is made through a dynamical equation of motion for the cell volume variable. When the cell volume changes there is a uniform spatial scaling of the particle positions. In the same paper a constant temperature method was also introduced. This, however, was based on a different mechanism in which the coupling between the heat reservoir and the system of

interest is stochastic and is implemented by randomly sampling particle velocities from a Maxwellian distribution. This method is of relevance when dealing with complex systems where there is a poor coupling between various degrees of freedom, for example molecular internal vibrational and translational or rotational degrees of freedom, and it is probably the best method of achieving adequate equilibrium sampling in these systems. However, here either the stochastic method or the constraint methods are not discussed. Instead, the concentration is focused on the extended system method for constant temperature dynamics first proposed by Nosé[46], and later modified by Hoover[58] and Nosé himself,[47] discussed extensively in Section 5.3.4. In this method, the reservoir is described by a single degree of freedom, which however does not have a simple physical meaning, and the coupling contains a scaling term which applies uniformly to all particle velocities and is in fact a scaling on simulation time. Hoover approach simplifies matters as it does not require the use of virtual variables and scaling concepts and also makes a direct link to constraint methods.
Therefore, the equations of motion for the constant temperature and pressure dynamics, which samples the NPT ensemble, are a combination between the Andersen constant pressure and the Nosé-Hoover constant temperature approaches.[13]
In the constant pressure molecular dynamics method the volume v of the MD box is allowed to fluctuate in time. Andersen replaced the nuclear coordinates $\{\boldsymbol{q}_i\}$ with virtual scaled coordinates $\{\tilde{\boldsymbol{q}}_i\}$ defined in the following way

$$\tilde{\boldsymbol{q}}_i = v^{-1/3}\boldsymbol{q}_i \tag{5.443}$$

Each component of the virtual positions $\{\tilde{\boldsymbol{q}}_i\}$ is therefore a dimensionless number between zero and one (taking care to consider always the image inside the MD box). Similarly, Nosé introduces a scaled time $\tilde{t}$ which is related to the real time variable t through the reservoir coordinate s in the following way

$$d\tilde{t} = s\,dt \tag{5.444}$$

Now the Lagrangian for the extended system can be introduced following Ref. [78] as

$$\tilde{L} = \frac{1}{2}\sum_{i=1}^{N} m_i\,\tilde{s}^2\,\tilde{v}^{2/3}\,\dot{\tilde{\boldsymbol{q}}}_i^2 - \phi(\{\tilde{v}^{1/3}\tilde{\boldsymbol{q}}_i\},\tilde{v}) + \frac{1}{2}W\,\tilde{s}^2\,\dot{\tilde{v}}^2 - P_0\tilde{v} + \frac{1}{2}Q\dot{\tilde{s}}^2 - gk_bT_0\ln\tilde{s} \tag{5.445}$$

in which two new variables $\tilde{v}$ and $\tilde{s}$ appear, and the potential $\phi(\{\tilde{v}^{1/3}\tilde{\boldsymbol{q}}_i\},\tilde{v})$ depends also on the variable $\tilde{v}$. If the variable $\tilde{v}$ is interpreted as the volume v, then the first two terms in (5.445) are just the Lagrangian of the system of interest. The third term is a kinetic energy term, followed by a potential energy term for the motion of $\tilde{v}$ in which two constants appear, the inertia factor W for the barostat degrees of freedom and the external pressure P_0. The firth term is the kinetic term for the thermostat variable $\tilde{s}$, and the last term is the corresponding potential energy. The constants appearing in the last two terms are another inertia factor Q for the thermostat degrees of freedom and the temperature of the reservoir T_0 with Boltzmann constant k_b, while g is the number of degrees of freedom which are computed so that the statistical distribution function for the isobaric-isothermal ensemble in the real system frame of reference has been recovered, starting from the conserved Hamiltonian form. Andersen gives a physical interpretation of the $\tilde{v}$ terms. Imagine that the fluid to be simulated can be compressed in a container whose volume is changed by a piston. Thus the variable W represents the mass of the piston whose motion is described by the coordinate $\tilde{v}$, and $P_0\tilde{v}$ is the potential derived from an external pressure P_0 acting on the piston. The piston is a virtual object and produces an isotropic expansion or contraction of the fluid. However, this interpretation is not completely consistent as, writing down the expected particle velocities

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{d}{dt}(v^{1/3}\tilde{\boldsymbol{q}}_i) = v^{1/3}\frac{d\tilde{\boldsymbol{q}}_i}{dt} + \frac{1}{3}v^{-2/3}\,\tilde{\boldsymbol{q}}_i\frac{dv}{dt} = v^{1/3}s\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} + \frac{\boldsymbol{q}_i}{3v}\frac{dv}{dt} = \tilde{v}^{1/3}\,\tilde{s}\,\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} + \frac{\boldsymbol{q}_i}{3\tilde{v}}\frac{d\tilde{v}}{dt} \tag{5.446}$$

where in the last equality the identities $\tilde{v} = v$ and $\tilde{s} = s$ have been used, one finds that the second term of this derivative does not appear in the kinetic terms of the Lagrangian. However equation (5.445) gives a well-defined Lagrangian for the extended system in terms of virtual coordinates. The next step is

[13] As a word of caution, it must be mentioned that different Lagrangians (and therefore different Hamiltonians) can be set up, all leading to the same equilibrium distribution but with different definitions and dynamics for the reservoir variables.

therefore to derive from it the conjugate momenta. First of all, the momentum conjugate to the particle positions $\tilde{\boldsymbol{q}}_i$ will be denoted as $\tilde{\boldsymbol{p}}_i$ and it is computed as

$$\tilde{\boldsymbol{p}}_i = \frac{\partial \tilde{L}}{\partial \dot{\tilde{\boldsymbol{q}}}_i} = m_i \, \tilde{s}^2 \, \tilde{v}^{2/3} \dot{\tilde{\boldsymbol{q}}}_i \tag{5.447}$$

The momentum conjugate to the variable $\tilde{v}$ will be denoted as $\tilde{p}_v$ and it is given by

$$\tilde{p}_v = \frac{\partial \tilde{L}}{\partial \dot{\tilde{v}}} = W \, \tilde{s}^2 \, \dot{\tilde{v}} \tag{5.448}$$

and finally, the momentum conjugate to the variable $\tilde{s}$ will be denoted as $\tilde{p}_s$ and it is defined by

$$\tilde{p}_s = \frac{\partial \tilde{L}}{\partial \dot{\tilde{s}}} = Q \, \dot{\tilde{s}} \tag{5.449}$$

The Hamiltonian of the extended system can now be written down as

$$\begin{aligned}
\tilde{\mathcal{H}} &= \sum_{i=1}^{N} \dot{\tilde{\boldsymbol{q}}} i \cdot \tilde{\boldsymbol{p}}_i + \dot{\tilde{v}} \tilde{p}_v + \dot{\tilde{s}} \tilde{p}_s - \tilde{L} \\
&= \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i \cdot \tilde{\boldsymbol{p}}_i}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} + W \, \tilde{s}^2 \, \dot{\tilde{v}}^2 + Q \dot{\tilde{s}}^2 - \frac{1}{2} \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i \cdot \tilde{\boldsymbol{p}}_i}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3} \tilde{\boldsymbol{q}}_i\}, \tilde{v}) - \frac{1}{2} W \, \tilde{s}^2 \, \dot{\tilde{v}}^2 + P_0 \tilde{v} - \frac{1}{2} Q \dot{\tilde{s}}^2 + g k_b T_0 \ln \tilde{s} \\
&= \frac{1}{2} \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i \cdot \tilde{\boldsymbol{p}}_i}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3} \tilde{\boldsymbol{q}}_i\}, \tilde{v}) + \frac{1}{2} \, W \, \tilde{s}^2 \, \dot{\tilde{v}}^2 + P_0 \tilde{v} + \frac{1}{2} \, Q \dot{\tilde{s}}^2 + g k_b T_0 \ln \tilde{s}
\end{aligned}$$

Therefore, using the relations (5.448) and (5.449) for the conjugate momenta, the Hamiltonian form for the extended system in virtual variables frame of reference is given by

$$\tilde{\mathcal{H}}(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}, \tilde{p}_s, \tilde{s}, \tilde{p}_v, \tilde{v}) = \frac{1}{2} \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3} \tilde{\boldsymbol{q}}_i\}, \tilde{v}) + \frac{\tilde{p}_v^2}{2 \tilde{s}^2 W} + P_0 \tilde{v} + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln \tilde{s} \tag{5.450}$$

At this point, the equations of motion for the extended system in virtual variables can be derived in the canonical way from the Hamiltonian, using the Hamilton form of the equations of motion, so that

$$\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{\boldsymbol{p}}_i} = \frac{\tilde{\boldsymbol{p}}_i}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} \qquad i = 1, ..., N \tag{5.451}$$

$$\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} = -\frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{\boldsymbol{q}}_i} = -\tilde{v}^{1/3} \, \frac{\partial \phi}{\partial \boldsymbol{q}_i} \qquad i = 1, ..., N \tag{5.452}$$

$$\frac{d\tilde{v}}{d\tilde{t}} = \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{p}_v} = \frac{\tilde{p}_v}{\tilde{s}^2 W} \tag{5.453}$$

$$\frac{d\tilde{p}_v}{d\tilde{t}} = -\frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{v}} = \frac{1}{3\tilde{v}} \sum_{i=1}^{N} \left(\frac{\tilde{\boldsymbol{p}}_i^2}{m_i \, \tilde{s}^2 \, \tilde{v}^{2/3}} - \tilde{v}^{1/3} \, \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \tilde{\boldsymbol{q}}_i \right) - \frac{\partial \phi}{\partial \tilde{v}} - P_0 \tag{5.454}$$

$$\frac{d\tilde{s}}{d\tilde{t}} = \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{p}_s} = \frac{\tilde{p}_s}{Q} \tag{5.455}$$

$$\frac{d\tilde{p}_s}{d\tilde{t}} = -\frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{s}} = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i \, \tilde{s}^3 \, \tilde{v}^{2/3}} + \frac{\tilde{p}_v^2}{\tilde{s}^3 W} - \frac{g k_b T_0}{\tilde{s}} \tag{5.456}$$

The conserved quantities are the Hamiltonian $\tilde{\mathcal{H}}$ expressed in virtual coordinates as in (5.450), the total linear momentum $\sum_{i=1}^{N} \tilde{\boldsymbol{p}}_i$ and the total angular momentum $\sum_{i=1}^{N} \tilde{\boldsymbol{q}}_i \wedge \tilde{\boldsymbol{p}}_i$. Indeed, the total time derivative of the virtual Hamiltonian is equal to zero,

$$\frac{d\tilde{\mathcal{H}}}{d\tilde{t}} = \sum_{i=1}^{N} \left(\frac{\partial \tilde{H}}{\partial \tilde{\boldsymbol{p}}_i} \frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} + \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{\boldsymbol{q}}_i} \frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} \right) + \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{p}_v} \frac{d\tilde{p}_v}{d\tilde{t}} + \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{v}} \frac{d\tilde{v}}{d\tilde{t}} + \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{p}_s} \frac{d\tilde{p}_s}{d\tilde{t}} + \frac{\partial \tilde{\mathcal{H}}}{\partial \tilde{s}} \frac{d\tilde{s}}{d\tilde{t}} = 0 \tag{5.457}$$

The conservation laws for the total linear and angular momenta are derived from equation (5.452) and the properties satisfied by the potential

$$\sum_{i=1}^{N} \frac{\partial \phi}{\partial \tilde{\boldsymbol{q}}_i} = 0 \qquad\qquad \sum_{i=1}^{N} \tilde{\boldsymbol{q}}_i \wedge \frac{\partial \phi}{\partial \tilde{\boldsymbol{q}}_i} = 0 \tag{5.458}$$

However, it should be noted here that during the ordinary molecular dynamics simulations with periodic boundary conditions the angular momentum is not conserved.[29] In the theoretical framework, the equations lead to the conservation of both the linear and angular momenta, thus producing a theoretical ensemble slightly different from that generated by imposing only the conservation of the total linear momentum due to the presence of periodic boundary conditions. These differences are ignored in the following discussion, but they are introduced and discussed later in paragraph A.8.1.

5.4.1.1 Equations of motion in real variables

The equations of motion (5.451)-(5.456) could be solved numerically to produce a time trajectory in the virtual phase space of the extended system. However this turns out not to be the best choice, because integrating with a constant time step $\Delta\tilde{t}$ over the virtual time implies producing a trajectory unevenly spaced in the real time of the system of interest. It is therefore more rewarding to go back from the positions and momenta in virtual space to the positions and momenta in the real space

$$(\tilde{\boldsymbol{p}}_i, \tilde{\boldsymbol{q}}_i, \tilde{p}_v, \tilde{v}, \tilde{p}_s, \tilde{s}, \tilde{t}) \quad \rightarrow \quad (\boldsymbol{p}_i, \boldsymbol{q}_i, p_v, v, p_s, s, t) \tag{5.459}$$

by means of the following transform relations

$$\begin{gathered} \boldsymbol{p}_i = \frac{\tilde{\boldsymbol{p}}_i}{s v^{1/3}} \qquad \boldsymbol{q}_i = v^{1/3}\, \tilde{\boldsymbol{q}}_i \qquad p_v = \frac{\tilde{p}_v}{s} \qquad v = \tilde{v} \\ p_s = \tilde{p}_s \qquad s = \tilde{s} \qquad t = \int^{t} \frac{d\tilde{t}}{s} \end{gathered} \tag{5.460}$$

that have been previously introduced and used in Section 5.4.1 and are explicitly given in Ref. [78]. By applying these transformations to the equations of motion in virtual variables, the correspondent equations of motion in the real frame of reference are obtained to be

$$\begin{aligned} \frac{d\boldsymbol{q}_i}{dt} = s\,\frac{d}{d\tilde{t}}(v^{1/3}\,\tilde{\boldsymbol{q}}_i) = s v^{1/3}\,\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} + \frac{1}{3}\frac{s\tilde{\boldsymbol{q}}_i}{v^{2/3}}\frac{d\tilde{v}}{d\tilde{t}} &= s v^{1/3}\,\frac{\tilde{\boldsymbol{p}}_i}{m_i\, s^2\, v^{2/3}} + \frac{1}{3}\frac{\boldsymbol{q}_i}{v}\, s\frac{d\tilde{v}}{d\tilde{t}} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\,\frac{dv}{dt} \\ &= \frac{\boldsymbol{p}_i}{m_i} + \frac{1}{3}\frac{\boldsymbol{q}_i}{v}\, s\,\frac{\tilde{p}_v}{s^2 W} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\,\frac{p_v}{W} \end{aligned} \tag{5.461}$$

$$\begin{aligned} \frac{d\boldsymbol{p}_i}{dt} = s\,\frac{d\boldsymbol{p}_i}{d\tilde{t}} = s\,\frac{d}{d\tilde{t}}\left(\frac{\tilde{\boldsymbol{p}}_i}{s v^{1/3}}\right) &= \frac{1}{v^{1/3}}\,\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} - \frac{\tilde{\boldsymbol{p}}_i}{s v^{1/3}}\,\frac{ds}{d\tilde{t}} - \frac{1}{3}\frac{\tilde{\boldsymbol{p}}_i}{v^{4/3}}\,\frac{dv}{d\tilde{t}} \\ &= \frac{1}{v^{1/3}}\left(-\tilde{v}^{1/3}\,\frac{\partial \phi}{\partial \boldsymbol{q}_i}\right) - \frac{\boldsymbol{p}_i}{s}\,\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{3v}\,\frac{dv}{dt} \\ &= -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \boldsymbol{p}_i\,\frac{ds}{d\tilde{t}} - \frac{s\boldsymbol{p}_i}{3v}\,\frac{dv}{d\tilde{t}} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \boldsymbol{p}_i\,\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{3v}\,\frac{p_v}{W} \end{aligned} \tag{5.462}$$

$$\frac{dv}{dt} = s\,\frac{dv}{d\tilde{t}} = s\left(\frac{\tilde{p}_v}{s^2 W}\right) = s\left(\frac{s\,p_v}{s^2 W}\right) = \frac{p_v}{W} \tag{5.463}$$

$$\begin{aligned} \frac{dp_v}{dt} = s\,\frac{dp_v}{d\tilde{t}} = s\,\frac{d}{d\tilde{t}}\left(\frac{\tilde{p}_v}{s}\right) = \frac{d\tilde{p}_v}{d\tilde{t}} - \frac{\tilde{p}_v}{s}\,\frac{ds}{d\tilde{t}} &= \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial \phi}{\partial \boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial \phi}{\partial v} - P_0 - \frac{p_v}{s}\left(\frac{ds}{dt}\right) \\ &= \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial \phi}{\partial \boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial \phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \end{aligned} \tag{5.464}$$

$$\frac{ds}{dt} = s\,\frac{ds}{d\tilde{t}} = s\,\frac{p_s}{Q} \tag{5.465}$$

$$\frac{dp_s}{dt} = s\frac{dp_s}{d\tilde{t}} = s\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{sm_i} + \frac{p_v^2}{sW} - \frac{gk_bT_0}{s}\right) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \tag{5.466}$$

Resuming the expression derived above, the equations of motion in real variables (i.e. in the variable associated to the real physical system), that lead to a partition function of the form (A.174) or, including the linear momentum conservation, to a partition function with the form (A.204), well reproducing the NPT ensemble equilibrium distribution function (see Section A.8.1), are given by

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{dv}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.467}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{\boldsymbol{p}_i}{s}\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{3v}\frac{dv}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \boldsymbol{p}_i\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.468}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{5.469}$$

$$\frac{dp_v}{dt} = \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \tag{5.470}$$

$$\frac{ds}{dt} = s\frac{p_s}{Q} \tag{5.471}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \tag{5.472}$$

It is known that, for a system without severe timescale separations, these equations are robust with respect to the choice of the inertial parameters and that their integration does yield the correct sampling of the statistical mechanical ensemble.[78] Comparing the Ferrario equations of motion with the Nosé-Hoover forms (5.258)-(5.261), it can be noted that the first two equations (5.467) and (5.468) are similar to (5.258) and (5.259), respectively, except for an additional term due to the barostat degrees of freedom. Obviously, equations (5.469) and (5.470) do not compare in the Nosé-Hoover form of the equations of motion, since they are related to the evolution in time of the barostat degrees of freedom. On the other hand, equation (5.471) is very similar to (5.260), except for a multiplicative factor s, and equation (5.472) is similar to (5.261) except for a multiplicative factor s and for a kinetic term related to the barostat degree of freedom instead of the thermostat one.
The Hamiltonian (5.450), when expressed using real non-canonical variables, transforms into the conserved energy of the extended system as

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_s, s, p_v, v) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\{\boldsymbol{q}_i\}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + gk_bT_0\ln s \tag{5.473}$$

However, it must be noticed that because the transformation between virtual and real set of coordinates is not canonical, the function (5.473), obtained applying virtual to real reference frame transformations to the Hamiltonian (5.450), cannot be used to derive the equations of motion for the real positions and momenta, given by (5.467)-(5.472). Indeed, starting from the function (5.473), the equations of motion obtained applying Hamilton formulation are

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i} = \frac{\boldsymbol{p}_i}{m_i} \qquad i = 1, ..., N \tag{5.474}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} \qquad i = 1, ..., N \tag{5.475}$$

$$\frac{dv}{dt} = \frac{\partial\mathcal{H}}{\partial p_v} = \frac{p_v}{W} \tag{5.476}$$

$$\frac{dp_v}{dt} = -\frac{\partial\mathcal{H}}{\partial v} = -\frac{\partial\phi}{\partial v} - P_0 \tag{5.477}$$

$$\frac{ds}{dt} = \frac{\partial\mathcal{H}}{\partial p_s} = \frac{p_s}{Q} \tag{5.478}$$

$$\frac{dp_s}{dt} = -\frac{\partial \mathcal{H}}{\partial s} = -\frac{g k_b T_0}{s} \tag{5.479}$$

which are clearly different from the previously derived equations of motion (5.467)-(5.472), that are computed by applying the virtual to real reference frame transformations (5.460) to the virtual variables equations of motion (5.451)-(5.456).
At the same time, the function (5.473) still remains a constant of motion for the time evolution defined by equations (5.467)-(5.472), so that its value can be monitored during a dynamical simulation and it can be used to judge the reliability of time trajectory and ensemble sampling.

5.4.1.2 Integration of the equations of motion

Using the equations of motion in the real reference frame, derived in Section 5.4.1.1, a time development operator can be formulated by the decomposed Liouville operators, as explained in Section 4.2. By changing the order of these decomposed operators, several time integrators can be thought. In the case of constant temperature and pressure, since the degrees of freedom associated to a thermostat and a barostat have to be added, the vector $\mathbf{\Gamma}$ contains $2dN + 4$ variables in the phase space of the extended system, which is defined by $\mathbf{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t), p_v(t), v(t), p_s(t), s(t))$. Its time development (4.65) from time t_0 to a generic time $t = t_0 + P\,\tau$, where P is an integer representing the number of dynamics steps the phase space coordinates are allowed to propagate and τ is the simulation time step unit, is written as

$$\mathbf{\Gamma}(t) = \mathbf{\Gamma}(t_0 + P\,\tau) = e^{i\mathcal{L}\,P\,\tau}\,\mathbf{\Gamma}(t_0) \tag{5.480}$$

where the Liouville operator is given by equation (4.62) reported here below for the case of the extended system of interest

$$i\mathcal{L} = \dot{\mathbf{\Gamma}} \cdot \frac{\partial}{\partial \mathbf{\Gamma}} \qquad \text{with} \quad \mathbf{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t), p_v(t), v(t), p_s(t), s(t)) \tag{5.481}$$

where the dot over the phase space vector variable stands for the time derivative. The equations of motion (5.467)-(5.472) can be rewritten using the variable transformation (5.498), that modifies only the form of equation (5.471) involving the thermostat degree of freedom s, so that the new set of equations of motion to be considered is

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{dv}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.482}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \frac{\boldsymbol{p}_i}{s}\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{3v}\frac{dv}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{q}_i} - \boldsymbol{p}_i\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.483}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{5.484}$$

$$\frac{dp_v}{dt} = \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial \phi}{\partial \boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial \phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \equiv F_v - \frac{p_v p_s}{Q} \tag{5.485}$$

$$\frac{d\eta}{dt} = \frac{p_s}{Q} \tag{5.486}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - g k_b T_0 \equiv F_s \tag{5.487}$$

where two new force variables have been introduced in equations (5.485) and (5.487) to further simplify the notation, namely,

$$F_v \equiv \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial \phi}{\partial \boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial \phi}{\partial v} - P_0 \qquad F_s \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - g k_b T_0 \tag{5.488}$$

Therefore, using the new variable defined in (5.498), the evolution in phase space of the system vector (5.481) becomes

$$i\mathcal{L} = \dot{\mathbf{\Gamma}} \cdot \frac{\partial}{\partial \mathbf{\Gamma}} \qquad \text{with} \quad \mathbf{\Gamma}(t) = (\boldsymbol{p}(t), \boldsymbol{q}(t), p_v(t), v(t), p_s(t), \eta(t)) \tag{5.489}$$

The conserved quantity corresponding to the Hamiltonian (5.473) expressed in real coordinates (physical system frame of reference) can be rewritten using the transformation (5.498) as in equation (5.534). Starting from the formula (5.489), and using the Ferrario equations of motion (5.482)-(5.487) just obtained, the form of the Liouville operator can be easily derived to be

$$\begin{aligned} i\mathcal{L} &= \sum_{i=1}^{Nd} \frac{d\boldsymbol{q}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \sum_{i=1}^{Nd} \frac{d\boldsymbol{p}_i}{dt} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} + \frac{dv}{dt}\frac{\partial}{\partial v} + \frac{dp_v}{dt}\frac{\partial}{\partial p_v} + \frac{d\eta}{dt}\frac{\partial}{\partial \eta} + \frac{dp_s}{dt}\frac{\partial}{\partial p_s} \\ &= \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \sum_{i=1}^{Nd} \frac{\boldsymbol{q}_i}{3v}\frac{p_v}{W} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} - \sum_{i=1}^{Nd} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} - \sum_{i=1}^{Nd} \boldsymbol{p}_i \frac{p_s}{Q} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} - \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{3v}\frac{p_v}{W} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \\ &\quad + \frac{p_v}{W}\frac{\partial}{\partial v} + F_v \frac{\partial}{\partial p_v} - \frac{p_v p_s}{Q}\frac{\partial}{\partial p_v} + \frac{p_s}{Q}\frac{\partial}{\partial \eta} + F_s \frac{\partial}{\partial p_s} \end{aligned} \tag{5.490}$$

The evolution in time of real system and thermostat and barostat degrees of freedom can be modeled through a velocity Verlet-like reversible scheme of integration derived using the Trotter-Suzuki factorization of the Liouville operator (see Section 4.2.1). Using a ten-terms decomposition of the Liouville operator (with the generalized factorization (4.93) derived in Section 4.2.1), the Liouville operator can be split into the sum of ten operators as follows

$$i\mathcal{L} = i\mathcal{L}_1 + i\mathcal{L}_2 + i\mathcal{L}_3 + i\mathcal{L}_4 + i\mathcal{L}_5 + i\mathcal{L}_6 + i\mathcal{L}_7 + i\mathcal{L}_8 + i\mathcal{L}_9 + i\mathcal{L}_{10} \tag{5.491}$$

where

$$i\mathcal{L}_1 = \sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{m_i} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} \qquad i\mathcal{L}_2 = \sum_{i=1}^{Nd} \frac{\boldsymbol{q}_i}{3v}\frac{p_v}{W} \cdot \frac{\partial}{\partial \boldsymbol{q}_i} \qquad i\mathcal{L}_3 = \frac{p_v}{W}\frac{\partial}{\partial v} \tag{5.492}$$

$$i\mathcal{L}_4 = -\sum_{i=1}^{Nd} \frac{\partial \phi}{\partial \boldsymbol{q}_i} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \qquad i\mathcal{L}_5 = -\sum_{i=1}^{Nd} \boldsymbol{p}_i \frac{p_s}{Q} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \qquad i\mathcal{L}_6 = -\sum_{i=1}^{Nd} \frac{\boldsymbol{p}_i}{3v}\frac{p_v}{W} \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \tag{5.493}$$

$$i\mathcal{L}_7 = F_v \frac{\partial}{\partial p_v} \qquad i\mathcal{L}_8 = -\frac{p_v p_s}{Q}\frac{\partial}{\partial p_v} \qquad i\mathcal{L}_9 = \frac{p_s}{Q}\frac{\partial}{\partial \eta} \qquad i\mathcal{L}_{10} = F_s \frac{\partial}{\partial p_s} \tag{5.494}$$

The separated operators $i\mathcal{L}_1$ and $i\mathcal{L}_4$ are the same as in the NVE ensemble (see Section 5.1.2), while the operators $i\mathcal{L}_5$ and $i\mathcal{L}_9$ have a similar form with respect to Nosé-Hoover operators $i\mathcal{L}_3$ and $i\mathcal{L}_4$ in (5.284), respectively (the Nosé-Hoover equations involve the variable $p_\eta = sp_s$, while the expressions defined above use the variable p_s). Furthermore, the operator $i\mathcal{L}_5$ in (5.284) is associated to the operator $i\mathcal{L}_{10}$ in (5.494) and it is reproduced by the two terms in F_s (see the second identity in (5.488)). All the other operators, namely, $i\mathcal{L}_2$, $i\mathcal{L}_3$, $i\mathcal{L}_6$, $i\mathcal{L}_7$, $i\mathcal{L}_8$ and a term in $i\mathcal{L}_{10}$ are additional operators which come from the variables associated to the barostat degrees of freedom. From the operator separations in equations (5.492)-(5.494), several decomposition forms of the time development operator $\exp(i\mathcal{L}\tau)$ can be considered as time integrators. Indeed, different variants of the time integrator form can be obtained by the permutation of the ten separated Liouville operators. However, because some operators in the decomposition commute to each other, the variants number reduces, and it is less than the total number of possible permutations of the ten operators.

Starting from the ten-term decomposition of the Liouville operator introduced above, the approximate discrete time propagator can now be derived by means of the Suzuki-Trotter factorization formula (4.93), with $P = 1$ (one time step) and a number of Liouville operators equal to $n = 10$ (see Section 4.2.1), following the decomposition order applied by A. Sergi *et al.*[84], given by

$$\begin{aligned} e^{i\mathcal{L}\tau} \approx\ & e^{i\mathcal{L}_{10}\tau/2}\, e^{i\mathcal{L}_9\tau/2}\, e^{i\mathcal{L}_8\tau/2}\, e^{i\mathcal{L}_7\tau/2}\, e^{i\mathcal{L}_6\tau/2}\, e^{i\mathcal{L}_5\tau/2}\, e^{i\mathcal{L}_4\tau/2}\, e^{i\mathcal{L}_3\tau/2}\, e^{i\mathcal{L}_2\tau/2}\, e^{i\mathcal{L}_1\tau} \cdot \\ & \cdot\, e^{i\mathcal{L}_2\tau/2}\, e^{i\mathcal{L}_3\tau/2}\, e^{i\mathcal{L}_4\tau/2}\, e^{i\mathcal{L}_5\tau/2}\, e^{i\mathcal{L}_6\tau/2}\, e^{i\mathcal{L}_7\tau/2}\, e^{i\mathcal{L}_8\tau/2}\, e^{i\mathcal{L}_9\tau/2}\, e^{i\mathcal{L}_{10}\tau/2} + O(\tau^{19}) \end{aligned} \tag{5.495}$$

which can be more conveniently rewritten using the shortest notation

$$e^{i\mathcal{L}\tau} \approx \left(\prod_{i=10}^{2} e^{i\mathcal{L}_i\tau/2} \right) e^{i\mathcal{L}_1\tau} \left(\prod_{i=2}^{10} e^{i\mathcal{L}_i\tau/2} \right) + O(\tau^{19}) \tag{5.496}$$

where the product order follows the index value with multiplication factors applied from left to right. The above defined time propagator has to be used in equation (5.480) to compute the time propagation of phase space variables from an initial time t_0 to $t_0 + \tau$ (one time step, i.e. $P = 1$) in the extended phase space, so that

$$\begin{aligned}\boldsymbol{\Gamma}(t_0+\tau) \approx\ & e^{i\mathcal{L}_{10}\,\tau/2}\, e^{i\mathcal{L}_{9}\,\tau/2}\, e^{i\mathcal{L}_{8}\,\tau/2}\, e^{i\mathcal{L}_{7}\,\tau/2}\, e^{i\mathcal{L}_{6}\,\tau/2}\, e^{i\mathcal{L}_{5}\,\tau/2}\, e^{i\mathcal{L}_{4}\,\tau/2}\, e^{i\mathcal{L}_{3}\,\tau/2}\, e^{i\mathcal{L}_{2}\,\tau/2}\, e^{i\mathcal{L}_{1}\,\tau}\,\cdot \\ & \cdot\ e^{i\mathcal{L}_{2}\,\tau/2}\, e^{i\mathcal{L}_{3}\,\tau/2}\, e^{i\mathcal{L}_{4}\,\tau/2}\, e^{i\mathcal{L}_{5}\,\tau/2}\, e^{i\mathcal{L}_{6}\,\tau/2}\, e^{i\mathcal{L}_{7}\,\tau/2}\, e^{i\mathcal{L}_{8}\,\tau/2}\, e^{i\mathcal{L}_{9}\,\tau/2}\, e^{i\mathcal{L}_{10}\,\tau/2}\, \boldsymbol{\Gamma}(t_0) + O(\tau^{19})\end{aligned}$$

or, equivalently, using (5.496) to shorten the notation,

$$\boldsymbol{\Gamma}(t_0+\tau) \approx \left[\left(\prod_{i=10}^{2} e^{i\mathcal{L}_i\,\tau/2}\right) e^{i\mathcal{L}_1\,\tau} \left(\prod_{i=2}^{10} e^{i\mathcal{L}_i\,\tau/2}\right)\right] \boldsymbol{\Gamma}(t_0) + O(\tau^{19}) \tag{5.497}$$

Obviously, this phase space propagation formula has been defined for a single time step ($P = 1$), starting from an initial time t_0. The number of steps P defined in a molecular dynamics simulation is in generally very high, so that P is a large but finite integer which approximates the condition $P \to \infty$ defined in the exact formulation of Suzuki-Trotter decomposition, equation (4.92). Indeed, in a dynamical simulation the formula (5.497) has to be applied sequentially P times, propagating the phase space point step by step until a final time $t = t_0 + P\,\tau$ (see equation (5.480)), with a discretized time unit of $\tau = \Delta t$.
The action of operators with the forms (5.492)-(5.494) on a phase space point has been demonstrated in Section 4.2.1, and it is therefore applied for each operator in the order given by equation (5.497). The resulting algorithm has the structure reported in Table 5.8, for a time step equal to $\tau = \Delta t$.

5.4.1.3 Conserved quantities

The search for conserved quantities can be started from the equations of motion in real variables, given by (5.467) - (5.472), and by rewriting them using the variable transformation

$$\eta = \ln s \tag{5.498}$$

that modifies only the form of equation (5.471) involving the thermostat degree of freedom s, so that the new set of equations of motion to be considered in the real variables frame of reference is

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{dv}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.499}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{\boldsymbol{p}_i}{s}\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{3v}\frac{dv}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \boldsymbol{p}_i\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{3v}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.500}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{5.501}$$

$$\frac{dp_v}{dt} = \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \equiv F_v - \frac{p_v p_s}{Q} \tag{5.502}$$

$$\frac{d\eta}{dt} = \frac{p_s}{Q} \tag{5.503}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \equiv F_s \tag{5.504}$$

where two new force variables have been introduced in equations (5.502) and (5.504) to further simplify the notation, namely,

$$F_v \equiv \frac{1}{3v}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 \qquad F_s \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \tag{5.505}$$

In the previous set of equations of motion (5.499) - (5.504), a three dimensional space ($d = 3$) has been considered. However, the generalization of the equations (5.499) - (5.504) at an arbitrary dimension d of the physical space leads to the set of equations of motion

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{vd}\frac{dv}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{vd}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.506}$$

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{p}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}],\ v(t),\ p_v(t),\ \eta(t),\ p_s(t)$
Step 1. Propagator $e^{i\mathcal{L}_{10}\Delta t/2}$	:	$p_s(t+\Delta t/2) \leftarrow p_s(t) + \dfrac{\Delta t}{2} F_s(t)$
Step 2. Propagator $e^{i\mathcal{L}_{9}\Delta t/2}$	:	$\eta(t+\Delta t/2) \leftarrow \eta(t) + \dfrac{\Delta t}{2Q} p_s(t+\Delta t/2)$
Step 3. Propagator $e^{i(\mathcal{L}_{8}+\mathcal{L}_7)\Delta t/2}$	:	$p_v(t+\Delta t/2) \leftarrow p_v(t) \exp\left[-\dfrac{\Delta t}{2Q} p_s(t+\Delta t/2)\right] + \dfrac{\Delta t}{2} F_v(t)$
Step 4. Propagator $e^{i(\mathcal{L}_{6}+\mathcal{L}_5)\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t) \exp\left[-\dfrac{\Delta t}{2W} \dfrac{p_v(t+\Delta t/2)}{3v(t)} - \dfrac{\Delta t}{2Q} p_s(t+\Delta t/2)\right]$
Step 5. Propagator $e^{i\mathcal{L}_{4}\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \dfrac{\Delta t}{2} \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 6. Propagator $e^{i\mathcal{L}_{3}\Delta t/2}$	:	$v(t+\Delta t/2) \leftarrow v(t) + \dfrac{\Delta t}{2W} p_v(t+\Delta t/2)$
Step 7. Propagator $e^{i\mathcal{L}_{2}\Delta t/2}$	:	$\boldsymbol{q}_i(t+\Delta t/2) \leftarrow \boldsymbol{q}_i(t) \exp\left[\dfrac{\Delta t}{2W} \dfrac{p_v(t+\Delta t/2)}{3v(t+\Delta t/2)}\right]$
Step 8. Propagator $e^{i\mathcal{L}_{1}\Delta t}$	:	$\boldsymbol{q}_i(t+\Delta t/2) \leftarrow \boldsymbol{q}_i(t+\Delta t/2) + \dfrac{\Delta t}{m_i} \boldsymbol{p}_i(t+\Delta t/2)$
Step 9. Propagator $e^{i\mathcal{L}_{2}\Delta t/2}$	:	$\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t+\Delta t/2) \exp\left[\dfrac{\Delta t}{2W} \dfrac{p_v(t+\Delta t/2)}{3v(t+\Delta t/2)}\right]$
Step 10. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 11. Propagator $e^{i\mathcal{L}_{3}\Delta t/2}$	:	$v(t+\Delta t) \leftarrow v(t+\Delta t/2) + \dfrac{\Delta t}{2W} p_v(t+\Delta t/2)$
Step 12. Propagator $e^{i\mathcal{L}_{4}\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \dfrac{\Delta t}{2} \mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 13. Propagator $e^{i(\mathcal{L}_{5}+\mathcal{L}_6)\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) \exp\left[-\dfrac{\Delta t}{2Q} p_s(t+\Delta t/2) - \dfrac{\Delta t}{2W} \dfrac{p_v(t+\Delta t/2)}{3v(t+\Delta t)}\right]$
Step 14. Propagator $e^{i(\mathcal{L}_{7}+\mathcal{L}_8)\Delta t/2}$	:	$p_v(t+\Delta t) \leftarrow \left(p_v(t+\Delta t/2) + \dfrac{\Delta t}{2} F_v(t+\Delta t)\right) \exp\left[-\dfrac{\Delta t}{2Q} p_s(t+\Delta t/2)\right]$
Step 15. Propagator $e^{i\mathcal{L}_{9}\Delta t/2}$	:	$\eta(t+\Delta t) \leftarrow \eta(t+\Delta t/2) + \dfrac{\Delta t}{2Q} p_s(t+\Delta t/2)$
Step 16. Propagator $e^{i\mathcal{L}_{10}\Delta t/2}$	:	$p_s(t+\Delta t) \leftarrow p_s(t+\Delta t/2) + \dfrac{\Delta t}{2} F_s(t+\Delta t)$

Table 5.8: Ferrario thermostat and barostat integrator algorithm, applied $\forall\, i = 1, ..., N$.

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{\boldsymbol{p}_i}{s}\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{vd}\frac{vd}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \boldsymbol{p}_i\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{vd}\frac{p_v}{W} \qquad i = 1, ..., N \tag{5.507}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{5.508}$$

$$\frac{dp_v}{dt} = \frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \equiv F_v - \frac{p_v p_s}{Q} \tag{5.509}$$

$$\frac{d\eta}{dt} = \frac{p_s}{Q} \tag{5.510}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \equiv F_s \tag{5.511}$$

where the two force variables used in equations (5.509) and (5.511) are given by

$$F_v \equiv \frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 \qquad F_s \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \tag{5.512}$$

where all the positions and momenta in (5.506) - (5.511) and (5.512) are d dimensional vectors. Obviously, taking $d = 3$, the set of equations (5.506) - (5.512) reduces to the set (5.499) - (5.505).

In order to determine the statistical mechanical ensemble generated by the equations of motion (5.506) - (5.511), following the analysis outlined in Section 4.3.2.1, the constants of motion have to be found. The simplest one is the total Hamiltonian (5.473), written using the change of variables (5.498), that is

$$\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_v, v, p_s, \eta) = \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + gk_bT_0\eta = C_1 \tag{5.513}$$

where the Hamiltonian on the right hand side of the previous equation has been defined as

$$\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\{\boldsymbol{q}_i\}, v) \tag{5.514}$$

If there exists a non zero net force acting on the system, then (5.513) is the only constant of motion preserved by the non Hamiltonian flow of equations (5.506) - (5.511). In general, however, there will be more conserved quantities. For instance, considering a system in the absence of external forces, namely, under the hypothesis

$$\sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.515}$$

then there are d additional conservation laws satisfied by the equations of motion (5.506) - (5.511), which take the form

$$\mathbf{P}\, v^{1/d}\, e^{\eta} = \mathbf{K} \tag{5.516}$$

where, $\mathbf{K}$ is an arbitrary constant vector in d dimensions (with d the dimension of the physical space) and $\mathbf{P}$ is the total linear momentum defined as

$$\mathbf{P} = \sum_{i=1}^{N}\boldsymbol{p}_i \tag{5.517}$$

with the origin taken at the center of mass of the system. Indeed, the conservation of the quantities in

(5.516) can be demonstrated by taking explicitly the time derivative, that is

$$\begin{aligned}\frac{d}{dt}(\mathbf{P}\,v^{1/d}\,e^{\eta}) &= \dot{\mathbf{P}}\,v^{1/d}\,e^{\eta} + \frac{1}{d}\,\mathbf{P}\,v^{1/d-1}\,e^{\eta}\,\dot{v} + \mathbf{P}\,v^{1/d}\,e^{\eta}\,\dot{\eta} \\ &= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\frac{d\boldsymbol{p}_i}{dt} + \frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P} + \frac{p_s}{Q}\,\mathbf{P}\right] \\ &= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\left(-\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \boldsymbol{p}_i\,\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{vd}\,\frac{p_v}{W}\right) + \frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P} + \frac{p_s}{Q}\,\mathbf{P}\right] \\ &= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_s}{Q}\sum_{i=1}^{N}\boldsymbol{p}_i - \frac{1}{vd}\,\frac{p_v}{W}\sum_{i=1}^{N}\boldsymbol{p}_i + \frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P} + \frac{p_s}{Q}\,\mathbf{P}\right] \\ &= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\mathbf{F}_i - \frac{p_s}{Q}\,\mathbf{P} - \frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P} + \frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P} + \frac{p_s}{Q}\,\mathbf{P}\right] = e^{\eta}\,v^{1/d}\sum_{i=1}^{N}\mathbf{F}_i\end{aligned} \tag{5.518}$$

where the equations of motion (5.507), (5.508) and (5.510) have been used. Therefore, as demonstrated in (5.518), the total time derivative of the quantity (5.516) is equal to zero iff the condition (5.515) is satisfied. In this case, an additional conservative law has been individuated, that is effectively conserved if the value of the total force acting of the system is equal to zero. The conservation law (5.516) can be written using a single independent variable (note that the d components of the total linear momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.516), so that

$$\|\mathbf{P}\,v^{1/d}\,e^{\eta}\| = P\,v^{1/d}\,e^{\eta} = \|\mathbf{K}\| = K \tag{5.519}$$

Hence, the conservation law in the absence of net forces on the system becomes

$$P\,v^{1/d}\,e^{\eta} = K = C_2 \qquad \leftrightarrow \qquad \mathbf{F} = \sum_{i=1}^{N}\mathbf{F}_i = 0 \tag{5.520}$$

Therefore, the quantity conserved by Ferrario equations of motion in the absence of external forces, namely when the condition (5.515) holds, is given by (5.520).
Furthermore, if the following condition is fulfilled

$$\sum_{i=1}^{N}\boldsymbol{q}_i \wedge \mathbf{F}_i = 0 \tag{5.521}$$

then there are d additional conservation laws satisfied by the equations of motion (5.506) - (5.511), which take the form

$$\mathbf{L}\,e^{\eta} = \mathbf{K} \tag{5.522}$$

where $\mathbf{K}$ is an arbitrary constant vector in d dimensions and $\mathbf{L}$ is the total angular momentum of the system defined as

$$\mathbf{L} = \sum_{i=1}^{N}\boldsymbol{q}_i \wedge \boldsymbol{p}_i \tag{5.523}$$

with the origin taken at the center of mass of the system. Equation (5.522) can be verified by taking

the time derivative explicitly, that is

$$\begin{aligned}\frac{d}{dt}(\mathbf{L}\,e^{\eta}) &= \dot{\mathbf{L}}\,e^{\eta} + \mathbf{L}\,e^{\eta}\,\dot{\eta} \\ &= e^{\eta}\left[\sum_{i=1}^{N}\left(\frac{d\boldsymbol{q}_i}{dt}\wedge\boldsymbol{p}_i + \boldsymbol{q}_i\wedge\frac{d\boldsymbol{p}_i}{dt}\right) + \frac{p_s}{Q}\,\mathbf{L}\right] \\ &= e^{\eta}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i}{m_i}\wedge\boldsymbol{p}_i + \frac{p_v}{vdW}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i - \boldsymbol{q}_i\wedge\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_s}{Q}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i - \frac{p_v}{vdW}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i\right) + \frac{p_s}{Q}\,\mathbf{L}\right] \\ &= e^{\eta}\left(-\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{p_s}{Q}\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\boldsymbol{p}_i + \frac{p_s}{Q}\,\mathbf{L}\right) \\ &= e^{\eta}\left(\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i - \frac{p_s}{Q}\,\mathbf{L} + \frac{p_s}{Q}\,\mathbf{L}\right) = e^{\eta}\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i\end{aligned} \tag{5.524}$$

where the equations of motion (5.506), (5.507) and (5.510) have been used, together with the definition of the total angular momentum (5.523). Therefore, as demonstrated in (5.524), the total time derivative of the quantity (5.522) is equal to zero iff the condition (5.521) is satisfied. The conservation law (5.522) can be written using a single independent variable (note that the d components of the total angular momentum are linearly dependent), by taking the Euclidean norm on both sides of equation (5.522), so that

$$\|\mathbf{L}\,e^{\eta}\| = L\,e^{\eta} = \|\mathbf{K}\| = K \tag{5.525}$$

Hence, the conservation law is given by

$$L\,e^{\eta} = K = C_3 \qquad \leftrightarrow \qquad \sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i = 0 \tag{5.526}$$

Therefore, the quantity conserved by Ferrario equations of motion (5.506) - (5.511) when the condition (5.521) holds, is given by (5.526). Note that the conservation quantity that involves the total linear momentum includes both the thermostat and the barostat degrees of freedom, while the conserved quantity related to the total angular momentum includes the thermostat degrees of freedom only. There is a sort of symmetry breaking, a consequence of the form of the equations of motion (5.506) - (5.511). Other quantities (related to total angular momentum with only thermostat degrees of freedom and total linear momentum with both thermostat and barostat degrees of freedom) are analyzed as possible candidates for a conservation law (see Appendix A, Section A.8.3), but the only two quantities which results conserved are those previously analyzed, namely (5.520) and 5.526, as a consequence of the form of the equations of motion (5.506) - (5.511).
A more general criterion for the existence of these conservation laws is provided by symmetry considerations (see Noether theorem, Ref. [33]). If the system is invariant to translation in a particular direction, then the corresponding momentum component is conserved. If the system is invariant to rotation about an axis, then the corresponding angular momentum component is conserved. Thus, the three quantities (5.513), (5.520) and (5.526) are conserved for a completely isolated set of interacting molecules subject to the equations of motion (5.506) - (5.511). In practice, however, a completely isolated system is rarely considered, except for the case of an isolated molecule. Indeed, when systems are enclosed in a spherical box are encounter, then all the three components of total angular momentum about the centre of symmetry will be conserved, but total translational momentum will not be. If the surrounding walls formed a cubical box, none of these quantities would be conserved. This is an important case in practice, since for the practical simulations, periodic boundary conditions are often used, i.e. the system is enclosed in a periodically repeated box. In the case of such periodic boundary conditions, it is easy to see that translational invariance is preserved, and hence total linear momentum is conserved. Several different box geometries can be considered (e.g. cubic, orthorhombic, or truncated octahedron), but none of them can be spherically symmetrical; in fact it is impossible (in Euclidean space) to construct a spherically symmetric periodic system. Hence, total angular momentum is not conserved in most molecular dynamics simulations. As a consequence, if periodic boundary conditions are included in the simulation, then

there are only two quantities conserved by the non Hamiltonian flow of the equations of motion (5.506) - (5.511), given by (5.513) and (5.520).
In order to resume these concepts, in Table 5.9, the constants of motion for the Ferrario equations (5.506) - (5.511) are reported, together with the simulation conditions that are necessary to preserve their values. Note that the total Hamiltonian (5.513) is always a constant of motion, independently of the simulation conditions.

conservation laws of Ferrario equations (5.506) - (5.511)
$\mathcal{H}(\boldsymbol{p},\boldsymbol{q},p_v,v,p_s,\eta)=\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}+\phi(\{\boldsymbol{q}_i\},v)+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta=C_1$
$P\,v^{1/d}\,e^{\eta}=K=C_2 \qquad \text{iff} \qquad \mathbf{F}=\sum_{i=1}^{N}\mathbf{F}_i=0$
$L\,e^{\eta}=K=C_3 \qquad \text{iff} \qquad \sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i=0 \qquad \text{and} \qquad \text{NO PBC}$

Table 5.9: Constants of motion for the Ferrario equations of motion (5.506) - (5.511), together with the simulation conditions that are necessary to preserve these quantities.

5.4.1.4 Statistical mechanical ensemble

Solving numerically equations (5.451) - (5.456), one can produce a time trajectory for the extended system and therefore calculate time averages as in the standard molecular dynamics to estimate the macroscopic properties of the system of interest. The question is now what kind of averages can be computed in this way with respect to statistical mechanics ensembles. The ensemble generated can be studied starting from the equations of motion in real variables, given by (5.467) - (5.472), rewriting them using the variable transformation (5.498), that modifies only the form of equation (5.471) involving the thermostat degree of freedom s, and generalizing the equations to a d dimensional space. In this way, the new set of equations of motion to be considered in the real variables frame of reference is given by (5.506) - (5.511), and it is reported here below

$$\frac{d\boldsymbol{q}_i}{dt}=\frac{\boldsymbol{p}_i}{m_i}+\frac{\boldsymbol{q}_i}{vd}\frac{dv}{dt}=\frac{\boldsymbol{p}_i}{m_i}+\frac{\boldsymbol{q}_i}{vd}\frac{p_v}{W} \qquad i=1,...,N \tag{5.527}$$

$$\frac{d\boldsymbol{p}_i}{dt}=-\frac{\partial\phi}{\partial\boldsymbol{q}_i}-\frac{\boldsymbol{p}_i}{s}\frac{ds}{dt}-\frac{\boldsymbol{p}_i}{vd}\frac{vd}{dt}=-\frac{\partial\phi}{\partial\boldsymbol{q}_i}-\boldsymbol{p}_i\frac{p_s}{Q}-\frac{\boldsymbol{p}_i}{vd}\frac{p_v}{W} \qquad i=1,...,N \tag{5.528}$$

$$\frac{dv}{dt}=\frac{p_v}{W} \tag{5.529}$$

$$\frac{dp_v}{dt}=\frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i}-\frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right)-\frac{\partial\phi}{\partial v}-P_0-\frac{p_vp_s}{Q}\equiv F_v-\frac{p_vp_s}{Q} \tag{5.530}$$

$$\frac{d\eta}{dt}=\frac{p_s}{Q} \tag{5.531}$$

$$\frac{dp_s}{dt}=\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}+\frac{p_v^2}{W}-gk_bT_0\equiv F_s \tag{5.532}$$

where all the positions and momenta are d dimensional vectors and the two force variables used in equations (5.530) and (5.532) are given by

$$F_v\equiv\frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i}-\frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right)-\frac{\partial\phi}{\partial v}-P_0 \qquad F_s\equiv\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}+\frac{p_v^2}{W}-gk_bT_0 \tag{5.533}$$

Note that, in equations (5.527) - (5.532), both the particles and the barostat are coupled to a single Nosé-Hoover thermostat. The equations of motion (5.527) - (5.532) produce the conserved energy

$$\begin{aligned}\mathcal{H}(\boldsymbol{p},\boldsymbol{q},p_v,v,p_s,\eta) &= \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}+\phi(\{\boldsymbol{q}_i\},v)+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta \\ &\equiv \mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta = C_1\end{aligned} \tag{5.534}$$

which corresponds to the Hamiltonian (5.473) expressed in real coordinates, rewritten using the transformation (5.498), and where the Hamiltonian introduced in the second line has been defined as

$$\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}+\phi(\{\boldsymbol{q}_i\},v) \tag{5.535}$$

As demonstrated at the end of Section 5.4.1.1, the Hamiltonian in equation (5.534) cannot be used to derive the equations of motion (5.527) - (5.532) for the real positions and momenta, while it still remains a constant of motion for the time evolution equations (5.527) - (5.532). This is due to the fact that the change of variables from the virtual to the real set of coordinates is not a canonical transformation.
At this point, the procedure described in Section 4.3.2.1 for constructing the microcanonical partition corresponding to the equations of motion (5.506) - (5.512) will now be carried on, assuming that the Hamiltonian (5.534) is the only conserved quantity. First of all, the compressibility, defined as in (4.123) with $\boldsymbol{\Gamma}=(\boldsymbol{p},\boldsymbol{q},p_v,v,p_s,\eta)$, can be computed using the equations of motion (5.506) - (5.511) as

$$\begin{aligned}\kappa(\boldsymbol{\Gamma},t) &= \nabla_{\boldsymbol{\Gamma}}\cdot\dot{\boldsymbol{\Gamma}} = \sum_{i=1}^{N}[\nabla_{\boldsymbol{p}_i}\cdot\dot{\boldsymbol{p}}_i+\nabla_{\boldsymbol{q}_i}\cdot\dot{\boldsymbol{q}}_i]+\frac{\partial\dot{p}_v}{\partial p_v}+\frac{\partial\dot{v}}{\partial v}+\frac{\partial\dot{p}_s}{\partial p_s}+\frac{\partial\dot{\eta}}{\partial\eta} \\ &= \sum_{i=1}^{N}\left[-d\,\frac{p_s}{Q}-\frac{1}{v}\,\frac{p_v}{W}+\frac{1}{v}\,\frac{p_v}{W}\right]-\frac{p_s}{Q} = -Nd\,\frac{p_s}{Q}-\frac{p_s}{Q} = -(Nd+1)\,\frac{p_s}{Q} = -(Nd+1)\,\dot{\eta}\end{aligned} \tag{5.536}$$

The metric can then be computed as

$$\begin{aligned}\sqrt{g(\boldsymbol{\Gamma},t)} &= \exp\left(-\int\kappa(\boldsymbol{\Gamma},t)\,dt\right) = \exp\left(\int(Nd+1)\,\frac{d\eta}{dt}\,dt\right) \\ &= \exp\left\{(Nd+1)\int\frac{d\eta}{dt}\,dt\right\} = \exp\{(Nd+1)\,\eta\}\end{aligned} \tag{5.537}$$

Thus, if there are no other conservation laws apart from equation (5.534), the microcanonical partition function becomes

$$Z(N,C_1) = \zeta\int d\boldsymbol{p}\int d\boldsymbol{q}\int dp_v\int dv\int dp_s\int d\eta\; e^{(Nd+1)\,\eta}\;\delta\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)$$

By introducing the relation

$$\delta[f(x)] = \frac{\delta(x-x_0)}{f'(x_0)} \tag{5.538}$$

where x_0 is the zero of function $f(x)$ in the argument of the Dirac function, and $f'(x_0)$ is the derivative of the function $f(x)$ evaluated in the point x_0, and applying this relation to the variable η in the argument of the delta function in the previous partition function, with

$$\eta_0 = \frac{1}{gk_bT_0}\left[C_1-\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})-\frac{p_v^2}{2W}-P_0v-\frac{p_s^2}{2Q}\right] \qquad \text{and} \qquad f'(\eta) = f'(\eta_0) = gk_bT_0 \tag{5.539}$$

the partition function becomes

$$Z(N,C_1) = \zeta\int d\boldsymbol{p}\int d\boldsymbol{q}\int dp_v\int dv\int dp_s\int d\eta\; e^{(Nd+1)\,\eta}\;\frac{\delta(\eta-\eta_0)}{gk_bT_0}$$

$$= \frac{\zeta}{g k_b T_0} \int d\boldsymbol{p} \int d\boldsymbol{q} \int dp_v \int dv \int dp_s \; e^{(Nd+1)\,\eta_0}$$

Then, the substitution of the explicit expression for the variable η_0 in the previous integrand, with η_0 given by the first equation in (5.539), leads to a partition function of the form

$$Z(N, C_1) = \frac{\zeta}{g k_b T_0} \int d\boldsymbol{p} \int d\boldsymbol{q} \int dp_v \int dv \int dp_s \; \exp\left\{ \frac{(Nd+1)}{g k_b T_0} \left[C_1 - \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) - \frac{p_v^2}{2W} - P_0 v - \frac{p_s^2}{2Q} \right] \right\}$$

Collecting all the constant factors (i.e. the terms in the integrals that do not depend on the variables $\boldsymbol{p}$, $\boldsymbol{q}$ and v), the partition function can be written as

$$Z(N, C_1) = \xi \int dv \int d\boldsymbol{p} \int d\boldsymbol{q} \; \exp\left\{ -\frac{(Nd+1)}{g k_b T_0} \left[\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + P_0 v \right] \right\} \tag{5.540}$$

where the constant factor ξ is expressed as

$$\begin{aligned} \xi &= \frac{\zeta}{g k_b T_0} \exp\left[\frac{(Nd+1)}{g k_b T_0} C_1 \right] \int dp_v \exp\left[-\frac{(Nd+1)}{g k_b T_0} \frac{p_v^2}{2W} \right] \int dp_s \exp\left[-\frac{(Nd+1)}{g k_b T_0} \frac{p_s^2}{2Q} \right] \\ &= \frac{\zeta}{g k_b T_0} \exp\left[\frac{(Nd+1)}{g k_b T_0} C_1 \right] \sqrt{\frac{2\pi W g k_b T_0}{Nd+1}} \sqrt{\frac{2\pi Q\, g k_b T_0}{Nd+1}} = \frac{2\pi\zeta}{Nd+1} \sqrt{WQ} \; \exp\left[\frac{(Nd+1)}{g k_b T_0} C_1 \right] \end{aligned} \tag{5.541}$$

where the integrals with respect to the variables p_v and p_s in the first line of the previous expression are Gaussian integrals, and they can be easily resolved using equation (5.321) (see footnote 9), as done in the second line of the previous expression.
Taking $g = Nd + 1$, then the partition function (5.540) generated by the equations of motion (5.506) - (5.511) if the only conserved quantity is given by (5.534) can be written as

$$g = Nd + 1 \qquad \rightarrow \qquad Z(N, C_1) = \xi \int d\boldsymbol{p} \int d\boldsymbol{q} \int dv \; \exp\left\{ -\frac{1}{k_b T_0} [\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + P_0 v] \right\} \tag{5.542}$$

where the constant factors have been collected outside of the integrals in the constant ξ that is given by

$$\xi = \frac{\zeta \exp(\beta C_1)}{(Nd+1) k_b T_0} \int dp_v \; \exp\left(-\beta \frac{p_v^2}{2W} \right) \int dp_s \; \exp\left(-\beta \frac{p_s^2}{2Q} \right) = \frac{2\pi\zeta}{Nd+1} \sqrt{WQ} \; e^{\beta C_1} \quad \text{with} \quad \beta = \frac{1}{k_b T_0}$$

From equation (5.542) it is clear that, by taking $g = Nd + 1$, the correct isothermal-isobaric ensemble distribution function is generated by the equations of motion (5.506) - (5.511), considering as the only conserved quantity the Hamiltonian given by equation (5.534). The non Hamiltonian equations of motion (5.506) - (5.511) do have the desired compressibility which, in turn, leads to the correct phase space metric and distribution function.

The previous derivation demonstrates that the Ferrario equations are capable of generating an isothermal-isobaric distribution in the physical subsystem variables when the Hamiltonian $\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_v, v, p_s, \eta)$, given by (5.534), is the *only* conserved quantity. Note that the basic assumption used is that there is only a single conservation law, namely the conservation of the Hamiltonian (5.534).
However, as demonstrated and discussed in Section 5.4.1.3, for the case of simulations with periodic boundary conditions, also the quantity (5.520) related to the total linear momentum is conserved, while in the absence of periodic boundary conditions, both the quantities (5.520) and (5.526), where the last one is related to the total angular momentum, are conserved (see Table 5.9). The conservation of these additional quantities will affect the phase space distribution. Therefore, a separate treatment of these two cases has to be performed, to study the form of the partition function with these additional conservation laws. The case of periodic boundary conditions, with the conserved quantities (5.513) and (5.520), will be first analyzed in Section 5.4.1.5, since it is the most important case in practical simulations.

5.4.1.5 Statistical mechanical ensemble under periodic boundary conditions

In order to perform the analysis following Section 4.3.2.1, the driven variables have to be eliminated from the system. The center of mass position $\mathbf{R}$ is a driven variable (its dynamics does not effect other variables, and it does not contribute to a nontrivial conserved quantity) and must also be eliminated in the formal analysis. On the other hand, the magnitude of the center of mass momentum is coupled to the other variables through a conservation law and cannot be eliminated from the analysis. At the same time, equation (5.516) implies that the d components of the center of mass momentum $\mathbf{P}$ are linearly dependent. Thus, $d-1$ components of the center of mass momentum must be eliminated. Therefore of the d components only one component can be chosen independently, otherwise the variable

$$\Pi = \|\mathbf{P}\| = \left(\sum_{\alpha=1}^{d} P_\alpha^2 \right)^{1/2} \tag{5.543}$$

can be taken as the independent variable. Before proceeding with the analysis, the equation of motion for this new variable can be easily found by taking into account the definition of the conserved quantity, equation (5.516), and rewriting it in terms of the new variable Π defined in (5.543), by taking the Euclidean norm on both sides of equation (5.516), so that

$$\|\mathbf{P}\, v^{1/d}\, e^{\eta}\| = \|\mathbf{P}\|\, v^{1/d}\, e^{\eta} = \Pi\, v^{1/d}\, e^{\eta} = \|\mathbf{K}\| = K \tag{5.544}$$

where K is a constant, being $\mathbf{K}$ an arbitrary constant vector. Therefore, the norm of the vector $\mathbf{P}$ defined in (5.543) can be written as

$$\Pi = v^{-1/d}\, e^{-\eta}\, K \tag{5.545}$$

Taking the time derivative on both member of the previous equivalence, and using the equations of motion (5.508) and (5.510) leads to

$$\dot{\Pi} = -\dot{\eta}\, v^{-1/d}\, e^{-\eta}\, K - \frac{1}{d}\, v^{-1}\, \dot{v}\, v^{-1/d}\, e^{-\eta}\, K = -\dot{\eta}\, \Pi - \frac{\dot{v}}{vd}\, \Pi = -\frac{p_s}{Q}\, \Pi - \frac{1}{vd}\, \frac{p_v}{W}\, \Pi \tag{5.546}$$

that is the equation of motion which rules the evolution in time of the norm of the total linear momentum of the system. Proceeding further with the analysis, the two variables $\mathbf{R}$ and $\mathbf{P}$ can be eliminated by considering the positions and momenta relative to the center of mass of the system, $\boldsymbol{\rho}$ and $\boldsymbol{\pi}$, respectively. At this point, a transformation of variable can be performed, from the set $\{\boldsymbol{q}, \boldsymbol{p}, v, p_v, \eta, p_s\}$ to the set $\{\boldsymbol{\rho}, \boldsymbol{\pi}, v, p_v, \eta, p_s\}$. The equations of motion (5.506) - (5.512) transform as follows

$$\frac{d\boldsymbol{\rho}_i}{dt} = \frac{\boldsymbol{\pi}_i}{\mu_i} + \frac{\boldsymbol{\rho}_i}{vd}\, \frac{dv}{dt} = \frac{\boldsymbol{\pi}_i}{\mu_i} + \frac{\boldsymbol{\rho}_i}{vd}\, \frac{p_v}{W} \qquad i = 1, ..., N-1 \tag{5.547}$$

$$\frac{d\boldsymbol{\pi}_i}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \frac{\boldsymbol{\pi}_i}{s}\, \frac{ds}{dt} - \frac{\boldsymbol{\pi}_i}{vd}\, \frac{vd}{dt} = -\frac{\partial \phi}{\partial \boldsymbol{\rho}_i} - \boldsymbol{\pi}_i\, \frac{p_s}{Q} - \frac{\boldsymbol{\pi}_i}{vd}\, \frac{p_v}{W} \qquad i = 1, ..., N-1 \tag{5.548}$$

$$\frac{d\Pi}{dt} = -\left(\frac{p_s}{Q} + \frac{1}{vd}\, \frac{p_v}{W} \right) \Pi \tag{5.549}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{5.550}$$

$$\frac{dp_v}{dt} = \frac{1}{vd} \sum_{i=1}^{N-1} \left(\frac{\boldsymbol{\pi}_i^2}{\mu_i} - \frac{\partial \phi}{\partial \boldsymbol{\rho}_i} \cdot \boldsymbol{\rho}_i \right) + \frac{1}{vd}\, \frac{\Pi^2}{M} - \frac{\partial \phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \equiv F_v - \frac{p_v p_s}{Q} \tag{5.551}$$

$$\frac{d\eta}{dt} = \frac{p_s}{Q} \tag{5.552}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{\mu_i} + \frac{\Pi^2}{M} + \frac{p_v^2}{W} - g k_b T_0 \equiv F_s \tag{5.553}$$

where the two force variables used in equations (5.551) and (5.553) are given by

$$F_v \equiv \frac{1}{vd} \sum_{i=1}^{N-1} \left(\frac{\boldsymbol{\pi}_i^2}{\mu_i} - \frac{\partial \phi}{\partial \boldsymbol{\rho}_i} \cdot \boldsymbol{\rho}_i \right) + \frac{1}{vd}\, \frac{\Pi^2}{M} - \frac{\partial \phi}{\partial v} - P_0 \qquad F_s \equiv \sum_{i=1}^{N-1} \frac{\boldsymbol{\pi}_i^2}{\mu_i} + \frac{\Pi^2}{M} + \frac{p_v^2}{W} - g k_b T_0 \tag{5.554}$$

Therefore, when the condition (5.515) is introduced in the system, this causes the conservation laws given in equation (5.516), so that two quantities are now conserved, that are

$$\mathcal{H}(\boldsymbol{\pi},\Pi,\boldsymbol{\rho},p_v,v,p_s,\eta)=\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta=C_1 \tag{5.555}$$

$$\Pi\, v^{1/d}\, e^{\eta}=C_2 \tag{5.556}$$

where in (5.555) the Hamiltonian on the right hand side is given by

$$\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})=\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}+\frac{\Pi^2}{M}+\phi(\{\boldsymbol{\rho}_i\},v) \tag{5.557}$$

The compressibility, defined as in (4.123) with $\boldsymbol{\Xi}=(\boldsymbol{\pi},\Pi,\boldsymbol{\rho},p_v,v,p_s,\eta)$, can be computed using the equations of motion (5.547) - (5.553) as

$$\begin{aligned}\kappa(\boldsymbol{\Xi},t)=\nabla_{\boldsymbol{\Xi}}\cdot\dot{\boldsymbol{\Xi}}&=\sum_{i=1}^{N-1}[\nabla_{\boldsymbol{\pi}_i}\cdot\dot{\boldsymbol{\pi}}_i+\nabla_{\boldsymbol{\rho}_i}\cdot\dot{\boldsymbol{\rho}}_i]+\frac{\partial\dot{\Pi}}{\partial\Pi}+\frac{\partial\dot{p}_v}{\partial p_v}+\frac{\partial\dot{v}}{\partial v}+\frac{\partial\dot{p}_s}{\partial p_s}+\frac{\partial\dot{\eta}}{\partial\eta}\\&=\sum_{i=1}^{N-1}\left[-d\,\frac{p_s}{Q}-\frac{1}{v}\,\frac{p_v}{W}+\frac{1}{v}\,\frac{p_v}{W}\right]-\left(\frac{p_s}{Q}+\frac{1}{vd}\,\frac{p_v}{W}\right)-\frac{p_s}{Q}\\&=-(N-1)d\,\frac{p_s}{Q}-2\,\frac{p_s}{Q}-\frac{1}{vd}\,\frac{p_v}{W}=-[(N-1)d+2]\,\frac{p_s}{Q}-\frac{1}{vd}\,\frac{p_v}{W}=-[(N-1)d+2]\,\dot{\eta}-\frac{\dot{v}}{vd}\end{aligned} \tag{5.558}$$

where the equations of motion (5.550) and (5.552) have been used. The metric can then be computed as

$$\begin{aligned}\sqrt{g(\boldsymbol{\Xi},t)}&=\exp\left(-\int\kappa(\boldsymbol{\Xi},t)\,dt\right)=\exp\left\{\int[(N-1)d+2]\,\frac{d\eta}{dt}\,dt+\int\frac{1}{vd}\,\frac{dv}{dt}\,dt\right\}\\&=\exp\left\{[(N-1)d+2]\int\frac{d\eta}{dt}\,dt+\frac{1}{d}\int\frac{d\ln(v)}{dt}\,dt\right\}\\&=\exp\{[(N-1)d+2]\,\eta\}\,\exp\left\{\frac{1}{d}\,\ln(v)\right\}=v^{1/d}\,\exp\{[(N-1)d+2]\,\eta\}\end{aligned} \tag{5.559}$$

The partition function can be written as

$$\begin{aligned}Z(N,C_1,C_2)=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\int d\eta\;v^{1/d}\,e^{[(N-1)d+2]\,\eta}\;\times\\\times\,\delta\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\delta(\Pi\,v^{1/d}\,e^{\eta}-C_2)\end{aligned} \tag{5.560}$$

where $d\boldsymbol{\pi}=d\boldsymbol{\pi}_1\cdots d\boldsymbol{\pi}_{N-1}$ and $d\boldsymbol{\rho}=d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1}$ and the Hamiltonian $\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})$ is given by equation (5.557). Using the relation (5.538) in the second delta function of the integral, with respect to the variable η, where

$$f(\eta)=\Pi\,v^{1/d}\,e^{\eta}-C_2 \qquad\qquad f'(\eta)=\Pi\,v^{1/d}\,e^{\eta} \tag{5.561}$$

$$\eta_0=\ln\left(\frac{C_2}{\Pi\,v^{1/d}}\right) \qquad\qquad f'(\eta_0)=C_2 \tag{5.562}$$

the integral of the delta function over the variable η gives the result

$$\begin{aligned}Z(N,C_1,C_2)&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\int d\eta\;v^{1/d}\,e^{[(N-1)d+2]\,\eta}\;\times\\&\quad\times\,\delta\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\frac{\delta(\eta-\eta_0)}{\Pi\,v^{1/d}\,e^{\eta}}\\&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\;v^{1/d}\,e^{[(N-1)d+2]\,\eta_0}\;\times\\&\quad\times\,\delta\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta_0-C_1\right)\frac{1}{\Pi\,v^{1/d}\,e^{\eta_0}}\end{aligned} \tag{5.563}$$

Then, the substitution of the explicit expression for the variable η_0 in the previous integrand, with η_0 given by the first equation in (5.562), leads to a partition function of the form

$$Z(N,C_1,C_2) = \frac{\zeta}{C_2}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\; v^{1/d}\left(\frac{C_2}{\Pi\, v^{1/d}}\right)^{(N-1)d+2}\times \\ \times\,\delta\left[\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0 v+\frac{p_s^2}{2Q}+gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)-C_1\right] \tag{5.564}$$

$$Z(N,C_1,C_2) = \frac{\zeta}{C_2}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}\times \\ \times\,\delta\left[\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0 v+\frac{p_s^2}{2Q}+gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)-C_1\right] \tag{5.565}$$

The previous partition function can be further simplified using the remaining delta function to perform the integration over the variable p_s by means of the relation (5.538) with

$$f(p_s) = \mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0 v+\frac{p_s^2}{2Q}+gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)-C_1 \qquad \text{and} \qquad f'(p_s)=\frac{p_s}{Q} \tag{5.566}$$

The previous function $f(p_s)$ has only one zero $p_{s,0}$ with respect to the variable p_s, that is given by

$$p_{s,0} = \sqrt{2Q}\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0 v-gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)\right]^{1/2} \tag{5.567}$$

$$f'(p_{s,0}) = \frac{p_{s,0}}{Q} = \sqrt{\frac{2}{Q}}\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0 v-gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)\right]^{1/2} \tag{5.568}$$

Therefore, the partition function (5.565) becomes

$$Z(N,C_1,C_2) = \frac{\zeta}{C_2}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2} Q\,\frac{\delta(p_s-p_{s,0})}{p_s} \\ = \frac{\zeta Q}{C_2}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\;\frac{1}{v^{(N-1)+1/d}}\;\frac{1}{p_{s,0}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}$$

and substituting the explicit expression for the variable $p_{s,0}$ in the previous integrand, with $p_{s,0}$ given by equation (5.567), leads to a partition function of the form

$$Z(N,C_1,C_2) = \frac{\zeta}{C_2}\sqrt{\frac{Q}{2}}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}\times \\ \times\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0 v-gk_bT_0\ln\left(\frac{C_2}{\Pi\, v^{1/d}}\right)\right]^{-1/2} \tag{5.569}$$

Finally, the integration over the variable p_v can be performed via a simple substitution (see Appendix A.8.2), leading to the following form for the partition function

$$Z(N,P_0,T,C_1,C_2) = \frac{\pi\zeta}{C_2}\sqrt{QW}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dv\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2} \tag{5.570}$$

The integration over the $N-1$ particle positions and momenta can be easily performed as

$$\int d\boldsymbol{\pi}\int d\boldsymbol{\rho} = \int d\boldsymbol{\pi}_1\cdots d\boldsymbol{\pi}_{N-1}\int d\boldsymbol{\rho}_1\cdots d\boldsymbol{\rho}_{N-1} = v^{N-1} \tag{5.571}$$

so that the partition function can be finally written as

$$Z(N,P_0,T,C_1,C_2) = \frac{\pi\zeta}{C_2}\sqrt{QW}\int d\Pi\int dv\;\frac{v^{N-1}}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2} \tag{5.572}$$

$$Z(N,P_0,T,C_1,C_2)=\frac{\pi\zeta}{C_2}\sqrt{QW}\int d\Pi\int dv\;\frac{1}{v^{1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2} \tag{5.573}$$

The resultant partition function is independent of the potential, and it is not the partition function of the NPT ensemble.
However, in case the condition $\Pi\, v^{1/d}\, e^{\eta}=C_2=0$ is satisfied, then equation (5.560) can be rewritten as

$$\begin{aligned}Z(N,C_1,0)=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\int d\eta\; v^{1/d}\, e^{[(N-1)d+2]\,\eta}\;\times\\ \times\,\delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\delta(\Pi\, v^{1/d}\, e^{\eta}-0)\end{aligned} \tag{5.574}$$

The integration with respect to the variable Π can be then performed using the relation (5.538) with

$$f(\Pi)=\Pi\, v^{1/d}\, e^{\eta}\qquad\qquad f(\Pi_0)=0\;\leftrightarrow\;\Pi_0=0\qquad\qquad f'(\Pi)=f'(\Pi_0)=v^{1/d}\, e^{\eta} \tag{5.575}$$

leading to a partition function with the form

$$\begin{aligned}Z(N,C_1,0)&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\int dp_s\int d\eta\; v^{1/d}\, e^{[(N-1)d+2]\,\eta}\;\times\\ &\quad\times\,\delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\frac{\delta(\Pi-\Pi_0)}{v^{1/d}\, e^{\eta}}\\ Z(N,C_1,0)&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int dp_v\int dv\int dp_s\int d\eta\; v^{1/d}\, e^{[(N-1)d+2]\,\eta}\;\times\\ &\quad\times\,\delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\Pi_0,\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)v^{-1/d}\, e^{-\eta}\end{aligned} \tag{5.576}$$

where the Hamiltonian $\mathcal{H}_0(\boldsymbol{\pi},\Pi_0=0,\boldsymbol{\rho})\equiv\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})$ is given by equation (5.557) with $\Pi=0$, that is

$$\mathcal{H}_0(\boldsymbol{\pi},\Pi_0=0,\boldsymbol{\rho})\equiv\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})=\sum_{i=1}^{N-1}\frac{\boldsymbol{\pi}_i^2}{2\mu_i}+\phi(\{\boldsymbol{\rho}_i\},v) \tag{5.577}$$

Then, the partition function (5.576) can be rewritten as

$$\begin{aligned}Z(N,C_1,0)&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int dp_v\int dv\int dp_s\int d\eta\; v^{1/d}\, v^{-1/d}\, e^{-\eta}\, e^{[(N-1)d+2]\,\eta}\;\times\\ &\quad\times\,\delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\\ &=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int dp_v\int dv\int dp_s\int d\eta\; e^{[(N-1)d+1]\,\eta}\;\times\\ &\quad\times\,\delta\!\left(\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\right)\end{aligned}$$

Using the relation (5.538) on the remaining delta function of the partition function, with respect to the variable η, so that

$$f(\eta)=\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}+gk_bT_0\eta-C_1\qquad\qquad f'(\eta)=f'(\eta_0)=gk_bT_0 \tag{5.578}$$

$$\eta_0=\frac{1}{gk_bT_0}\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0v-\frac{p_s^2}{2Q}\right] \tag{5.579}$$

the partition function becomes

$$\begin{aligned}Z(N,C_1,0)&=\zeta\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int dp_v\int dv\int dp_s\int d\eta\; e^{[(N-1)d+1]\,\eta}\;\frac{\delta(\eta-\eta_0)}{gk_bT_0}\\ &=\frac{\zeta}{gk_bT_0}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int dp_v\int dv\int dp_s\; e^{[(N-1)d+1]\,\eta_0}\end{aligned}$$

and substituting the explicit expression for the variable η_0 in the previous integrand, with η_0 given by the equation (5.579), then the partition function can be written as

$$\begin{aligned} &Z(N, C_1, 0) \\ &= \frac{\zeta}{g k_b T_0} \int d\boldsymbol{\pi} \int d\boldsymbol{\rho} \int dp_v \int dv \int dp_s \ \exp\left\{ \frac{(N-1)d+1}{g k_b T_0} \left[C_1 - \mathcal{H}_0(\boldsymbol{\pi}, \boldsymbol{\rho}) - \frac{p_v^2}{2W} - P_0 v - \frac{p_s^2}{2Q} \right] \right\} \end{aligned}$$

Collecting all the constant terms (i.e. the terms that do not depend on the variables $\boldsymbol{\pi}$, $\boldsymbol{\rho}$ and v) in the constant ξ, the the partition function can be written as

$$Z(N, C_1, 0) = \xi \int d\boldsymbol{\pi} \int d\boldsymbol{\rho} \int dv \ \exp\left\{ -\frac{(N-1)d+1}{g k_b T_0} \left[\mathcal{H}_0(\boldsymbol{\pi}, \boldsymbol{\rho}) + P_0 v \right] \right\} \tag{5.580}$$

with the constant factor given by

$$\begin{aligned} \xi &= \frac{\zeta}{g k_b T_0} \exp\left[\frac{(N-1)d+1}{g k_b T_0} \ C_1 \right] \int dp_v \exp\left[-\frac{(N-1)d+1}{g k_b T_0} \ \frac{p_v^2}{2W} \right] \int dp_s \exp\left[-\frac{(N-1)d+1}{g k_b T_0} \ \frac{p_s^2}{2Q} \right] \\ &= \frac{\zeta}{g k_b T_0} \exp\left[\frac{(N-1)d+1}{g k_b T_0} \ C_1 \right] \sqrt{\frac{2\pi W g k_b T_0}{(N-1)d+1}} \sqrt{\frac{2\pi Q \, g k_b T_0}{(N-1)d+1}} \\ &= \frac{2\pi\zeta}{(N-1)d+1} \ \sqrt{WQ} \ \exp\left[\frac{(N-1)d+1}{g k_b T_0} C_1 \right] \end{aligned}$$

where the integrals with respect to the variables p_v and p_s in the first line of the previous expression are Gaussian integrals, and they have been easily resolved in the the second line using equation (5.321) (see footnote 9).

Taking $g = (N-1)d + 1$, the partition function (5.580) and the constant factor ξ contained in its expression become

$$g = (N-1)d + 1 \quad \rightarrow \quad Z(N, C_1, 0) = \xi \int d\boldsymbol{\pi} \int d\boldsymbol{\rho} \int dv \ \exp\left\{ -\frac{1}{k_b T_0} \left[\mathcal{H}_0(\boldsymbol{\pi}, \boldsymbol{\rho}) + P_0 v \right] \right\} \tag{5.581}$$

$$\xi = \frac{\zeta \, \exp(\beta C_1)}{[(N-1)d+1] k_b T_0} \int dp_v \exp\left(-\beta \, \frac{p_v^2}{2W} \right) \int dp_s \exp\left(-\beta \, \frac{p_s^2}{2Q} \right) = \frac{2\pi\zeta}{(N-1)d+1} \ \sqrt{WQ} \ e^{\beta C_1}$$

where $\beta = 1/(k_b T_0)$.

If a three dimensional space is considered, so that $d = 3$, then the value of g becomes equal to $g = 3N - 3 + 1 = 3N - 2$. The same result for the value of the factor g has been obtained in Appendix A.8, Section A.8.1. However, in this case the partition function obtained, equation (A.204), and as a consequence the distribution function (A.205), do not describe correctly an isothermal-isobaric ensemble. This is due to the additional term $1/v$ derives from the Jacobian computed in (A.195), and used for the change of variable from the virtual to the real set of coordinates. Indeed, the calculation of the Jacobian has been based on the integration over $3N - 3$ momenta variables, but also on the integration over $3N$ position variables. Therefore, the error was due to the fact that the center of mass positions $\mathbf{R}$ were not identified as driven variables, and they have not been eliminated from the analysis. This is clear also noting that, in the calculation of the Jacobian in (A.195), if only $3N - 3$ positions would be considered, then the factor $1/v$ would cancel out with that from the momenta change of variable, so that it is not carried out inside the partition function integrals.

Some comments about the differences between the results obtained for the form of the partition function starting from the Ferrario equations of motion (5.506) - (5.511) and the Hoover algorithm (see equations (5.2) in Ref. [38]) can be done. In the case of non zero total force acting on the system (i.e. when only the total Hamiltonian is conserved), the significant difference is that in the equation of motion for the variable v, that depends on the variable v itself in the Hoover algorithm, while it is independent on the variable v in the case of equation (5.508). This leads to a different compressibility, namely, for the Ferrario equations of motion the form of the compressibility obtained is that in (5.536), while for the Hoover algorithm it assumes the form in equation (5.6) in Ref. [38], which differs from (5.536) for an

additive term that derives from the non zero contribution of the derivative $\partial\dot{v}/\partial v$ to the compressibility. As a consequence, the metric (5.7) in Ref. [38] differs from (5.537) by a multiplicative term $1/v$, whose presence in the volume integration of the partition function derived from the Hoover equations of motion leads to a form of the distribution function that does not correspond to an isothermal-isobaric ensemble. The same considerations are also valid in the case of a zero total force acting on the system. Also in this case, the compressibility derived from the Hoover algorithm has an additional term with respect to the compressibility in equation (5.558), that derives from the non zero contribution of the derivative $\partial\dot{v}/\partial v$. As a consequence, the metric in the Hoover algorithm (see equation (5.10) in Ref. [38]) has a additional factor of $1/v$ with respect to the metric computed in (5.559), whose presence in the integral leads to a partition function form different from that of an isothermal-isobaric ensemble, even in the case $C_2 = 0$, condition that generates the correct isothermal-isobaric partition function in the absence of a net force acting on the system for the case of equations of motion (5.547) - (5.553).

5.5 Summary: ensembles and equations of motion

In the following, a summary of the ensembles whose theoretical derivations have been made in this chapter and that have been implemented in the CRYSTAL code is reported.
In Table 5.10, the list of conserved quantities for each ensemble is given.

Ensemble	**Conserved quantity**
Standard molecular dynamics NVE ensemble, Section 5.1	$H(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q})$
Simple scaling thermostat Section 5.3.2	$H(\boldsymbol{p}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}$
Berendsen thermostat Section 5.3.3	none (but in the code : kinetic energy $H(\boldsymbol{p}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}$ is taken as conserved)
Nosé-Hoover thermostat NVT ensemble, Section 5.3.4	$H(\boldsymbol{p},\boldsymbol{q},p_\eta,\eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{p_\eta^2}{2Q} + gk_bT_0\eta$
Ferrario thermostat and barostat NPT ensemble, Section 5.4.1	$H(\boldsymbol{p},\boldsymbol{q},p_v,v,p_s,\eta) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\{\boldsymbol{q}_i\},v) + \frac{p_v^2}{2W} + P_0v + \frac{p_s^2}{2Q} + gk_bT_0\eta$

Table 5.10: Conserved quantity for each ensemble and thermostat method implemented in the CRYSTAL code.

Table 5.11 reports, for each ensemble and thermostat type, the logical variables involved, the name of the subroutines which implement the correspondent time propagator algorithm, as well as the keywords and the integer i_e to be inserted in the input file to perform molecular dynamics simulations for each particular ensemble or using different thermostats. For more details about the keywords that can be used in the molecular dynamics input block, see Appendix E, Section E.1.

1. Code internal distinction among ensembles

Ensemble	Logical variables	thstat (i_e)	Keyword	EOM integrator subroutine
NVE	nve = T [default: T]	0 [default]	none	vverlet_NVE
RESCALING	nve = T [default: T]	1	THERMOST	vverlet_NVE
BERENDSEN	nve = T [default: T]	2	THERMOST	
NVT (NOSE)	nve = F [default: T] nvt = T [default: F]	0 [default]	NOSE	vverlet_nose_I0 (if nose_integ = 0) vverlet_nose_I1 (if nose_integ = 1) [default]
NPT	nve = F [default: T] npt = T [default: F]	0 [default]	NPT	vverlet_npt_I0 (if npt_integ = 0) [default]

Table 5.11: Logical variables set to true (T) or false (F), value of the variable thstat, keyword used to activate the ensemble and Equations of Motion (EOM) integrator subroutine for the different implemented molecular dynamics ensembles. The equations of motion integrators subroutines vverlet_nose_I0, vverlet_nose_I1 and vverlet_npt_I0 refer to the algorithms reported in Tables A.1, 5.5 and 5.8, respectively.

Finally, since a lot of attention has been devoted in this chapter in finding the correct value of the number of degrees of freedom g to be used in a molecular dynamics simulation to correctly sample a given ensemble under a certain set of equations of motion, in Table 5.12 the number of degrees of freedom g for each ensemble and kind of thermostat implemented in the CRYSTAL code is reported. In the case

of geometrical constraint, as for example the removal of the system rotations (activated by the keyword NOROT), a further decreasing of the number of degrees of freedom has to be applied, depending on the dimensionality of the system. This last issue has been also discussed in more details in Chapter 3, Section 3.2.

2. Number of degrees of freedom for each ensemble

Ensemble	Logical variables	thstat (i_e)	Keyword	g
NVE	nve = T [default: T]	0 [default]	none	$3N-3$
RESCALING	nve = T [default: T]	1	THERMOST	$3N-3$
BERENDSEN	nve = T [default: T]	2	THERMOST	$3N-3$
NVT (NOSE)	nve = F [default: T] nvt = T [default: F]	0 [default]	NOSE	$3N-3$
NPT	nve = F [default: T] npt = T [default: F]	0 [default]	NPT	$3N-2$

Geometry Constraint	Logical variables	g
Keyword NOROT	norot = T [default: F]	0D systems : $g \leftarrow g-3$ 1D systems : $g \leftarrow g-1$

Table 5.12: Number of degrees of freedom g for a given ensemble and kind of thermostat, logical variables that activates the number of degrees of freedom to the correspondent value g, value of the integer variable thstat and keywords used to activate the molecular dynamics calculation in a given ensemble.

Chapter 6

Post processing of molecular dynamics trajectory

6.1 Radial Pair Correlation Function

6.1.1 Theory

The relative dispositions of atoms and molecules in liquids and solids are often predicted accurately using classical statistical mechanics. The electronic system has to be described with quantum mechanical equations, but these quantal fluctuations can be averaged inside the physical system. The remaining problem is to sample the statistical configurations of the nuclei in the effective interaction induced by the electrons, whose degrees of freedom have been already integrated out. From a mathematical point of view, this procedure can be achieved considering the partition function in the canonical ensemble:

$$Z = \sum_{\mu} e^{-\beta E_{\mu}} \tag{6.1}$$

where $\beta = 1/(k_b T_0)$, and μ identifies the state of the system with a particular nuclear arrangement, defined by the symbol R, and with a particular electronic state parametrized by R, defined by the symbol $i(R)$, which indicates the i-th state of the electrons when the nuclei are fixed in the configuration R. The partition function (6.1) can be factorized as follows:

$$Z = \sum_{R} \left[\sum_{i(R)} e^{-\beta E_{R,i(R)}} \right] = \sum_{R} e^{-\beta \tilde{E}_R} \tag{6.2}$$

The quantity $\tilde{E}_R$ is obtained by performing the Boltzmann weighted sum in the square brackets, and it represents the effective energy governing the statistics for the configurations of the nuclei. In general, $\tilde{E}_R$ is a free energy that depends on the temperature. Often, however, the electronic states of a system are dominated by the lowest energy level. In that case, the averaging out of the electronic degrees of freedom yields an $\tilde{E}_R$ that must be the ground state Born-Oppenheimer energy surface for all the nuclei. After that the electronic contribution to the partition function has been averaged out, the spatial configurations of the nuclei can be usually well studied with a classical mechanical approach. The reason is that the nuclei are heavier than the electrons, and the relatively high mass implies that the quantum uncertainties in the positions of the nuclei are relatively small and, as a result, the quantum dispersion becomes unimportant when considering the spatial fluctuations of the nuclei in atomic systems.

6.1.1.1 Averages in phase space

When adopting a classical model, the microscopic state of the system is characterized by a point in the phase space. A point in phase space for a system with n particles is given by a list of the coordinates and conjugate momenta of all the classical degrees of freedom in the system:

$$(\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_n, \boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n) \equiv (\boldsymbol{q}, \boldsymbol{p})$$

where $\boldsymbol{q}_i$ and $\boldsymbol{p}_i$ are, respectively, the position and the momentum of the i-th particle, and $\boldsymbol{q}$ and $\boldsymbol{p}$ are abbreviations for points in configuration space and momentum space, respectively. The probability of a state in a classical system is $f(\boldsymbol{p},\boldsymbol{q})\,d\boldsymbol{q}\,d\boldsymbol{p}$, where :

$$f(\boldsymbol{p},\boldsymbol{q}) = \text{probability distribution for observing the system at phase space point } (\boldsymbol{q},\boldsymbol{p})$$

The analytical form for the probability distribution function $f(\boldsymbol{p},\boldsymbol{q})$ can be found by analogy with the statistical mechanical probability to find a system in a particular state μ, whose partition function is given by equation (6.1). From a classical point of view, the probability distribution function $f(\boldsymbol{p},\boldsymbol{q})$ that gives the probability to find the system at point $(\boldsymbol{q},\boldsymbol{p})$ in phase space is given by

$$f(\boldsymbol{p},\boldsymbol{q}) = \frac{e^{-\beta\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}}{\int d\boldsymbol{p}\int d\boldsymbol{q}\; e^{-\beta\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}} \tag{6.3}$$

where $d\boldsymbol{p} = d\boldsymbol{p}_1\cdots d\boldsymbol{p}_n$, $d\boldsymbol{q} = d\boldsymbol{q}_1\cdots d\boldsymbol{q}_n$ have been used as integration variables, and the Hamiltonian $\mathcal{H}(\boldsymbol{p},\boldsymbol{q})$ can be written as the sum of the kinetic energy $E_k(\boldsymbol{p})$ of the classical degrees of freedom and the potential energy $\phi(\boldsymbol{q})$, namely,

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q}) = E_k(\boldsymbol{p}) + \phi(\boldsymbol{q}) = \sum_{i=1}^{n}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{6.4}$$

Here the kinetic energy is a function of the momenta only, and the potential energy is a function of position coordinates only, since it is supposed that the system is a conservative Newtonian system. The potential energy $\phi(\boldsymbol{q})$ is obtained by averaging over all the quantum degrees of freedom that are not treated explicitly in the classical model. Thus, the potential energy has to be determined from a quantum electronic structure calculation, using for example Hartree-Fock theory or Density Functional Theory. Since the Hamiltonian can be written as a sum of two contributions, and the probability distribution function (6.3) includes the Hamiltonian in the exponential part, then the phase space distribution can be exactly factorized as

$$f(\boldsymbol{p},\boldsymbol{q}) = Q(\boldsymbol{p})P(\boldsymbol{q}) \tag{6.5}$$

where

$$Q(\boldsymbol{p}) = \frac{e^{-\beta E_k(\boldsymbol{p})}}{\int d\boldsymbol{p}\, e^{-\beta E_k(\boldsymbol{p})}} \qquad \text{and} \qquad P(\boldsymbol{q}) = \frac{e^{-\beta\phi(\boldsymbol{q})}}{\int d\boldsymbol{q}\, e^{-\beta\phi(\boldsymbol{q})}} \tag{6.6}$$

are, respectively, the probability distribution for observing the system at momentum space point $\boldsymbol{p}$, and the probability distribution for observing the system at configuration space point $\boldsymbol{q}$. Further factorization of the momentum distribution is possible, since the kinetic energy can be written as a sum of single particle kinetic energies, so that

$$Q(\boldsymbol{p}) = \prod_{i=1}^{n} h(\boldsymbol{p}_i) \tag{6.7}$$

where

$$h(\boldsymbol{p}_i) = \frac{e^{-\beta\boldsymbol{p}_i^2/2m}}{\int d\boldsymbol{p}_i\; e^{-\beta\boldsymbol{p}_i^2/2m}} \tag{6.8}$$

in which also the integral on the momenta $d\boldsymbol{p}$ has been factorized as a product of single particle integrals in the momentum variable: this has been possible because the momenta of the nuclei are each one independent from each other. The single particle momentum distribution $h(\boldsymbol{p}_i)$ in (6.8) is usually called the normalized Maxwell-Boltzmann distribution, and it is the correct momentum distribution function for a particle with mass m in thermal equilibrium with the system at a fixed temperature $T_0 = 1/(k_b\beta) = 2/(n_d\,k_b)E_k(\boldsymbol{p})$, where n_d is the number of degrees of freedom of the system and k_b is the Boltzmann constant (see Appendix A, Section A.1).

6.1.1.2 Reduced Configurational Distribution Functions

The configurational distribution $P(\boldsymbol{q})$ does not factor into single particle functions because the potential energy $\phi(\boldsymbol{q})$ couples together all the coordinates. However, the distribution functions for a small number

of particles can be defined, by integrating over all coordinates except those corresponding to the particles of interest. For example:

$$P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \int d\boldsymbol{q}_3 \int d\boldsymbol{q}_4 \ \cdots \int d\boldsymbol{q}_n \ P(\boldsymbol{q}) \tag{6.9}$$

is the *specific reduced joint probability distribution* for finding the particle 1 at position $\boldsymbol{q}_1$ and the particle 2 at position $\boldsymbol{q}_2$. The function is called specific because it requires particle 1 (and no other particle) to be at $\boldsymbol{q}_1$ and, similarly, it must be particle 2 at $\boldsymbol{q}_2$. Such requirements are not physically relevant for systems composed of n indistinguishable particles. A more meaningful quantity is the *generic reduced joint probability distribution* function. The generic counterpart of the previously defined specific distribution function $P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ is indicated with $\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$. It is the joint distribution function for finding a particle (any one) at position $\boldsymbol{q}_1$ and any other particle (in the remaining $n-1$ particle system) at position $\boldsymbol{q}_2$. Since there are n possible ways of picking the first particle (the one at $\boldsymbol{q}_1$) and there are $n-1$ ways of picking the second, the generic reduced joint probability can be written as a function of the specific reduced joint probability as follows

$$\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) = n(n-1)\, P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) \tag{6.10}$$

Generalizing the concept, the generic reduced joint distribution function that describes the probability that, in a system with n particles, a particle will be found at $\boldsymbol{q}_1$, another at $\boldsymbol{q}_2$, ..., and another at $\boldsymbol{q}_m$ is

$$\begin{aligned} \rho_{(m/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_m) &= \frac{n!}{(n-m)!} \int d\boldsymbol{q}_{m+1} \int d\boldsymbol{q}_{m+2} \ \cdots \int d\boldsymbol{q}_n \ P(\boldsymbol{q}) \\ &= \frac{n!}{(n-m)!} \int d\boldsymbol{q}_{n-m} \ \frac{e^{-\beta\phi(\boldsymbol{q})}}{\int d\boldsymbol{q}\, e^{-\beta\phi(\boldsymbol{q})}} \equiv \ \frac{n!}{(n-m)!} \int d\boldsymbol{q}_{n-m} \ \frac{e^{-\beta\phi(\boldsymbol{q})}}{Z_n} \end{aligned} \tag{6.11}$$

where Z_n is the partition function and $d\boldsymbol{q}_{n-m}$ is an abbreviation for $d\boldsymbol{q}_{m+1} d\boldsymbol{q}_{m+2} \cdots d\boldsymbol{q}_n$. For a homogeneous system, the generic reduced single-particle joint probability for the i-th particle is given by

$$\rho_{(1/n)}(\boldsymbol{q}_i) = \rho = \frac{n}{v} \qquad\qquad \forall\ i = 1, ..., n \tag{6.12}$$

where v is the volume of the system. In an ideal gas, the different particles in the system are uncorrelated. As a result, for an ideal gas, the specific reduced joint two-particles distribution $P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ factors as $P_{(1/n)}(\boldsymbol{q}_1) P_{(1/n)}(\boldsymbol{q}_2)$. As a consequence, the generic reduced joint two-particles distribution for an ideal gas can be written as

$$\begin{aligned} \rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) &= n(n-1) P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) \\ &= n(n-1) P_{(1/n)}(\boldsymbol{q}_1) P_{(1/n)}(\boldsymbol{q}_2) = \ n(n-1)\ \frac{\rho^2}{n^2} = \frac{n-1}{n}\ \rho^2 \approx \rho^2 \end{aligned} \tag{6.13}$$

where in the last approximation the difference between n and $(n-1)$ is neglected. In view of the ideal homogeneous gas result for $\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$, given by equation (6.13), it seems appropriate to introduce the quantity

$$g(\boldsymbol{q}_1, \boldsymbol{q}_2) = \frac{\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)}{\rho^2} \tag{6.14}$$

which is the ratio between the generic reduced joint two-particles distribution function for the real system and that for an ideal homogeneous gas. If the system is isotropic, the previous function (6.14) depends only upon coordinates distances $q \equiv q_{12} = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|$, that is,

$$g(\boldsymbol{q}_1, \boldsymbol{q}_2) = g(q) \qquad \text{for isotropic homogeneous systems} \tag{6.15}$$

The function $g(q)$ is called the *radial distribution function*, but it is also referred to as the *pair correlation function* or *pair distribution function*. In the following, the function $g(q)$ will be addressed as the total radial pair correlation function.

Starting from the knowledge of the analytical form for $\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$, that is the generic reduced joint probability for finding a particle at position $\boldsymbol{q}_2$ and any other particle of the system at position $\boldsymbol{q}_1$, the conditional probability can be found thanks to the following theorem:

Theorem If x and y are random variables with a joint probability distribution $P(x, y)$, then the conditional probability distribution for y given a specific value of x is $P(x, y)/p(x)$, where $p(x)$ is the probability distribution for x.

Therefore, the *conditional probability density* that a particle will be found at $\boldsymbol{q}_2$ given the specific position of another particle at $\boldsymbol{q}_1$ is given by

$$\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)/\rho = \rho\, g(q) \tag{6.16}$$

Alternatively, $\rho\, g(q)$ can be seen as the *average density* of particles at $\boldsymbol{q}_2$ given that another particle is at the origin. Now, $\rho\, g(q)\, dq$ is the probability of observing a second particle at a certain distance dq (where dq is a radial distance) given that there is another particle at position $\boldsymbol{q}_1$ (which can be taken to be, for example, the origin of the coordinate system). Note that this probability is not normalized to unity, but instead

$$\int_0^\infty \rho\, g(q)\; 4\pi q^2 dq = n - 1 \tag{6.17}$$

This equation shows that $\rho\, g(q)\, 4\pi q^2$ is really the number of particles between q and $q + dq$ about a central particle. In this view, the function $g(q)$ can also be thought of as the factor that multiplies the bulk density ρ to give a local density $\rho(q) = \rho\, g(q)$ about some fixed particle. Clearly, $g(q) \to 0$ as the radial distance $q \to 0$, because the particles become effectively "hard" as $q \to 0$ (and, obviously, because of the repulsion among particles increases when the particles approach each other). On the other hand, since the influence of the particle at position $\boldsymbol{q}_1$ (at the origin) decreases as q becomes large, then $g(q) \to 1$ as $q \to \infty$. Finally, consider the first minimum q_m of the total radial pair correlation function $g(q)$ (the minimum can be local or global, depending on the atomic system). The integral from 0 to q_m of the integrand function in (6.17) is the so-called *coordination number* n_c of the system, that is

$$\int_0^{q_m} \rho\, g(q)\; 4\pi q^2 dq = n_c \tag{6.18}$$

6.1.1.3 Analytical expression of the radial pair correlation function

Generalizing previous concepts, the radial pair correlation function for a group of m particles can be defined in terms of the corresponding generic reduced single-particle joint probability distributions as

$$g_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) = \rho_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) \Big/ \prod_{i=1}^{m} \rho_{(1/n)}(\boldsymbol{q}_i) \tag{6.19}$$

and, using the definition (6.14) where the radial pair correlation distribution is written as a function of the generic reduced single-particle joint probability for a homogeneous gas (6.12), the previous expression becomes

$$g_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) = \frac{\rho_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m)}{\rho^m} \tag{6.20}$$

Note that $g_{(m/n)}$ is called correlation function because, if the particles were independent from each other (as in an ideal gas), $\rho_{(m/n)}$ for the real system would equal simply the correspondent quantity for an ideal homogeneous gas, ρ^m. Rewriting the previous formula as

$$\rho_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) = \rho^m\, g_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) \tag{6.21}$$

the role of $g_{(m/n)}$ in correcting for the non-independence can be clearly seen, i.e. $g_{(m/n)}$ corrects for the correlation among particles acting on a real atomic system.
Using the expression (6.11), along with the relation $\rho^m = (n/v)^m$ (see equation (6.12) for this last identity), equation (6.20) becomes

$$g_{(m/n)}(\boldsymbol{q}_1, ..., \boldsymbol{q}_m) = \frac{v^m\, n!}{n^m\, (n-m)!} \int d\boldsymbol{q}_{n-m}\; \frac{e^{-\beta\phi(\boldsymbol{q})}}{Z_n} \tag{6.22}$$

Working in the canonical ensemble, it is possible to obtain useful expressions for the particles densities in terms of the delta functions of positions. Starting from the mean value of the delta function of position, the following relation holds

$$\begin{aligned} < \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_1) > &= \int d\boldsymbol{q}\ \frac{e^{-\beta\phi(\boldsymbol{q})}}{Z_n}\ \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_1) \\ &= \int d\boldsymbol{q}_2\ \cdots \int d\boldsymbol{q}_n\ \frac{e^{-\beta\phi(\boldsymbol{q}_\alpha, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n)}}{Z_n} \end{aligned} \tag{6.23}$$

The statistical average in (6.23) is a function of the coordinate $\boldsymbol{q}_\alpha$, but is independent on the particle label (here taken to be 1). The sum over all the particle labels can therefore be written as n times the contribution from a single particle (6.23), that is

$$\begin{aligned} < \sum_{i=1}^{n} \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_i) > \ &= \int d\boldsymbol{q}\ \frac{e^{-\beta\phi(\boldsymbol{q})}}{Z_n}\ \sum_{i=1}^{n} \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_i) = \sum_{i=1}^{n} \int d\hat{\boldsymbol{q}}_i\ \frac{e^{-\beta\phi(\boldsymbol{q}_{i\to\alpha})}}{Z_n} \\ &= n \int d\boldsymbol{q}_2\ \cdots \int d\boldsymbol{q}_n\ \frac{e^{-\beta\phi(\boldsymbol{q}_\alpha, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n)}}{Z_n} \quad \overset{(6.11)}{=} \quad \rho_{(1/n)}(\boldsymbol{q}_\alpha) \end{aligned} \tag{6.24}$$

where $d\hat{\boldsymbol{q}}_i$ is the integration over all the coordinate variables $\boldsymbol{q}_1 \cdots \boldsymbol{q}_n$ except the variable $\boldsymbol{q}_i$, while the notation $\phi(\boldsymbol{q}_{i\to\alpha})$ means that the potential depends on all the coordinates variables set where the coordinate $\boldsymbol{q}_i$ has been substituted with $\boldsymbol{q}_\alpha$. If the function form of the potential $\phi(\boldsymbol{q})$ is the same for all the coordinates degrees of freedom (as it is for the atomic systems where electrostatic interactions are considered), then all the integrals in the sum in the last expression of the first line in (6.24) are equal to each other, so that the sum can be written as n times one form of the integral, equivalent to the others. Therefore, from the definition (6.11), it follows that the generic reduced single-particle joint distribution function is given by

$$\rho_{(1/n)}(\boldsymbol{q}_\alpha) = < \sum_{i=1}^{n} \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_i) > \tag{6.25}$$

Similarly, the statistical average of the product of two delta functions of the position is

$$\begin{aligned} < \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_1)\delta(\boldsymbol{q}_\beta - \boldsymbol{q}_2) > &= \int d\boldsymbol{q}\ \frac{e^{-\beta\phi(\boldsymbol{q})}}{Z_n}\ \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_1)\,\delta(\boldsymbol{q}_\beta - \boldsymbol{q}_2) \\ &= \int d\boldsymbol{q}_3\ \cdots \int d\boldsymbol{q}_n\ \frac{e^{-\beta\phi(\boldsymbol{q}_\alpha, \boldsymbol{q}_\beta, \boldsymbol{q}_3, \ldots, \boldsymbol{q}_n)}}{Z_n} \end{aligned} \tag{6.26}$$

and hence

$$\rho_{(2/n)}(\boldsymbol{q}_\alpha, \boldsymbol{q}_\beta) = < \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i\neq j}}^{n} \delta(\boldsymbol{q}_\alpha - \boldsymbol{q}_i)\,\delta(\boldsymbol{q}_\beta - \boldsymbol{q}_j) > \tag{6.27}$$

Finally, for a system that is homogeneous, $g(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta, \boldsymbol{q}_\beta)$ is independent on the position $\boldsymbol{q}_\beta$. The expression for $g(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta, \boldsymbol{q}_\beta) = g(\boldsymbol{q}_\alpha)$ can be obtained by computing the following quantity

$$\begin{aligned} &< \frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i\neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > = < \int d\boldsymbol{q}_\beta\ \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i\neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta - \boldsymbol{q}_i)\,\delta(\boldsymbol{q}_\beta - \boldsymbol{q}_j) \right] > \\ &= \frac{1}{n} \int d\boldsymbol{q}_\beta\ < \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i\neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta - \boldsymbol{q}_i)\,\delta(\boldsymbol{q}_\beta - \boldsymbol{q}_j) > = \frac{1}{n} \int d\boldsymbol{q}_\beta\ \rho_{(2/n)}(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta, \boldsymbol{q}_\beta) \\ &= \frac{1}{\rho} \left[\frac{1}{v} \int d\boldsymbol{q}_\beta\ \rho_{(2/n)}(\boldsymbol{q}_\alpha + \boldsymbol{q}_\beta, \boldsymbol{q}_\beta) \right] = \frac{\rho_{(2/n)}(\boldsymbol{q}_\alpha)}{\rho} = \rho\ g_{(2/n)}(\boldsymbol{q}_\alpha) \equiv \rho\ g(\boldsymbol{q}_\alpha) \end{aligned} \tag{6.28}$$

where in the first equality an elementary property of delta functions has been exploited, and in the fourth equality the expression $n = \rho\, v$ for a homogeneous system has been used. Considering the first and last

expression of the previous derivation, the following equality can be written

$$g(\boldsymbol{q}_\alpha) = \frac{1}{n\,\rho} \; < \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > = \; \frac{v}{n^2} \; < \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > \tag{6.29}$$

If the system is isotropic as well as homogeneous, the pair distribution function $g_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ is a function of the distance $q \equiv q_{12} = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|$ only, for this reason it is usually called pair correlation function as previously explained, and it is indicated simply with $g(q)$ as defined in (6.15). When the separation q_{12} is much larger than the range of interparticle potential, then $g_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ approaches its ideal gas limit. The expression for the total pair correlation distribution function in this limit can be computed as follows. The definition for $g_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)$ is

$$g_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \frac{\rho_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)}{\rho^2} = \frac{n(n-1) P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2)}{\rho^2} \tag{6.30}$$

As the separation q_{12} is much larger than the range of interparticle potential, the interaction between particles approaches zero, i.e. $\phi(\boldsymbol{q}) \to 0$, so that

$$P_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \int d\boldsymbol{q}_3 \; \cdots \int d\boldsymbol{q}_n \; \frac{e^{-\beta\phi(\boldsymbol{q})}}{\int d\boldsymbol{q}\, e^{-\beta\phi(\boldsymbol{q})}} \to \frac{v^{n-2}}{v^n} = \frac{1}{v^2} \quad \text{for } \phi(\boldsymbol{q}) \to 0 \tag{6.31}$$

and finally, using (6.30)

$$g_{(2/n)}(\boldsymbol{q}_1, \boldsymbol{q}_2) = n(n-1)\frac{1}{(\rho\, v)^2} = n(n-1)\left(\frac{v^2}{n^2}\right)\frac{1}{v^2} = 1 - \frac{1}{n} \quad \text{for } q \equiv q_{12} = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\| \to \infty \tag{6.32}$$

Thus, for a system with a large number of particles, the pair correlation function for $q \to \infty$ approaches the asymptotic value of one.

6.1.1.4 Radial pair correlation function resolved per species

The radial pair correlation functions resolved per species are computed according to the formula

$$x_\mu x_\nu \rho\, g_{\mu\nu}(\boldsymbol{q}_\alpha) = \frac{1}{n} \; < \sum_{i=1}^{n_\mu} \sum_{j=1}^{n'_\nu} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > \tag{6.33}$$

where $x_{\mu,\nu} = n_{\mu,\nu}/n$ is the fraction of species μ or ν, the coordinates $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ are respectively the positions of the i-th particle with species μ and of the j-th particle with species ν, and the prime n'_ν indicates that the terms for which $i = j$ are omitted when $\mu = \nu$. Substituting the terms $x_{\mu,\nu}$ with their expressions as functions of n and $n_{\mu,\nu}$ in the left hand side of equation (6.33), the pair correlation function for the species (μ, ν) can be rewritten more explicitly as

$$g_{\mu\nu}(\boldsymbol{q}_\alpha) = \; \frac{n}{\rho\, n_\mu\, n_\nu} \; < \sum_{i=1}^{n_\mu} \sum_{j=1}^{n'_\nu} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > = \; \frac{v}{n_\mu\, n_\nu} \; < \sum_{i=1}^{n_\mu} \sum_{j=1}^{n'_\nu} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > \tag{6.34}$$

The radial pair correlation function resolved per species is defined so that the normalized sum over all couples of possible atomic species of the system is equal to the total radial pair correlation function $g(\boldsymbol{q}_\alpha)$, i.e. so that

$$g(\boldsymbol{q}_\alpha) = \frac{1}{n^2} \sum_{\mu\nu} n_\mu n_\nu\, g_{\mu\nu}(\boldsymbol{q}_\alpha) \tag{6.35}$$

Indeed, substituting the first expression of equation (6.34) in the previous relation, the total $g(\boldsymbol{q}_\alpha)$ of equation (6.29) can be recovered as follows

$$\begin{aligned} g(\boldsymbol{q}_\alpha) &= \frac{1}{n^2} \sum_{\mu\nu} n_\mu n_\nu \left[\frac{n}{\rho\, n_\mu\, n_\nu} \; < \sum_{i=1}^{n_\mu} \sum_{j=1}^{n'_\nu} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > \right] \\ &= \frac{1}{n\,\rho} \sum_{\mu\nu} \; < \sum_{i=1}^{n_\mu} \sum_{j=1}^{n'_\nu} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > = \; \frac{1}{n\,\rho} \; < \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} \delta(\boldsymbol{q}_\alpha + \boldsymbol{q}_j - \boldsymbol{q}_i) > \end{aligned} \qquad \text{q.e.d.}$$

6.1.2 Implementation

In this section the implementation of the radial pair correlation function in the CRYSTAL code is described. The calculation of the pair correlation function is a post-processing analysis, which starts from the results of a molecular dynamics simulation.

6.1.2.1 Requirements

For the calculation of the pair correlation function, an input file with the atomic configurations per each step of a molecular dynamics simulation is needed. Furthermore, if the system is a three dimensional crystalline periodic bulk, the direct lattice vectors has to be known, in order to compute the lattice parameters (fort.34 file is required). When the Module Moldyn is activated through the keyword MOLDYN the input file, and a molecular dynamics simulation is performed, the code generates by default the files POSITIONS.DAT and VELOCITIES.DAT, which contains the positions and velocities of each atom during the molecular dynamics simulation. At the same time, the code updates at each step the file fort.34 (by overwriting it at each step), where the direct lattice vectors as well as the MD final geometry are written. Since the pair correlation function deals with the spatial configuration of the particles, a file POSITIONS.DAT must exist in the same directory where the post-processing analysis is carried out. For these reasons, the code check for the existence of a file named POSITIONS.DAT; if the file exists in the directory of execution, everything is fine and the calculation can proceed, otherwise, if the file does not exist, the code call the subroutine errvrs with ierr = 0 (this subroutine is implemented in library both4.f), it stops and prints in the output file the following string:

ERROR **** PCF_MD **** FILE POSITIONS.DAT NOT FOUND!

6.1.2.2 The subroutine pcf_md: description and methods

The subroutine implemented for the calculation of the radial pair correlation function is the subroutine pcf_md, contained in the module moldyn_post in the library moldyn_post.f90. The execution section of the subroutine pcf_md can be divided into the following different parts:

1. Reading of the file POSITIONS.DAT, saving the information needed
2. Modification of the vector with the atomic species for chemically irreducible atoms
3. Creation of a radial regular mesh (binned space)
4. Initialization of the vectors and matrices needed for the calculation
5. Filling of the histogram with the number of atomic couples
6. Computation of the radial pair correlation functions (total and resolved per species)
7. Creation of the output files with the results

Then, in the final part of the subroutine, the files used are closed and the dynamical vectors and matrices are deallocated. In the following part of this chapter, the different execution subsections of the code listed above are described, taking particular attention on the way the data are saved and manipulated within vectors and matrices to obtain the results, and on the practical way to implement the calculation for the radial pair correlation function, whose underneath theoretical basis have been explained in Section 6.1.1.

Reading of the file POSITIONS.DAT and saving information The subroutine pcf_md calls the subroutine readPOSfile of the moldyn_post module in order to accomplish the task of reading the file POSITIONS.DAT with the atomic positions per each molecular dynamics step. The file POSITIONS.DAT, generated by default during a molecular dynamics calculation (and described in Appendix B.3), is read by the subroutine readPOSfile as follows:

- The first row contains the number of atoms in the system, saved in the integer variable naf.

- The following naf rows are divided into three columns: the first one for the row index (from 1 to naf), the second one for the atomic number and the third one for the atomic mass of each atom, respectively. Only the atomic number is saved in the dynamical integer vector atos_tmp(naf).
- The following nstp·naf rows contains the Cartesian atomic positions (x, y, z) [in Angstrom] of each atom in the initial atomic configuration and in the successive nstp−1 steps of the molecular dynamics simulation. These positions are saved in matrices with dimension nstp·naf in the following way

$$\boldsymbol{q}_x = \begin{pmatrix} x_1^{s_1} & x_1^{s_2} & ... & x_1^{s_m} \\ x_2^{s_1} & x_2^{s_2} & ... & x_2^{s_m} \\ ... & ... & ... & \\ x_{\mathsf{naf}}^{s_1} & x_{\mathsf{naf}}^{s_2} & ... & x_{\mathsf{naf}}^{s_m} \end{pmatrix} \qquad \boldsymbol{q}_y = \begin{pmatrix} y_1^{s_1} & y_1^{s_2} & ... & y_1^{s_m} \\ y_2^{s_1} & y_2^{s_2} & ... & y_2^{s_m} \\ ... & ... & ... & \\ y_{\mathsf{naf}}^{s_1} & y_{\mathsf{naf}}^{s_2} & ... & y_{\mathsf{naf}}^{s_m} \end{pmatrix} \qquad \boldsymbol{q}_z = \begin{pmatrix} z_1^{s_1} & z_1^{s_2} & ... & z_1^{s_m} \\ z_2^{s_1} & z_2^{s_2} & ... & z_2^{s_m} \\ ... & ... & ... & \\ z_{\mathsf{naf}}^{s_1} & z_{\mathsf{naf}}^{s_2} & ... & z_{\mathsf{naf}}^{s_m} \end{pmatrix}$$

  where the subscripts 1,...,naf are referred to the indexes of the atoms while the superscripts s_1,...,s_m are the indexes of the molecular dynamics step (with s_m = nstp). These matrices are therefore filled column-by-column while reading the file, so that all (x, y, z) positions per each atom of the system are given for the initial configuration (first column), then for the first step (second column), and so on.

The atomic number of each atomic species in the system has been saved in vector atos_tmp(naf). In order to generalize the calculation of the pair correlation function for couples of atoms of the same atomic species but with a different structural (chemical) environment, it is useful to work with atomic symbols instead of atomic numbers. For this reason, after the file with positions has been read, the atomic numbers saved in vector atos_tmp(naf) are converted in the correspondent atomic symbols, and these are saved in the characters-type vector atos(naf), using the function symbat(at_number):

$$\mathsf{atos(i)} = \mathsf{symbat(mod(atos\_tmp(i),100))},\ \mathrm{i} = 1,...,\mathsf{naf}$$

so that

$$\mathsf{atos\_tmp} = \begin{pmatrix} X_1 & X_2 & ... & X_{\mathsf{naf}} \end{pmatrix} \quad \rightarrow \quad \mathsf{atos} = \begin{pmatrix} A_1 & A_2 & ... & A_{\mathsf{naf}} \end{pmatrix}$$

where X_i is the atomic number of the i-th atom, while A_i is the symbol of the correspondent atomic species. Obviously, since the atos vector (as well as the atos_tmp vector) contains a list of *all* the atoms in the system, it can happen that $A_i = A_j$ for some elements i and j.

Modification of the vector with the atomic symbols (for chemically irreducible atoms) If two or more atoms have the same atomic symbol but two different structural (chemical) environments, and the user wants to study these atoms with distinct radial pair correlation functions, than the functionality of the keyword IRRCHIM can be exploited (see manual in Appendix E). If this keyword is activated, the label of the atoms in the vector atos is modified consequently. For example, consider the case of water H_2O. If the two atoms of hydrogen H have to be considered differently from each other and the two chemical environments around each of them have to be analyzed, one can insert in the CRYSTAL input file the list of atom indexes accordingly to the order of insertion of the atoms in the geometry section of the input file (this order also corresponds to the sequence order of atoms as printed in the CRYSTAL output file). The insertion of the atom indexes in the correct order in the input file is not mandatory, the code works well even if the correct input sequence is not respected. In this way, the atoms whose indexes are called in the input file through the keyword IRRCHIM are considered different from each other even if they have the same atomic number. The only modification in the code is the vector atos, which from now on labels those atoms in a different way, i.e. with a name that is a concatenation of the atomic symbol with the correspondent index inserted in input by the user. In the case of water, for example, if the atomic coordinates are inserted in the input file (and printed in the output file) with the order H H O, than the three atoms have indexes 1 2 3, respectively. If the keyword IRRCHIM is used in the input file, and the indexes 1 and 2 are selected, the atos vector is modified as follows:

$$\mathrm{IRRCHIM} \quad \rightarrow \quad \mathsf{atos} = (\mathrm{H1},\ \mathrm{H2},\ \mathrm{O})$$

so that four radial pair correlation functions will be computed: the total radial pair correlation function $g(q)$, together with the pair correlation functions for the three atomic couples H1-H2, H1-O and H2-O.

Creation of a radial regular mesh Equations (6.29) and (6.34) are the formal definitions of the total and partial pair correlation functions, respectively. In practice, the delta function is implemented by filling an histogram, counting all the pairs of atoms of two given species falling within each bin of the histogram, i.e. counting the number of atoms of a given species at a certain distance from atoms of another given species. In order to define and fill such an histogram, a radial regular mesh is created, which divides the space in nbin bins, each of one is a spherical shell in three dimensions. The number of bin nbin is defined by the user, otherwise it is equal to its default value (see see manual in Appendix E). Consider a system with the side of the sampling cubic box equal to ℓ_s (the radial pair correlation function will be sampled and calculated in this cubic box). The radial bin width $\mathsf{dq} \equiv \delta q$ is defined as:

$$\mathsf{dq} \equiv \delta q = \frac{\ell_s}{2 \cdot \mathsf{nbin}} \qquad \text{bin width} \tag{6.36}$$

In order to identify each bin with a unique distance parameter, the following definition is given:

$$\mathsf{rad}(i) \equiv q(i) = ((i-1) + 0.5) \cdot \delta q \quad \text{with } i = 1,..., \mathsf{nbin} \tag{6.37}$$

In this way, the vector rad contains the elements:

$$\mathsf{rad} = \begin{pmatrix} 0.5\,\delta q & 1.5\,\delta q & 2.5\,\delta q & ... & (\mathsf{nbin} - 0.5)\,\delta q \end{pmatrix}$$

so that the i-th element of the vector uniquely identifies the i-th bin of the system and it *associates to each i-th bin a radial distance equal to the radius of the sphere that intersect the i-th bin in its central point*. This construction is very useful in each loop on the number of bins, in which the radial bin distance has to be involved, because in this way all the quantities in the loop are identified with a unique index, that is the bin index, referred to a unique radial distance associated to a given bin. When the radial pair distribution function is computed, it has to be calculated inside each i-th bin, so that there are nbin different values for each particular radial pair correlation function, each referred to a particular bin. The radial distance associated to the i-th values of the radial pair correlation function correspondent to the i-th bin will be defined to be equal to $\mathsf{rad}(i) \equiv q(i)$. It can be said that $\mathsf{rad}(i)$ is the radial distance that represents the i-th bin, and the radial pair distribution function computed for that bin is associated to that radial distance. Table 6.1 resumes the division of the three-dimensional space in spherical shell bins with radius q. The radial extension of each bin identifies the range of values for the radius q that spanned the spherical shell associated with that bin. The radial pair correlation function will be computed for each one of these volume elements (bins).

bin index	bin radial extension	representative radial distance
1	$0 < q \leq q(1) + h$	$q(1) \equiv \mathsf{rad}(1)$
2	$q(2) - h < q \leq q(2) + h$	$q(2) \equiv \mathsf{rad}(2)$
..	...	...
nbin	$q(\mathsf{nbin}) - h < q \leq q(\mathsf{nbin}) + h$	$q(\mathsf{nbin}) \equiv \mathsf{rad}(\mathsf{nbin})$

Table 6.1: Bin division defined by the radial regular mesh. The radial extension and the representative radial distance for each bin are reported. The quantity h is equal to half of the bin width: $h = \delta q/2$.

—— definition of ℓ_s

Initialization of vectors and matrices needed for the calculation The radial pair correlation function (RPCF) can be computed for the entire system, without distinction of the atomic species, or by taking into account only two particular atomic species, μ and ν. In the first case it is called *total* RPCF $g(q)$, while in the second case it is called *partial* RPCF $g_{\mu\nu}(q)$. The definition of the radial pair distribution function lies on the assumption that the system is *homogeneous* as well as *isotropic*, so that the pair correlation function depends only on the separation $q_{12} = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|$. The partial radial pair distribution function is also characterized by a two-particles exchange symmetry, i.e. $g_{\mu\nu}(q) = g_{\nu\mu}(q)$. The correspondent matrix $\boldsymbol{g} = \{g_{\mu\nu}(q)\}$ is then a square neqatos×neqatos symmetric matrix, completely determined by

$$\frac{\mathsf{neqatos} \cdot (\mathsf{neqatos} + 1)}{2}$$

elements, where neqatos is the number of non-equivalent atoms (physically or chemically speaking). First of all, it is necessary to determine the number of non-equivalent atoms, the atomic species of these atoms and the total number of atoms per each atomic species. This is accomplished with an analysis of the elements contained in the atos(naf) vector. The results of the analysis are saved in the following vectors:

$$\mathsf{atyp\_neq} = \begin{pmatrix} A_{n_1} & A_{n_2} & ... & A_{n_s} \end{pmatrix} \qquad s = \mathsf{neqatos}$$

which contains only the name of the atomic species of the neqatos non-equivalent atoms, and

$$\mathsf{ntyp\_neq} = \begin{pmatrix} N_{n_1} & N_{n_2} & ... & N_{n_s} \end{pmatrix} \qquad s = \mathsf{neqatos}$$

whose i-th element is the number of atoms of the i-th non-equivalent atomic species saved in the correspondent i-th element of the atyp_neq vector. Starting from the atyp_neq vector, a matrix can be constructed, that contains all the possible non-equivalent atomic couples for which the partial $g_{\mu\nu}(q)$ has to be computed. The couple (μ, ν) is equivalent to the couple (ν, μ), because the correspondent pair correlation functions $g_{\mu\nu}(q)$ and $g_{\nu\mu}(q)$ are equal. The number of non-equivalent couples is therefore

$$\mathsf{ncpl} = \frac{\mathsf{neqatos} \cdot (\mathsf{neqatos} + 1)}{2}$$

However, if there is only one atom of a certain atomic species μ in the system, it makes no sense to compute the partial pair correlation function $g_{\mu\mu}(q)$ for that particular atomic species μ: no other atoms except it exists with the same atomic species, so that computing $g_{\mu\mu}(q)$ would mean to compute a partial pair correlation function of an atom with itself, a case which has to be excluded by default. Thus, although (μ, μ) is in principle an acceptable couple to be taken into account, it has to be excluded when there is only one atom in the system with that atomic species μ. As a consequence, the variable ncpl is reduced by the number of atomic species N_{alone} that have only one atom in the system:

$$\mathsf{ncpl} \leftarrow \mathsf{ncpl} - N_{\text{alone}}$$

Now, a 2×ncpl matrix can be constructed, with the atomic species of all the possible *non-equivalent* couples, in this way

$$\mathsf{cpl} = \begin{pmatrix} A_1^{c_1} & A_1^{c_2} & ... & A_1^{c_m} \\ A_2^{c_1} & A_2^{c_2} & ... & A_2^{c_m} \end{pmatrix} \qquad m = \mathsf{ncpl}$$

where $A_1^{c_i}$ and $A_2^{c_i}$ are respectively the first and the second atomic species of the i-th couple. Thus, each column of the matrix contains the symbols of the two atomic species of a possible non-equivalent atomic couple. At the same time, the ntyp_neq vector can be used to construct a 2×ncpl matrix whose element (i, j) is equal to the number of atoms in the system with that particular cpl(i, j) atomic species, i.e.

$$\mathsf{cpl\_num} = \begin{pmatrix} N_1^{c_1} & N_1^{c_2} & ... & N_1^{c_m} \\ N_2^{c_1} & N_2^{c_2} & ... & N_2^{c_m} \end{pmatrix} \qquad m = \mathsf{ncpl}$$

where $N_1^{c_i}$ and $N_2^{c_i}$ are respectively the number of atoms in the system with atomic species $A_1^{c_i}$ and $A_2^{c_i}$. The elements of the matrices cpl and cpl_num must have a one-to-one correspondence. The matrix cpl_num is useful for the normalization procedure of the partial radial pair correlation functions.

Note: by default, the code computes the total radial pair correlation function $g(q)$ and all the partial radial pair correlation functions $g_{\mu\nu}(q)$ for each couple in the matrix cpl.

Example
Consider the crystalline system $CsBa_2I_5$. The vector atos, that contains a list of *all* the atoms in the system, is given by

$$\mathsf{atos} = \begin{pmatrix} \text{Cs} & \text{Ba} & \text{Ba} & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \end{pmatrix}$$

Physically speaking, the number of non-equivalent atoms in the system is neqatos = $s = 3$. The atomic species of these non-equivalent atoms are Cs, Ba and I, and the number of atoms per each non-equivalent atomic species is respectively 1, 2 and 5, so that the following vectors are defined:

$$\mathsf{atyp\_neq} = \begin{pmatrix} \text{Cs} & \text{Ba} & \text{I} \end{pmatrix}$$

which contains only the symbols of the atomic species of the neqatos non-equivalent atoms, and

$$\mathsf{ntyp\_neq} = \begin{pmatrix} 1 & 2 & 5 \end{pmatrix}$$

that contains the total number of each non-equivalent atom in the system. The number of non-equivalent couples are:

$$\mathsf{ncpl} = \frac{3 \cdot 4}{2} = 6$$

However, in this case it makes no sense to compute the partial radial pair correlation function $g_{\mathrm{Cs,Cs}}(q)$ for the atomic species Cs, because there is only one Cs atom in the entire system. Indeed, computing $g_{\mathrm{Cs,Cs}}(q)$ would mean to compute the partial pair correlation function of an atom with itself, which is not allowed. Therefore, the number of possible couples ncpl has to be decreased by $N_{\mathrm{alone}} = 1$, so that

$$\mathsf{ncpl} = 5$$

Finally, the 2×5 matrix cpl, containing all the possible non-equivalent atomic couples for which the partial radial pair correlation function is computed, is defined as

$$\mathsf{cpl} = \begin{pmatrix} \mathrm{Cs} & \mathrm{Cs} & \mathrm{Ba} & \mathrm{Ba} & \mathrm{I} \\ \mathrm{Ba} & \mathrm{I} & \mathrm{Ba} & \mathrm{I} & \mathrm{I} \end{pmatrix}$$

and the correspondent 2×5 matrix cpl_num, with the number of atoms in the system for each correspondent atomic species in the matrix cpl, is given by

$$\mathsf{cpl\_num} = \begin{pmatrix} 1 & 1 & 2 & 2 & 5 \\ 2 & 5 & 2 & 5 & 5 \end{pmatrix}$$

Filling of the histogram with the number of atomic couples As previously explained, equations (6.29) and (6.34) are replaced, in practice, by a step function which is non-zero in a small range of separations, and the pair correlation function is most simply thought as the number of atoms of a given species at a distance q from the atoms of a second given species in the system, compared with the same quantity computed for an ideal gas. The most computationally expensive part of the calculation of the radial pair distribution function is the loop over the bins and the calculation of the number of particles n_p inside each bin, given a reference central atomic species. The pseudocode in Table 6.2 explains the way in which this calculation is performed.
An important point to underline is that, for periodic systems (3D, 2D or 1D) the distances are computed using periodic boundary conditions along the directions of periodicity, using the minimum image convention (i.e. each individual particle in the simulation undergoes an interaction with the *closest* image of the remaining particles in the system).
Finally, the (ncpl + 1)×nbin matrix gr(0:ncpl,nbin), which contains the computed number of particles per each bin, is defined as follows

$$\mathsf{gr} = \begin{pmatrix} n_c(b_1) & n_c(b_2) & ... & n_c(b_\lambda) \\ n_c^{c_1}(b_1) & n_c^{c_1}(b_2) & ... & n_c^{c_1}(b_\lambda) \\ ... & ... & ... & ... \\ n_c^{c_m}(b_1) & n_c^{c_m}(b_2) & ... & n_c^{c_m}(b_\lambda) \end{pmatrix} \qquad m = \mathsf{ncpl},\ \lambda = \mathsf{nbin}$$

Each i-th element of the first row of the matrix gr is the total number of particles in the i-th bin b_i, so that the first row contains the values of the total radial pair correlation function $g(q)$. From the second to the (ncpl + 1)-th row, the number of particles per each bin for a given couple c_k (which corresponds to the k-th column of the matrix cpl) is listed, and these elements are used row-by-row to compute the partial radial pair correlation function $g_{\mu\nu}(q)$ for the k-th couple of atoms with species μ and ν. In this way, each column identifies a different bin, and each row of the matrix corresponds to a partial radial pair correlation function for a given non-equivalent couple of atomic species (except the first row which sums up all the partial radial pair correlation functions in order to compute the total radial pair correlation function).

```
fixloop: do i = 1,naf    ! fix an atom i (reference central atom)
   spi = atomic species of the atom i
   varloop: do j = 1,naf    ! loop on the other naf atoms given a fixed central i atom
      if( i == j ) cycle
      spj = atomic species of the atom j
      compute the distance d_ij between atoms i and j, using periodic boundary conditions (PBC)
      binloop: do ib = 1,nbin    ! loop over the bins to count the number of atoms per each bin
         distance: if (d_ij > q(ib) - δq/2 .and. d_ij ≤ q(ib) + δq/2) then
            increase total n_c(q) of bin ib: nc(0,ib) = nc(0,ib) + 1
            species: do k = 1,ncpl    ! loop on the cpl matrix columns (on each non-equivalent couple)
               if (spi == cpl(1,k) .and. spj == cpl(2,k)) then
                  increase partial n_c^ij(q) of bin ib: nc(k,ib) = nc(k,ib) + 1
               endif
            enddo species
         endif distance
      enddo binloop
   enddo varloop
enddo fixloop
```

Table 6.2: Pseudocode for the computation of the non-normalized total and partial radial pair correlations functions (RPCF histogram filling).

Note: in the pseudocode previously reported, the case $i = j$ (i.e. the case in which the atom selected in the loop fixloop is the same as the atom selected in the inner loop varloop, namely, the self-counted case) has been discarded by the code with an if statement inside the inner loop.

The procedure described by the pseudocode is repeated for a certain number τ_{run} of different atomic configurations. Each atomic configuration (sometimes also called snapshot) is defined as the collection of the nuclear coordinates for a given step of a molecular dynamics simulation. In the code, the information about the nuclear positions is stored in the matrices q_x, q_y and q_z (see paragraph 6.1.2.2). The way in which the atomic configurations for the calculation of the pair correlation functions are chosen among all those obtained from the molecular dynamics simulation is fixed by the user with the parameters t_{in}, t_{fin} and δ_t (see the input keywords manual in Appendix E). The indexes of the initial and final atomic configurations (corresponding to an initial and a final molecular dynamics steps) are defined by the parameters t_{in} and t_{fin}, respectively. The δ_t parameter is instead the frequency of the selection of the atomic configurations among the t_{in} and t_{fin} steps. Therefore, the atomic configurations selected from the molecular dynamics trajectory are those whose step indexes are $t_1 = t_{in} + \delta_t$, $t_2 = t_1 + \delta_t$,..., $t_n = t_{n-1} + \delta_t$, for each $t_n \leq t_{fin}$. For example, if $t_{in} = 2$, $t_{fin} = 15$ and $\delta_t = 3$, the atomic configurations selected are those corresponding to the steps 2, 5, 8, 11, 14 of the molecular dynamics simulation. The pair correlation functions obtained for each one of these atomic configurations are then appropriately averaged out to obtain the final results, as explained in paragraph 6.1.2.2.

Computation of the normalized radial pair correlation functions When all the configurations have been processed, the (total and partial) pair correlation functions must be finally computed and normalized. Suppose that τ_{run} atomic configurations of the molecular dynamics trajectory have been analyzed to compute the pair correlation functions, and that a particular bin b_i of the histogram, corresponding to the interval $(q(b_i) - \delta q/2,\ q(b_i) + \delta q/2)$, contains $n_{\text{c}}(b_i)$ pairs. Then the average number of atoms in the system whose distance from a given atom lies in this interval is

$$n(b_i) = \frac{n_c(b_i)}{n \cdot \tau_{\text{run}}} \tag{6.38}$$

where $n_{\text{c}}(b_i)$ is the number of atoms detected in the i-th bin (computed through the algorithm described in Table 6.2), n is the total number of atoms in the system and τ_{run} is the number of atomic configurations

(snapshots) used to compute $n_c(b_i)$. The average number of atoms in the same interval of space in an ideal gas with uniform density ρ is

$$n^{\text{gas}}(b_i) = \rho \cdot v_s(b_i) \tag{6.39}$$

where $v_s(b_i)$ is the volume of the i-th bin. In the present case each bin divides the three-dimensional space in a spherical shell and the number of particles inside this shell are counted. The volume of the spherical shell can be computed (considering the definition of the radial regular grid previously descrb˙ied) as the difference between two spheres, one with radius $q(b_i) - \delta q/2$ and the other with radius $q(b_i) + \delta q/2$, where $q(b_i)$ is the distance of the center of the i-th bin spherical shell from the origin (i.e. from the atom of reference), and δq is the bin width. Then,

$$v_s(b_i) = \frac{4}{3}\pi\left[\left(q(b_i) + \frac{\delta q}{2}\right)^3 - \left(q(b_i) - \frac{\delta q}{2}\right)^3\right] = 4\pi[q(b_i)]^2\,\delta q + \frac{\pi}{3}(\delta q)^3 \;\approx\; 4\pi[q(b_i)]^2\,\delta q \tag{6.40}$$

where the term $\pi(\delta q)^3/3$ can be neglected iff the bin width δq is sufficiently small. Substituting the last expression in equation (6.39), the result is

$$n^{\text{gas}}(b_i) = 4\pi\rho\,[q(b_i)]^2\,\delta q \tag{6.41}$$

By definition, the total pair correlation function of the i-th bin is the ratio between (6.38) and (6.41), i.e.

$$g(b_i) = \frac{n(b_i)}{n^{\text{gas}}(b_i)} = \frac{n_c(b_i)}{4\pi\rho\,[q(b_i)]^2\,\delta q \cdot n \cdot \tau_{\text{run}}} \tag{6.42}$$

In the same way as (6.38), the average number of atoms $n_{\mu\nu}(b_i)$, that is the average number of atoms in the system with species ν whose distance from a given atom of species μ (taken as the reference central atom) lies within the spherical shell defined by the i-th bin, can be computed as follows

$$n_{\mu\nu}(b_i) = \frac{n_c^{\mu\nu}(b_i)}{\left(\frac{n_\mu n_\nu}{n}\right)\tau_{\text{run}}} = n\,\frac{n_c^{\mu\nu}(b_i)}{n_\mu n_\nu \tau_{\text{run}}} = \frac{1}{n}\left(\frac{n_c^{\mu\nu}(b_i)}{x_\mu x_\nu \tau_{\text{run}}}\right) \tag{6.43}$$

where $n_c^{\mu\nu}(b_i)$ is the number of the couple of atomic species (μ, ν) detected in the i-th bin (computed through the algorithm described in Table 6.2), $x_\mu = n_\mu/n$ ($x_\nu = n_\nu/n$) is the fraction of atoms with species μ (or ν) with respect to the total number of atoms n, and τ_{run} is the number of configurations along the molecular dynamics trajectory (snapshots) for which the partial pair correlation functions have been computed.
Analogously to the total pair correlation function (6.42), the partial radial pair distribution function resolved per species, formally defined by equation (6.34), can be given by the following expression:

$$g_{\mu\nu}(b_i) = \frac{n_{\mu\nu}(b_i)}{n^{\text{gas}}(b_i)} \tag{6.44}$$

where $n_{\mu\nu}(b_i)$ is defined by equation (6.43) as the average number of atoms in the system with species ν whose distance from a given atom of species μ (taken as the reference central atom) lies within the spherical shell defined by the i-th bin. The volume of this spherical shell is equal to $v_s(b_i)$, and it is given by the equation (6.40). Then the partial pair correlation function resolved per species is the ratio between expression (6.43) and the average number of atoms in the same i-th bin for an ideal gas with uniform density ρ, given by the expression (6.41), that is

$$g_{\mu\nu}(b_i) = \frac{1}{n} \cdot \frac{n_c^{\mu\nu}(b_i)}{4\pi\rho\,[q(b_i)]^2\,\delta q \cdot x_\mu x_\nu \cdot \tau_{\text{run}}} \tag{6.45}$$

Substituting x_μ (and x_ν) with the correspondent expressions n_μ/n (and n_ν/n), and using the relation for the density $\rho = n/v$, the following expression for the partial pair correlation function in the i-th bin can be obtained

$$g_{\mu\nu}(b_i) = \frac{v \cdot n_c^{\mu\nu}(b_i)}{4\pi\,[q(b_i)]^2\,\delta q \cdot n_\mu n_\nu \cdot \tau_{\text{run}}} \tag{6.46}$$

where v is the volume of the unitary simulation cell, and n_μ (n_ν) is the number of atomic species μ (and ν) in the system. From the definition in equation (6.46), the total pair correlation function for the i-th bin can be recovered using the relation (6.35), so that

$$g(b_i) = \frac{1}{n^2} \sum_{\mu\nu} n_\mu n_\nu \, g_{\mu\nu}(b_i) \tag{6.47}$$

Indeed, substituting the expression (6.46) in equation (6.47), the formula (6.42) for the total pair correlation function is recovered, as demonstrated in the following.

$$\begin{aligned} g(b_i) &= \frac{1}{n^2} \sum_{\mu\nu} n_\mu n_\nu \; \frac{v \cdot n_c^{\mu\nu}(b_i)}{4\pi \, [q(b_i)]^2 \, \delta q \cdot n_\mu n_\nu \cdot \tau_{\text{run}}} \\ &= \frac{v}{n^2} \cdot \frac{\sum_{\mu\nu} n_c^{\mu\nu}(b_i)}{4\pi \, [q(b_i)]^2 \, \delta q \cdot \tau_{\text{run}}} = \frac{1}{n\rho} \cdot \frac{n_c(b_i)}{4\pi \, [q(b_i)]^2 \, \delta q \cdot \tau_{\text{run}}} \\ &= \frac{n_c(b_i)}{4\pi\rho \, [q(b_i)]^2 \, \delta q \cdot n \cdot \tau_{\text{run}}} \qquad\qquad \text{q.e.d.} \end{aligned} \tag{6.48}$$

where the expression for the density $\rho = n/v$ has been used in the second row, together with the identity

$$\sum_{\mu\nu} n_c^{\mu\nu}(b_i) = n_c(b_i) \tag{6.49}$$

In this way, the equation (6.42) for the total pair correlation function $g(b_i)$ of the i-th bin has been recovered, starting from the relation (6.47) and the definition (6.46) for the partial pair correlation function resolved per species $g_{\mu\nu}(b_i)$ of the correspondent i-th bin. The computed (ncpl + 1)×nbin matrix for the radial pair correlation function is then:

$$\mathsf{gr} = \begin{pmatrix} gr(b_1) & gr(b_2) & ... & gr(b_\lambda) \\ gr_{c_1}(b_1) & gr_{c_1}(b_2) & ... & gr_{c_1}(b_\lambda) \\ ... & ... & ... & ... \\ gr_{c_m}(b_1) & gr_{c_m}(b_2) & ... & gr_{c_m}(b_\lambda) \end{pmatrix} \qquad m = \mathsf{ncpl},\ \lambda = \mathsf{nbin}$$

The first row of the matrix contains the total radial pair correlation function $g(b_i)$ per each bin b_i, $i = 1, ..., \mathsf{nbin}$. From the second to the (ncpl + 1)-th row, the partial radial pair correlation functions $g_{c_k}(b_i) \equiv g_{\mu\nu}(b_i)$ are reported per each bin b_i for a given couple $c_k = (\mu, \nu)$ of species μ and ν, that corresponds to the two elements in the k-th column of the matrix cpl.

Consider now the equation (6.17), that represents the number of particles inside a sphere of radius q about a central particle, as q varies from 0 to ∞. In the code, the integral over the radial distance is substituted by a summation over the number of bin, so that equation (6.17) becomes

$$\sum_{i=1}^{\mathsf{nbin}} \rho \, g(b_i) \; 4\pi [q(b_i)]^2 \, \delta q = N - 1 \tag{6.50}$$

Substituting the definition (6.42) of the total radial pair correlation function in the previous equation, it can be written

$$\sum_{i=1}^{\mathsf{nbin}} \frac{n_c(b_i)}{n \cdot \tau_{\text{run}}} = \sum_{i=1}^{\mathsf{nbin}} n(b_i) = N - 1 \tag{6.51}$$

where the equation (6.38) has been used. Thus, per each bin b_i, the total number of particles $(N(b_i) - 1)$ contained within a sphere of radius equal to the external radius correspondent to that bin b_i can be computed, summing up all the values of the total pair correlation function $g(b_i)$ multiplied by the factor $4\pi[q(b_i)]^2 \rho \, \delta q$, up to that i-th bin, namely,

$$\mathsf{intgr}(b_i) = \sum_{n=1}^{i} \rho \, g(b_n) \; 4\pi [q(b_n)]^2 \, \delta q = N(b_i) - 1 \qquad\qquad b_i = 1, ..., \mathsf{nbin} \tag{6.52}$$

As regards the integrated partial pair correlation function $g_{\mu\nu}(q)$, here the symmetry is broken, i.e. $\mathsf{intgr}_{\mu\nu} \neq \mathsf{intgr}_{\nu\mu}$, at least for a multiplication factor. Indeed, the two integrals which defines the integrated partial radial pair correlation functions are:

$$\mathsf{intgr}_{\mu\nu} = \int_0^\infty \rho_\mu \, g_{\mu\nu}(q) \; 4\pi q^2 dq \qquad \text{and} \qquad \mathsf{intgr}_{\nu\mu} = \int_0^\infty \rho_\nu \, g_{\nu\mu}(q) \; 4\pi q^2 dq \tag{6.53}$$

Exploiting the symmetry of the functions in the previous integrals, i.e. $g_{\mu\nu}(q) = g_{\nu\mu}(q)$, it can be written:

$$\mathsf{intgr}_{\mu\nu} = \rho_\mu \left[\int_0^\infty g_{\mu\nu}(q) \; 4\pi q^2 dq \right] = \rho_\mu \left[\int_0^\infty g_{\nu\mu}(q) \; 4\pi q^2 dq \right] = \rho_\mu \frac{\mathsf{intgr}_{\nu\mu}}{\rho_\nu} \tag{6.54}$$

so that it is verified that $\mathsf{intgr}_{\mu\nu}$ differs from $\mathsf{intgr}_{\nu\mu}$ by a multiplication factor ρ_μ/ρ_ν. Therefore, computing one of the two integrals lead to the knowledge of the other one, if the multiplication factor is known. In order to go into the details of equations (6.53), return to the operative definition of the partial radial pair correlation function (6.46), transforming the integrals in summation over the bins

$$\mathsf{intgr}_{\mu\nu} = \sum_{i=1}^{\mathsf{nbin}} \rho_\mu \, g_{\mu\nu}(b_i) \; 4\pi [q(b_i)]^2 \delta q \qquad \text{and} \qquad \mathsf{intgr}_{\nu\mu} = \sum_{i=1}^{\mathsf{nbin}} \rho_\nu \, g_{\nu\mu}(b_i) \; 4\pi [q(b_i)]^2 \delta q \tag{6.55}$$

and, substituting equation (6.46) into the first relation of the previous definitions, this can be written as

$$\mathsf{intgr}_{\mu\nu} = \sum_{i=1}^{\mathsf{nbin}} \rho_\mu \, \frac{v \cdot n_c^{\mu\nu}(b_i)}{n_\mu n_\nu \cdot \tau_{\text{run}}} = \sum_{i=1}^{\mathsf{nbin}} \frac{n_c^{\mu\nu}(b_i)}{n_\nu \cdot \tau_{\text{run}}} \tag{6.56}$$

where in the last equality the relation for the density $\rho_\mu = n_\mu / v$ has been used.
At the same time, making the same procedure on the second equation in (6.55), it becomes

$$\mathsf{intgr}_{\nu\mu} = \sum_{i=1}^{\mathsf{nbin}} \rho_\nu \, \frac{v \cdot n_c^{\nu\mu}(b_i)}{n_\nu n_\mu \cdot \tau_{\text{run}}} = \sum_{i=1}^{\mathsf{nbin}} \frac{n_c^{\nu\mu}(b_i)}{n_\mu \cdot \tau_{\text{run}}} = \sum_{i=1}^{\mathsf{nbin}} \frac{n_c^{\mu\nu}(b_i)}{n_\mu \cdot \tau_{\text{run}}} \tag{6.57}$$

where the relation for the density $\rho_\nu = n_\nu / v$ has been used, and the symmetry for the number of couples in each bin $n_c^{\mu\nu}(b_i) = n_c^{\nu\mu}(b_i)$ can be easily derived from the equations (6.43) and (6.44) and from the exchange symmetry of the partial radial pair correlation function, $g_{\mu\nu}(b_i) = g_{\nu\mu}(b_i)$. Therefore, the equations (6.56) and (6.57) lead to the relation

$$\mathsf{intgr}_{\mu\nu} = \frac{n_\mu}{n_\nu} \, \mathsf{intgr}_{\nu\mu} \tag{6.58}$$

Furthermore, the integrated partial radial pair correlation function up to the i-th bin can be defined as

$$\mathsf{intgr}_{\mu\nu}(b_i) = \sum_{n=1}^{i} \rho_\mu \, g_{\mu\nu}(n) \; 4\pi [q(n)]^2 \, \delta q \qquad\qquad b_i = 1, ..., \, \mathsf{nbin} \tag{6.59}$$

The relation between the two summations is therefore the same that relates the integrals (6.53), as shown in (6.54). Thus, operatively speaking, all the information needed in order to compute all the values $\mathsf{intgr}_{\mu\nu}$, with ($\mu = 1, ..., n_\mu$ and $\nu = 1, ..., n_\nu$) can be stored in a $(\mathsf{ncpl}+1)\times\mathsf{nbin}$ matrix as reported in the following.

$$\mathsf{intgr} = \begin{pmatrix} intgr(b_1) & intgr(b_2) & ... & intgr(b_\lambda) \\ intgr_{c_1}(b_1) & intgr_{c_1}(b_2) & ... & intgr_{c_1}(b_\lambda) \\ ... & ... & ... & ... \\ intgr_{c_m}(b_1) & intgr_{c_m}(b_2) & ... & intgr_{c_m}(b_\lambda) \end{pmatrix} \qquad m = \mathsf{ncpl}, \; \lambda = \mathsf{nbin}$$

The first row contains the integrated total radial pair correlation function $\mathsf{intgr}(b_i)$ per each bin b_i, $i = 1, ..., \mathsf{nbin}$, as defined in equation (6.52). From the second to the $(\mathsf{ncpl}+1)$-th row, the integrated partial radial pair correlation functions $\mathsf{intgr}_{c_k}(b_i) \equiv \mathsf{intgr}_{\mu\nu}(b_i)$ are reported per each bin b_i for a given couple $c_k = (\mu, \nu)$ with species μ and ν, that corresponds to the two elements in the k-th column of the

matrix ncpl. These elements $\mathsf{intgr}_{\mu\nu}(b_i)$ are defined by a summation over the partial radial pair correlation function $g_{\mu\nu}(b_i)$ up to a certain radial distance, identified by the i-th bin index, as expressed in equation (6.59).
All the other elements needed, that regards the integrated partial pair correlation functions with the same atomic species but with the order inverted (i.e. the exchange symmetry related partial pair correlation functions), are computed from the previous ones by a multiplication factor, as in equation (6.58). For example, knowing the elements $\mathsf{intgr}_{\mu\nu}(b_i)$ for the couple of atomic species (μ, ν), the elements $\mathsf{intgr}_{\nu\mu}(b_i)$ for the couple of atomic species (ν, μ) can be computed with equation (6.58) by multiplying the elements $\mathsf{intgr}_{\mu\nu}(b_i)$ with a factor n_ν/n_μ. This procedure is obviously general for each couple of atomic species (μ, ν), with $\mu = 1, ..., n_\mu$ and $\nu = 1, ..., n_\nu$.

Creation of the output files The results obtained are printed in the output files. Two output files are created, namely, *pcf.dat* and *integrated_pcf.dat*.

The main file *pcf.dat* reports the total and the pair correlation functions for each spatial bin ib = 1,..., nbin. It has a number of columns equal to the number of pair correlation functions computed plus one, and is structured as follows:
1. first column: radial distance for each ib-th bin: $r(ib)$ (spatial mesh points)
2. second column: total pair correlation function for each ib-th bin: gr^{ib}
3. third-(ncpl+2) columns: partial (μ, ν) pair correlation functions for each ib-th bin: $gr^{ib}_{\mu\nu}$ → the name of the atomic species of the couple (μ, ν) is specified for each column in the first line of the file, respectively.

The second output file *integrated_pcf.dat* contains the integrated pair correlation functions for each spatial bin ib = 1,..., nbin. It has a number of columns that depends on the number of pair correlation functions computed, and it is structured as follows:
1. first column: radial distance for each ib-th bin: $r(ib)$ (spatial mesh points)
2. second column: total pair correlation function for each ib-th bin: $intgr^{ib}$
3. third-(ncpl+n_neq+2) columns: partial (μ, ν) integrated pair correlation functions for each ib-th bin: $intgr^{ib}_{\mu\nu}$ → the name of the atomic species of the couple (μ, ν) is specified for each column in the first line of the file, respectively.

6.2 Power Spectrum and Diffusion Coefficient

6.2.1 Theory

The aim of this chapter is to explain and demonstrate the Wiener-Khinchin theorem,[85, 86] that relates the power spectrum to the Fourier transform of the autocorrelation function $G(\tau)$, in the following way

$$P(\omega) = \mathcal{F}[G(\tau)] = \mathcal{F}[\langle f^*(t)\, f(t+\tau)\rangle] = \lim_{T\to\infty} \frac{1}{T} \left| \int_{-T/2}^{T/2} f(t)\, e^{-i\omega t}\, dt \right|^2 \tag{6.60}$$

where $\omega = 2\pi\nu$ and $G(\tau) = \langle f^*(t)\, f(t+\tau)\rangle$ is the autocorrelation function of the signal $f(t)$. In order to derive this theorem, the convolution theorem will be introduced and demonstrate in Section 6.2.1.1, then the Wiener-Khinchin will be proved in Section 6.2.1.2 and some properties of the autocorrelation function will be analyzed in Section 6.2.1.3. Finally, in Section 6.2.1.4, the calculation of the power spectra for the study of the vibrational frequencies in condensed matter systems through the post processing of nuclear velocities derived from molecular dynamics trajectory is outlined.

6.2.1.1 The convolution theorem

The convolution integral is defined as

$$(f * g)(\tau) = \int_{-\infty}^{\infty} f(t)\, g(\tau - t)\, dt \tag{6.61}$$

The convolution theorem gives an easy way to evaluate the convolution integral in equation (6.61), both in an intuitive and a computational point of view. The convolution theorem states that the Fourier transform of the convolution is the product of the Fourier transforms of the individual functions, namely,

$$\mathcal{F}(f * g) = \mathcal{F}(f)\, \mathcal{F}(g) \tag{6.62}$$

In the following, the assumption that the functions involved are well-behaved and nice enough that all the integrals simply exist is underpinned. To prove equation (6.62), the explicit form of $\mathcal{F}(f * g)$ has to be computed. First of all, the Fourier and inverse transforms have the form

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\omega)\, e^{i\omega t}\, d\omega \qquad \text{and} \qquad \mathcal{F}[f(t)] \equiv f(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt \tag{6.63}$$

It is important to make these definitions explicit, since the result depends on the normalization convention chosen for the Fourier transform. Then computing the Fourier transform of $\mathcal{F}(f * g)$ the following expression can be obtained

$$\begin{aligned} \mathcal{F}(f * g) &= \mathcal{F}\left[\int_{-\infty}^{\infty} f(t)\, g(\tau - t)\, dt \right] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\, g(\tau - t)\, e^{-i\omega\tau}\, dt\, d\tau \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, g(\tau - t)\, e^{-i\omega(\tau - t)}\, dt\, d\tau \end{aligned} \tag{6.64}$$

Letting $\tau \to \tau + t$

$$\begin{aligned} \mathcal{F}(f * g) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, g(\tau)\, e^{-i\omega\tau}\, dt\, d\tau \\ &= \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt \int_{-\infty}^{\infty} g(\tau)\, e^{-i\omega\tau}\, d\tau = \mathcal{F}[f]\, \mathcal{F}[g] \end{aligned} \tag{6.65}$$

Thus, to convolve two functions, just follow this recipe: Fourier transform both functions, multiply them together, then compute the inverse Fourier transform. Mathematically, it can be written

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}[f]\, \mathcal{F}[g]\} \tag{6.66}$$

Since Fourier transforms of common function are usually already known, the convolution theorem provides a shortcut for evaluating the full convolution integral.

6.2.1.2 The Wiener-Khinchin theorem

Recall the convolution theorem for two functions $f(t)$ and $g(t)$ (see Section 6.2.1.1). Writing out the convolution integral explicitly,

$$(f * g)(\tau) = \int_{-\infty}^{\infty} f(t)\, g(\tau - t)\, dt = \mathcal{F}^{-1}\{\mathcal{F}[f]\, \mathcal{F}[g]\} \tag{6.67}$$

Consider the particular choice $g(t) = f^*(-t)$, then

$$\mathcal{F}[g(t)] = \mathcal{F}[f^*(-t)] = \int_{-\infty}^{\infty} f^*(-t)\, e^{-i\omega t}\, dt = \int_{-\infty}^{\infty} f^*(t)\, e^{i\omega t}\, dt = (\mathcal{F}[f(t)])^* \tag{6.68}$$

Thus, equation (6.67) becomes

$$\int_{-\infty}^{\infty} f(t)\, f^*(t - \tau)\, dt = \mathcal{F}^{-1}\{\mathcal{F}[f]\, (\mathcal{F}[f(t)])^*\} = \mathcal{F}^{-1}\{|\mathcal{F}[f]|^2\} \tag{6.69}$$

where $g(\tau - t)$ in (6.67) has been transformed, following the condition $g(t) = f^*(-t)$ previously introduced, as $g(\tau - t) = f^*(t - \tau)$. Inverting the transform in (6.69) and letting $t \to t + \tau$, the Wiener-Khinchin theorem is recovered

$$\mathcal{F}\left[\int_{-\infty}^{\infty} f^*(t)\, f(t + \tau)\, dt\right] = |\mathcal{F}[f]|^2 \tag{6.70}$$

The function on the left hand side in (6.70) is the autocorrelation function of $f(t)$, that is defined as

$$G(\tau) = \int_{-\infty}^{\infty} f^*(t)\, f(t + \tau)\, dt \tag{6.71}$$

Using this definition, together with the Fourier transform convention introduced in equation (6.63), then the Wiener-Khinchin theorem in (6.70) can be rewritten as

$$\mathcal{F}[G(\tau)] = \left| \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt \right|^2 \tag{6.72}$$

Essentially, it compares the function $f(t)$ to itself but shifted by an amount τ by computing an overlap integral. The right hand side of equation (6.70) can be understand by noting that $\mathcal{F}[f(t)]$ is the spectrum of $f(t)$, and so $|\mathcal{F}[f]|^2$ is the energy spectral density of $f(t)$. Essentially, the energy spectrum is the square of the usual spectrum, with the phase information removed. This is consistent with the notion of energy going as the square of a signal amplitude. Thus, the Wiener-Khinchin theorem states that the Fourier transform of the autocorrelation function gives the energy spectrum.
There is one subtle point to these definitions: for some signals, such as steady optical signals, the correlation integral diverges,

$$G(\tau) = \int_{-\infty}^{\infty} f^*(t)\, f(t + \tau)\, dt \quad \to \quad \infty \tag{6.73}$$

In this case, a time average instead of the normal integral has to be considered. For an averaging time of T, the average in time is defined as

$$\langle f^*(t)\, f(t + \tau)\rangle_T = \frac{1}{T} \int_{-T/2}^{T/2} f^*(t)\, f(t + \tau)\, dt \tag{6.74}$$

For bounded signals, this integral is guaranteed to converge. To be physically sensible, T should be a suitably long observation time (e.g., long enough to resolve the frequency spectrum). For such signals, equation (6.64) becomes

$$\begin{aligned} \mathcal{F}[\langle f^*(t)\, f(t + \tau)\rangle_T] &= \mathcal{F}\left[\frac{1}{T} \int_{-T/2}^{T/2} f^*(t)\, f(t + \tau)\, dt\right] = \int_{-\infty}^{\infty} \frac{1}{T} \int_{-T/2}^{T/2} f^*(t)\, f(t + \tau)\, e^{-i\omega\tau}\, dt\, d\tau \\ &= \frac{1}{T} \int_{-T/2}^{T/2} f^*(t) \int_{-\infty}^{\infty} f(t + \tau)\, e^{-i\omega\tau}\, dt\, d\tau \end{aligned} \tag{6.75}$$

Letting $\tau \to \tau - t$

$$\begin{aligned}\mathcal{F}[\langle f^*(t)\, f(t+\tau)\rangle_T] &= \frac{1}{T}\int_{-T/2}^{T/2} f^*(t) \int_{-\infty}^{\infty} f(\tau)\, e^{-i\omega(\tau-t)}\, dt\, d\tau \\ &= \frac{1}{T}\int_{-T/2}^{T/2} f^*(t)\, e^{i\omega t}\, dt \int_{-\infty}^{\infty} f(\tau)\, e^{-i\omega\tau}\, d\tau\end{aligned} \tag{6.76}$$

Now the Wiener-Khinchin theorem says that the Fourier transform of the (time averaged) correlation function is the power spectral density, or the energy spectral density per unit time. For a stationary process, the correlation function is independent of t (generally for a sufficiently long averaging time T). Therefore, the averaging time can be extended $T \to \infty$. Denoting this long-time average limit as

$$\langle f^*(t)\, f(t+\tau)\rangle = \lim_{T\to\infty} \langle f^*(t)\, f(t+\tau)\rangle_T \tag{6.77}$$

In this limit, equation (6.76) becomes

$$\begin{aligned}\lim_{T\to\infty} \mathcal{F}[\langle f^*(t)\, f(t+\tau)\rangle_T] &= \mathcal{F}\left[\lim_{T\to\infty} \langle f^*(t)\, f(t+\tau)\rangle_T\right] = \mathcal{F}[\langle f^*(t)\, f(t+\tau)\rangle] \\ &= \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} f^*(t)\, e^{i\omega t}\, dt \int_{-\infty}^{\infty} f(\tau)\, e^{-i\omega\tau}\, d\tau\end{aligned} \tag{6.78}$$

and the Wiener-Khinchin theorem can be rewritten as

$$P(\omega) = \mathcal{F}[G(\tau)] = \int_{-\infty}^{\infty} G(\tau)\, e^{-i\omega\tau}\, d\tau = \lim_{T\to\infty} \frac{1}{T}\left|\int_{-T/2}^{T/2} f(t)\, e^{-i\omega t}\, dt\right|^2 \tag{6.79}$$

where the autocorrelation function $G(\tau)$ is defined as

$$G(\tau) = \langle f^*(t)\, f(t+\tau)\rangle = \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} f^*(t)\, f(t+\tau)\, dt \tag{6.80}$$

Again, the right hand side of equation (6.79) is the power spectral density $P(\omega)$, and in this form it is more clear that this is the energy density per unit time.

6.2.1.3 The autocorrelation function

Consider a real time series, i.e. a time dependent signal describe by a real-valued function $f(t)$ so that $f(t) : \mathbb{R} \to \mathbb{R}$. Assume that this signal is known over an infinitely long interval of time $[-T, T]$, with $T \to \infty$. The general time autocorrelation function between the two time series is given by

$$G(\tau) = \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} f(t)\, f(t+\tau)\, dt = \langle f(t)\, f(t+\tau)\rangle \tag{6.81}$$

that is known as the autocorrelation function of the signal $f(t)$. It can be proven that the autocorrelation function (6.81) is even. Indeed,

$$\begin{aligned}G(-\tau) &= \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} f(t)\, f(t-\tau)\, dt = \lim_{T\to\infty} \frac{1}{T}\int_{-T/2-\tau}^{T/2-\tau} f(t'+\tau)\, f(t')\, dt' \\ &= \lim_{T\to\infty} \frac{1}{T}\left[\int_{-T/2-\tau}^{-T/2} f(t'+\tau)\, f(t')\, dt' + \int_{-T/2}^{T/2} f(t'+\tau)\, f(t')\, dt'\right] = G(\tau)\end{aligned} \tag{6.82}$$

where the last equality holds in the limit $T \to \infty$, and in the first line the substitution

$$t - \tau = t' \qquad \text{so that} \qquad t = -\frac{T}{2} \quad \to \quad t' = -\frac{T}{2} - \tau \quad \text{and} \quad t = \frac{T}{2} \quad \to \quad t' = \frac{T}{2} - \tau \tag{6.83}$$

has been used.

6.2.1.4 Velocity autocorrelation function and power spectrum

An alternative way to compute vibrational properties from molecular dynamics trajectory is given by the power spectra, that describe all present vibrational and rotational movements of the system and do not only the infrared or Raman active ones. The power spectra feature peaks for each normal mode and allow the investigation of vibrational frequencies independently of infrared and Raman selection rules. The main aspect of this calculation is that the nuclear velocities evolving in time during a molecular dynamics trajectory are treated as a signal. The power spectrum $P(\omega)$ (also called power spectral density) is defined as the Fourier transform of the velocity autocorrelation function $G(\tau)$, namely,

$$P(\omega) = \mathcal{F}[G(\tau)] = \int_{-\infty}^{\infty} G(\tau)\, e^{-i\omega\tau}\, d\tau \tag{6.84}$$

where the velocity autocorrelation function $G(\tau)$ is defined as

$$G(\tau) = \langle \boldsymbol{v}(t) \cdot \boldsymbol{v}(t+\tau) \rangle = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} \boldsymbol{v}(t) \cdot \boldsymbol{v}(t+\tau)\, dt \tag{6.85}$$

where the velocity vector $\boldsymbol{v}(t)$ at time t in a three-dimensional system with N atoms has $3N$ components. It is also interesting to analyze the velocity autocorrelation function (6.85) derived from a molecular dynamics trajectory. Indeed, the form of this function can give an indication about the strength of the nuclear forces. The calculation of the velocity autocorrelation function (6.85) is performed using the formula

$$\begin{aligned} G(\tau) &= \frac{1}{3N(t_{end} - t_{beg} - \tau)} \sum_{\alpha=1}^{3N} \sum_{t=t_{beg}}^{t_{end}-\tau} v_\alpha(t)\, v_\alpha(t+\tau) \\ &= \frac{1}{3N(t_{end} - t_{beg} - \tau)} \sum_{i=1}^{N} \sum_{j=1}^{3} \sum_{t=t_{beg}}^{t_{end}-\tau} v_{i,j}(t)\, v_{i,j}(t+\tau) \end{aligned} \tag{6.86}$$

where $\tau = \Delta t$ is the time step, t_{end} is the time length of the whole trajectory, $v_\alpha(t)$ with $\alpha = 1, ..., 3N$ are the components of the $\boldsymbol{v}(t)$ velocity vectors, containing three components per each nucleus, and $v_{i,j}(t)$ with $i = 1, ..., N$ and $j = 1, 2, 3$ is another way of writing the nuclear velocities scalar product in the left hand side of equation (6.85), that indicates explicitly the j-th Cartesian component of the i-th nucleus. It would be also useful to compare the form of the velocity autocorrelation function computed using molecular dynamics simulations at different temperatures. In this case, the autocorrelation function (6.85) can be normalized by the average of the scalar product of the velocity vector at time $t = 0$, namely,

$$G(\tau) = \frac{\langle \boldsymbol{v}(t) \cdot \boldsymbol{v}(t+\tau) \rangle}{\langle \boldsymbol{v}(0) \cdot \boldsymbol{v}(0) \rangle} \tag{6.87}$$

so that the previous formula (6.86) becomes

$$\begin{aligned} G(\tau) &= \frac{1}{3N(t_{end} - t_{beg} - \tau)} \sum_{\alpha=1}^{3N} \sum_{t=t_{beg}}^{t_{end}-\tau} v_\alpha(t)\, v_\alpha(t+\tau) \left[\frac{1}{3N} \sum_{\alpha=1}^{3N} v_\alpha(t_{beg})\, v_\alpha(t_{beg}) \right]^{-1} \\ &= \frac{1}{(t_{end} - t_{beg} - \tau)} \sum_{\alpha=1}^{3N} \sum_{t=t_{beg}}^{t_{end}-\tau} v_\alpha(t)\, v_\alpha(t+\tau) \left[\sum_{\alpha=1}^{3N} v_\alpha(t_{beg})\, v_\alpha(t_{beg}) \right]^{-1} \end{aligned} \tag{6.88}$$

Instead of computing the Fourier Transform of the velocity autocorrelation function, the power spectrum can be evaluated more easily using the Wiener-Khinchin theorem,[85, 86] which stated that the Fourier transform of the (time averaged) autocorrelation function is the power spectral density, or the energy spectral density per unit time. Therefore, the power spectrum (6.84) can be also computed as

$$P(\omega) = \mathcal{F}[G(\tau)] = \lim_{T\to\infty} \frac{1}{T} \sum_{\alpha=1}^{3N} \left| \int_{-T/2}^{T/2} v_\alpha(t)\, e^{-i\omega t}\, dt \right|^2 \tag{6.89}$$

Operatively, the power spectrum is implemented using a finite time version of the formula in (6.89), that is

$$P(\omega) = \frac{1}{t_{end} - t_{beg}} \sum_{\alpha=1}^{3N} \left\{ \left[\sum_{t=t_{beg}}^{t_{end}} v_\alpha(t)\, \cos(\omega t) \right]^2 + \left[\sum_{t=t_{beg}}^{t_{end}} v_\alpha(t)\, \sin(\omega t) \right]^2 \right\} \tag{6.90}$$

This calculation has the advantage that it does not affect the computational cost of an ab initio molecular dynamics simulation, as it is evaluated as a post processing quantity, after the simulation has been performed. At the same time, the reliability of a power spectrum calculation increases as the system size and the simulation time increase, so that long molecular dynamics runs are required to obtain a good quality power spectrum and thus a good estimation of vibrational frequencies. The power spectra can be a useful tool to interpret the signals found in experimental spectra, without the necessity to compute infrared or Raman spectra. Recent studies have demonstrated that it can be a unique tool in the study of short time dynamics in ionic liquids, allowing to obtain power spectra with peaks at wavenumbers in agreement with experimental data.[87] However, it has to be noted that the intensities derived from theoretical power spectra may not match the infrared or Raman intensities.[87] Therefore, the power spectra can be used as a tool of validation of the molecular dynamics approach used, and at the same time, they provide important information for computing the vibrational properties of a system and for interpreting the associated experimental data.

6.2.1.5 Velocity autocorrelation function and diffusion coefficient

Using the Green-Kubo formula, the diffusion coefficient can be defined in terms of the velocity autocorrelation function (VACF), denoted as $G_v(\tau)$, in the following way

$$D = \int_0^\infty G(\tau)\, d\tau \equiv \int_0^\infty \langle \boldsymbol{v}(t) \cdot \boldsymbol{v}(t+\tau) \rangle \; d\tau \tag{6.91}$$

where the velocity vector $\boldsymbol{v}(t) = \{v_\alpha(t)\}$ with $\alpha = 1, ..., Nd$ at time t in a d dimensional system of N atoms has Nd components. The VACF can be easily computed by discretizing the time coordinate in the time average over the dynamical evolution contained in the definition of $G(\tau)$, as follows

$$\begin{aligned} G(\tau) &= G(k\Delta t) \\ &= \frac{1}{Nd(n_t - k + 1)} \sum_{\alpha=1}^{Nd} \sum_{n=0}^{n_t - k} v_\alpha(n\Delta t)\, v_\alpha[(n+k)\Delta t] \end{aligned} \tag{6.92}$$

where k is an integer defined in the range $(0, n_t)$, while Δt is the time step and n_t is the total number of molecular dynamics steps carried out from a given initial state. In practical implementations, the integral over time in equation (6.91) has to be discretized and it is computed using τ_{cut} as an upper bound in the integration, that is

$$D(\tau_{cut}) = \sum_{\tau=0}^{\tau_{cut}} G(\tau)\, \Delta t \tag{6.93}$$

where Δt is the time step of the molecular dynamics simulation and τ_{cut} is chosen as the time at which the normalized velocity autocorrelation function $G(\tau)$ begins to oscillate around zero. The diffusion coefficient D is a collective observable describing a process where, in the absence of external forces, an initial concentration gradient is canceled through molecular motions. This phenomenon is specific to the liquid and gaseous phases in which, starting from a non-uniform concentration, self-diffusion causes the system to evolve towards a uniform equilibrium concentration. In the case of solids, self-diffusion is less interesting where instead it is possible to calculate the migration of interstitial defects, vacancies or molecules trapped inside the lattice. Thus, in order to estimate the diffusion coefficient of an atom or a group of atoms along one or more directions, equations (6.92) and (6.93) can be used, with the summation in (6.92) restricted to the velocities of the atoms (and directions) of interest.

6.2.2 Implementation

In this section the implementation of the calculation of normal modes, frequencies and power spectrum (or power spectral density) as a post-processing of the molecular dynamics velocities in the CRYSTAL code is described.

6.2.2.1 Requirements

For the calculation of the normal modes, the frequencies and the power spectrum, an input file with the nuclear velocities per each step of the molecular dynamics simulation is required. When the Module Moldyn is activated through the keyword MOLDYN the input file, and a molecular dynamics simulation is performed, the code generates by default the files POSITIONS.DAT and VELOCITIES.DAT, which contains the positions and velocities of each atom during the molecular dynamics simulation. At the same time, the code updates at each step the file fort.34 (by overwriting it at each step), where the direct lattice vectors as well as the MD final geometry are written. Since the normal modes and related quantities rely on the velocities of the particles, a file VELOCITIES.DAT must exist in the same directory where the post-processing analysis is carried out. For these reasons, the code check for the existence of a file named VELOCITIES.DAT; if the file exists in the directory of execution, everything is fine and the calculation can proceed, otherwise, if the file does not exist, the code call the subroutine errvrs with ierr = 0 (this subroutine is implemented in library both4.f), it stops and prints in the output file the following string:

ERROR **** FREQUENCIES_MD **** FILE VELOCITIES.DAT NOT FOUND!

6.2.2.2 The subroutine frequencies_md: description and methods

The subroutine implemented for the calculation of the normal modes, the frequencies and the power spectrum is the subroutine frequencies_md, contained in the module moldyn_post in the library moldyn_post.f90. The execution section of the subroutine frequencies_md can be divided into the following different parts:

1. Reading of the file VELOCITIES.DAT, saving the information needed
2. Creation of the output files with the results

Then, in the final part of the subroutine, the files used are closed and the dynamical vectors and matrices are deallocated. In the following part of this chapter, the different execution subsections of the code listed above are described, taking particular attention on the way the data are saved and manipulated within vectors and matrices to obtain the results, and on the practical way to implement the calculation of vibrational modes and frequencies and their intensities, whose underneath theoretical basis have been explained in Section 6.2.1.

Reading of the file VELOCITIES.DAT and saving information The subroutine frequencies_md calls the subroutine readVELfile of the moldyn_post module in order to accomplish the task of reading the file VELOCITIES.DAT with the nuclear velocities per each molecular dynamics step. The file VELOCITIES.DAT, generated by default during a molecular dynamics calculation (and described in Appendix B.3), is read by the subroutine readVELfile as follows:

- The first row contains the number of atoms in the system, saved in the integer variable nats.
- The following nats rows are divided into three columns: the first one for the row index (from 1 to nats), the second one for the atomic number and the third one for the atomic mass of each atom, respectively. Only the atomic number is saved in the dynamical integer vector atos_tmp(nats).
- The following nstp·nats rows contains the Cartesian nuclear velocities (v_x, v_y, v_z) [in Angstrom/fs] of each atom in the initial atomic configuration and in the successive nstp−1 steps of the molecular dynamics simulation. These velocities are saved in matrices with dimension nstp·nats in the following way

$$\boldsymbol{v} = \begin{pmatrix} v_{x,1}^{s_1} & v_{x,1}^{s_2} & \dots & v_{x,1}^{s_m} \\ v_{y,1}^{s_1} & v_{y,1}^{s_2} & \dots & v_{y,1}^{s_m} \\ v_{z,1}^{s_1} & v_{z,1}^{s_2} & \dots & v_{z,1}^{s_m} \\ v_{x,2}^{s_1} & v_{x,2}^{s_2} & \dots & v_{x,2}^{s_m} \\ v_{y,2}^{s_1} & v_{y,2}^{s_2} & \dots & v_{y,2}^{s_m} \\ v_{z,2}^{s_1} & v_{z,2}^{s_2} & \dots & v_{z,2}^{s_m} \\ \dots & \dots & \dots & \\ v_{x,\mathsf{nats}}^{s_1} & v_{x,\mathsf{nats}}^{s_2} & \dots & v_{x,\mathsf{nats}}^{s_m} \\ v_{y,\mathsf{nats}}^{s_1} & v_{y,\mathsf{nats}}^{s_2} & \dots & v_{y,\mathsf{nats}}^{s_m} \\ v_{z,\mathsf{nats}}^{s_1} & v_{z,\mathsf{nats}}^{s_2} & \dots & v_{z,\mathsf{nats}}^{s_m} \end{pmatrix} \qquad \text{[Bohr / (au time)]}$$

where the subscripts 1,...,naf are referred to the indexes of the atoms while the superscripts s_1,...,s_m are the indexes of the molecular dynamics step (with s_m = nstp). These matrices are therefore filled column-by-column while reading the file, so that all (v_x, v_y, v_z) velocities per each atom of the system are given for the initial configuration (first column), then for the first step (second column), and so on.

The atomic number of each atomic species in the system has been saved in vector atos_tmp(nats). Since the atomic masses are important for the calculation of the vibrational modes and frequencies, the atomic masses are computed starting from the atomic numbers saved in atos_tmp(nats) using the function aisotop(at_number) in the following way

$$\mathsf{mass\_nc(i)} = \mathsf{aisotop(mod(atos\_tmp(i),100))\cdot amu} \qquad \text{with} \qquad \mathsf{i} = 1,...,\mathsf{nats}$$

where amu is the conversion factor from atomic units to Hartree units used in the code.

Calculation of the velocity autocorrelation function The calculation of the velocity autocorrelation function is implemented in the code using formula (6.86). In particular, considering the set of velocity components v_α^i at the i-th molecular dynamics steps for each degrees of freedom $\alpha = 1, ..., 3N$, where N is the number of atoms in the system, the velocity autocorrelation function associated to the α-th degrees of freedom at a time t associated to the index τ can be computed as

$$\tilde{G}_{\alpha,\tau} = \sum_{s=s_b}^{s_e-\tau+1} v_\alpha^s \, v_\alpha^{s+\tau-1} \qquad \alpha = 1, ..., 3N \qquad \tau = 1, ..., s_e \tag{6.94}$$

$$G_{\alpha,\tau} = \frac{1}{3N(n_s - \tau + 1)} \sum_{s=s_b}^{s_e-\tau+1} v_\alpha^s \, v_\alpha^{s+\tau-1} \qquad \alpha = 1, ..., 3N \qquad \tau = 1, ..., s_e \tag{6.95}$$

where s_b and s_e are respectively the indexes of the initial and final molecular dynamics steps for which the velocity autocorrelation function has to be computed, $n_s = s_e - s_b + 1$ is the number of molecular dynamics steps between s_e and s_b (considering a step of one between a the i-th and the i-th step, i.e. considering all the molecular dynamical steps) and τ is the time index, which ranges up to a maximum index of s_e with a step of one, i.e. $\tau = 1, 2, ..., s_e - 1, s_e$.
Therefore, the velocity autocorrelation function can be stored in a $3N \times n_s$ matrix auto_vcorr($3N$, n_s) defined in the time domain, where n_s is the number of molecular dynamics steps considered.

At the same time, the velocity autocorrelation matrix in frequency domain can be stored in a $3N \times n_\omega$ matrix auto_vcorr_fourier($3N$, n_ω), where $n_\omega = n_s$ is the number of frequency values spanned (equal to the number of molecular dynamics step for which the velocity autocorrelation function is computed). Therefore, the velocity autocorrelation function associated to the α-th degrees of freedom at a frequency ω can be computed using the Wiener-Khinchin theorem as applied in formula (6.90) for each single

degrees of freedom, leading to the implemented equation

$$\begin{aligned} G_{\alpha,\omega} = &\sum_{s=s_b}^{s_e} v_\alpha^s \cos(t_s \cdot \omega \cdot d\omega) \cdot \sum_{s=s_b}^{s_e} v_\alpha^s \cos(t_s \cdot \omega \cdot d\omega) \\ &+ \sum_{s=s_b}^{s_e} v_\alpha^s \sin(t_s \cdot \omega \cdot d\omega) \cdot \sum_{s=s_b}^{s_e} v_\alpha^s \sin(t_s \cdot \omega \cdot d\omega) \qquad \alpha = 1, ..., 3N \qquad \omega = 1, ..., n_\omega \end{aligned} \tag{6.96}$$

where t_s is the time value in units of ps and $d\omega$ is expressed in THz units. Finally, to properly obtain the power spectrum of the system in equation (6.90), all the components related to each degrees of freedom have to be summed up, so that

$$P(\omega) = \sum_{\alpha=1}^{3N} G_{\alpha,\omega} \qquad \omega = 1, ..., n_\omega \tag{6.97}$$

Table 6.3 reports some details about the implemented algorithm just mentioned for the calculation of the power spectrum by the application of the Wiener-Khinchin theorem and for the computation of the velocity autocorrelation function, used in formula (6.93) for the estimation of the diffusion coefficient.

Fourier Transform $t \to \omega$	$n_\alpha = 3N$ where N = number of atoms
$\omega_i = \omega_{min} + (i-1)\,\delta_\omega$	$i = 1, ..., n_\omega$
$t_k = t_{min} + (k-1)\,\delta_t$	$k = 1, ..., n_t$

$$\Re[\mathcal{F}_i[\{v(1:n_\alpha, 1:t_{n_t})\}]] = \mathcal{F}_i(1:n_\alpha, 1) = \sum_{k=1}^{n_t} v(1:n_\alpha, t_k) \cos(\omega_i\, t_k)$$

$$\Im[\mathcal{F}_i[\{v(1:n_\alpha, 1:t_{n_t})\}]] = \mathcal{F}_i(1:n_\alpha, 2) = \sum_{k=1}^{n_t} v(1:n_\alpha, t_k) \sin(\omega_i\, t_k)$$

$$p(1:n_\alpha, i) = \mathcal{F}_i(1:n_\alpha, 1) \cdot \mathcal{F}_i(1:n_\alpha, 1) + \mathcal{F}_i(1:n_\alpha, 2) \cdot \mathcal{F}_i(1:n_\alpha, 2) \qquad i = 1, ..., n_\omega$$

$$P(i) = \sum_{\mu=1}^{n_\alpha} p(\mu, i) \qquad i = 1, ..., n_\omega$$

Velocity Autocorrelation Function	$n_\alpha = 3N$ where N = number of atoms

$$G(\alpha, m-n) = \sum_{m=t_i}^{t_e} \sum_{n=t_i}^{m} v(\alpha, m) \cdot v(\alpha, n) \qquad \alpha = 1, ..., n_\alpha \quad \text{and} \quad m - n = 1, ..., n_t$$

$$G(\alpha, \tau) = \sum_{n=0}^{n_t - 1} v(\alpha, t_i) \cdot v(\alpha, t_i + n\tau) \qquad \alpha = 1, ..., n_\alpha \quad \text{and} \quad \tau = 1, ..., n_t$$

$$G(\tau) = \sum_{\alpha=1}^{n_\alpha} A(\alpha, \tau) \qquad \tau = 1, ..., n_t$$

Table 6.3: Implemented steps for the calculation of the power spectrum through the Wiener-Khinchin theorem (upper panel) and for the calculation of the velocity autocorrelation function (lower panel).

Chapter 7

Molecular dynamics simulations in microcanonical and canonical ensembles: results and discussion

For the purpose of demonstrating the reliability of molecular dynamics module in the CRYSTAL code, calculations on proton-ordered ice with space group $Pna2_1$ (P-ice) and $(H_2O)_{32}$ water 3D periodic systems are presented. The first structure is one of the hypothetical proton-ordered model which satisfy the Bernal-Fowler-Pauling rules[88, 89] among the 17 possible orthorhombic symmetry-independent structures that can exist with eight water molecules per unit cell,[90] and it has been heralded as a reasonable and convenient reference model for the simulation of ordinary ice.[91] The second system considered consists in 32 water molecules inside a periodic cubic cell of size $a = b = c = 9.855$ Å, which corresponds to liquid water at a density of 1.00 g·cm^{-3}. The computational details of simulations are outlined in Section 7.2, while further information about the initial nuclear coordinates and cell parameters are reported for both systems in Sections D and in the Supplementary Information of Ref. [92]. These two models are representative of hydrogen bonded environments, that are known to be very difficult to characterize, thus being optimal starting test cases to validate computational methods and the corresponding practical implementation in quantum mechanical packages. At the same time, these systems are very well known and extensively studied, thus permitting a strict comparison with the results found in existing literature.
In the microcanonical ensemble, the modeling of P-ice and liquid water has been done using a GGA functional (PBE)[93] and two global hybrid functionals (B3LYP and PBE0). Simulations with PBE and B3LYP functionals enriched with the a posteriori dispersion correction as proposed by Grimme (PBE-D3 and B3LYP-D3)[94, 95] were also carried out, for both systems. In the canonical ensemble, instead, the modeling of both P-ice and liquid water structures has been performed with the hybrid functional B3LYP, which is known to provide a more accurate description of hydrogen bonded water based materials.[96]
First of all, the accuracy in terms of the drift in the conserved quantities and the related standard deviation has been evaluated, for all the molecular dynamics simulations in both the microcanonical (NVE) and the canonical (NVT) ensembles, where the conserved quantities are, respectively, the total energy (5.39) and the quantity (5.296) that also includes the thermostat degrees of freedom. The drift in the conserved quantities has been computed by performing a least-squares fit of the correspondent trajectory to a straight line, so that the reported total drift is the absolute value of the difference at the first and last point of the fitting line. The results obtained are reported in Table 7.1, together with the mean value of the temperature and the corresponding standard deviation.
The data demonstrate the presence of a very small drift in the conserved quantity, and a good temperature control reproduced by the NVT ensemble. In order to obtain an accurate sampling of the canonical ensemble, the parameter Q related to the mass of the thermostat has to be accurately chosen, as discussed in Section 5.3.4.7. To this aim, the parameter Q has been varied using a period τ_s in the range $(10, 100)$ fs for the P-ice and $(22, 266)$ fs for the $(H_2O)_{32}$ liquid water systems (see equation (5.384) for the relation between the thermostat mass and the period). The drift and standard deviation in the conserved quantity, the mean temperature and the standard deviation of the temperature for each different value of Q have been evaluated using PBE functional, the results are reported in Table 7.2 for both P-ice and $(H_2O)_{32}$ systems.

System	Ensemble	Functional	C_{drift} [eV/ps/atom]	σ_C [Hartree]	$\bar{T}$ [K]	σ_T [K]	D [10^{-9} m^2/s]
P-ice	NVE	PBE	$1.39998 \cdot 10^{-5}$	$1.58244 \cdot 10^{-5}$	148.857	23.235	0.73
	NVE	PBE-D3	$2.04989 \cdot 10^{-5}$	$1.42484 \cdot 10^{-5}$	148.765	23.108	0.23
	NVE	B3LYP	$3.65062 \cdot 10^{-5}$	$1.54663 \cdot 10^{-5}$	148.561	22.558	0.17
	NVE	B3LYP-D3	$2.40323 \cdot 10^{-5}$	$1.52647 \cdot 10^{-5}$	148.404	21.927	0.01
	NVE	PBE0	$2.05324 \cdot 10^{-5}$	$1.49972 \cdot 10^{-5}$	149.211	22.865	0.18
	NVT	B3LYP	$7.92477 \cdot 10^{-5}$	$3.43785 \cdot 10^{-5}$	300.103	45.367	0.79
	NVT	B3LYP	$1.86164 \cdot 10^{-4}$	$7.76731 \cdot 10^{-5}$	600.037	86.383	3.53
$(H_2O)_{32}$	NVE	PBE	$4.34625 \cdot 10^{-5}$	$9.70295 \cdot 10^{-5}$	427.077	28.579	3.06
	NVE	PBE-D3	$6.16852 \cdot 10^{-5}$	$1.07775 \cdot 10^{-4}$	430.797	31.161	2.81
	NVE	B3LYP	$7.57273 \cdot 10^{-5}$	$1.29620 \cdot 10^{-4}$	397.799	24.126	4.46
	NVE	B3LYP-D3	$7.04336 \cdot 10^{-5}$	$1.12112 \cdot 10^{-4}$	399.862	25.810	4.64
	NVE	PBE0	$3.46565 \cdot 10^{-5}$	$1.11460 \cdot 10^{-4}$	403.606	27.976	3.97
	NVT	B3LYP	$1.39504 \cdot 10^{-5}$	$6.80806 \cdot 10^{-5}$	300.058	29.339	3.83
	NVT	B3LYP	$2.10581 \cdot 10^{-5}$	$1.91238 \cdot 10^{-4}$	599.879	47.198	11.11

Table 7.1: Drift in the conserved quantity C_{drift}, standard deviation of the conserved quantity σ_C, mean temperature $\bar{T}$ with the corresponding standard deviation σ_T and diffusion coefficient D obtained for all the molecular dynamics simulations performed on the two test cases systems. The drift, standard deviations and mean values are computed on the whole 1 ps trajectory, with initial temperature of $T = 300$ K for the microcanonical (NVE) ensemble and initial temperatures of $T = 300$ K and $T = 600$ K for the two simulations in the canonical (NVT) ensemble, and a time step equal to 0.15 fs and 0.25 fs, respectively for the P-ice and the $(H_2O)_{32}$ liquid-like systems. The diffusion coefficients are obtained using formula (6.93) with $\tau_{cut} = 600$ fs for P-ice and $\tau_{cut} = 200$ fs for $(H_2O)_{32}$ cubic system.

The calculation of dynamical properties from the post-processing of molecular dynamics trajectory is also very useful to evaluate the accuracy and capabilities of the Born-Oppenheimer molecular dynamics implementation.

The pair correlation functions of the $(H_2O)_{32}$ liquid water is first computed from the post-processing of NVE molecular dynamics simulations using PBE and B3LYP functionals, as reported respectively in the left and right panel of Figure 7.1. The comparison with the experimental data demonstrates that the hybrid functional reproduces the experimental patterns with much more precision, matching the experimental radius of each coordination shell (corresponding to a maximum in the pair correlation function) with increased accuracy with respect to PBE functional, as already known from other theoretical studies.[97]

Then, the pair correlation function of the $(H_2O)_{32}$ computed from a canonical trajectory at 300 K and 600 K are compared, in Figure 7.2, with the experimental data. As already highlighted in literature,[97] it is plausible that some amount of the observed overstructure is due to the neglect of the quantum motion of the hydrogen atoms. The effect of the increase in average temperature, which is claimed to compensate for this lack,[97] leads to a striking agreement with experimental data.

Moreover, the power spectra obtained at the B3LYP level from the NVT molecular dynamics of both P-ice and $(H_2O)_{32}$ systems at $T = 300$ K are shown in Figure 7.3. The broadening of the peaks and the appearance of a broad band in the librational zone is a direct consequence of the transition from a crystalline structure to a liquid-like material that our dynamics manages to reproduce perfectly.

Finally, the diffusion coefficients are computed and the values are listed in Table 7.1. In particular, the result obtained in the case of the $(H_2O)_{32}$ liquid-like system at 300 K using the B3LYP functional in the canonical ensemble reproduce the correct order of magnitude of the experimental data, which also present a certain degree of variability, ranging from $2.3 \cdot 10^{-9}$ m^2/s (Ref. [98]) to $2.4 \cdot 10^{-9}$ m^2/s (Ref. [99]) at 298.2 K. Moreover, this value for the diffusion coefficient is also in agreement with the result

System	τ_s [fs]	Q [kJ·ps^2/mol]	C_{drift} [eV/ps/atom]	σ_C [Hartree]	$\bar{T}$ [K]	σ_T [K]
P-ice	10.00	$8.71916 \cdot 10^{-4}$	$1.20277 \cdot 10^{-4}$	$3.09166 \cdot 10^{-5}$	299.660	44.475
	20.00	$3.48767 \cdot 10^{-3}$	$1.88581 \cdot 10^{-4}$	$3.22775 \cdot 10^{-5}$	300.004	50.878
	30.00	$7.84725 \cdot 10^{-3}$	$7.72230 \cdot 10^{-5}$	$2.83148 \cdot 10^{-5}$	298.067	50.722
	100.00	$8.71916 \cdot 10^{-2}$	$2.08887 \cdot 10^{-4}$	$5.16090 \cdot 10^{-5}$	302.522	82.868
$(H_2O)_{32}$	22.00	$1.74307 \cdot 10^{-2}$	$1.08989 \cdot 10^{-4}$	$8.11815 \cdot 10^{-5}$	300.101	26.575
	38.00	$5.20041 \cdot 10^{-2}$	$9.61499 \cdot 10^{-5}$	$7.81892 \cdot 10^{-5}$	299.628	34.960
	55.00	$1.08942 \cdot 10^{-1}$	$1.46431 \cdot 10^{-4}$	$9.25325 \cdot 10^{-5}$	299.703	33.472
	266.00	$25.48202 \cdot 10^{-1}$	$3.58336 \cdot 10^{-5}$	$7.22603 \cdot 10^{-5}$	303.720	56.555

Table 7.2: Drift in the conserved quantity C_{drift} and corresponding standard deviation σ_C, mean temperature $\bar{T}$ and corresponding standard deviation σ_T for all the NVT (Nosé-Hoover thermostat) molecular dynamics simulations performed on the two test cases systems using PBE functional, associated to the different values of the coupling period τ_s and the thermostat mass Q. The drift, standard deviations and mean values are computed for a trajectory of 2000 steps in the NVT ensemble (Nosé-Hoover thermostat) with initial temperature of $T = 300$ K and time step equal to 0.15 fs and 0.25 fs, respectively for the P-ice and the $(H_2O)_{32}$ liquid-like systems.

of $1.0 \cdot 10^{-9}$ m^2/s found by Silvestrelli et al.[100] using Car-Parrinello molecular dynamics with BLYP functional.
These results demonstrate that the dynamical properties can be accurately determined from both the microcanonical and the canonical (Nosé-Hoover thermostat) ensembles.

7.1 Scaling efficiency of molecular dynamics algorithm

In order to assess the computational performances of the molecular dynamics algorithm, a strong and a weak scaling analyses were investigated. In the first case, the number of processing units n_p is varied while maintaining fixed the system size; in the second case, the system size (expressed in terms of number of atoms in the reference cell n_{at}) is varied while maintaining fixed the ratio of n_p to system size, $\zeta = n_p/n_{at}$. The crystalline ice belonging to the $Pna2_1$ space group (P-ice), with 8 water molecules in the reference cell, is chosen as the test system and a set of supercells were constructed, containing 16, 24 and 32 water molecules, respectively. A series of 16 different molecular dynamics simulations were performed, in which the system size is varied to include $n_{at} = 24, 48, 72, 96$ atoms, that correspond respectively to $8, 16, 24, 32$ water molecules, and the ratio is varied so that $\zeta = 1/6, 1/3, 1, 2$. The optimized structure of P-ice, whose lattice parameters are reported in Table D.3 and in the Supporting Information of Ref. [92], is used as a starting point to construct supercells of increasing size. In this way, the initial atomic configuration is the same for the dynamics of all supercell models, and the possible effect on the scaling analysis due to different initial conditions is minimized. The computational parameters used for all the simulations are reported in Section 7.2.
The scaling efficiency has been evaluated by averaging over 200 molecular dynamics steps, using a time step of 0.15 fs in the microcanonical (NVE) ensemble, with an initial temperature equal to 300 K. The average timings obtained using the PBE and B3LYP hybrid functionals are reported in Table 7.3 (in units of s/step) and are meant to reflect the mean computational time required to perform a self-consistent cycle ($\langle t_{\mathrm{SCF}} \rangle$) and the associated analytical nuclear forces calculation ($\langle t_{\mathrm{GRAD}} \rangle$).
All the calculations presented in this section are performed on a x86_64 Debian GNU/Linux cluster with AMD EPYC 7742 64-Core @ 2552.484 MHz processors grouped into 128 processing units per each node. The strong scaling behavior can be assessed by analyzing how $\langle t_{\mathrm{SCF}} \rangle$ and $\langle t_{\mathrm{GRAD}} \rangle$ change as the number of processors is varied for a fixed problem size. For each system with a given number of atoms, this is accomplished using the formula

$$\eta^m_{\mathrm{strong}}(\zeta) \equiv \frac{\zeta_{ref} \cdot \langle t_m \rangle_{\zeta_{\mathrm{ref}}}}{\zeta \cdot \langle t_m \rangle_{\zeta}} = \frac{\frac{1}{6} \cdot \langle t_m \rangle_{\zeta=1/6}}{\zeta \cdot \langle t_m \rangle_{\zeta}} \tag{7.1}$$

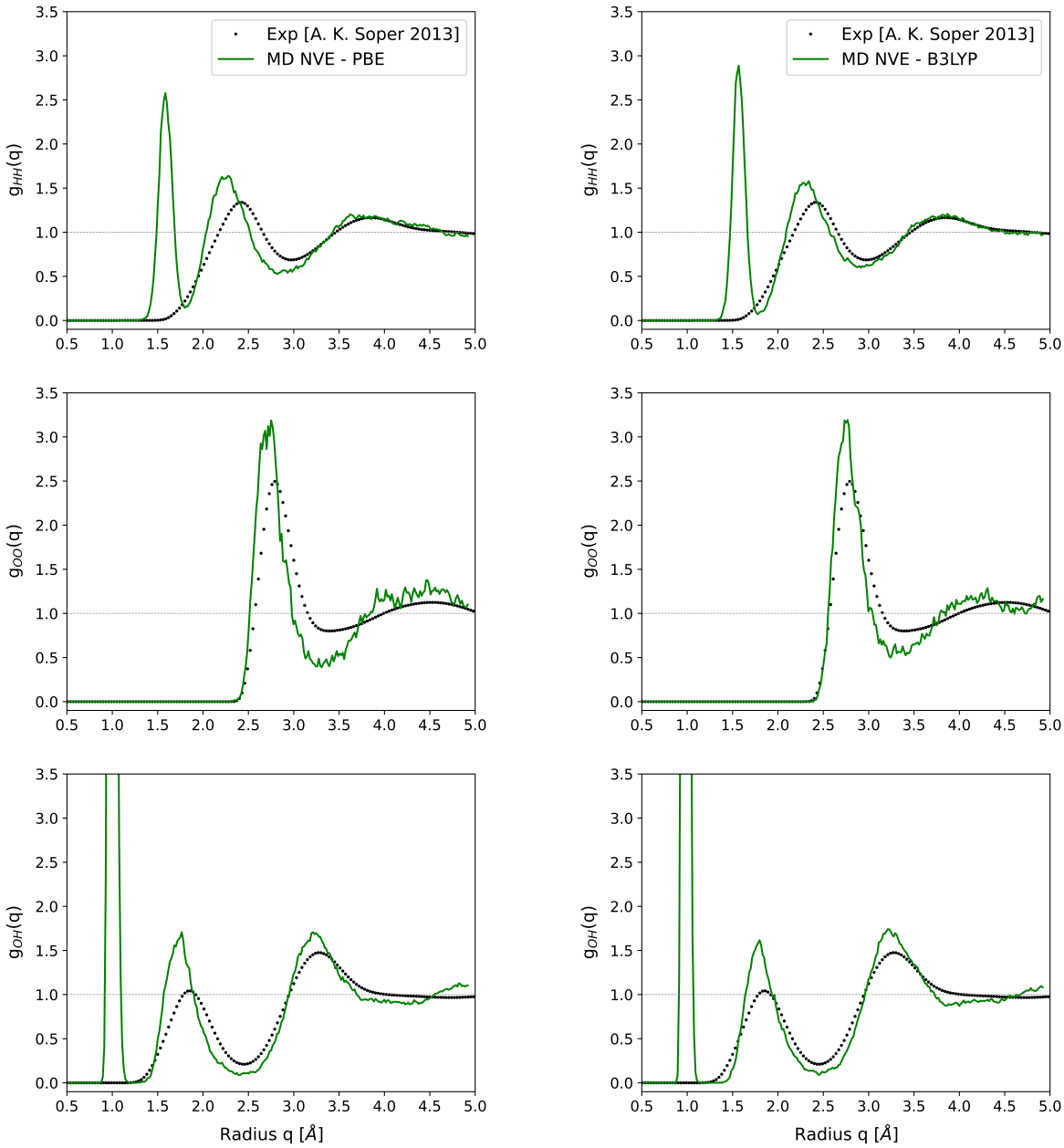

Figure 7.1: Pair correlation functions of the periodic $(H_2O)_{32}$ cubic system obtained from a post-processing of 1 ps NVE molecular dynamics trajectory using PBE (green lines, left panel) and B3LYP (green lines, right panel) functionals, together with the experimental counterpart (black dots) by A. K. Soper.[101] See Table 7.1 for details about molecular dynamics simulations.

where $\zeta_{\text{ref}} = 1/6$ is chosen as the reference value for the ratio ζ of processing units to system size and $\langle t_m \rangle_\zeta$ is the average time spent in the m = SCF or m = GRAD module for a given ζ. The efficiency factors for all the systems is reported in Table 7.3. The weak scaling behavior, namely, the changes in $\langle t_{\text{SCF}} \rangle$ and $\langle t_{\text{GRAD}} \rangle$ as the system size is varied for a fixed ratio ζ is computed as

$$\eta^m_{\text{weak}}(n_{at}) \equiv \frac{\langle t_m \rangle_{n^{\text{ref}}_{at}}}{\langle t_m \rangle_{n_{at}}} = \frac{\langle t_m \rangle_{n_{at}=24}}{\langle t_m \rangle_{n_{at}}} \tag{7.2}$$

where $n^{\text{ref}}_{at} = 24$ is chosen as the reference system size, and $\langle t_m \rangle_{n_{at}}$ is the average time spent in the m = SCF or m = GRAD module for a given number of atoms n_{at}.
The last parameter considered is the speedup Θ, that permits to assess the scaling efficiency of a generic algorithm, by means of the formula

$$\Theta \equiv \frac{T(n^{\text{ref}}_p)}{T(n_p)} \tag{7.3}$$

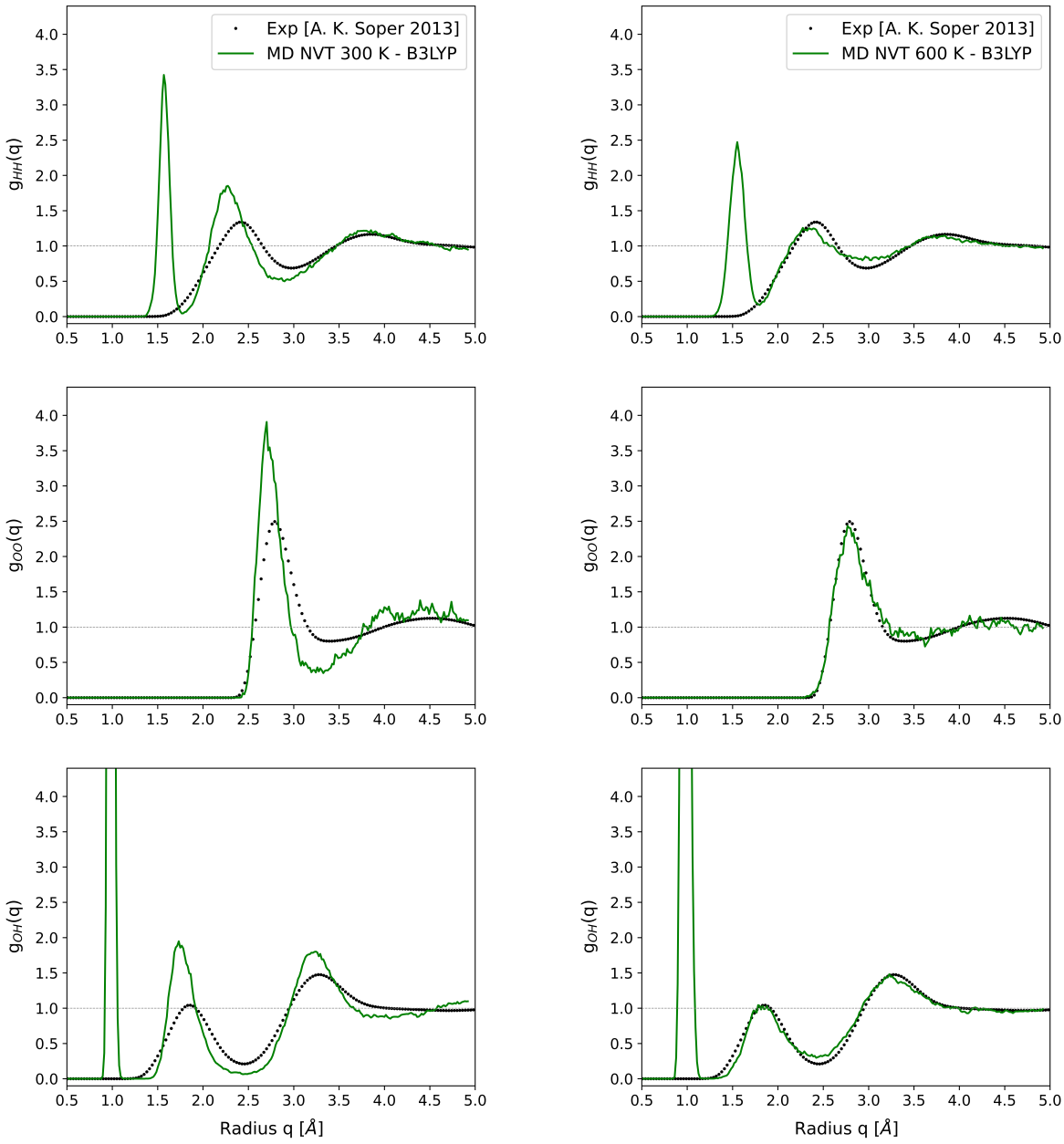

Figure 7.2: Pair correlation functions of the periodic $(H_2O)_{32}$ cubic system obtained from a post-processing of 1 ps NVT (Nosé-Hoover) molecular dynamics trajectory at 300 K (green lines, left panel) and 600 K (green lines, right panel) using B3LYP functionals, together with the experimental counterpart (black dots) by A. K. Soper.[101] See Table 7.1 for details about molecular dynamics simulations.

in which $T(n_p^{\text{ref}})$ is the total computational time required for a molecular dynamics calculation on a reference number of cores n_p^{ref} (chosen equal to the lower number of processors used for each system), while $T(n_p)$ is the time obtained for the calculation on n_p processors. The plots representing the speedup as a function of the number of processors n_p for PBE and B3LYP functionals, shown in Figure 7.4, exhibit a trend very close to the ideal linear scaling, represented by the diagonal line, for all the four systems. Furthermore, the total computational time (SCF + GRAD) for all the simulations is reported as a function of the number of processors (strong scaling) and as a function of the number of atoms (weak scaling) in Figure 7.5, for both the PBE and B3LYP functionals. As shown in these figures, the molecular dynamics module is quite scalable as the system size increases, and the time to the solution can be kept relatively constant for systems as large as 96 atoms, provided that a consistent (i.e. fixed ζ ratio) amount of computational resources are available. Moreover, the scalability of the B3LYP functional is comparable to that obtained using PBE and the prefactor is about only double, thus demonstrating the possibility of routinely performing molecular dynamics with hybrid functionals.

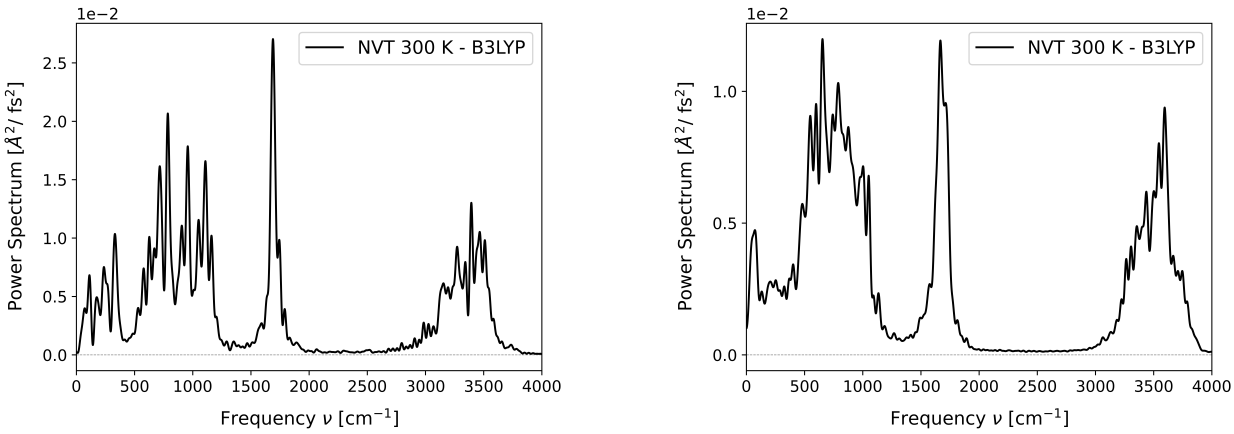

Figure 7.3: Power spectrum obtained as a post-processing of 1 ps NVT (Nosé-Hoover) molecular dynamics trajectory at 300 K using B3LYP functional for the P-ice crystalline structure (left panel) and the periodic $(H_2O)_{32}$ cubic system (right panel). Details about the molecular dynamics simulations are reported in Table 7.1.

				PBE						B3LYP					
n_{at}	n_{ao}	n_p	ζ	$\langle t_{\mathrm{SCF}} \rangle$	$\langle t_{\mathrm{GRAD}} \rangle$	$\eta^{\mathrm{SCF}}_{\mathrm{strong}}$	$\eta^{\mathrm{GRAD}}_{\mathrm{strong}}$	$\eta^{\mathrm{SCF}}_{\mathrm{weak}}$	$\eta^{\mathrm{GRAD}}_{\mathrm{weak}}$	$\langle t_{\mathrm{SCF}} \rangle$	$\langle t_{\mathrm{GRAD}} \rangle$	$\eta^{\mathrm{SCF}}_{\mathrm{strong}}$	$\eta^{\mathrm{GRAD}}_{\mathrm{strong}}$	$\eta^{\mathrm{SCF}}_{\mathrm{weak}}$	$\eta^{\mathrm{GRAD}}_{\mathrm{weak}}$
24	192	4	1/6	334.16	199.63	1.0000	1.0000	1.0000	1.0000	7277.16	3194.07	1.0000	1.0000	1.0000	1.0000
24	192	8	1/3	169.04	100.87	0.9884	0.9896	1.0000	1.0000	4374.50	1883.98	0.8318	0.8477	1.0000	1.0000
24	192	24	1	62.33	35.46	0.8935	0.9382	1.0000	1.0000	1585.11	692.50	0.7652	0.7687	1.0000	1.0000
24	192	48	2	36.29	18.52	0.7674	0.8985	1.0000	1.0000	756.08	334.85	0.8021	0.7949	1.0000	1.0000
48	384	8	1/6	500.76	223.65	1.0000	1.0000	0.6673	0.8926	7933.75	3156.85	1.0000	1.0000	0.6673	0.8926
48	384	16	1/3	256.25	115.14	0.9771	0.9712	0.6597	0.8761	4713.74	1868.09	0.8416	0.8449	0.6597	0.8761
48	384	48	1	99.41	40.80	0.8396	0.9136	0.6270	0.8692	1730.72	685.98	0.7640	0.7670	0.6270	0.8692
48	384	96	2	56.70	20.61	0.7360	0.9044	0.6400	0.8985	879.19	346.93	0.7520	0.7583	0.6400	0.8985
72	576	12	1/6	602.84	228.39	1.0000	1.0000	0.5543	0.8741	8294.50	3148.30	1.0000	1.0000	0.5543	0.8741
72	576	24	1/3	319.08	118.44	0.9447	0.9641	0.5298	0.8516	4971.23	1872.38	0.8342	0.8407	0.5298	0.8516
72	576	72	1	124.88	42.48	0.8046	0.8960	0.4991	0.8348	1838.99	686.72	0.7517	0.7641	0.4991	0.8348
72	576	144	2	80.00	24.01	0.6279	0.7926	0.4536	0.7711	978.78	371.28	0.7062	0.7066	0.4536	0.7711
96	768	16	1/6	827.07	257.65	1.0000	1.0000	0.4040	0.7748	9625.43	3287.34	1.0000	1.0000	0.4040	0.7748
96	768	32	1/3	452.93	132.64	0.9130	0.9713	0.3732	0.7605	5606.58	1975.27	0.8584	0.8321	0.3732	0.7605
96	768	96	1	208.22	48.19	0.6620	0.8911	0.2994	0.7359	1938.37	660.57	0.8276	0.8294	0.2994	0.7359
96	768	192	2	121.02	27.67	0.5695	0.7760	0.2998	0.6692	1066.49	365.37	0.7521	0.7498	0.2998	0.6692

Table 7.3: Number of atoms n_{at} and atomic orbitals n_{ao} in the unit cell, number of processors n_p and scaling factor $\zeta = n_p/n_{at}$, together with the computational timings (in units of s/step, obtained as an average over 200 molecular dynamics steps) for the calculation of the converged energy (t_{SCF}) and the related nuclear forces (t_{GRAD}), for a NVE BOMD performed using two different DFT functionals. The corresponding strong and weak scaling efficiencies for the SCF module ($\eta^{\mathrm{SCF}}_{\mathrm{strong}}$, $\eta^{\mathrm{SCF}}_{\mathrm{weak}}$) and for the GRAD module ($\eta^{\mathrm{GRAD}}_{\mathrm{strong}}$, $\eta^{\mathrm{GRAD}}_{\mathrm{weak}}$), which are computed using formulae (7.1) and (7.2), respectively, are also reported.

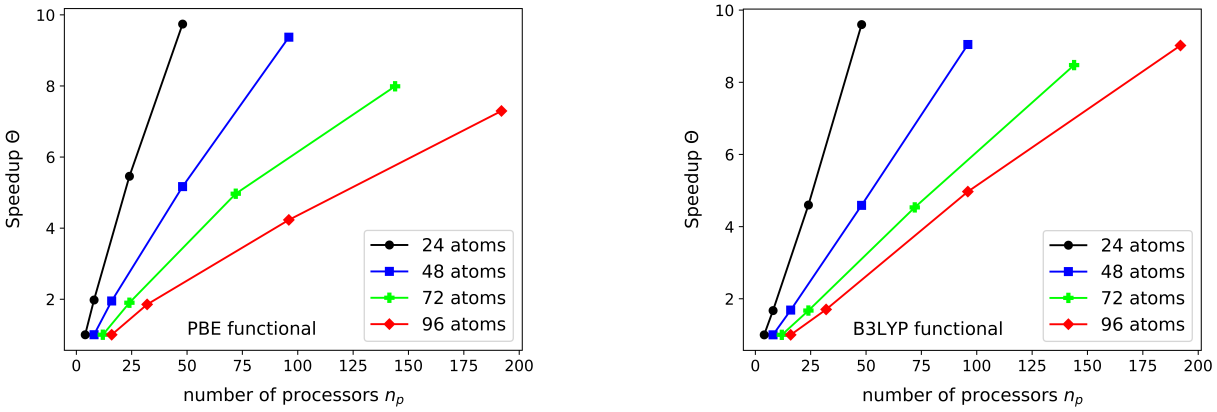

Figure 7.4: Speedup of the molecular dynamics module (computed with respect to the total computational time over 200 steps) as a function of the number of processors n_p used to perform the calculation, with PBE (left panel) and B3LYP (right panel) functionals.

7.2 Computational details

The two model systems consisted of a crystalline ice with $Pna2_1$ space group (P-ice) and a liquid-like system $(H_2O)_{32}$ with 32 water molecules, treated under periodic boundary conditions in the framework of density functional theory (DFT). Both the crystalline P-ice and the liquid-like $(H_2O)_{32}$ system are described using the gradient-corrected Perdew-Burke-Ernzerhof (PBE) functional,[93] and two hybrid functionals such as B3LYP[102] and PBE0.[103] Moreover, in conjunction with PBE and B3LYP functionals, a London-type pairwise empirical correction (D3) to the energy for dispersive interactions is also considered, in order to include long-range dispersion contributions to the computed ab initio total energy and gradients, as proposed by Grimme[94] and as implemented in CRYSTAL, including three-body dispersion contributions with fast analytical gradients.[95]
The all-electron basis sets introduced by Gatti et al.,[104] consisting of contracted Gaussian-type atomic orbital functions (AOs), are used for both the hydrogen and the oxygen atoms. The number of orbitals used for the description of each hydrogen and oxygen atom is given respectively by 5 and 14 AOs, with a resultant number of 192 and 768 AOs for the P-ice crystal and the $(H_2O)_{32}$ liquid-like system, respectively.
The DFT exchange-correlation contribution was evaluated by numerical integration over the unit cell volume, using a pruned grid,[105, 106] consisting (for both systems) of 99 radial points, with a maximum number of 1454 angular points in the region relevant for chemical bonding.
The diagonalization of the Hamiltonian matrix and the integration over the reciprocal space is carried out using the Monkhorst-Pack mesh,[107] consisting in a grid of 112 and 36 $\boldsymbol{k}$-points defined in the irreducible part of the first Brillouin Zone (IBZ) for the P-ice crystal and the $(H_2O)_{32}$ liquid-like system, respectively. The Coulomb and exchange series, summed in direct space, are truncated using overlap criteria, with thresholds of [10,10,20,20,20] for both the P-ice and $(H_2O)_{32}$ systems with all the functionals correspondingly used. The threshold for the self-consistent field algorithm convergence on the total energy per unit cell is chosen equal to 10^{-8} Ha for all the considered simulations.
Molecular dynamics calculations are performed using the previous listed computational parameters, both in the NVE and NVT ensemble. The initial nuclear configuration for all the molecular dynamics simulations on the P-ice crystal is obtained though a structural optimization using the corresponding exchange-correlation functional, the fully relaxed cell parameters obtained are reported in Appendix D, while the related initial nuclear positions are listed in the Supporting Information of Ref. [92]. The initial nuclear positions of the $(H_2O)_{32}$ liquid-like system are specified (together with the cell parameters) in the Supporting Information of Ref. [92]. The initial nuclear velocities are determined by the input temperature, that is given by $T = 300$ K for all the simulations on both of the two systems in the microcanonical ensemble, while it is equal to $T = 300$ K and $T = 600$ K for the two molecular dynamics simulations performed in the canonical ensemble using B3LYP functional. The total time of each molecular dynamics run is 1 ps, with a time step equal to 0.15 fs and 0.25 fs, respectively for P-ice and $(H_2O)_{32}$ periodic systems, for both the ensembles considered.

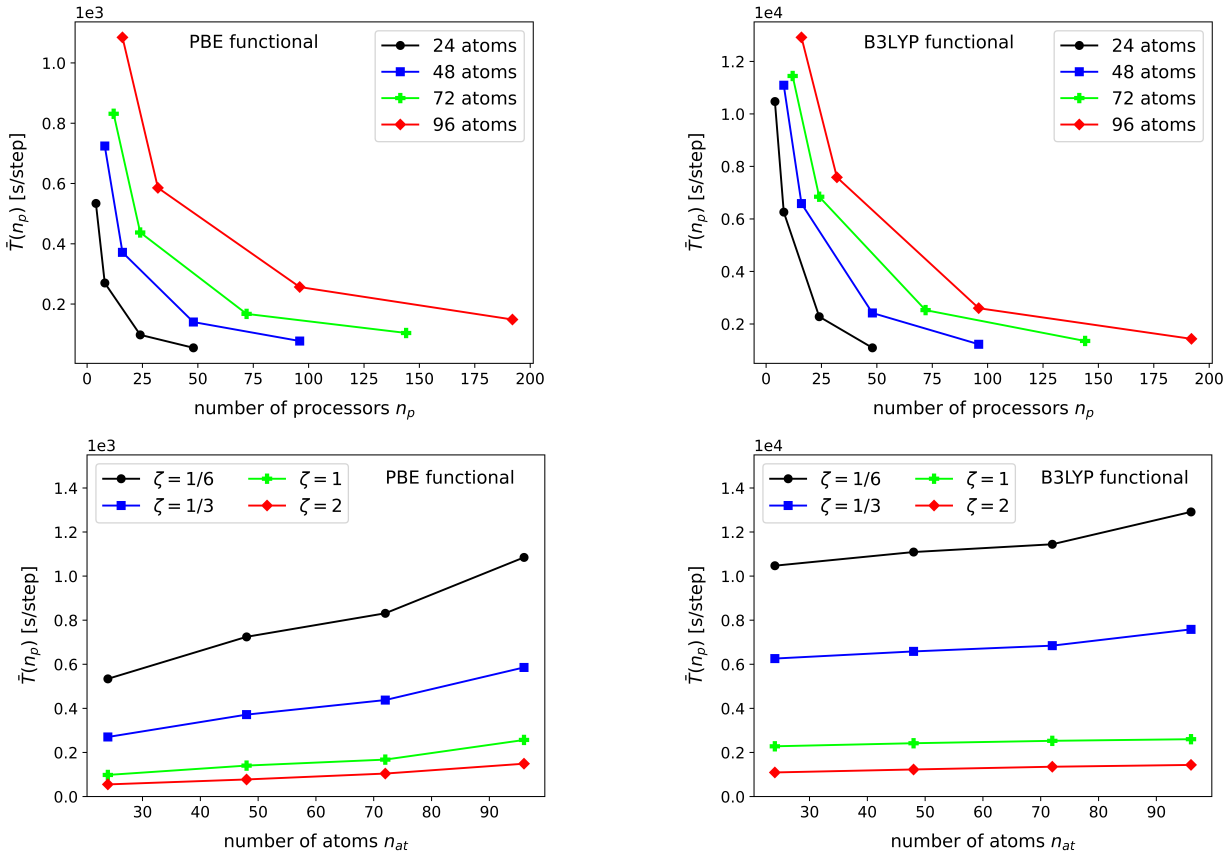

Figure 7.5: Average total computational time per step as a function of the number of processors n_p, for all the systems considered ($n_{at} = 24, 48, 72, 96$), using PBE (top left panel) and B3LYP (top right panel) functionals. Average total computational time per step as a function of the number of atoms n_{at}, for a given ratio $\zeta = n_p/n_{at}$, using PBE (down left panel) and B3LYP (down right panel) functionals.

A comment apart has to be reserved for the molecular dynamics simulation performed on the P-ice crystal to study the scaling efficiency of the algorithm (see Section 7.1). In this case, the initial nuclear positions are the ones obtained though a PBE-D3 fully structural optimization (see Appendix D and the Supporting Information of Ref. [92]), and the supercells with increasing size are derived from this initial configuration, as described in Section 7.1. Molecular dynamics simulations are then performed in the microcanonical (NVE) ensemble, using the PBE functional, with an initial temperature of $T = 300$ K and a time step of 0.15 fs, for a total run time of 30.0 fs (200 steps).
Geometry optimizations are performed using analytical gradients with respect to nuclear coordinates, within a quasi-Newton algorithm combined with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formula[108, 109, 110, 111, 112] The computational parameters (DFT functional, basis set, Coulomb and exchange integrals thresholds, reciprocal space sampling and energy thresholds for self-consistent cycle) used for structural optimization are the same as previously reported, for both systems. Convergence criteria for BFGS structural optimizations are based on the root mean square and absolute value of the largest component of both the estimated displacements and the gradients of energy functional with respect to the nuclear positions, computed in normal coordinates. In this framework, CRYSTAL23 default convergence thresholds and minimization parameters have been adopted.[113, 114]

Chapter 8

Fast Inertial Relaxation Engine (FIRE)

Structural minimization of molecular and solid state atomic systems is a fundamental step in the study and the comprehension of condensed matter properties at a microscopic level, becoming one of the most common tasks in computational solid state physics, chemistry, materials science and biology. The problem of structural optimization of atomic systems is a multi-dimensional minimization problem, that corresponds to finding the atomic structures nearest to a local or a global minimum of the potential energy surface, starting from a given initial configuration. The resolution of such a problem can be quite complicated from a mathematical point of view and its implementation in the framework of *ab initio* quantum-mechanical codes can lead to algorithms with high computational costs. Therefore, every effort that goes in the direction of a possible simplification of the methodology to perform structural optimization could provide novel robust and efficient minimization methods, that could in principle be applied on a large variety of atomic systems. In this framework, a novel minimization method for structural optimization based on Molecular Dynamics concepts, namely, the Fast Inertial Relaxation Engine (FIRE), is introduced in this Chapter. This algorithm relies on molecular dynamics concepts, introducing an equation of motion that associates to each atomic nucleus a velocity and a force, bringing the system to a minimum in the potential energy surface.
In order to understand the main differences between this novel method and the quasi-Newton algorithms which are the most used structural optimization schemes in the field of *ab initio* simulations, a brief review of the two well-known quasi-Newton minimization schemes for multidimensional functions, namely, the Conjugate Gradient (CG) and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods, is outlined in Section 8.1. Then, in Section 8.2, the (FIRE) structural minimization method is introduced, the theoretical details about the algorithm are described and discussed, the power and the limits of this novel method are investigated, also in comparison with the other two minimization scheme previously illustrated. Finally, in Section 8.3, the implementation of the FIRE method in the CRYSTAL code is delineated. The efficiency and reliability of the algorithm is tested on systems with different dimensionalities and type of bonding and the results are reported in Section 8.4.

8.1 A review on quasi-Newton structural minimization methods

The minimization of a function of one variable is a well known procedure. If we start at a point $\mathbf{P}$ in N-dimensional space, and proceed from there in some vector direction $\mathbf{n}$, then any function of N variables $f(\mathbf{P})$ can be minimized along the line $\mathbf{n}$ by the one-dimensional methods. One can dream up various multidimensional minimization methods that consist of sequences of such line minimizations. Different methods will differ only by how, at each stage, they choose the next direction $\mathbf{n}$ to try. All such methods presume the existence of a black box sub-algorithm, which is a line minimization algorithm, whose definition can be taken as follows.

Line minimization algorithm: Given as input the vectors $\mathbf{P}$ and $\mathbf{n}$, and the function f, find the scalar λ that minimizes $f(\mathbf{P} + \lambda\mathbf{n})$. Replace $\mathbf{P}$ by $\mathbf{P} + \lambda\mathbf{n}$. Replace $\mathbf{n}$ by $\lambda\mathbf{n}$. Done.

All the minimization methods in the following two sections fall under this general schema of successive line minimizations. It is not necessary to specify whether the line minimization algorithm uses gradient information or not. That choice is up to the method used, and its optimization depends on the particular

function considered.
But what if, in some application, calculation of the gradient is out of the question. You might first think of this simple method: take the unit vectors $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N$ as a set of directions. Using the line minimization, move along the first direction to its minimum, then from there along the second direction to its minimum, and so on, cycling through the whole set of directions as many times as necessary, until the function stops decreasing. This simple method is actually not too bad for many functions. Even more interesting is why it is bad, i.e. very inefficient, for some other functions. Consider a function of two dimensions whose contour map (level lines) happens to define a long, narrow valley at some angle to the coordinate basis vectors. Then the only way down the length of the valley going along the basis vectors at each stage is by a series of many tiny steps. More generally, in N dimensions, if the function second derivatives are much larger in magnitude in some directions than in others, then many cycles through all N basis vectors will be required in order to get anywhere. This condition is not all unusual. Obviously what we need is a better set of directions than the $\mathbf{e}_i$. All direction set methods consist of prescriptions for updating the set of directions as the method proceeds, attempting to come up with a set which either (i) includes some very good directions that will take us far along narrow valleys, or else (more subtly) (ii) includes some number of non-interfering directions with the special property that minimization along one is not spoiled by subsequent minimization along another, so that interminable cycling through the set of directions can be avoided.

8.1.1 Conjugate Gradient method

This concept of non-interfering directions, more conventionally called conjugate directions, is worth making mathematically explicit. First, note that if we minimize a function along some direction $\mathbf{u}$, then the gradient of the function must be perpendicular to $\mathbf{u}$ at the line minimum; if not, then there would still be a nonzero directional derivative along $\mathbf{u}$. Take some particular point $\mathbf{P}$ as the origin of the coordinate system with coordinates $\mathbf{x}$. Consider the case where it is possible to compute, at a given N dimensional point $\mathbf{P}$, not just the value of a function $f(\mathbf{P})$ but also the gradient $\nabla f(\mathbf{P})$ of the function in that point. Then any function f can be approximated by its Taylor series:

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + ... \tag{8.1}$$

Suppose that the function f can be roughly approximated by a quadratic form, which we can write, in matrix notation, as

$$f(\mathbf{x}) \approx f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j \approx c - \mathbf{b} \cdot \mathbf{x} + \frac{1}{2}\, \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x} \tag{8.2}$$

where

$$c \equiv f(\mathbf{P}) \qquad \mathbf{b} \equiv -\nabla f|_{\mathbf{P}} \qquad [\mathbf{A}]_{ij} \equiv \left. \frac{\partial^2 f}{\partial x_i \partial x_j} \right|_{\mathbf{P}} \tag{8.3}$$

The matrix $\mathbf{A}$ whose components are the second partial derivative of the function is called the Hessian matrix of the function at the point $\mathbf{P}$. In the approximation (8.2), the gradient of the function f is easily calculated as

$$\nabla f(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} - \mathbf{b} \tag{8.4}$$

This implies that the gradient will vanish (the function will be at an extremum) at a value of $\mathbf{x}$ obtained by solving the system of equations

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{8.5}$$

To compute how does the gradient ∇f change as we move along some direction, the following calculation can be performed

$$\delta(\nabla f) = \mathbf{A} \cdot (\delta \mathbf{x}) \tag{8.6}$$

Suppose that we have moved along some direction $\mathbf{u}$ to a minimum, and now suppose to move along some new direction $\mathbf{v}$. The condition that the motion along $\mathbf{v}$ not spoil our minimization along the

direction $\mathbf{u}$ is just that the gradient stay perpendicular to $\mathbf{u}$, i.e. that the change in the gradient be perpendicular to $\mathbf{u}$. By equation (8.6) this is expressed by the formula

$$0 = \mathbf{u} \cdot \delta(\nabla f) = \mathbf{u} \cdot \mathbf{A} \cdot (\delta \mathbf{u}) = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} \tag{8.7}$$

When the condition (8.7) holds for two vectors $\mathbf{u}$ and $\mathbf{v}$, they are said to be conjugate. When the relation holds pairwise for all members of a set of vectors, they are said to be a conjugate set. If a successive line minimization of a function along a conjugate set of directions is done, then it is not necessary to redo any of those directions (unless, of course, things are spoiled by minimizing along a direction that they are not conjugate to). A triumph for a direction set method is to come up with a set of N linearly independent, mutually conjugate directions. Then, one step along each of the N line minimizations will bring exactly at the minimum of a quadratic form like (8.2). For functions f that are not exactly quadratic forms, it won't be exactly at the minimum; but repeated cycles of N line minimizations will in due course converge quadratically to the minimum.

A rough counting argument will show how advantageous it is to use the gradient information: suppose that the function f is roughly approximated as a quadratic form as in (8.2). Then the number of unknown parameters in f is equal to the number of free parameters in the vector $\mathbf{b}$ and in the matrix $\mathbf{A}$, which is $\frac{1}{2}N(N+1)$, that is of order N^2. Changing any one of these parameters can move the location of the minimum. Therefore, we should not expect to be able to find the minimum until an equivalent information content has been collected, of order N^2 numbers. In the direction set method based on conjugate line minimizations described below, we collected the necessary information by making on the order of N^2 separate line minimizations, each requiring a few (but sometimes a big few!) function evaluations. Now, each evaluation of the gradient will bring us N new components of information. If we use them wisely, we should need to make only of order N separate line minimizations. That is in fact the case for the algorithms in this section and the next, namely, for the Conjugate Gradient (CG) and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods.

A factor of N improvement in computational speed is not necessarily implied. As a rough estimate, we might imagine that the calculation of *each component* of the gradient takes about as long as evaluating the function itself. In that case there will be of order N^2 equivalent function evaluations both with and without gradient information. Even if the advantage is not of order N, however, it is nevertheless quite substantial: (i) each calculated component of the gradient will typically save not just one function evaluation, but a number of them, equivalent to say, a whole line minimization. (ii) There is often a high degree of redundancy in the formulas for the various components of a function's gradient; when this is so, especially when there is also redundancy with the calculation of the function, then the calculation of the gradient may cost significantly less than N function evaluations.

However, not any reasonable way of incorporating gradient information is as good as any other. For example, think about the following not very good algorithm, the steepest descent method:

Steepest Descent: Start at a point $\mathbf{P}_0$. As many times as needed, move from point $\mathbf{P}_i$ to the point $\mathbf{P}_{i+1}$ by minimizing along the line from $\mathbf{P}_i$ in the direction of the local downhill gradient $-\nabla f(\mathbf{P}_i)$.

The problem with the steepest descent method is that the method will perform many small steps in going down a long, narrow valley, even if the valley is a perfect quadratic form. You might have hoped that, say in two dimensions, your first step would take you to the valley floor, the second step directly down the long axis; but remember that the new gradient at the minimum point of any line minimization is perpendicular to the direction just traversed. Therefore, with the steepest descent method, you must make a right angle turn, which does not, in general, take you to the minimum (see Figure 8.1).

We really want a way of proceeding not down the new gradient, but rather in a direction that is somehow constructed to be conjugate to the old gradient, and, insofar as possible, to all previous directions traversed. Methods that accomplish this construction are called *conjugate gradient methods*.

First of all, we have to build a method to find conjugate directions. We can perform this task thinking that the conjugate gradient method is also studied as a technique for solving linear algebraic equations by minimizing a quadratic form.[115] That formalism can also be applied to the problem of minimizing a function approximated by the quadratic form (8.2). Recall that, starting with an arbitrary initial vector $\mathbf{g}_0$, and letting $\mathbf{h}_0 = \mathbf{g}_0$, the conjugate gradient method constructs two sequences of vectors from the recurrence formula

$$\mathbf{g}_{i+1} = \mathbf{g}_i - \lambda_i \mathbf{A} \cdot \mathbf{h}_i \qquad \mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i \qquad i = 0, 1, 2, \ldots \tag{8.8}$$

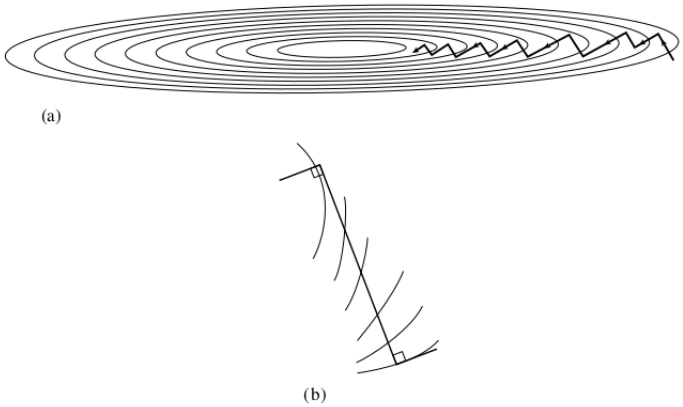

Figure 8.1: (a) Steepest descent method in a long, narrow valley. In this case, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

The vectors satisfy the orthogonality and conjugacy conditions

$$\mathbf{g}_i \cdot \mathbf{g}_j = 0 \qquad \mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_j = 0 \qquad \mathbf{g}_i \cdot \mathbf{h}_j = 0 \qquad \text{with } j < i \tag{8.9}$$

The scalars λ_i and γ_i are given by

$$\lambda_i = \frac{\mathbf{g}_i \cdot \mathbf{g}_i}{\mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i} = \frac{\mathbf{g}_i \cdot \mathbf{h}_i}{\mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i} \tag{8.10}$$

$$\gamma_i = \frac{\mathbf{g}_{i+1} \cdot \mathbf{g}_{i+1}}{\mathbf{g}_i \cdot \mathbf{g}_i} \tag{8.11}$$

A self-contained derivation of these results in the context of function minimization is given by Polak.[116] Now suppose that the Hessian matrix $\mathbf{A}$ in equation (8.2) is known. Then we could use the construction (8.8) to find successively conjugate directions $\mathbf{h}_i$ along which to line-minimize. After N such, we would efficiently have arrived at the minimum of the quadratic form. But we don't know the matrix $\mathbf{A}$.

Here is a remarkable theorem to save the day: Suppose that $\mathbf{g}_i = -\nabla f(\mathbf{P}_i)$ for some point $\mathbf{P}_i$, where the function f is of the form (8.2). Suppose that we proceed from $\mathbf{P}_i$ along the direction $\mathbf{h}_i$ to the local minimum of f located at some point $\mathbf{P}_{i+1}$ and then set $\mathbf{g}_{i+1} = -\nabla f(\mathbf{P}_{i+1})$. Then, this $\mathbf{g}_{i+1}$ is the same vector as would have been constructed by equation (8.8). (And we have constructed it without knowledge of the matrix $\mathbf{A}$)

Proof: By equation (8.4) we have

$$\mathbf{g}_i = -\nabla f(\mathbf{P}_i) = -\mathbf{A} \cdot \mathbf{P}_i + \mathbf{b} \tag{8.12}$$

$$\mathbf{g}_{i+1} = -\nabla f(\mathbf{P}_{i+1}) = -\mathbf{A} \cdot (\mathbf{P}_i + \lambda \mathbf{h}_i) + \mathbf{b} = (-\mathbf{A} \cdot \mathbf{P}_i + \mathbf{b}) - \lambda \mathbf{A} \cdot \mathbf{h}_i = \mathbf{g}_i - \lambda \mathbf{A} \cdot \mathbf{h}_i \tag{8.13}$$

with λ chosen to take us to the line minimum. But at the line minimum

$$\mathbf{h}_i \cdot \nabla f(\mathbf{P}_{i+1}) = -\mathbf{h}_i \cdot \mathbf{g}_{i+1} = 0 \tag{8.14}$$

This latter equation is easily combined with (8.13) to solve for λ:

$$\mathbf{h}_i \cdot \mathbf{g}_{i+1} = \mathbf{h}_i \cdot \mathbf{g}_i - \lambda \mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i = 0 \qquad \rightarrow \qquad \lambda = \frac{\mathbf{g}_i \cdot \mathbf{h}_i}{\mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i} \tag{8.15}$$

This result is exactly the expression (8.10). But with this value of λ, the vector $\mathbf{g}_{i+1}$ in (8.13) becomes the same as (8.8), q.e.d.

We have, then, the basis of an algorithm that requires neither knowledge of the Hessian matrix $\mathbf{A}$, nor even the storage necessary to store such a matrix. A sequence of directions $\mathbf{h}_i$ is constructed, using only line minimizations, evaluations of the gradient vector, and an auxiliary vector to store the latest in the

sequence of the vectors $\mathbf{g}_i$. The algorithm described so far is the original Fletcher-Reeves version of the conjugate gradient algorithm. Later, Polak and Ribiere introduced one tiny, but sometimes significant, change. They proposed using the form

$$\gamma_i = \frac{(\mathbf{g}_{i+1} - \mathbf{g}_i) \cdot \mathbf{g}_{i+1}}{\mathbf{g}_i \cdot \mathbf{g}_i} \tag{8.16}$$

instead of equation (8.11). This expression seems to be exactly equivalent to (8.11) because of the orthogonality conditions (8.9). Indeed, the two expressions are exactly equal for exact quadratic forms. In the real world, however, your function is not exactly a quadratic form. Arriving at the supposed minimum of the quadratic form, you may still need to proceed for another set of iterations. There is some evidence[117] that the Polak-Ribiere formula accomplishes the transition to further iterations more gracefully: when it runs out of steam, it tends to reset $\mathbf{h}$ to be down the local gradient, which is equivalent to beginning the conjugate-gradient procedure anew.

8.1.2 Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) minimization scheme is one of the variable metric methods in multidimensions. The goal of variable metric methods, which are sometimes called quasi-Newton methods, is not different from the goal of conjugate gradient methods: to accumulate information from successive line minimizations so that N such line minimizations lead to the exact minimum of a quadratic form in N dimensions. In that case, the method will also be quadratically convergent for more general smooth functions.
Both variable metric and conjugate gradient methods require that you are able to compute the gradient of the involved function, or first partial derivatives, at arbitrary points. The variable metric approach differs from the conjugate gradient in the way that it stores and updates the information that is accumulated. Instead of requiring intermediate storage on the order of N, the number of dimensions, it requires a matrix of size $N \times N$. Generally, for any moderate N, this is an entirely trivial disadvantage.
Variable metric methods come in two main flavors. One is the *Davidon-Fletcher-Powell* (DFP) algorithm (sometimes referred to as simply Fletcher-Powell). The other goes by the name *Broyden-Fletcher-Goldfarb-Shanno* (BFGS). The BFGS and DFP schemes differ only in details of their roundoff error, convergence tolerances, and similar dirty issues which are outside of our scope. However, it has become generally recognized that, empirically, the BFGS scheme is superior in these details. In the following, the BFGS method will be described.

As before, imagine that the arbitrary function $f(\mathbf{x})$ can be locally approximated by the quadratic form of equation (8.2). However, no information about the values of the quadratic form parameters $\mathbf{A}$ and $\mathbf{b}$ are given, except insofar as we can glean such information from the function evaluations and line minimizations.
The basic idea of the variable metric method is to build up, iteratively, a good approximation to the inverse Hessian matrix $\mathbf{A}^{-1}$, that is, to construct a sequence of matrices $\mathbf{H}_i$ with the property

$$\lim_{i \to \infty} \mathbf{H}_i = \mathbf{A}^{-1} \tag{8.17}$$

Even better if the limit is achieved after N iterations instead of ∞.
The reason that variable metric methods are sometimes called quasi-Newton methods can now be explained. Consider finding a minimum by using Newton method to search for a zero of the gradient of the function. Near the current point $\mathbf{x}_i$, we have, up to second order

$$f(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i) \tag{8.18}$$

and the gradient is

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_i) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i) \tag{8.19}$$

In Newton method the gradient of $f(\mathbf{x})$ is set to zero, $\nabla f(\mathbf{x}) = 0$, to determine the next iteration point:

$$\mathbf{x} - \mathbf{x}_i = -\mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_i) \tag{8.20}$$

The left-hand side in (8.20) is the finite step we need take to get to the exact minimum; the right-hand side is known once we have accumulated an accurate $\mathbf{H} \approx \mathbf{A}^{-1}$. The *quasi* in quasi-Newton is because the actual Hessian matrix is not used in the method, but instead the current approximation of it is used. This is often better than using the true Hessian. This paradoxical result can be understood by considering the descent directions of f at $\mathbf{x}_i$. These are the directions $\mathbf{p}$ along which f decreases, so that $\nabla f \cdot \mathbf{p} < 0$. For the Newton direction (8.20) to be a descent direction, it must holds

$$\nabla f(\mathbf{x}_i) \cdot (\mathbf{x} - \mathbf{x}_i) = -(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i) < 0 \tag{8.21}$$

that is, $\mathbf{A}$ must be positive definite. In general, far from a minimum, we have no guarantee that the Hessian is positive definite. Taking the actual Newton step with the real Hessian can move us to points where the function is increasing in value. The idea behind quasi-Newton methods is to start with a positive definite, symmetric approximation to $\mathbf{A}$ (usually the unit matrix) and build up the approximating $\mathbf{H}_i$ matrices in such a way that the matrix $\mathbf{H}_i$ remains positive definite and symmetric. Far from the minimum, this guarantees that we always move in a downhill direction. Close to the minimum, the updating formula approaches the true Hessian and we enjoy the quadratic convergence of Newton method. When we are not close enough to the minimum, taking the full Newton step $\mathbf{p}$ even with a positive definite $\mathbf{A}$ need not decrease the function; we may move too far for the quadratic approximation to be valid. All we are guaranteed is that *initially* f decreases as we move in the Newton direction. A backtracking strategy can be used to choose a step along the direction of the Newton step $\mathbf{p}$, but not necessarily all the way.
The DFP algorithm for taking $\mathbf{H}_i$ into $\mathbf{H}_{i+1}$ is not derived rigorously here; reminding to Ref. [118] for clear derivations. Following Brodlie (see Ref. [117]), we will give the following heuristic motivation of the procedure. Following equation (8.20), the following vector can be defined:

$$\mathbf{x} - \mathbf{x}_{i+1} = -\mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_{i+1}) \tag{8.22}$$

and subtracting the previous equation to the equation (8.20) results

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{A}^{-1} \cdot (\nabla f_{i+1} - \nabla f_i) \tag{8.23}$$

where $\nabla f_j \equiv \nabla f(\mathbf{x}_j)$. Having made the step from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$, we might reasonably want to require that the new approximation $\mathbf{H}_{i+1}$ satisfies (8.23) as if it were actually $\mathbf{A}^{-1}$, that is,

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{H}_{i+1} \cdot (\nabla f_{i+1} - \nabla f_i) \tag{8.24}$$

We might also imagine that the updating formula should be of the form $\mathbf{H}_{i+1} = \mathbf{H}_i +$ correction. Now, the question regards what objects are around out of which to construct a correction term. Most notable are the two vectors $(\mathbf{x}_{i+1} - \mathbf{x}_i)$ and $(\nabla f_{i+1} - \nabla f_i)$, as well as $\mathbf{H}_i$. There are not infinitely many natural ways of making a matrix out of these objects, especially if (8.24) must hold! One such way, the DFP updating formula, is:

$$\mathbf{H}_{i+1}^{\mathrm{DFP}} = \mathbf{H}_i + \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i+1} - \mathbf{x}_i)}{(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\nabla f_{i+1} - \nabla f_i)} - \frac{[\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)] \otimes [\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)]}{(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)} \tag{8.25}$$

where the symbol $\otimes$ denotes the outer or direct product of two vectors, which defines a matrix so that the ij component of $\mathbf{u} \otimes \mathbf{v}$ is $u_i v_j$.
The BFGS updating formula is exactly the same, but with one additional term, that is:

$$\mathbf{H}_{i+1}^{\mathrm{BFGS}} = \mathbf{H}_{i+1}^{\mathrm{DFP}} + [(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)]\ \mathbf{u} \otimes \mathbf{u} \tag{8.26}$$

where $\mathbf{u}$ is defined as the vector

$$\mathbf{u} \equiv \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i)}{(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\nabla f_{i+1} - \nabla f_i)} - \frac{\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)}{(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)} \tag{8.27}$$

You will have to take on faith (or else consult Ref. [118] for details of) the deep result that equation (8.25), with or without the additional term of equation (8.26), does in fact converge to $\mathbf{A}^{-1}$ in N steps, if the function f is a quadratic form.

8.1.3 Structural optimization methods in the CRYSTAL code

In the CRYSTAL code, geometry optimization calculations are performed using the nuclear forces computed through the Hellmann-Feynman theorem[30] as analytical gradient with respect to atomic coordinates, within a quasi-Newton algorithm combined with two kinds of Hessian updating schemes: the Schlegel's (SH)[119] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formulae.[108, 109, 110, 111, 112] Convergence criteria for SH and BFGS structural optimizations are based on (i) the root mean square and (ii) the absolute value of the largest component of both the estimated displacements and the gradients of energy functional with respect to the nuclear positions. All these four quantities, which are used to check the convergence of the optimization process, are computed in normal coordinates.[120, 121]

8.2 The FIRE structural optimization method

The current state-of-the-art methods for structural optimization are mostly based on some approximate representation for the Hessian matrix to determine line search directions, as described in Section 8.1. Also variants of molecular dynamics (MD) concepts which systematically remove kinetic energy from the system are commonly applied for minimization purposes.[122, 123, 124] Interestingly, relaxation methods based on MD has been thought to be good for practical realization, but not very competitive with the aforementioned sophisticated algorithms, and for this reason they have often been introduced as by-products of secondary importance in regular articles,[122, 123, 124] not receiving the attention they deserve. However, in 2006 Erik Bitzek et al.[125] introduced a simple, yet powerful MD scheme for structural relaxation, which has been called Fast Inertial Relaxation Engine (FIRE). The FIRE algorithm is based on molecular dynamics concepts, with additional velocity modifications and adaptive time steps. In their work, E. Bitzek et al.[125, 126] arrived at the conclusion that, in atomistic simulations, pseudo-dynamics relaxation schemes often exhibit better performance and accuracy in finding local minima than line-search-based descent algorithms like steepest descent or conjugate gradient. Furthermore, FIRE has been proved to be significantly faster than standard implementations of the conjugate gradient method and often competitive with more sophisticated quasi-Newton schemes typically used in *ab initio* calculations.[125] Therefore the FIRE algorithm, in its basic[125] and improved[126] implementation, seems to be a promising structural optimization method. The advantages of FIRE approach with respect to the quasi-Newton algorithms are further discussed in Section 8.2.3. In the following, the basic theoretical framework on which the FIRE scheme relies is outlined.

8.2.1 The FIRE algorithm

In a system with N nuclei of mass m, whose positions is associated to the coordinates $\boldsymbol{x} = (x_1, x_2, ..., x_{3N})$, the Potential Energy Surface (PES) described by $E(\boldsymbol{x})$ is a map $E : \mathbb{R}^{3N} \to \mathbb{R}$ that assigns to each atomic configuration $\boldsymbol{x}$ a potential energy value $E(\boldsymbol{x})$. Finding a minimum of the multidimensional function $E(\boldsymbol{x})$ with respect to the atomic coordinates $\boldsymbol{x}$ corresponds to finding a minimum in the PES, that is a set of coordinates for the atomic systems representing the configuration $\boldsymbol{x}$ which minimizes the potential energy of the system at 0 K. The FIRE algorithm is based on the definition of an equation of motion for the nuclei with the form

$$\dot{\boldsymbol{v}}(t) = \frac{\mathbf{F}(\boldsymbol{x}(t))}{m} - \gamma(t)\|\boldsymbol{v}(t)\|\{\hat{\boldsymbol{v}}(t) - \mathbf{F}(\boldsymbol{x}(t))\} \tag{8.28}$$

where boldface quantities denote $3N$ vectors, hats indicate unit vectors, $\|\boldsymbol{v}(t)\|$ is the Euclidean norm of the velocity vector, and $\gamma(t)$ is a scalar function of time t. In particular, $\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t)$ is the vector containing the $3N$ velocity components along the (x, y, z) direction of the N nuclei at time t, $\hat{\boldsymbol{v}}(t)$ is the velocity versor, which can be written as

$$\hat{\boldsymbol{v}}(t) = \frac{\boldsymbol{v}(t)}{\|\boldsymbol{v}(t)\|} \tag{8.29}$$

i.e. it is a unit vector with the same direction of the velocity vector. $\mathbf{F}(\boldsymbol{x}(t))$ is the force vector, containing the $3N$ force components acting on the nuclei at time t for a given nuclear configuration $\boldsymbol{x}(t)$, and it is computed as the analytical gradient of the potential energy functional, following the Hellmann-Feynman theorem.[30]

The equation of motion (8.28) has to be intended as an equation which predicts the position and the velocity at a certain time t of the *entire* system of N nuclei in the PES. Obviously, the equation of motion modifies *each* position and velocity component of the N nuclei, bringing the atomic system from the configuration $\boldsymbol{x}(t) = (x_1(t), x_2(t), ..., x_{3N}(t))$ at time t to a configuration $\boldsymbol{x}(t') = (x_1(t'), x_2(t'), ..., x_{3N}(t'))$ at time $t' > t$.
The first term on the right hand side of equation (8.28) represents regular Newtonian dynamics. In order to understand the effect of the second term in (8.28), the physical meaning of the force vector has to be analyzed. Since the gradient of the energy functional $E(\boldsymbol{x}(t))$ points in the maximum energy direction in the PES, the nuclear forces computed as $\mathbf{F}(\boldsymbol{x}(t)) = -\nabla E(\boldsymbol{x}(t))$ points in the direction that minimizes the potential energy. Taking this interpretation into account, two consequences can be constructed:

(i) the effect of the second term in equation (8.28) is to reduce the angle between $\boldsymbol{v}(t)$ and $\mathbf{F}(\boldsymbol{x}(t))$, which is the direction of the steepest descent at $\boldsymbol{x}(t)$;

(ii) uphill and downhill motion of the system in the PES can be detected by evaluating the direction of the system trajectory on the potential energy surface, identified by the velocity vector $\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t)$, and computing the scalar product between this velocity and the nuclear forces $\mathbf{F}(\boldsymbol{x}(t))$. This quantity, called the power factor $P(t)$ and defined as

$$P(t) = \mathbf{F}(\boldsymbol{x}(t)) \cdot \boldsymbol{v}(t) \tag{8.30}$$

is greater than zero whenever the nuclear velocity points in a direction of energy minimization, i.e. whenever the angle between the nuclear force $\mathbf{F}(\boldsymbol{x}(t))$ and the velocity $\boldsymbol{v}(t)$ vectors is in the range $(0, \pi)$. Otherwise, if the power factor $P(t) = \mathbf{F}(\boldsymbol{x}(t)) \cdot \boldsymbol{v}(t) \leq 0$, the equation of motion brings the system in a direction opposite to the direction of the steepest descent identified by $\mathbf{F}(\boldsymbol{x}(t))$, i.e. the system is brought in a point $\boldsymbol{x}(t')$ with energy greater than the potential energy in the point $\boldsymbol{x}(t)$ at a previous time $t < t'$, so that this kind of motion has to be avoided if the final goal is to minimize the energy.

On the base of these considerations, the recommended strategy related to the equation of motion (8.28) is to introduce acceleration in a direction that is steeper than the current direction of motion, via the function $\gamma(t)$, if the power $P(t)$ is positive, otherwise, in order to avoid uphill trajectories, the algorithm simply stops the motion as soon as the power factor becomes negative, by setting all the velocities equal to zero ($\boldsymbol{v} = \mathbf{0}$). On the other hand, $\gamma(t)$ should not be too large, because the current velocities carry information about the reasonable average descent direction and energy scale.

In order to obtain an equation for the new positions and velocities at a certain time $t' > t$, knowing the values of these quantities at a previous time t, the differential equation (8.28), rewritten as

$$\dot{\boldsymbol{v}}(t) \equiv \frac{d\boldsymbol{v}(t)}{dt} = \frac{\mathbf{F}(\boldsymbol{x}(t))}{m} - \gamma(t)\|\boldsymbol{v}(t)\|\{\hat{\boldsymbol{v}}(t) - \hat{\mathbf{F}}(\boldsymbol{x}(t))\} \quad \text{with the initial condition } \boldsymbol{v}(t=0) = \mathbf{0} \tag{8.31}$$

has to be resolved. This is an ordinary differential equation, which, together with a specified value (called the initial condition) of the unknown function at a given point in the domain of the solution, defines an initial value problem. The simplest method for solving numerically an initial value problem is the explicit Euler method, which is a first order method consisting in the replacement of the temporal derivative by finite differences:

$$\begin{aligned}
\frac{d\boldsymbol{v}(t)}{dt} = \frac{\boldsymbol{v}(t+dt) - \boldsymbol{v}(t)}{dt} &= \frac{\mathbf{F}(\boldsymbol{x}(t))}{m} - \gamma(t)\|\boldsymbol{v}(t)\|\{\hat{\boldsymbol{v}}(t) - \hat{\mathbf{F}}(\boldsymbol{x}(t))\} \\
&= \frac{\mathbf{F}(\boldsymbol{x}(t))}{m} - \gamma(t)\|\boldsymbol{v}(t)\|\left\{\frac{\boldsymbol{v}(t)}{\|\boldsymbol{v}(t)\|} - \frac{\mathbf{F}(\boldsymbol{x}(t))}{\|\mathbf{F}(\boldsymbol{x}(t))\|}\right\} \\
&= \frac{\mathbf{F}(\boldsymbol{x}(t))}{m} - \gamma(t)\boldsymbol{v}(t) + \gamma(t)\frac{\|\boldsymbol{v}(t)\|}{\|\mathbf{F}(\boldsymbol{x}(t))\|}\mathbf{F}(\boldsymbol{x}(t))
\end{aligned} \tag{8.32}$$

So that we can obtain the set of velocities of the nuclei in the system at time $t + dt$ with the following formula

$$\boldsymbol{v}(t+dt) = \frac{\mathbf{F}(\boldsymbol{x}(t))}{m}\,dt - \gamma(t)dt\,\boldsymbol{v}(t) + \boldsymbol{v}(t) + \gamma(t)dt\,\frac{\|\boldsymbol{v}(t)\|}{\|\mathbf{F}(\boldsymbol{x}(t))\|}\mathbf{F}(\boldsymbol{x}(t))$$

$$= \frac{\mathbf{F}(\boldsymbol{x}(t))}{m}\, dt + (1-\gamma(t)dt)\, \boldsymbol{v}(t) + \gamma(t)dt\, \frac{\|\boldsymbol{v}(t)\|}{\|\mathbf{F}(\boldsymbol{x}(t))\|}\mathbf{F}(\boldsymbol{x}(t))$$
$$= \frac{\mathbf{F}(\boldsymbol{x}(t))}{m}\, dt + (1-\alpha)\, \boldsymbol{v}(t) + \alpha\, \frac{\|\boldsymbol{v}(t)\|}{\|\mathbf{F}(\boldsymbol{x}(t))\|}\mathbf{F}(\boldsymbol{x}(t)) \tag{8.33}$$

in which the dimensionless quantity $\alpha = \gamma(t)\, dt$ has been defined.
It was shown that combining a discretized version of equation (8.28) with an adaptive time step scheme yields a simple and competitive optimization algorithm,[127] which results in a minimization scheme for multidimensional functions $E(x_1, ..., x_{3N})$ that is competitive in speed with the above mentioned sophisticated optimizers.
The numerical treatment of the algorithm is simple. Both the timestep dt and the mixing factor α are treated as dynamically adaptive quantities. The parameter α is particularly important since it is the mixing factor between velocities and forces, stating the contribution of the forces on the nuclei in velocity modification. The MD trajectory is therefore continuously readjusted by two kinds of velocity modifications, driven by the value of the power factor $P(t)$:

a) if $P(t) > 0$ (uphill motion) the system is frozen setting all velocities to zero ($\boldsymbol{v} = \mathbf{0}$), the timestep dt is substantially reduced by a factor dt_{dec} in order to have a smooth restart and the mixing factor α is reset to its initial value α_{in};

b) if $P(t) \leq 0$ (downhill motion) a simple mixing of the global $3N$-dimensional velocity and force vectors is done, following the equation

$$\boldsymbol{v} \rightarrow (1-\alpha)\boldsymbol{v} + \alpha\hat{\mathbf{F}}\|\boldsymbol{v}\| \tag{8.34}$$

derived in (8.33) from the Euler-discretization of the last term in equation (8.28) with timestep dt. If the number of steps until the case $P(t) \leq 0$ is encountered is greater than N_{del}, the dynamics is accelerated by increasing the timestep by a factor dt_{inc} (up to a maximum value dt_{max}) and by decreasing the mixing parameter α through a factor f_α.

Algorithm 1 FIRE

Initialize $\mathbf{x}(t)$ and $\mathbf{F}(\mathbf{x}(t))$
$\mathbf{v}(t) \leftarrow 0$
$\alpha \leftarrow \alpha_{\text{start}}$
$\Delta t \leftarrow \Delta t_{\text{start}}$
$N_{P>0} \leftarrow 0$
for $i \leftarrow 1, N_{\max}$ **do**
  $P(t) \leftarrow \mathbf{F}(\mathbf{x}(t)) \cdot \mathbf{v}(t)$
  if $P(t) > 0$ **then**
    $N_{P>0} \leftarrow N_{P>0} + 1$
    $\mathbf{v}(t) \leftarrow (1-\alpha)\mathbf{v}(t) + \alpha\mathbf{F}(\mathbf{x}(t))|\mathbf{v}(t)|/|\mathbf{F}(\mathbf{x}(t))|$
    if $N_{P>0} > N_{delay}$ **then**
      $\Delta t \leftarrow \min(\Delta t f_{inc}, \Delta t_{max})$
      $\alpha \leftarrow \alpha f_\alpha$
    end if
  else if $P(t) \leq 0$ **then**
    $N_{P>0} \leftarrow 0$
    $\mathbf{v}(t) \leftarrow 0$
    $\Delta t \leftarrow \Delta t f_{dec}$
    $\alpha \leftarrow \alpha_{start}$
  end if
  Calculate $\mathbf{x}(t+\Delta t)$, $\mathbf{v}(t+\Delta t)$, $\mathbf{F}(\mathbf{x}(t+\Delta t))$, $E(\mathbf{x}(t+\Delta t))$ ▷ MD integration
  $t \leftarrow t + \Delta t$
  if converged **then**
    break
  end if
end for
$\Delta t \leftarrow \Delta t_{\text{start}}$

Figure 8.2: Pseudo-code for FIRE algorithm, taken from Ref. [126].

The pseudo-code for the algorithm is reported in Figure 8.2. It has to be noted that in this kind of structural relaxation algorithm an accurate calculation of the atomic trajectories is not necessary, and the adaptive timestep allows FIRE to increase dt until either the largest stable timestep dt_{max} is reached, or an energy minimum along the current direction of motion ($P(t) \leq 0$) is encountered. Furthermore,

the short latency time of N_{del} FIRE steps before accelerating the dynamics is important for the stability of the algorithm.

Special attention needs to be paid to the global nature of the algorithm, which assumes that all degrees of freedom are comparable. In particular, the power factor $P(t)$ is a global parameter for the atomic system, and it is at the same time very important because on the basis of its value the dynamics is accelerated, decelerated or it is suddenly frozen. The power factor, defined in equation (8.30) as the scalar product of the force and the velocity vectors, is explicitly computed as the summation over all the atoms of the scalar product of the force acting on a specific atom times the velocity of that specific atom (scalar product of the $3N$ dimensional force and velocity vectors), so that it can be expressed more precisely as

$$P(t) = \sum_{i=1}^{N} \mathbf{F}_i(\boldsymbol{x}(t)) \cdot \boldsymbol{v}_i(t) = \sum_{i=1}^{N} \Big[F(x_i(t))\, v_{x,i} + F(y_i(t))\, v_{y,i} + F(z_i(t))\, v_{z,i} \Big] \tag{8.35}$$

with

$$\mathbf{F}_i(\boldsymbol{x}(t)) = (F(x_i(t)), F(y_i(t)), F(z_i(t))) \quad \text{and} \quad \boldsymbol{v}_i(t) = (v_{x,i}, v_{y,i}, v_{z,i}) \tag{8.36}$$

where $\mathbf{F}_i(\boldsymbol{x}(t))$ is the force applied on the i-th atom and $\boldsymbol{v}_i(t)$ is the velocity vector which contains the velocity components of a given i-th atom. Then the power factor seems to be a global scalar property of an atomic systems, and a quantity driven by the intensity of each contribute rather than by the statistics, e.g. if there are four atoms for which $P_i(t)$ is negative, and only one for which $P_i(t)$ is positive, but the last contribute is greater than the sum of the first 4 negative terms, the global power $P(t)$ results positive.

8.2.2 FIRE2.0 algorithm

The FIRE2.0 algorithm is an improvement of the previously described basic version of the FIRE scheme. Starting from the basic FIRE, three important modifications are performed with this new FIRE2.0 algorithm, which are described in the following.

(i) **Correcting uphill motion**
An important principle of FIRE is to set the velocity vector to zero as soon as the power factor $P(t)$ becomes negative. However, due to discrete time integration, the system will have already gone uphill before $P(t) < 0$ is detected. One could correct this uphill motion by moving backwards for one entire timestep dt and then restarting the motion at time $t - dt$. This will undo the uphill motion as expected, but could keep the trajectory too far from where $P(t) = 0$. A less aggressive correction is to move backward for half a timestep ($0.5dt$).

(ii) **Adjustments for improved stability**
The second modification is to perform the mixing of velocity and force vectors $\boldsymbol{v} \to (1-\alpha)\boldsymbol{v} + \alpha\hat{\mathbf{F}}\|\boldsymbol{v}\|$ just before the last part of the time integration scheme, instead of at the beginning of the step. Note that this modification has no effect if FIRE is used together with the Euler explicit integrator.

(iii) **Additional stopping criteria**
An additional stopping criteria has been implemented in FIRE2.0 in order to avoid unnecessary looping, when it appears that further relaxation is impossible. This could happen when the system is stuck in a narrow valley, bouncing back and forth from the walls but never reaching the bottom. The criterion is the number of consecutive iterations with $P(t) < 0$. The minimization is stopped if this number exceeds a threshold.

8.2.3 Advantages in using FIRE algorithm

Quasi-Newton optimization methods as BFGS has been widely used and proved to be effective in a wide range of different applications. Nevertheless, it is worth to note that in these methods (i) the potential energy surface is locally described with a function expanded up to its second order derivative (i.e. the PES is approximated to be of quadratic form), (ii) the calculation of the approximated Hessian matrix requires a computational cost which scales approximately linearly with the system size and (iii) the

search for transition states turned out to be particularly thorny.
At the same time, the FIRE method has been proved to be a simple and robust algorithm based on molecular dynamics concepts and seems to be promising both in its basic[125] and improved[126] implementation. It has been demonstrated to be a useful tool in numerous atomistic studies, including DFT based simulations of chemically complex systems,[128, 129, 130, 131] significantly faster than standard implementation of the Conjugate Gradient (CG) algorithm and often competitive with more sophisticated quasi-Newton schemes as typified by the BFGS method.[125, 132] Moreover, it seems very well suited for transition state calculations in conjunction with the Nudged Elastic Band method,[133] and apparently it works efficiently in the case of noisy potential energy surface, where BFGS minimization often fails.[134]
Furthermore, the FIRE scheme (i) does not require any approximation of the shape of the potential energy surface, possibly resulting in better performances for systems which are far from the equilibrium, (ii) does not involve the approximation of the Hessian matrix, conceivably leading to a reduction of the computational cost as the size of the system increases and (iii) could eventually take into account temperature effects by exploiting the whole Born-Oppenheimer MD machinary.
On the heels of these results, FIRE algorithm has been implemented in several atomistic simulation packages, like LAMMPS,[135] GROMACS,[136] IMD,[137] DL_POLY,[138] EON,[139] or ASE.[140]
In order to complement CRYSTAL's abilities and overcome some of its limitations, we turned our attention to this novel FIRE method, with the aim of implementing and testing the efficiency of the algorithm in the framework of the CRYSTAL code, expanding the code capabilities in performing structural optimization calculations.

8.3 Implementation of FIRE in the CRYSTAL code

The FIRE algorithm for structural minimization has been implemented in the Molecular Dynamics module of the CRYSTAL code. For technical details about the code, see Appendix B, Section B.5. In the following, the MD integrator chosen for the velocities and positions updating and the stopping criteria adopted for FIRE optimization are described and justified. Then, a screening of the main FIRE adjustable parameters has been performed, in order to select the best set of values and establish default parameters for the algorithm.

8.3.1 Molecular Dynamics integrator

In Ref. [125] it was suggested that FIRE can be used in conjunction with any common MD integrator. However, FIRE implements a variable time-stepping. Therefore, the integrator must be robust against a change of timestep during integration. As highlighted also by F. Shuang et al.,[132] the MD integrator chosen to update positions and velocities at each iteration has a relevant influence on the performance of FIRE minimization procedure. For example, a simple Euler explicit integration scheme is not suitable. Symplectic schemes like Euler semi-implicit (also called symplectic Euler), Leapfrog or Velocity Verlet are more robust against varying timesteps. In particular, the Velocity Verlet integrator has been demonstrated to be an optimal choice along with FIRE optimization, showing good efficiency and convergence.[132, 126] Thanks to its robust behavior, the Velocity Verlet integrator has been adopted as default molecular dynamics integrator in the implementation of FIRE algorithm in the CRYSTAL code.

8.3.2 Convergence criteria

Four kind of convergence criteria are implemented in the CRYSTAL code for FIRE structural minimization, respectively based on (i) the normalized euclidean norm (i.e. the root mean square) of nuclear forces vector, given by

$$F_{rms} = \frac{\|\mathbf{F}\|}{\sqrt{3N}} = \frac{1}{\sqrt{3N}} \left[\sum_{i=1}^{N} (F_{x,i}^2 + F_{y,i}^2 + F_{z,i}^2) \right]^{1/2} \tag{8.37}$$

and computed in cartesian coordinates, (ii) the difference in energy between two consecutive optimization steps, (iii) the maximum and (iv) root-mean-square displacement of the atomic positions between the n-th step and the previous $(n-1)$-th one, both computed in cartesian coordinates. By default, when

these four conditions are all satisfied at a time, FIRE optimization is considered complete. However, it is also possible for the users to adopt only one of these convergence criteria, and to modify the default values of the thresholds, using specific input keywords reported in the manual in Appendix E, Section E.2. Nevertheless, the normalized euclidean norm of cartesian nuclear forces, computed as in Eq. (8.37), has proven to be a reliable and sufficiently strict check to assess the degree of relaxation in FIRE,[126] and it is therefore used as the *only* convergence criterion in all the simulations performed in this work.

8.3.3 Setting of FIRE default parameters

The adjustable parameters in FIRE algorithm are
(i) the initial value α_{in} of the mixing parameter to be used in equation (8.33),
(ii) the initial timestep dt_{in},
(iii) its maximum value $dt_{max} = dt_{in} \cdot t_{max}$,
(iv) the number N_{del} of self-consistent cycles (SCF) and force evaluations (GRAD) to be performed, after a stop due to $P(t) \leq 0$, before accelerating the dynamics again.
It is worth to note that a good assessment of the set of parameters, namely (α_{in}, dt_{in}, t_{max}, N_{del}), is essential to fully exploit the efficiency of FIRE optimization scheme, but, within a wide range of variability of these parameters, it does not affect the reliability of the final results. Based on our experience, and with reference to four systems of different dimensionality, i.e. a water molecule, a water polymer, a 2D slab of ice and a urea crystal, modeled through a PBE functional, a 3-steps procedure is proposed to disentangled the effect of the different parameters and adjust their values. In all cases, the goal is to minimize the number of iterations N_{step} (in each step, a self-consistent cycle (SCF) procedure and an energy gradient calculation (GRAD) are performed), as follows:
(i) for a given dt_{in} and N_{del}, different values of t_{max} are explored (as in Figs. 8.3a,d and 8.4a,d) in order to choose its optimal value;
(ii) then, given the best value of t_{max} (and fixing N_{del}), different dt_{in} are evaluated in the range $[0.1, 3.0]$ fs, to define its best value (Figs. 8.3b,e and 8.4b,e);
(iii) finally, having fixed the optimal values of t_{max} and dt_{in}, N_{del} can be varied in the range $[0, 10]$, as shown in Figs. 8.3c,f and 8.4c,f. The best value for α_{in} is automatically derived from the overall analysis of all the trends obtained.
A very similar behavior of the number of total iterations N_{step} as a functions of the initial mixing factor α_{in} for points (i), (ii) and (iii) is obtained with all the atomic systems considered, suggesting that the best set of FIRE parameters does not depend on the dimensionality of the system. Furthermore, despite the difference in the number of iterations necessary to reach convergence, the reliability of FIRE in finding the minimum energy structure is preserved. Therefore, a set of parameters that provides excellent performance and is optimal for a wide range of systems, different in size, dimension and chemical-physical properties, can be defined. These default values, obviously modifiable from Input, are shown in Table 8.1.

Parameter	Keyword	Default value
α_{in}	ALPHASTART	0.30
dt_{in}	DTSTART	1.0 (fs)
t_{max}	TMAX	1.0
N_{del}	NDELAY	0
α_{dec}	ALPHASHRINK	0.99
dt_{inc}	DTGROW	1.1
dt_{dec}	DTSHRINK	0.5

Table 8.1: CRYSTAL default values for FIRE algorithm parameters, with the correspondent keywords that can be used in Input to modify their values.

It is worth to note that all the FIRE parameters screening is here performed using PBE functional. However, from a preliminary analysis in the case of water molecule and urea crystal with B3LYP functional, the best set of FIRE parameters results slightly dependent on the kind of exchange-correlation functional employed. For the case of water molecule, for instance, the best set of FIRE parameters ($\alpha_{in}, dt_{in}, t_{max}, N_{del}$)

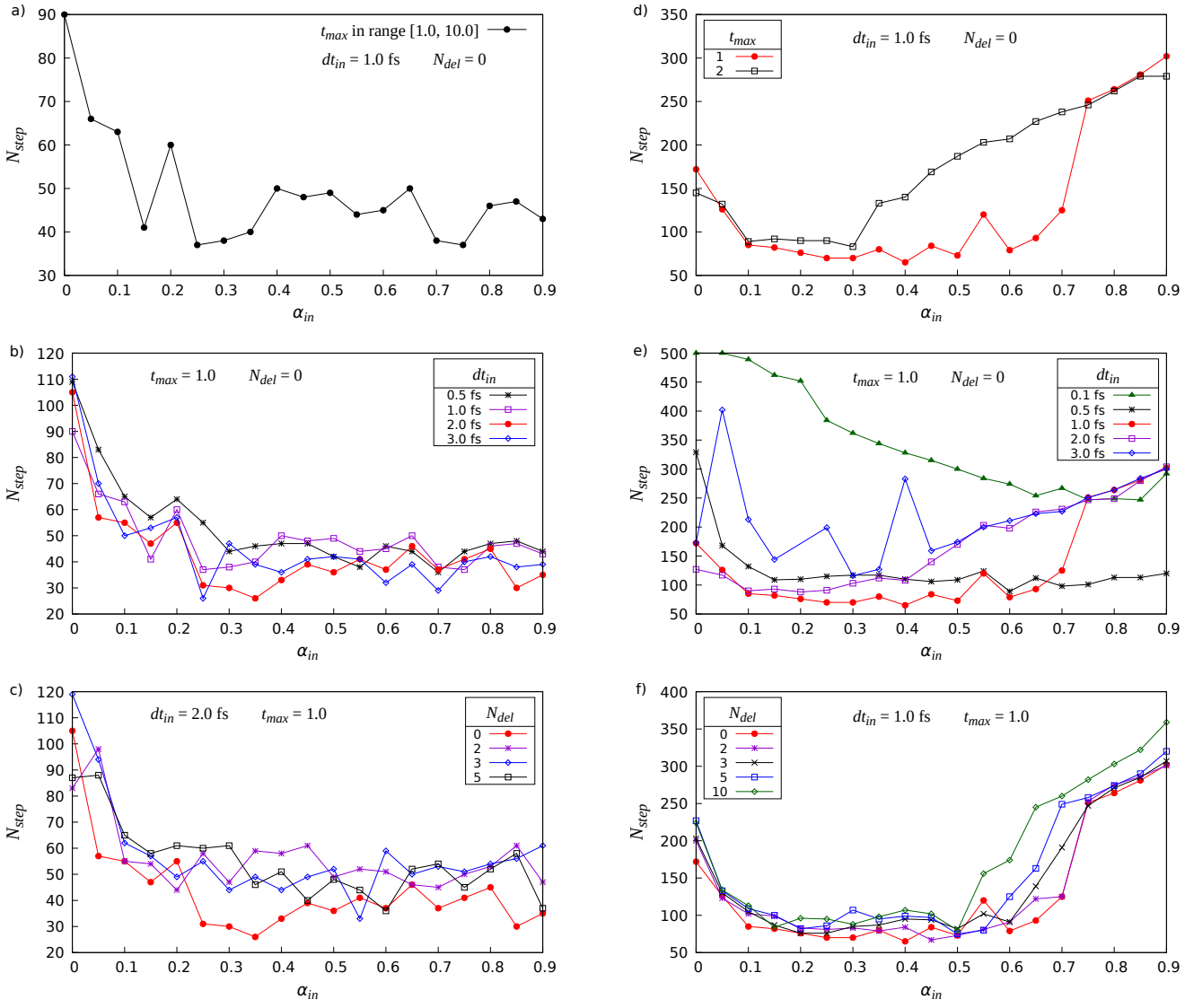

Figure 8.3: FIRE parameters $(\alpha_{in}, dt_{in}, t_{max}, N_{del})$ for (left panel) water molecule (0D system, 3 atoms) and (right panel) water polymer (1D system, 6 atoms): screening and setting of the default values.

with PBE and B3LYP functional is, respectively, $(0.35, 2.0, 1.0, 0)$ and $(0.20, 1.0, 1.0, 0)$, leading to a corresponding total number of optimization steps equal to 26 and 32.
Finally, the default values for the α_{dec}, dt_{inc} and dt_{dec} factors, used to accelerate or decrease the entity of the atomic motion, are those originally proposed by Bitzek et al.[125, 126]

8.3.4 Computational details

All the calculations reported in this work are performed with a beta version of the CRYSTAL program for *ab initio* quantum chemistry and physics of solid state, based on the public version of the code.[120] All the considered atomic systems are treated in the frame of the Density Functional Theory (DFT), adopting the gradient-corrected Perdew-Burke-Ernzerhof (PBE) functional,[93] except the crambin molecule that is described through Hartree-Fock (HF) Hamiltonian with three semiclassical corrections (D3, gCP, SRB), which are added to the HF energies (and atomic and cell gradients) within the so called HF3C method.[141, 142, 143] Moreover, to better describe the ice slab with CO molecules and the amorphous ice, in conjunction with PBE functional, London-type pairwise empirical correction to the energy for dispersive interactions as proposed by Grimme[94] and as modified for molecular crystals,[144] is considered, in order to include long-range dispersion contributions to the computed *ab initio* total energy and gradients. Furthermore, hybrid functionals such as B3LYP,[102] HSE06[145, 93] and PBE0[103] have been used to model the crystalline urea system.

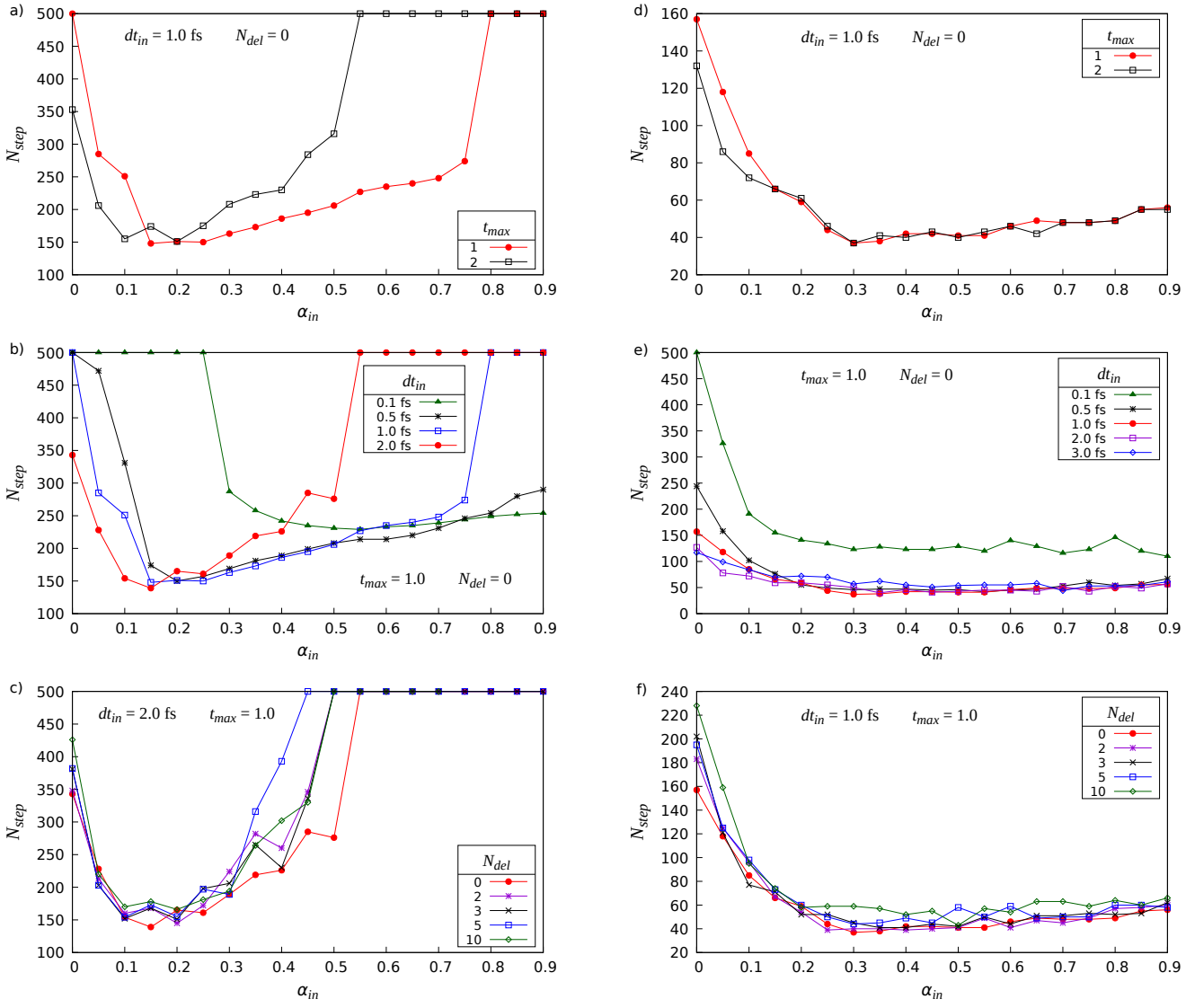

Figure 8.4: FIRE parameters ($\alpha_{in}, dt_{in}, t_{max}, N_{del}$) for (left panel) water slab (2D system, 24 atoms) and (right panel) urea molecular crystal (3D system, 16 atoms): screening and setting of the default values.

All-electron basis sets, consisting of contracted Gaussian-type atomic orbital functions (A.O.) are used for all the atoms and are reported in the SM.

The DFT exchange-correlation contribution was evaluated by numerical integration over the unit cell volume, using a pruned grid,[105, 106] with a number of radial and angular points reported in Table 2 of the SM for the different analyzed atomic systems. The diagonalization of the Hamiltonian matrix and the integration over the reciprocal space is carried out using the Monkhorst-Pack mesh,[107] consisting in a grid of $\boldsymbol{k}$-points defined in the Irreducible part of the first Brillouin Zone (IBZ). The Coulomb and exchange series, summed in direct space, are truncated using overlap criteria thresholds. The number of $\boldsymbol{k}$-points in the IBZ, together with the overlap criteria thresholds for Coulomb and exchange series, and the thresholds for the self-consistent field algorithm convergence on the total energy per unit cell, used for the different atomic systems, are reported in Table 2 of the SM.

The tempered ice 3D system has been obtained using a beta develop version of CRYSTAL Molecular Dynamics module (derived from the CRYSTAL17 public release[120]),[127] starting from the initial configuration of crystalline ice and performing a 80 steps (16 fs) NVE MD simulation, with a timestep of 0.2 fs and an initial temperature of 1800 K (the temperature at the 80-th step is equal to 816.5 K).

Geometry optimization is performed using analytical gradients with respect to atomic coordinates, within a quasi-Newtonian algorithm combined with two kinds of Hessian updating schemes: the Schlegel's (SH)[119] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formulae.[108, 109, 110, 111, 112] Only atomic coordinates are optimized, to allow comparison with FIRE structural minimization. Convergence

criteria for SH and BFGS structural optimizations are based on the root mean square and absolute value of the largest component of both the estimated displacements and the gradients of energy functional with respect to the nuclear positions, both computed in normal coordinates. In this framework, the CRYSTAL17 default convergence thresholds and minimization parameters have been adopted.[120, 121]

8.4 Results and Discussions

The performance of FIRE algorithm, at constant volume and lattice parameters, has been evaluated for systems with different dimensionality, number of atoms and kind of chemical bond. The geometry optimization on the same set of systems has been also performed with quasi-Newton SH and BFGS updating schemes, removing the symmetry of the systems, if necessary, to allow a reliable comparison with FIRE procedure. Information on each system (geometry structures, basis sets and computational settings) are reported in the Supplementary Material (SM). The number of optimization steps up to convergence, N_{step}, the root mean square of final forces, F_{rms}, of last iteration atomic displacements, d_{rms}, and of bond lengths difference between SH or FIRE and BFGS, Δb_{rms}, are summarized in Table 8.2. The number of degrees of freedom in FIRE structural optimizations is equal to three times the number of atoms N_{at} in the reference cell unit, while for SH and BFGS equals $(3N_{at} - 6)$ for molecules and $(3N_{at} - 3)$ for periodic systems.
The specific parameters adopted in FIRE for each system are reported in Table 3 of the SM. As regards the general performances, the results show that each single FIRE optimization cycle (SCF $\oplus$ GRAD) has a less computational cost than the corresponding SH and BFGS one which involves also the updating of the Hessian matrix, and the saving in CPU time increases with the system size. At the same time, FIRE takes more iterations to reach convergence, so that its efficiency is comparable with the BFGS one when the increasing in the number of iterations is counterbalanced by the decreasing in the computational cost of each step. Nevertheless, for almost all the systems, FIRE performs better than quasi-Newton SH updating scheme.

As for the reliability of the results, i.e. the capability of finding stationary minima in the PES, the agreement on the final energies and atomic positions between FIRE and BFGS is within the accuracy of the computational set up. The overall root mean square differences between bonds of the structures obtained with FIRE and BFGS, Δb_{rms}, are of the same order of magnitude of the final root mean square displacements, d_{rms}, suggesting that the optimized geometries obtained with the two procedures can be considered equivalent.
Different behaviors in the minimization process were observed and deserve few comments. With reference to the prototypical cases of tempered ice and urea bulk, the mixing factor α, timestep dt, euclidean norm of the $3N$ velocities vector and the power factor are plotted in Fig. 8.5 as a functions of the number of minimization steps.

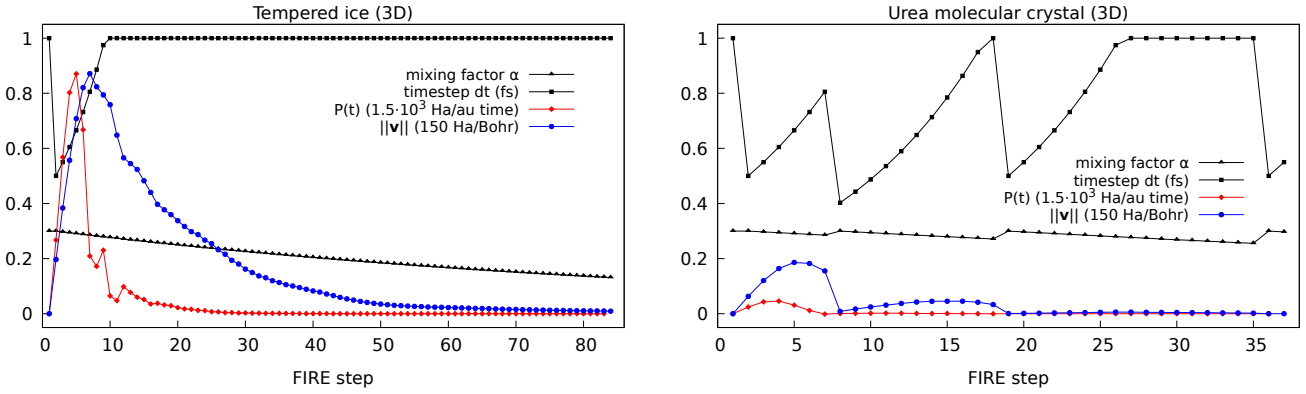

Figure 8.5: Behavior of FIRE parameters during the structural optimization of the tempered ice (left panel) and urea molecular crystal (right panel). The values of power factor and of euclidean norm of velocities vector have been amplified by a factor $1.5 \cdot 10^3$ and 150, respectively, to allow comparison with the other parameters.

In the tempered ice, left panel, after the initial balancing of the system obtained in 10 iterations by

System	N_{at}		N_{step}	T_{ratio}	F_{rms} [Ha/Bohr]	d_{rms} [Bohr]	Δb_{rms} [Bohr]	ΔE [eV/atom]
H_2O (0D)	3	SH	7	1.1	$1.76 \cdot 10^{-7}$	$1.0 \cdot 10^{-5}$	$1.9 \cdot 10^{-5}$	$1.0 \cdot 10^{-12}$
		BFGS	6	–	$3.74 \cdot 10^{-6}$	$4.3 \cdot 10^{-4}$	–	–
		FIRE	26	2.8	$4.31 \cdot 10^{-7}$	$2.8 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$-9.1 \cdot 10^{-10}$
Water Polymer (1D)	6	SH	26	1.3	$1.60 \cdot 10^{-5}$	$1.0 \cdot 10^{-3}$	$7.0 \cdot 10^{-4}$	$1.8 \cdot 10^{-7}$
		BFGS	21	–	$1.32 \cdot 10^{-5}$	$8.6 \cdot 10^{-4}$	–	–
		FIRE	65	2.6	$5.62 \cdot 10^{-6}$	$2.4 \cdot 10^{-5}$	$2.2 \cdot 10^{-4}$	$3.4 \cdot 10^{-7}$
Urea (0D)	8	SH	42	9.0	$9.04 \cdot 10^{-5}$	$8.1 \cdot 10^{-4}$	$1.6 \cdot 10^{-2}$	$-1.0 \cdot 10^{-2}$
		BFGS	5	–	$1.39 \cdot 10^{-5}$	$2.8 \cdot 10^{-4}$	–	–
		FIRE	55	8.2	$9.29 \cdot 10^{-6}$	$9.8 \cdot 10^{-7}$	$1.0 \cdot 10^{-4}$	$-4.0 \cdot 10^{-8}$
Urea (3D)	16	SH	43	2.7	$3.13 \cdot 10^{-6}$	$1.9 \cdot 10^{-4}$	$5.7 \cdot 10^{-5}$	$7.8 \cdot 10^{-7}$
		BFGS	17	–	$1.24 \cdot 10^{-5}$	$4.6 \cdot 10^{-4}$	–	–
		FIRE	37	2.1	$9.85 \cdot 10^{-6}$	$5.4 \cdot 10^{-6}$	$5.0 \cdot 10^{-4}$	$-5.2 \cdot 10^{-8}$
Water Slab (2D)	24	SH	78	2.5	$1.10 \cdot 10^{-5}$	$5.9 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	$2.9 \cdot 10^{-7}$
		BFGS	32	–	$1.42 \cdot 10^{-5}$	$1.1 \cdot 10^{-3}$	–	–
		FIRE	139	4.1	$9.93 \cdot 10^{-6}$	$4.1 \cdot 10^{-5}$	$2.2 \cdot 10^{-4}$	$-5.6 \cdot 10^{-7}$
Tempered Ice (3D)	24	SH	106	2.6	$9.14 \cdot 10^{-6}$	$4.1 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	$-7.6 \cdot 10^{-8}$
		BFGS	41	–	$1.12 \cdot 10^{-5}$	$9.2 \cdot 10^{-4}$	–	–
		FIRE	84	1.9	$9.85 \cdot 10^{-6}$	$5.2 \cdot 10^{-4}$	$2.6 \cdot 10^{-4}$	$-6.7 \cdot 10^{-6}$
Ice Slab with CO (2D)	32	BFGS	234	–	$5.31 \cdot 10^{-5}$	$5.0 \cdot 10^{-4}$	–	–
		FIRE	1180	4.1	$1.27 \cdot 10^{-4}$	$8.6 \cdot 10^{-4}$	$5.3 \cdot 10^{-3}$	$-2.6 \cdot 10^{-4}$
Amorphous Ice (3D)	126	BFGS	125	–	$2.90 \cdot 10^{-5}$	$8.1 \cdot 10^{-4}$	–	–
		FIRE	177	1.2	$9.80 \cdot 10^{-6}$	$1.0 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	$1.3 \cdot 10^{-4}$
Crambin (0D)	642	BFGS	530	–	$1.18 \cdot 10^{-5}$	$3.1 \cdot 10^{-4}$	–	–
		FIRE	530	0.8	$8.54 \cdot 10^{-5}$	$2.1 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$6.7 \cdot 10^{-4}$

Table 8.2: Number of optimization steps N_{step}, computational time ratio $T_{ratio} = T_{\text{FIRE,SH}}/T_{\text{BFGS}}$, root mean square of final forces $F_{rms} = \|\mathbf{F}\|/\sqrt{3N}$ and of atomic displacements d_{rms} in the last cycle, root mean square of bond lengths differences $\Delta b = b_{\text{FIRE,SH}} - b_{\text{BFGS}}$, and energy difference $\Delta E = E_{\text{FIRE}} - E_{\text{BFGS}}$ which results from structural optimizations performed within SH, BFGS and FIRE schemes for different atomic systems with N_{at} atoms.

efficiently accumulating the inertia, the maximum timestep is reached and both the mixing factor α and the velocities decrease monotonously till convergence. On the contrary, the velocity $\boldsymbol{v}(t)$ of the urea crystal, right panel, brings the trajectory to an uphill motion in the PES several times, so that the timestep is periodically decreased by a factor dt_{dec}, and reaches its maximum value in a nearly stable way only after 27 iterations. A change in the number of equilibration steps, N_{del}, from 0 to 5, does not modify the behavior of α and dt during the minimization of both systems but, within the accuracy on energies and forces, the number of iterations N_{step} increases slightly, passing from 84 to 86 and from 37 to 44, respectively, for tempered ice and urea crystal.

Then, in order to test the robustness and scalability of FIRE, given the same set of computational parameters (basis set, DFT functional, FIRE setting as in Table 8.1), structural optimizations of crystalline urea supercells of increasing size were performed. The results, reported in Table 8.3, show that in the case of FIRE the number of iterations does not depend on the system size, so that the efficiency of FIRE algorithm gradually reaches the BFGS one, succeeded in overtaking BFGS computational cost for a number of atoms greater than 192 (3×2×2 supercell). Interestingly enough, both the internal accuracy, $\Delta\tilde{E}$, and the agreement with BFGS results, ΔE, are preserved, moving from 16 to 192 atoms (with a corresponding number of atomic orbitals in the basis set equal to 152 and 1824, respectively).

Finally, the dependence of FIRE efficiency on different exchange-correlation functionals, which determine

Supercell	N_{at}	N_{AOs}		N_{step}	$\bar{N}_w$	T_{ratio}	$\bar{d}_{rms}$ [Bohr]	$\Delta\tilde{E}\left[\frac{\text{eV}}{\text{atom}}\right]$	$\Delta E\left[\frac{\text{eV}}{\text{atom}}\right]$
			BFGS	17	6	–	$1.20\cdot10^{-2}$	–	–
1×1×1	16	152	FIRE	37	5	2.11	$1.61\cdot10^{-3}$	–	$-5.95\cdot10^{-8}$
			FIRE2.0	36	5	1.93	$1.81\cdot10^{-3}$	–	$6.86\cdot10^{-7}$
			BFGS	21	7	–	$1.56\cdot10^{-2}$	$9.42\cdot10^{-7}$	–
2×1×1	32	304	FIRE	37	5	1.68	$1.60\cdot10^{-3}$	$-2.76\cdot10^{-7}$	$1.16\cdot10^{-6}$
			FIRE2.0	35	6	1.52	$1.85\cdot10^{-3}$	$2.79\cdot10^{-7}$	$1.35\cdot10^{-6}$
			BFGS	20	8	–	$2.03\cdot10^{-2}$	$3.38\cdot10^{-6}$	–
2×2×1	64	608	FIRE	37	6	1.59	$1.60\cdot10^{-3}$	$-5.21\cdot10^{-7}$	$3.84\cdot10^{-6}$
			FIRE2.0	35	6	1.47	$1.85\cdot10^{-3}$	$7.60\cdot10^{-8}$	$3.99\cdot10^{-6}$
			BFGS	29	7	–	$9.73\cdot10^{-3}$	$1.34\cdot10^{-6}$	–
3×2×1	96	912	FIRE	37	6	1.05	$1.61\cdot10^{-3}$	$-4.56\cdot10^{-7}$	$1.73\cdot10^{-6}$
			FIRE2.0	35	6	1.01	$1.85\cdot10^{-3}$	$4.59\cdot10^{-9}$	$2.02\cdot10^{-6}$
			BFGS	25	8	–	$1.48\cdot10^{-2}$	$6.01\cdot10^{-6}$	–
2×2×2	128	1216	FIRE	37	7	1.11	$1.61\cdot10^{-3}$	$1.92\cdot10^{-6}$	$4.03\cdot10^{-6}$
			FIRE2.0	33	7	1.04	$1.96\cdot10^{-3}$	$2.48\cdot10^{-6}$	$4.22\cdot10^{-6}$
			BFGS	29	9	–	$1.00\cdot10^{-2}$	$3.19\cdot10^{-6}$	–
3×2×2	192	1824	FIRE	37	7	0.95	$1.61\cdot10^{-3}$	$1.90\cdot10^{-6}$	$1.23\cdot10^{-6}$
			FIRE2.0	33	7	0.90	$1.96\cdot10^{-3}$	$2.51\cdot10^{-6}$	$1.37\cdot10^{-6}$

Table 8.3: Number of atoms N_{at} in the supercell, number of atomic orbitals N_{AOs} in the basis set, number of optimization steps N_{step}, average number of self-consistent cycles for ground state wavefunction calculation $\bar{N}_w$, ratio of total computational times $T_{ratio} = T_{\text{FIRE}}/T_{\text{BFGS}}$, mean of root-mean-square of atomic displacements $\bar{d}_{rms}$, internal check on energy computed by means of the formula $\Delta\tilde{E} = E^{16}_{\text{opt}}/16 - E^{N_{at}}_{\text{opt}}/N_{at}$ and final energy difference between the two methods $\Delta E = E_{\text{FIRE}} - E_{\text{BFGS}}$, for different urea molecular crystal (3D) supercell sizes. The mean values $\bar{N}_w$ and $\bar{d}_{rms}$ are computed by averaging over, respectively, the N_w and d_{rms} values for all the optimization steps. Cartesian coordinates of optimized atomic structures are reported in Figures 19-24 of the Supplementary Material of Ref. [127].

the shape of the PES, was investigated. The influence of the functional on the best setting of FIRE computational parameters (i.e. α_{in}, dt_{in}, t_{max} and N_{del}) was already pointed out in Section 8.3.3. Nevertheless, in order to perform a comparison on the same ground, the structural optimization of urea crystal was performed with PBE, B3LYP, HSE06 and PBE0 functionals, adopting the FIRE default values optimized for PBE (see Table 8.1). The results, summarized in Table 8.4, show that, despite the predictable increase in the computational time due to the non-optimal tuning of the FIRE parameters, the geometries and energies obtained with different functionals are in good agreement with the corresponding BFGS ones, confirming the general portability and robustness of the FIRE optimization procedure.

8.5 Conclusions and Perspectives

In this work, we have described the implementation of FIRE algorithm and assessed its efficiency and reliability in the CRYSTAL code. FIRE is a structural optimization method based on Molecular Dynamics concepts, introduced by E. Bitzek et al.[125] as an alternative scheme to quasi-Newton line-search based minimization algorithms. The interest for this novel method is threefold. First, it does not rely on any approximation on the shape of the PES, possibly resulting in good convergence behavior regardless the PES form. Secondly, it does not involve the Hessian approximation, maybe leading to a less computational cost than SH and BFGS schemes. Finally, being based on MD approach, the inclusion of temperature in structural optimization could be straightforward.
First of all, a screening of the four FIRE adjustable parameters has been realized through structural optimizations of atomic systems with different dimensionality, identifying a set of robust default values. Then, structural optimizations of atomic systems with different number of atoms, dimensionality and kind of bonds have been performed with FIRE algorithm. The accuracy of FIRE in finding energy minima

Functional		N_{step}	$\bar{N}_w$	T_{ratio}	$\bar{d}_{rms}$ [Bohr]	F_{rms} $\left[\frac{\text{Ha}}{\text{Bohr}}\right]$	d^e_{rms} [Bohr]	ΔE $\left[\frac{\text{eV}}{\text{atom}}\right]$
PBE	BFGS	17	6	–	$1.20 \cdot 10^{-2}$	$1.24 \cdot 10^{-5}$	$4.6 \cdot 10^{-4}$	–
	FIRE	37	5	2.1	$1.61 \cdot 10^{-3}$	$9.85 \cdot 10^{-6}$	$5.4 \cdot 10^{-6}$	$-5.20 \cdot 10^{-8}$
	FIRE2.0	36	5	1.9	$1.81 \cdot 10^{-3}$	$8.47 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$	$6.95 \cdot 10^{-7}$
B3LYP	BFGS	17	5	–	$3.62 \cdot 10^{-3}$	$2.81 \cdot 10^{-5}$	$3.1 \cdot 10^{-4}$	–
	FIRE	43	4	2.4	$7.15 \cdot 10^{-4}$	$8.54 \cdot 10^{-6}$	$1.5 \cdot 10^{-4}$	$1.56 \cdot 10^{-8}$
	FIRE2.0	43	4	2.4	$7.79 \cdot 10^{-4}$	$9.62 \cdot 10^{-6}$	$1.7 \cdot 10^{-4}$	$-2.64 \cdot 10^{-7}$
HSE06	BFGS	16	5	–	$4.43 \cdot 10^{-3}$	$2.44 \cdot 10^{-5}$	$5.6 \cdot 10^{-4}$	–
	FIRE	58	4	3.4	$4.61 \cdot 10^{-4}$	$9.61 \cdot 10^{-6}$	$1.8 \cdot 10^{-4}$	$1.95 \cdot 10^{-6}$
	FIRE2.0	50	4	3.0	$5.30 \cdot 10^{-4}$	$9.62 \cdot 10^{-6}$	$1.8 \cdot 10^{-4}$	$2.08 \cdot 10^{-6}$
PBE0	BFGS	14	6	–	$4.49 \cdot 10^{-3}$	$2.69 \cdot 10^{-5}$	$4.6 \cdot 10^{-4}$	–
	FIRE	56	4	3.7	$4.60 \cdot 10^{-4}$	$9.73 \cdot 10^{-6}$	$1.4 \cdot 10^{-4}$	$6.34 \cdot 10^{-7}$
	FIRE2.0	48	4	3.2	$5.31 \cdot 10^{-4}$	$9.37 \cdot 10^{-6}$	$1.5 \cdot 10^{-4}$	$9.41 \cdot 10^{-7}$

Table 8.4: Number of optimization steps N_{step}, average of self-consistent cycles for ground state wavefunction calculation $\bar{N}_w$, computational time ratio $T_{ratio} = T_{\text{FIRE}}/T_{\text{BFGS}}$, mean of root-mean-square of atomic displacements $\bar{d}_{rms}$, final root mean square of nuclear forces $F_{rms} = \|\mathbf{F}\|/\sqrt{3N}$ and of last step atomic displacements d^e_{rms}, and energy difference $\Delta E = E_{\text{FIRE}} - E_{\text{BFGS}}$ which results from structural optimizations of urea crystal (16 atoms) performed with FIRE and BFGS, for different exchange-correlation functionals. Cartesian coordinates of optimized atomic structures are reported in Figures 25-28 of the Supplementary Material of Ref. [127].

of the PES for these systems has been demonstrated comparing the total energy and geometry of FIRE final structures with those obtained with quasi-Newton SH and BFGS schemes. The reliability of FIRE in minimize the PES shaped by different functionals, such as PBE, B3LYP, HSE06 and PBE0 has been proven for the case of urea molecular crystal. Finally, as regards the computational time, we can conclude that FIRE structural optimization has generally a greater computational cost than the correspondent BFGS one, due to the fact that it implies more iterations to reach convergence. Nevertheless, a single FIRE step has a less computational cost than a SH or BFGS one, so that the overall FIRE minimization becomes the most efficient one when the increasing of the computational cost due to a greater number of iterations is compensated by a reduction of timings in performing each single step.
This study sets the ground for further improvements of the efficiency of FIRE in the CRYSTAL code, by means of its optimized version FIRE2.0,[126] which has demonstrated to be significantly faster than the original one.[126] Furthermore, the fact that FIRE algorithm is based on MD concepts could paved the way for other important implementations, namely, a finite temperature structural optimization algorithm and the Nudged Elastic Band method for transition states calculations.

Chapter 9

Ab initio calculation of electronic transport properties with semiclassical Boltzmann transport theory

This chapter is devoted to the description of the theoretical framework and the practical algorithms that can be used to compute ab initio electronic transport properties of condensed matter systems, with the aim to develop an efficient version of the code (in terms of memory usage), based on Massive Parallel Processing approach.
In this context, Section 9.1 contains a brief explanation of the semiclassical theory of transport processes in solids, which relied on Boltzmann equation. This equation can be easily demonstrated with the introduction an electronic distribution function and the calculation of its derivative with respect time and it can be used, together with semiclassical equations of motion, to describe transport properties of materials, such as current density or electrical conductivity. The theoretical framework underlying the calculation of ab initio electronic transport properties, based on the semiclassical approximation, is described in Section 9.2, as implemented in the CRYSTAL code. A new orbital rotational invariant formulation for the definition of the band velocities is discovered and justified. In Section 9.3, the Massive Parallel Processing algorithm applied to the electronic transport properties calculation is then described. Finally, Section 9.4 reports some results for the comparison of the new implemented parallelization with the strategy already developed in the public version of the code. The aim of this section is also to underline the importance of the novel invariant formulation for the band velocities definition, supporting this theoretical findings with calculations performed on Silicon bulk structure. The last part of this chapter, Section 9.5, is devoted to solve a problem related to the parallelized procedure already implemented in the public version of the code, due to the creation of input/output units which occupy large part of the disk space during the calculation. The description of the solution found to that problem and the associated implemented algorithm is then described, and calculations on some tests cases are performed in order to prove the accuracy and the computational time of the new version with respect to the public one.

9.1 Boltzmann transport theory

9.1.1 Distribution function and BBGKY hierarchy

Consider the Hamiltonian dynamics for N identical point particles, given by

$$\mathcal{H} = \frac{1}{2m}\sum_{i=1}^{N} \boldsymbol{p}_i^2 + \sum_{i=1}^{N} v_{ext}(\boldsymbol{r}_i) + \sum_{i<j} V_{int}(\boldsymbol{r}_i - \boldsymbol{r}_j) \tag{9.1}$$

where the potential v_{ext} illustrates the presence of an external force $\mathbf{F} = -\nabla v_{ext}$ acting equally on all particles and the term V_{int} is the potential associated to two-body interactions between particles. The quantity of interest is the probability distribution function $f(\boldsymbol{r}_i, \boldsymbol{p}_i; t)$, defined over the $2N$ dimensional phase space and its evolution in time under the Hamiltonian (9.1). The function $f(\boldsymbol{r}_i, \boldsymbol{p}_i; t)$ represents

the probability that the system will be found in the vicinity of the point $(\boldsymbol{r}_i, \boldsymbol{p}_i)$. As with all probabilities, the function is normalized as

$$\int dV f(\boldsymbol{r}_i, \boldsymbol{p}_i; t) = 1 \qquad \text{con } dV = \prod_{i=1}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \tag{9.2}$$

Furthermore, because the probability is locally conserved, it must obey a continuity equation in phase space. Inserting the Hamilton equations in the continuity formula, it is obtained that the evaluation of $f(\boldsymbol{r}_i, \boldsymbol{p}_i; t)$ with respect the Hamiltonian field $\bar{X}_H$ must be equal to zero, i.e. it must be verified the expression

$$\langle \bar{X}_{\mathcal{H}}, df \rangle = \frac{\partial f}{\partial t} + \sum_{i=1}^{N} \left(\frac{\partial f}{\partial \boldsymbol{r}_i} \cdot \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}_i} - \frac{\partial f}{\partial \boldsymbol{p}_i} \cdot \frac{\partial \mathcal{H}}{\partial \boldsymbol{r}_i} \right) = 0 \tag{9.3}$$

The formula (9.3) is called Liouville equation: it is the statement that probability does not change along any trajectory in phase space. The mathematical expression (9.3) is often written using the Poisson bracket

$$\frac{df}{dt} = \frac{\partial f}{\partial t} - \{\mathcal{H}, f\} = 0 \tag{9.4}$$

It is convenient to define a one-particle non equilibrium distribution function, that captures the expected number of parting lying at some point $(\boldsymbol{r}, \boldsymbol{p})$, such as the function

$$f_1(\boldsymbol{r}, \boldsymbol{p}; t) = N \int \prod_{i=2}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \, f(\boldsymbol{r}, \boldsymbol{r}_2, ..., \boldsymbol{r}_N, \boldsymbol{p}, \boldsymbol{p}_2, ..., \boldsymbol{p}_N; t) \tag{9.5}$$

and to make the calculations with this function instead of the probability distribution for all N particles. The distribution $f_1(\boldsymbol{r}, \boldsymbol{p}; t)$ is defined such as the quantity $f_1(\boldsymbol{r}, \boldsymbol{p}; t) \, d\boldsymbol{r} \, d\boldsymbol{p}$ counts the number of particles that can be found at time t in the infinitesimal phase space volume $d\boldsymbol{r} \, d\boldsymbol{p}$ in the neighborhood of the point $(\boldsymbol{r}, \boldsymbol{p})$. The factor N that sits in front of the equation (9.5) ensures that f_1 is normalized as

$$N = \int d\boldsymbol{r} \, d\boldsymbol{p} \, f_1(\boldsymbol{r}, \boldsymbol{p}; t) \tag{9.6}$$

To understand how the one-particle distribution function changes with time, the equation (9.5) can be derived with respect time, so it is obtained

$$\frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d\mathbf{r}_i \, d\boldsymbol{p}_i \, \frac{\partial f}{\partial t} = N \int \prod_{i=2}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \, \{\mathcal{H}, f\} \tag{9.7}$$

where the last identity is written by means of equation (9.4). The Poisson bracket can then be calculated explicitly using the Hamiltonian (4.6) as

$$\begin{aligned} \frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \Big[& - \sum_{j=1}^{N} \frac{\boldsymbol{p}_j}{m} \cdot \frac{\partial f}{\partial \boldsymbol{r}_j} + \sum_{j=1}^{N} \frac{\partial v_{ext}}{\partial \boldsymbol{r}_j} \cdot \frac{\partial f}{\partial \boldsymbol{p}_j} \\ & + \sum_{j=1}^{N} \sum_{k<l} \frac{\partial V_{int}(\boldsymbol{r}_k - \boldsymbol{r}_l)}{\partial \boldsymbol{r}_j} \cdot \frac{\partial f}{\partial \boldsymbol{p}_j} \Big] \end{aligned} \tag{9.8}$$

Now, whenever $j = 2, ..., N$ the previous equation can always be integrated by parts to move the derivative away from f and onto the other terms. In each case the result is simply zero because when the derivative is with respect to $\boldsymbol{r}_j$, the other terms depend only on $\boldsymbol{p}_i$ and vice-versa. Only the terms that involve derivatives with respect to $\boldsymbol{r}_1$ and $\boldsymbol{p}_1$ remain, because they can't be integrated by parts. The phase space coordinate subscript 1 becomes a free index, so the notation $\boldsymbol{r}_1 \equiv \boldsymbol{r}$ e $\boldsymbol{p}_1 \equiv \boldsymbol{p}$ is legitimized. Then equation (9.8) becomes

$$\frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \Big[- \frac{\boldsymbol{p}}{m} \cdot \frac{\partial f}{\partial \boldsymbol{r}} + \frac{\partial v_{ext}(\boldsymbol{r})}{\partial \boldsymbol{r}} \cdot \frac{\partial f}{\partial \boldsymbol{p}} + \sum_{k=2}^{N} \frac{\partial V_{int}(\boldsymbol{r} - \boldsymbol{r}_k)}{\partial \boldsymbol{r}} \cdot \frac{\partial f}{\partial \boldsymbol{p}} \Big] \tag{9.9}$$

Define the single particle hamiltonian:

$$H_1 = \frac{\boldsymbol{p}^2}{2m} + v_{ext}(\boldsymbol{r}) \tag{9.10}$$

that includes the interaction between one particle and the external force field, but it knows nothing about the interaction with the other particles. All of that information is contained in the last term of the right member of equation (9.9). Thanks to the definition (9.10) the first two terms of the right member in (9.9) can be seen as the Poisson bracket between H_1 and f_1, so (9.9) reduces to

$$\frac{\partial f_1}{\partial t} = \{H_1, f_1\} + N \int \prod_{i=2}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \sum_{k=2}^{N} \frac{\partial V_{int}(\boldsymbol{r} - \boldsymbol{r}_k)}{\partial \boldsymbol{r}} \cdot \frac{\partial f}{\partial \boldsymbol{p}}. \tag{9.11}$$

The evolution of the one-particle non equilibrium distribution function is then described by a Liouville-like equation, together with an extra term

$$\frac{\partial f_1}{\partial t} = \{\mathcal{H}_1, f_1\} + \left(\frac{\partial f_1}{\partial t}\right)_{coll} \tag{9.12}$$

The first term is sometimes referred to as the *streaming* term. It tells you how the particles move in the absence of collisions. The second term, known as the *collision integral*, is given by the second term in (9.11) and contains the summation of the interaction potential of one particle with each of the others. In fact, because all the particles are the same, each of the (N-1) terms in the summation in equation (9.11) are identical, so it can be written

$$\left(\frac{\partial f_1}{\partial t}\right)_{coll} = N(N-1) \int d\boldsymbol{r}_2 \, d\boldsymbol{p}_2 \, \frac{\partial V_{int}(\boldsymbol{r} - \boldsymbol{r}_2)}{\partial \boldsymbol{r}} \cdot \frac{\partial}{\partial \boldsymbol{p}} \int \prod_{i=3}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \, f(\boldsymbol{r}, \boldsymbol{r}_2, ..., \boldsymbol{p}, \boldsymbol{p}_2, ...; t) \tag{9.13}$$

It is now clear that the collision integral can't be expressed in terms of the one-particle distribution function. Indeed, as the name suggests, the collision integral captures the interactions (or collisions) of one particle with another. Yet f_1 contains no information about where any of the other particles are in relation to the first. However some of that information is contained in the two-particle distribution function

$$f_2(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{p}_1, \boldsymbol{p}_2; t) \equiv N(N-1) \int \prod_{i=3}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \, f(\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{p}_1, \boldsymbol{p}_2, ...; t) \tag{9.14}$$

With this definition, the collision integral is written simply as

$$\left(\frac{\partial f_1}{\partial t}\right)_{coll} = \int d\boldsymbol{r}_2 \, d\boldsymbol{p}_2 \, \frac{\partial V_{int}(\boldsymbol{r} - \boldsymbol{r}_2)}{\partial \boldsymbol{r}} \cdot \frac{\partial f_2}{\partial \boldsymbol{p}} \tag{9.15}$$

The upshot of all of this is that if you want to know how the one-particle distribution function evolves, you also need to know something about the two-particle distribution function. Repeating the same calculation, it is not hard to demonstrate that f_2 evolves in time by a Liouville-like equation with a corrective term that depends on the three particle distribution function f_3, and so on. In general, the n-particle distribution function

$$f_n(\boldsymbol{r}_1, ..., \boldsymbol{r}_n, \boldsymbol{p}_1, ..., \boldsymbol{p}_n; t) = \frac{N!}{(N-n)!} \int \prod_{i=n+1}^{N} d\boldsymbol{r}_i \, d\boldsymbol{p}_i \, f(\boldsymbol{r}_1, ..., \boldsymbol{r}_N, \boldsymbol{p}_1, ..., \boldsymbol{p}_N; t) \tag{9.16}$$

obeys the equation

$$\frac{\partial f_n}{\partial t} = \{H_n, f_n\} + \sum_{i=1}^{N} \int d\boldsymbol{r}_{n+1} \, d\boldsymbol{p}_{n+1} \, \frac{\partial V_{int}(\boldsymbol{r}_i - \boldsymbol{r}_{n+1})}{\partial \boldsymbol{r}_i} \cdot \frac{\partial f_{n+1}}{\partial \boldsymbol{p}_i} \tag{9.17}$$

where the effective n-body hamiltonian includes the external force and any interactions between the n particles, but neglects interactions with any particles outside of this set

$$\mathcal{H}_n = \sum_{i=1}^{n} \left(\frac{\boldsymbol{p}_i^2}{2m} + v_{ext}(\boldsymbol{r}_i) \right) + \sum_{i<j\leq n} V_{int}(\boldsymbol{r}_i - \boldsymbol{r}_j) \tag{9.18}$$

The equations (9.16)-(9.18) are known as the BBGKY hierarchy, and they are telling that any group of n particles evolves in a Hamiltonian fashion, corrected by the sum of the interactions of each one of the n particles in the group with one of the particles outside that group.

9.1.2 Expression for the collision integral in a crystal

The main problem in finding a solution for the equation (9.17) is to resolve the collision integral. In particular, it would be convenient to write the collision term in (9.12) as a function of the one-particle probability distribution f_1.
In order to deal with this problem, two different time scales can be distinguished. One is the time between collision, τ, known as the scattering time or relaxation time. The second is the collision time, τ_{coll}, which is roughly the time it takes for the process of collision between particles to occur. In situations where

$$\tau \gg \tau_{coll} \tag{9.19}$$

it should be expected that, for much of the time, f_1 simply follows its hamiltonian evolution with occasional perturbations by the collisions. The collisions can be seen, at the same time, as abrupt, instantaneous events. In this regime the collision integral should reflect the rate at which these collisions occur.
To define an expression for the collision integral in a crystal system the one particle distribution function f_1 needs to be rephrased in terms of position $\boldsymbol{r}$ and wave vector $\boldsymbol{k}$ of the electrons. Then the quantity $\tau(\boldsymbol{r},\boldsymbol{k}) \equiv \tau_\alpha(\boldsymbol{r},\boldsymbol{k})$ is the relaxation time of an electron in the band α in the phase space point $(\boldsymbol{r},\boldsymbol{k})$. It can be assumed that the collisions are well localized in space and time, so that the scatterings occurring at ($\boldsymbol{r}$, t) are determined by properties of the solid in the immediate vicinity of ($\boldsymbol{r}$, t).
The scattering probability is written in terms of the quantity $W_{\boldsymbol{k},\boldsymbol{k'}}$. The probability in an infinitesimal time interval dt that an electron with wave vector $\boldsymbol{k}$ is scattered into any one of the group of levels (with the same spin) contained in the infinitesimal k-space volume element $d\boldsymbol{k'}$ about $\boldsymbol{k'}$, assuming that these levels are all unoccupied, is

$$\frac{W_{\boldsymbol{k},\boldsymbol{k'}}\, dt\, d\boldsymbol{k'}}{(2\pi)^3} \tag{9.20}$$

The particular form taken by $W_{\boldsymbol{k},\boldsymbol{k'}}$ depends on the particular scattering mechanism being described, and in general it has a quite complex structure, that may also depend on the electronic distribution function f_1.
Since $W_{\boldsymbol{k},\boldsymbol{k'}}\ d\boldsymbol{k'}/(2\pi)^3$ is the probability per unit time that an electron with wave vector $\boldsymbol{k}$ will be scattered into the group of levels with the same spin contained in $d\boldsymbol{k'}$ about $\boldsymbol{k'}$, given that these levels are all unoccupied, the actual rate of transition must be reduced by the fraction of these levels that actually are unoccupied. The total probability per unit time for a collision is given by summing over all final wave vectors $\boldsymbol{k'}$ obtaining

$$\frac{1}{\tau(\boldsymbol{r},\boldsymbol{k})} = \int \frac{d\boldsymbol{k'}}{(2\pi)^3}\ W_{\boldsymbol{k},\boldsymbol{k'}}\,[1 - f_1(\boldsymbol{r},\boldsymbol{k'};t)] \tag{9.21}$$

It is clear from (9.21) that the relaxation time $\tau(\boldsymbol{r},\boldsymbol{k})$ depends on the particular form assumed by the non equilibrium distribution function f_1. In order to define the collision integral, two different contributions to the scattering can be analyzed:
(i) Since $dt/\tau(\boldsymbol{r},\boldsymbol{k})$ is the probability that any electron in the neighborhood of $\boldsymbol{k}$ is scattered in the time interval dt, then the total number of electrons per unit volume in $d\boldsymbol{k}$ about $\boldsymbol{k}$ that suffer a collision is just $dt/\tau(\boldsymbol{r},\boldsymbol{k})$ times the number of electrons per unit volume in $d\boldsymbol{k}$ about $\boldsymbol{k}$, equal to $f_1(\boldsymbol{r},\boldsymbol{k};t)d\boldsymbol{k}/(2\pi)^3$, so that

$$I_{out}^{coll} = -\frac{f_1(\boldsymbol{r},\boldsymbol{k'};t)}{\tau(\boldsymbol{r},\boldsymbol{p})} = -f_1(\boldsymbol{r},\boldsymbol{k'};t)\int \frac{d\boldsymbol{k'}}{(2\pi)^3}\ W_{\boldsymbol{k},\boldsymbol{k'}}\,[1 - f_1(\boldsymbol{r},\boldsymbol{k'};t)] \tag{9.22}$$

where the integral has been labeled with subscript *out* because, since $d\boldsymbol{k}$ is infinitesimal, the effect of any collision of an electron in the volume is to remove it from that volume element.
(ii) Electrons not only scatter out of the level $\boldsymbol{k}$ as seen before, but are also scattered into it from other levels thanks to collisions. To evaluate this contribution, consider the electrons that, just prior

to the collision, were in the volume element $d\boldsymbol{k}'$ about $\boldsymbol{k}'$. The above-mentioned contribution will be proportional to that number of electrons, to the fraction of non occupied levels with wave vector $\boldsymbol{k}$, and to the fraction of particles with wave vector $d\boldsymbol{k}'$ about $\boldsymbol{k}'$ that would be scattered into $d\boldsymbol{k}$ about $\boldsymbol{k}$. Therefore the total number of electrons arriving in the volume element $d\boldsymbol{k}$ about $\boldsymbol{k}$ from the volume element $d\boldsymbol{k}'$ about $\boldsymbol{k}'$ as the result of a collision in the time interval dt is

$$I_{in}^{coll} = [1 - f_1(\boldsymbol{r}, \boldsymbol{k}; t)] \int \frac{d\boldsymbol{k}'}{(2\pi)^3} \, W_{\boldsymbol{k}',\boldsymbol{k}} \, f_1(\boldsymbol{r}, \boldsymbol{k}'; t) \tag{9.23}$$

The sum of these two contributions is called collision integral. Omitting the spatial of the distribution function and of the relaxation time, as well as the time dependence of the distribution function, it can be adopted the notation $f_1(\boldsymbol{r}, \boldsymbol{k}; t) \equiv f_1(\boldsymbol{k})$, $\tau(\boldsymbol{r}, \boldsymbol{k}) \equiv \tau(\boldsymbol{k})$, and it can be written

$$\left(\frac{\partial f_1(\boldsymbol{k})}{\partial t}\right)_{coll} = -\int \frac{d\boldsymbol{k}'}{(2\pi)^3} \, \{W_{\boldsymbol{k},\boldsymbol{k}'} \, f_1(\boldsymbol{k})[1 - f_1(\boldsymbol{k}')] - W_{\boldsymbol{k}',\boldsymbol{k}} \, f_1(\boldsymbol{k}')[1 - f_1(\boldsymbol{k})]\} \tag{9.24}$$

In the majority of practical cases it is used the so called "relaxation-time approximation", so that the equation (9.24) is simplified to

$$\left(\frac{\partial f_1(\boldsymbol{k})}{\partial t}\right)_{coll} = -\frac{[f_1(\boldsymbol{k}) - f_{1,eq}(\boldsymbol{k})]}{\tau(\boldsymbol{k})} \tag{9.25}$$

This approximation represents quantitatively the fact that it is the role of collisions to restore local equilibrium after a collision in a time of the order of the relaxation time. In case of electrons in metals the equilibrium distribution is given by the Fermi-Dirac function for the electronic band with energy ε

$$f_1(\boldsymbol{k}) = f_{1,eq}(\varepsilon(\boldsymbol{k}), \mu(\boldsymbol{r}), T(\boldsymbol{r})) \equiv f_{1,eq}(\varepsilon, \mu, T) = \frac{1}{e^{(\varepsilon(\boldsymbol{k}) - \xi(\boldsymbol{r}))/kT(\boldsymbol{r})} + 1} \tag{9.26}$$

where ξ is the chemical potential, T is the temperature, k is the Boltzmann constant and the notation f_1 represents the fact that the Fermi-Dirac distribution must be seen as a single electron distribution function.

9.1.3 Boltzmann equation

Boltzmann equation is given by the previous expression (9.12), that is

$$\frac{\partial f_1}{\partial t} = \{\mathcal{H}_1, f_1\} + \left(\frac{\partial f_1}{\partial t}\right)_{coll} \tag{9.27}$$

where f_1 is the single particle distribution function, that in general depends on coordinates ($\boldsymbol{r}$, $\boldsymbol{k}$) in phase space and on time t. In the relaxation-time approximation the collision integral can be written as in (9.25). In this case the Boltzmann equation becomes:

$$\frac{\partial f_1}{\partial t} = \{\mathcal{H}_1, f_1\} - \frac{[f_1 - f_{1,eq}]}{\tau} \tag{9.28}$$

where τ in general depends on $\boldsymbol{r}$ and $\boldsymbol{k}$, while $f_{1,eq}$ is the equilibrium single particle probability distribution function. Rewriting the expression (9.28) as

$$\frac{\partial f_1}{\partial t} + \frac{f_1}{\tau} = \{\mathcal{H}_1, f_1\} + \frac{f_{1,eq}}{\tau} \tag{9.29}$$

it assumes explicitly the form of a first order non-homogeneous differential equation.
Consider a physical system of a solid. Equation (9.29) can be resolved for a distribution function f_1 depending on $\boldsymbol{r}$, $\boldsymbol{k}$, on time t and associated to a given energy ε, whose general expression is $f_1 \equiv f(\varepsilon, \boldsymbol{r}, \boldsymbol{k}, t)$ (the Fermi-Dirac equilibrium distribution form (9.26) is a time-independent example of this kind of function). In order to write the solution of (9.29) in a simple way the relaxation time τ_α can be considered as independent of time t' at which the collision takes place, and the distribution function f_1 appearing in the Poisson bracket can be considered as the equilibrium distribution function for the

electronic band α: $\{\mathcal{H}_1, f_1\} \equiv \{\mathcal{H}_1, f_{eq}\}$, since this is the *streaming* term describing the evolution of distribution function in the absence of collisions. Then the solution of the differential equation allows an explicit expression for the form of the non equilibrium electronic distribution function at time t, given by

$$f(\varepsilon, \boldsymbol{r}, \boldsymbol{k}, t) = f_{eq}(\boldsymbol{r}, \boldsymbol{k}) + \int_{-\infty}^{t} dt' \; e^{-(t-t')/\tau(\varepsilon(\boldsymbol{k}))} \{\mathcal{H}_1, f_{eq}\}_{\boldsymbol{r},\boldsymbol{p}} \tag{9.30}$$

where the relaxation time depends on wave vector only through the energy term $\varepsilon(\boldsymbol{k})$.
The Poisson bracket in (9.30) can be made explicit by means of the Hamilton equations, obtaining the following expression

$$\{\mathcal{H}_1, f_{eq}\}_{\boldsymbol{r},\boldsymbol{p}} = \frac{\partial \mathcal{H}_1}{\partial \boldsymbol{r}} \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{p}} - \frac{\partial \mathcal{H}_1}{\partial \boldsymbol{p}} \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{r}} = -\dot{\boldsymbol{p}} \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{p}} - \dot{\boldsymbol{r}} \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{r}} \tag{9.31}$$

Furthermore, the semiclassical equation of motion for the particles in solids can be introduced

$$\dot{\boldsymbol{r}} = \boldsymbol{v}_\alpha(\boldsymbol{k}) = \frac{1}{\hbar} \frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial \boldsymbol{k}} \qquad ; \qquad \hbar \dot{\boldsymbol{k}} = -e \left[\mathbf{E}(\boldsymbol{r}, t) + \frac{1}{c} \boldsymbol{v}_\alpha(\boldsymbol{k}) \times \mathbf{H}(\boldsymbol{r}, t) \right] = \mathbf{F}(\boldsymbol{r}, \boldsymbol{k}, t) \tag{9.32}$$

where $\mathbf{E}$ is the electric field, $\mathbf{H}$ is the magnetic field and $\boldsymbol{v}_\alpha$ is the group velocity of the electronic band α. Substituting these semiclassical equations in (9.31), it can be written as

$$\{\mathcal{H}_1, f_{eq}\} = -\frac{1}{\hbar} \mathbf{F}(\boldsymbol{r}, \boldsymbol{k}, t) \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{k}} - \boldsymbol{v}_\alpha(\boldsymbol{k}) \cdot \frac{\partial f_{eq}}{\partial \boldsymbol{r}} \tag{9.33}$$

If the distribution function $f_{eq}(\boldsymbol{r}, \boldsymbol{k})$ is given by the Fermi Dirac distribution (9.26), the changes of variables in the derivative of the distribution function in (9.33) permit to obtain the following expression

$$\{\mathcal{H}_1, f_{eq}\} = -\frac{\partial f_{eq}}{\partial \varepsilon} \boldsymbol{v}_\alpha(\boldsymbol{k}) \cdot \left[\mathbf{F}(\boldsymbol{r}, \boldsymbol{k}, t) - \nabla \xi - \left(\frac{\varepsilon(\boldsymbol{k}) - \xi}{T} \right) \nabla T \right] \tag{9.34}$$

Substituting (9.34) in (9.30) the previous equation (9.30) becomes

$$\begin{aligned} f(\boldsymbol{r}, \boldsymbol{k}, t) = f_{eq}(\boldsymbol{r}, \boldsymbol{k}) + \int_{-\infty}^{t} dt' \; e^{-(t-t')/\tau(\varepsilon(\boldsymbol{k}))} \left(-\frac{\partial f_{eq}}{\partial \varepsilon} \right) \boldsymbol{v}_\alpha(\boldsymbol{k}(t')) \cdot \\ \cdot \left[\mathbf{F}(\boldsymbol{r}, \boldsymbol{k}, t') - \nabla \xi(t') - \left(\frac{\varepsilon(\boldsymbol{k}) - \xi}{T} \right) \nabla T(t') \right] \end{aligned} \tag{9.35}$$

Restricting the equation (9.35) to the case of zero magnetic field, $\mathbf{H} = \mathbf{0}$, the force $\mathbf{F}(\boldsymbol{r}, \boldsymbol{k}, t')$ depends only on electric field. Furthermore, in the absence of magnetic fields the wave vector $\boldsymbol{k}(t')$ will be time independent, reducing to $\boldsymbol{k}$, and the integral in (9.35) becomes trivial for static electric field and temperature gradients. In addition, if the electric field is spatially uniform (so $\mathbf{E}(\mathbf{r}) = \mathbf{E}$) and the temperature is constant throughout the system, the previous equation can be reduced to

$$f(\boldsymbol{k}) \approx f_{eq}(\boldsymbol{k}) - e\mathbf{E} \cdot \boldsymbol{v}_\alpha(\boldsymbol{k}) \, \tau(\varepsilon(\boldsymbol{k})) \left(-\frac{\partial f_{eq}}{\partial \varepsilon} \right) \tag{9.36}$$

In the following this relaxation-time approximation formula will be used to calculate the transport properties of interest for solid state systems.

9.1.4 Current density and electrical conductivity

The number of electrons per unit volume in the infinitesimal three-dimensional element $d\boldsymbol{k}$ is equal to $f(\boldsymbol{k})d\boldsymbol{k}/4\pi^3$. The current density in the electronic band α is then given by

$$\boldsymbol{j}_\alpha = -e \int \frac{d\boldsymbol{k}}{4\pi^3} \boldsymbol{v}_\alpha(\boldsymbol{k}) \, f(\boldsymbol{k}) \tag{9.37}$$

where the band index dependence of the velocity is been transported in the parenthesis, i.e. $\boldsymbol{v}_\alpha(\boldsymbol{k})$, in order to avoid confusion with the velocity components. Since each partially filled band makes such a

contribution to the current density, the total current density is the sum of these contributions over all bands. Inserting the form (9.36) in the equation (9.37), the following relation between current density and electric field can be obtained

$$j_\mu^\alpha = \sum_\nu \sigma_{\mu\nu}^\alpha E_\nu \tag{9.38}$$

where $\boldsymbol{\sigma}^\alpha$ is a two-rank tensor. The explicit expression for the conductivity tensor components related to the electronic band α is given by

$$\sigma_{\mu\nu}^\alpha = e^2 \int \frac{d\boldsymbol{k}}{4\pi^3}\, \tau(\varepsilon_\alpha(\boldsymbol{k}))\, v_\mu(\alpha,\boldsymbol{k})\, v_\nu(\alpha,\boldsymbol{k}) \left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)_{\varepsilon=\varepsilon_\alpha(\boldsymbol{k})}. \tag{9.39}$$

where the velocities are defined by deriving the energy band structure with respect to a given $\boldsymbol{k}$ point in a given direction, that is

$$\dot{r}_\mu = v_\mu(\alpha,\boldsymbol{k}) = \frac{1}{\hbar}\frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial k_\mu} \tag{9.40}$$

The total electrical conductivity is the sum over all partially filled bands that contribute to the current density (9.37), i.e.

$$\sigma_{\mu\nu} = \sum_\alpha \sigma_{\mu\nu}(\alpha). \tag{9.41}$$

If $\mathbf{j}$ is parallel to $\mathbf{E}$ the electrical conductivity tensor $\boldsymbol{\sigma}$ is diagonal: $\sigma_{\mu\nu} = \sigma\delta_{\mu\nu}$. This is generally true in crystals of cubic symmetry. Furthermore, since the Fermi function f_α^{eq} has negligible derivative except when the energy ε is within kT of the Fermi energy ε_F, filled bands make no contribution to the conductivity, as expected. Finally, because of the dependence of the Fermi-Dirac distribution on the chemical potential and the temperature, the conductivity can be viewed as a function of these parameters, $\sigma(\mu, T)$. Mapping the integral (9.39) in the energy domain, the electrical conductivity results

$$\sigma_{\mu\nu}(\xi,T) = e^2 \int_{-\infty}^{\infty} d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial \varepsilon}\right) \Xi_{\mu\nu}(\varepsilon), \tag{9.42}$$

with the definition of the quantity

$$\Xi_{\mu\nu}(\varepsilon) = \frac{1}{v}\sum_{\alpha,\boldsymbol{k}} \tau(\varepsilon_\alpha(\boldsymbol{k}))\, v_\mu(\alpha,\boldsymbol{k})\, v_\nu(\alpha,\boldsymbol{k})\; \delta(\varepsilon - \varepsilon_\alpha(\boldsymbol{k})), \tag{9.43}$$

where v is the volume of the Brillouin zone, and the summation is over all bands α and over all the Brillouin zone. The quantity $\Xi_{\mu\nu}(\varepsilon)$ is called transport distribution function (TDF). In first-principles calculations is assumed that the lifetime $\tau(\varepsilon_\alpha(\boldsymbol{k}))$ is independent both of α and $\boldsymbol{k}$, and is chosen the value $\tau = \tau(\varepsilon_\alpha(\boldsymbol{k}))$ by fitting the experimental values for, e.g., the experimental electrical conductivity at a given temperature.

The derivative of the Fermi-Dirac equilibrium function is easily computed analytically starting from the form of the equilibrium function

$$f_{eq}(\varepsilon,\xi,T) = \frac{1}{e^{(\varepsilon(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}+1} \qquad \rightarrow \qquad \frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial \varepsilon} = -\frac{1}{kT}\left(\frac{e^{(\varepsilon(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}}{[e^{(\varepsilon(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}+1]^2}\right) \tag{9.44}$$

so that equation (9.42) can be written more explicitly as

$$\sigma_{\mu\nu}(\xi,T) = \frac{e^2}{kT}\int_{-\infty}^{\infty} d\varepsilon\; \frac{e^{(\varepsilon(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}}{[e^{(\varepsilon(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}+1]^2}\; \Sigma_{\mu\nu}(\varepsilon) \tag{9.45}$$

In practical implementation, the integral in the energy domain in equation (9.45) is replaced by a discrete summation, in the following way

$$\sigma_{\mu\nu}(\xi,T) = e^2 \sum_{i=1}^{n_\varepsilon}\left(-\frac{\partial f_{eq}(\varepsilon_i,\xi,T)}{\partial \varepsilon_i}\right)\Xi_{\mu\nu}(\varepsilon_i)\;\Delta\varepsilon_i = \frac{e^2}{kT}\;\sum_{i=1}^{n_\varepsilon}\frac{e^{(\varepsilon_i(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}}{[e^{(\varepsilon_i(\boldsymbol{k})-\xi(\boldsymbol{r}))/kT(\boldsymbol{r})}+1]^2}\;\Sigma_{\mu\nu}(\varepsilon_i)\;\Delta\varepsilon_i \tag{9.46}$$

9.1.4.1 Generalization to a multiband approach

Transport in materials is described assuming that electrons can be treated as free particles whose dispersion is the actual band structure of the material. One typically relies on the semiclassical Boltzmann equation within the standard long-wave approximation, for which Bloch electrons are presented as localized wave packets comprised of a single band near the Fermi level. The theory further relies on the equations of motion for the wave packet center in order to derive various transport properties. Realistic band structures of materials, however, can have several bands near the Fermi level and can contain crossing points or local degeneracies.
On the other hand, several researchers have considered Bloch electron wave packets extending over multiple energy bands (see, for example, Ref. [146]). Indeed, it has been shown that the equations of motion for the centers of such multiband wave packets are different than those in the single band wave packet case. In this framework, the semiclassical Boltzmann needs to be reconsidered and reviewed in light of these discoveries. To this end, research findings strongly suggest that the single band semiclassical transport theory may need to be expanded to capture multiband effects from the materials electronic structure.
The starting point is the approximate equation for the distribution function

$$f(\boldsymbol{k}) \approx f_{eq}(\boldsymbol{k}) - e\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right) \sum_{\alpha=1}^{n_d} \mathbf{E}\cdot \boldsymbol{v}_\alpha(\boldsymbol{k})\, \tau(\varepsilon_\alpha(\mathbf{k})) \tag{9.47}$$

where n_d is the number of degenerate states with energy $\varepsilon_\alpha(\boldsymbol{k})$. In the following, this relaxation-time approximation formula will be used to calculate the transport properties of interest for solid state systems. The number of electrons per unit volume in the infinitesimal three-dimensional element $d\boldsymbol{k}$ is equal to $f(\boldsymbol{k})d\boldsymbol{k}/4\pi^3$. The current density in the electronic band α is then given by

$$\boldsymbol{j} = -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f(\boldsymbol{k}) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k}) \tag{9.48}$$

Substituting the expression (9.47) for the distribution function in (9.48) gives

$$\boldsymbol{j} = -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + e^2\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\mathbf{E}\cdot\left[\sum_{\alpha=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k})\,\tau(\varepsilon_\alpha(\boldsymbol{k})) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k})\right] \tag{9.49}$$

The manipulation of the previous formula leads to the expression

$$\begin{aligned}
\boldsymbol{j} &= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + e^2\,\tau\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\mathbf{E}\cdot\left[\sum_{\alpha=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k})\right] \\
&= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + e^2\,\tau\,\mathbf{E}\cdot\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\sum_{\alpha=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k}) \\
&= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + \mathbf{E}\cdot\left[e^2\,\tau\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\sum_{\alpha=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k})\right]
\end{aligned} \tag{9.50}$$

Therefore, taking the component λ of the previous equation that describes the current density gives

$$\begin{aligned}
j_\lambda &= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + \left\{e^2\,\tau\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\left[\sum_{\alpha=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \sum_{\beta=1}^{n_d} \boldsymbol{v}_\beta(\boldsymbol{k})\right]_{\lambda\nu}\right\}\cdot E_\nu \\
&= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + \left\{e^2\,\tau\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\left[\sum_{\alpha=1}^{n_d}\sum_{\beta=1}^{n_d} \boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \boldsymbol{v}_\beta(\boldsymbol{k})\right]_{\lambda\nu}\right\}\cdot E_\nu \\
&= -e\int \frac{d\boldsymbol{k}}{4\pi^3}\, f_{eq}(\boldsymbol{k}) + \left\{e^2\,\tau\int \frac{d\boldsymbol{k}}{4\pi^3}\left(-\frac{\partial f_{eq}}{\partial \varepsilon}\right)\sum_{\alpha=1}^{n_d}\sum_{\beta=1}^{n_d}[\boldsymbol{v}_\alpha(\boldsymbol{k}) \otimes \boldsymbol{v}_\beta(\boldsymbol{k})]_{\lambda\nu}\right\}\cdot E_\nu
\end{aligned} \tag{9.51}$$

Finally, by applying the definition of the tensor product in the previous expression, a glimpse on the form of the current density component which takes into account also degenerate states can be given by the formula

$$j_\lambda = -e \int \frac{d\boldsymbol{k}}{4\pi^3} f_{eq}(\boldsymbol{k}) + \left\{ e^2 \tau \int \frac{d\boldsymbol{k}}{4\pi^3} \left(-\frac{\partial f_{eq}}{\partial \varepsilon} \right) \left[\sum_{\alpha=1}^{n_d} v_{\lambda,\alpha}(\boldsymbol{k}) \sum_{\beta=1}^{n_d} v_{\nu,\beta}(\boldsymbol{k}) \right] \right\} \cdot E_\nu \tag{9.52}$$

Since each partially filled band makes such a contribution to the current density, the total current density is the sum of these contributions over all bands. Inserting the form (9.47) in the equation (9.48), the following relation between current density and electric field can be obtained

$$j_\mu = \sum_\nu \sigma_{\mu\nu} E_\nu \tag{9.53}$$

where $\boldsymbol{\sigma}$ is a two-rank tensor.
The explicit expression for the conductivity tensor components is thus given by

$$\sigma_{\mu\nu} = e^2 \tau \int \frac{d\mathbf{k}}{4\pi^3} \left(-\frac{\partial f_{eq}}{\partial \varepsilon} \right) \left[\sum_{\alpha=1}^{n_d} v_{\lambda,\alpha}(\mathbf{k}) \sum_{\beta=1}^{n_d} v_{\nu,\beta}(\mathbf{k}) \right] \tag{9.54}$$

9.2 Band velocities calculation in the CRYSTAL code

Band velocities, defined in equation (9.40) and used in the transport distribution function (9.43) for the calculation of electronic properties, can be obtained in practical implementations according to the Pople analytical method,[147, 18] which consists of differentiating by the wave vector $\boldsymbol{k}$ the generalized eigenvalue problem and the orthogonalization condition. Therefore, the starting point are the expressions for the generalized eigenvalue problem and the orthogonalization condition associated, respectively given by

$$\mathbf{F}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) = \mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) \tag{9.55}$$

$$\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) = \mathbb{1} \tag{9.56}$$

Differentiating both equations (9.55) and (9.56) with respect to the wave vector $\boldsymbol{k}$ leads to

$$\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{F}(\boldsymbol{k})\dot{\mathbf{C}}(\boldsymbol{k}) = \dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \mathbf{S}(\boldsymbol{k})\dot{\mathbf{C}}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\dot{\mathbf{E}}(\boldsymbol{k}) \tag{9.57}$$

$$\dot{\mathbf{C}}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\dot{\mathbf{C}}(\boldsymbol{k}) = \mathbb{0} \tag{9.58}$$

where the dot over the matrices indicates the derivatives with respect to the wave vector $\boldsymbol{k}$. Using a localized Gaussian-type basis set for the expansion of the Kohn-Sham one-electron orbitals, the Fock and the overlap matrices are computed in the direct space. From the knowledge of the cell-dependent Fock and overlap elements in the direct space, their expression in the reciprocal space can be obtained by applying the Fourier transforms, so that the $\boldsymbol{k}$ vector derivatives of the Fock and overlap matrices can be easily obtained as

$$\dot{\mathbf{F}}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\}} i\,\boldsymbol{g}\,\mathbf{F}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} \tag{9.59}$$

and similarly,

$$\dot{\mathbf{S}}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\}} i\,\boldsymbol{g}\,\mathbf{S}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} \tag{9.60}$$

where $\boldsymbol{g}$ is a lattice vector in direct space, while $\mathbf{F}(\boldsymbol{g})$ and $\mathbf{S}(\boldsymbol{g})$ are the Fock and the overlap matrices in the direct space, respectively. By assuming that

$$\dot{\mathbf{C}}(\boldsymbol{k}) = \mathbf{C}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) \tag{9.61}$$

and by multiplying equation (9.57) on the left by $\mathbf{C}^{\dagger}(\boldsymbol{k})$, the adjoint of $\mathbf{C}(\boldsymbol{k})$, the differentiation of the eigenvalue problem (9.57) can be rewritten as

$$\begin{aligned}\mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\underbrace{\mathbf{F}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k})}\mathbf{U}(\boldsymbol{k}) &= \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) \\ &\quad + \mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\dot{\mathbf{E}}(\boldsymbol{k})\end{aligned}$$

$$\begin{aligned}\mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \underbrace{\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbb{1}}\mathbf{E}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) &= \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \underbrace{\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbb{1}}\mathbf{U}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) \\ &\quad + \underbrace{\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbb{1}}\dot{\mathbf{E}}(\boldsymbol{k})\end{aligned} \tag{9.62}$$

Using the assumption (9.61) also in equation (9.58), the differentiation of the orthogonalization condition (9.58) can be rewritten as

$$[\mathbf{C}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k})]^{\dagger}\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) = \mathbb{0} \tag{9.63}$$

$$\mathbf{U}^{\dagger}(\boldsymbol{k})\underbrace{\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbb{1}} + \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \underbrace{\mathbf{C}^{\dagger}(\boldsymbol{k})\mathbf{S}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})}_{=\mathbb{1}}\mathbf{U}(\boldsymbol{k}) = \mathbb{0} \tag{9.64}$$

The orthogonalization condition (9.56) permits to simplify the previous equations (9.62) and (9.64) respectively as follows

$$\mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{E}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) = \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \mathbf{U}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \dot{\mathbf{E}}(\boldsymbol{k}) \tag{9.65}$$

$$\mathbf{U}^{\dagger}(\boldsymbol{k}) + \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) + \mathbf{U}(\boldsymbol{k}) = \mathbb{0} \tag{9.66}$$

Using the following definitions

$$\mathbf{G}(\boldsymbol{k}) = \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) \qquad \text{and} \qquad \mathbf{R}(\boldsymbol{k}) = \mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k}) \tag{9.67}$$

the differential form of the eigenvalue problem (9.65) and the differential form of the orthogonalization condition (9.66) becomes

$$\mathbf{G}(\boldsymbol{k}) + \mathbf{E}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) = \mathbf{R}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \mathbf{U}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) + \dot{\mathbf{E}}(\boldsymbol{k}) \tag{9.68}$$

$$\mathbf{U}^{\dagger}(\boldsymbol{k}) + \mathbf{R}(\boldsymbol{k}) + \mathbf{U}(\boldsymbol{k}) = \mathbb{0} \tag{9.69}$$

Starting from these last two equations, the desired values in the matrices $\mathbf{U}(\boldsymbol{k})$ and $\dot{\mathbf{E}}(\boldsymbol{k})$ can thus be obtained. In particular, the diagonal terms of $\mathbf{U}(\boldsymbol{k})$ matrix can be derived from the differentiation of the orthogonalization condition (9.69) as

$$\mathbf{U}^{\dagger}(\boldsymbol{k}) + \mathbf{U}(\boldsymbol{k}) = -\mathbf{R}(\boldsymbol{k}) \qquad \rightarrow \qquad U_{\alpha\alpha}(\boldsymbol{k}) = -\frac{1}{2}\, R_{\alpha\alpha}(\boldsymbol{k})$$

while the off-diagonal terms of the same matrix $\mathbf{U}(\boldsymbol{k})$ can be derived from the differentiation expression of the generalized eigenvalue problem (9.68) as

$$\mathbf{U}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) - \mathbf{E}(\boldsymbol{k})\mathbf{U}(\boldsymbol{k}) = \mathbf{G}(\boldsymbol{k}) - \mathbf{R}(\boldsymbol{k})\mathbf{E}(\boldsymbol{k}) - \dot{\mathbf{E}}(\boldsymbol{k})$$

$$U_{\alpha\beta}(\boldsymbol{k})\varepsilon_{\beta}(\boldsymbol{k}) - \varepsilon_{\alpha}(\boldsymbol{k})U_{\alpha\beta}(\boldsymbol{k}) = G_{\alpha\beta}(\boldsymbol{k}) - R_{\alpha\beta}(\boldsymbol{k})\varepsilon_{\beta}(\boldsymbol{k}) - \dot{E}_{\alpha\beta}(\boldsymbol{k}) \quad \rightarrow \quad U_{\alpha\beta}(\boldsymbol{k}) = \frac{G_{\alpha\beta}(\boldsymbol{k}) - R_{\alpha\beta}(\boldsymbol{k})\varepsilon_{\beta}(\boldsymbol{k})}{\varepsilon_{\beta}(\boldsymbol{k}) - \varepsilon_{\alpha}(\boldsymbol{k})}$$

where α, β are the band indexes, and it has been supposed that the matrices $\mathbf{E}(\boldsymbol{k})$ and its derivative with respect to the wave vectors $\dot{\mathbf{E}}(\boldsymbol{k})$ are both diagonal matrices, so that the values $\varepsilon_n(\boldsymbol{k})$ (with $n = \alpha, \beta$) are the diagonal elements of the matrix $\mathbf{E}(\boldsymbol{k})$, and the off-diagonal elements $E_{\alpha\beta}(\boldsymbol{k})$ and $\dot{E}_{\alpha\beta}(\boldsymbol{k})$ are all equal to zero, that is $E_{\alpha\beta}(\boldsymbol{k}) = \dot{E}_{\alpha\beta}(\boldsymbol{k}) = 0$.
Therefore, the diagonal elements $\dot{\varepsilon}_n(\boldsymbol{k})$ (with $n = \alpha, \beta$) of the matrix $\dot{\mathbf{E}}(\boldsymbol{k})$, which are the band velocities defined in (9.40), can be computed considering the diagonal elements of the matrix expression for the differentiation of the generalized eigenvalue problem (9.68), so as

$$\begin{aligned} G_{\alpha\alpha}(\boldsymbol{k}) + \varepsilon_{\alpha}(\boldsymbol{k})U_{\alpha\alpha}(\boldsymbol{k}) &= R_{\alpha\alpha}(\boldsymbol{k})\varepsilon_{\alpha}(\boldsymbol{k}) + U_{\alpha\alpha}(\boldsymbol{k})\varepsilon_{\alpha}(\boldsymbol{k}) + \dot{\varepsilon}_{\alpha}(\boldsymbol{k}) \\ G_{\alpha\alpha}(\boldsymbol{k}) = R_{\alpha\alpha}(\boldsymbol{k})\varepsilon_{\alpha}(\boldsymbol{k}) + \dot{\varepsilon}_{\alpha}(\boldsymbol{k}) \qquad &\rightarrow \qquad \dot{\varepsilon}_{\alpha}(\boldsymbol{k}) = G_{\alpha\alpha}(\boldsymbol{k}) - R_{\alpha\alpha}(\boldsymbol{k})\varepsilon_{\alpha}(\boldsymbol{k}) \end{aligned}$$

Resuming the results obtained, the desired values for the elements of the matrices $\mathbf{U}(\boldsymbol{k})$ and $\dot{\mathbf{E}}(\boldsymbol{k})$ can be obtained using the following three equations

$$U_{\alpha\alpha}(\boldsymbol{k}) = -\frac{1}{2}\, R_{\alpha\alpha}(\boldsymbol{k}) \tag{9.70}$$

$$U_{\alpha\beta}(\boldsymbol{k}) = \frac{G_{\alpha\beta}(\boldsymbol{k}) - R_{\alpha\beta}(\boldsymbol{k})\varepsilon_{\beta}(\boldsymbol{k})}{\varepsilon_{\beta}(\boldsymbol{k}) - \varepsilon_{\alpha}(\boldsymbol{k})} \tag{9.71}$$

$$\dot{\varepsilon}_{\alpha}(\boldsymbol{k}) = G_{\alpha\alpha}(\boldsymbol{k}) - R_{\alpha\alpha}(\boldsymbol{k})\varepsilon_{\alpha}(\boldsymbol{k}) \tag{9.72}$$

where α, β are the band indexes (quantum numbers). Equation (9.72) is particularly important, because it defines a practical way to compute the band velocities (9.40) starting from the values of the diagonal elements of the Fock, the overlap and the coefficients matrices which are the results of the self-consistent procedure towards the electronic ground state. Indeed, using the definitions (9.67), equation (9.72) can be written more explicitly as

$$\dot{\varepsilon}_{\alpha}(\boldsymbol{k}) = [\mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{F}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^{\dagger}(\boldsymbol{k})\dot{\mathbf{S}}(\boldsymbol{k})\mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}\, \varepsilon_{\alpha}(\boldsymbol{k}) \tag{9.73}$$

Equation (9.73) is the general expression for the derivative of the band structure form with respect to a generic wave vector. Indeed, the band velocities expression (9.40) can be recovered by making the derivative functional form in equation (9.73) more explicit, that is

$$\frac{\partial \varepsilon_{\alpha}(\boldsymbol{k})}{\partial k_{\mu}} = \hbar\, v_{\mu,\alpha}(\boldsymbol{k}) = \left[\mathbf{C}^{\dagger}(\boldsymbol{k})\, \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_{\mu}}\, \mathbf{C}(\boldsymbol{k})\right]_{\alpha\alpha} - \left[\mathbf{C}^{\dagger}(\boldsymbol{k})\, \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_{\mu}}\, \mathbf{C}(\boldsymbol{k})\right]_{\alpha\alpha} \varepsilon_{\alpha}(\boldsymbol{k}) \qquad \mu = x, y, z \tag{9.74}$$

where $\mu = x, y, z$ are the three spatial Cartesian directions in the reciprocal space. Taking into account the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$, which is a diagonal matrix with the α-th elements on the diagonal equal to the eigenvalue $\varepsilon_\alpha(\boldsymbol{k})$, the previous expression can be equivalently rewritten as

$$\frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial k_\mu} = \hbar\, v_{\mu,\alpha}(\boldsymbol{k}) = \left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\right]_{\alpha\alpha} - \left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})\right]_{\alpha\alpha} \qquad \mu = x, y, z \quad (9.75)$$

Therefore, the band velocities are given by

$$v_{\mu,\alpha}(\boldsymbol{k}) = \frac{1}{\hbar}\frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial k_\mu} = \frac{1}{\hbar}\left\{\left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\right]_{\alpha\alpha} - \left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})\right]_{\alpha\alpha}\right\} \quad (9.76)$$

where $\mu = x, y, z$ are the three spatial Cartesian directions in the reciprocal space.

The band velocities are real numbers. In order to demonstrate this property of the band velocities, it is convenient to perform the calculation with the whole matrices in the expression, i.e. considering the formula

$$\hbar\, \boldsymbol{v}_\mu(\boldsymbol{k}) = \mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \qquad \mu = x, y, z \quad (9.77)$$

and then studying the final equations by restricting the case to only the diagonal elements of the matrices involved. Furthermore, the notation can be further simplified by expressing the derivatives of the Kohn-Sham and of the overlap matrices with respect to the k_μ components in reciprocal space (with $\mu = x, y, z$) in a shortest notation such as

$$\hbar\, \boldsymbol{v}_\mu(\boldsymbol{k}) = \mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \qquad \mu = x, y, z \quad (9.78)$$

The overlap $\mathbf{C}(\boldsymbol{k})$, the Kohn-Sham derivatives $\partial_\mu \mathbf{F}(\boldsymbol{k})$ and the overlap derivatives $\partial_\mu \mathbf{S}(\boldsymbol{k})$ matrices contain, in general, complex numbers, and can therefore be written as the sum of a real and an imaginary part, in the following way

$$\mathbf{C}(\boldsymbol{k}) = \mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k}) \qquad \partial_\mu \mathbf{F}(\boldsymbol{k}) = \partial_\mu \mathbf{F}_r(\boldsymbol{k}) + i\,\partial_\mu \mathbf{F}_i(\boldsymbol{k}) \qquad \partial_\mu \mathbf{S}(\boldsymbol{k}) = \partial_\mu \mathbf{S}_r(\boldsymbol{k}) + i\,\partial_\mu \mathbf{S}_i(\boldsymbol{k})$$

where the first term of each expression is a purely real matrix and the second term is a purely real matrix multiplied by the imaginary unit i to give the purely imaginary part of the associated matrix on the left hand sides. On the other hand, the eigenvalues are purely real, so that the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$ has to be considered as a purely real numbers diagonal matrix. Taking into account the formula (9.78) and expanding each term as a sum of its purely real and purely imaginary parts, as previously reported, leads to the following expression

$$\begin{aligned}
&\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \\
&= [\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})]^\dagger [\partial_\mu \mathbf{F}_r(\boldsymbol{k}) + i\,\partial_\mu \mathbf{F}_i(\boldsymbol{k})][\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})] \\
&\quad - [\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})]^\dagger [\partial_\mu \mathbf{S}_r(\boldsymbol{k}) + i\,\partial_\mu \mathbf{S}_i(\boldsymbol{k})][\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})]\ \mathbf{E}(\boldsymbol{k}) \\
&= [\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k}) + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k}) - i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k}) + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})][\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})] \\
&\quad - [\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k}) + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k}) - i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k}) + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})][\mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_i(\boldsymbol{k})]\ \mathbf{E}(\boldsymbol{k}) \\
&= \mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k}) + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k}) - i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k}) + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k}) \\
&\quad + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k}) - \mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k}) + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k}) + i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k}) \\
&\quad - [\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) - i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \\
&\quad + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) + i\,\mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) - \mathbf{C}_r^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \\
&\quad + \mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) + i\,\mathbf{C}_i^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]
\end{aligned} \quad (9.79)$$

where the property of the conjugate transpose operation applied on two general matrices

$$(\mathbf{A} + \mathbf{B})^\dagger = \mathbf{A}^\dagger + \mathbf{B}^\dagger \quad (9.80)$$

which is valid for any two matrices $\mathbf{A}$ and $\mathbf{B}$ of the same dimensions, has been used (indeed, the conjugate transpose respects addition). Therefore, collecting all the purely real matrix products in a square parenthesis and the remaining purely imaginary matrix products in a second square parenthesis results in

$$\begin{aligned}
&\mathbf{C}^{\dagger}(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k}) - \mathbf{C}^{\dagger}(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\
&= [\;{}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) + {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) - {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) + {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) \\
&\quad - {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) - {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) + {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\
&\quad - {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})\;] \\
&+ i\,[\;{}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) - {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) + {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) + {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) \\
&\quad - {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) + {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) - {}^{t}\mathbf{C}_r(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\
&\quad - {}^{t}\mathbf{C}_i(\boldsymbol{k})\,\partial_{\mu}\mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})\;]
\end{aligned} \tag{9.81}$$

where it is used the fact that the conjugate transpose operations in (9.79) coincides with the simpler transpose operation because all the matrices in the final expression in equation (9.79) (and so in (9.81)) contain purely real numbers representing the coefficients associated to the real and the imaginary part. Therefore, the first and second square brackets in equation (9.81) identify, respectively, the matrix products that give the purely real terms and the matrix products which give the purely imaginary part of the band velocities.

9.2.1 Properties of reciprocal space representation of $\partial_{\mu}\mathbf{F}$ and $\partial_{\mu}\mathbf{S}$ matrices

First of all, the properties of the derivative of the Kohn-Sham and overlap matrices has to be computed. In particular, it can be demonstrated that the derivative matrices are Hermitian. Indeed, let $F_{\gamma\rho}(\boldsymbol{g})$ and $S_{\gamma\rho}(\boldsymbol{g})$ be the (γ,ρ) elements of the direct space representation of respectively the Kohn-Sham and overlap matrices at lattice vector $\boldsymbol{g}$, and similarly for $F_{\gamma\rho}(\boldsymbol{k})$ and $S_{\gamma\rho}(\boldsymbol{k})$ for the reciprocal space representation. A couple of useful properties of the direct space representation are

$$F_{\gamma\rho}(\boldsymbol{g}) = F_{\rho\gamma}(-\boldsymbol{g}) \quad \text{and} \quad S_{\gamma\rho}(\boldsymbol{g}) = S_{\rho\gamma}(-\boldsymbol{g}) \qquad \text{i.e. simmetry} \tag{9.82}$$

$$F_{\gamma\rho}(\boldsymbol{g}) \in \mathcal{R} \quad \text{and} \quad S_{\gamma\rho}(\boldsymbol{g}) \in \mathcal{R} \quad \forall\ \gamma,\ \rho,\ \boldsymbol{g} \qquad \text{i.e. matrices reality} \tag{9.83}$$

that is, the direct space Kohn-Sham and overlap matrices are symmetric and purely real. In order to compute the properties of the matrices when a Fourier Transform operation is applied on them, the calculations in the following will be carried out for the Kohn-Sham matrix (the same result can be then applied to the overlap matrix). The summation for the Fourier Transform can be split into the zero $\boldsymbol{g}$ vector case, a sum over positive $\boldsymbol{g}$ vectors and a sum over negative $\boldsymbol{g}$ vectors. The last two parts are most conveniently done by both going over all $\boldsymbol{g}$ vectors except the zero vector, and halving the result, obtaining

$$\begin{aligned}
F_{\gamma\rho}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\}} F_{\gamma\rho}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} &= F_{\gamma\rho}(\boldsymbol{0})\,e^{i\boldsymbol{k}\cdot\boldsymbol{0}} + \frac{1}{2}\sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\,F_{\gamma\rho}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} + F_{\gamma\rho}(-\boldsymbol{g})\,e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\
&= F_{\gamma\rho}(\boldsymbol{0}) + \frac{1}{2}\sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\,F_{\gamma\rho}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} + F_{\gamma\rho}(-\boldsymbol{g})\,e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,]
\end{aligned} \tag{9.84}$$

Now consider the diagonal elements of the Kohn-Sham matrix above:

$$\begin{aligned}
F_{\gamma\gamma}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\}} F_{\gamma\gamma}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} &= F_{\gamma\gamma}(\boldsymbol{0}) + \frac{1}{2}\sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\,F_{\gamma\gamma}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} + F_{\gamma\gamma}(-\boldsymbol{g})\,e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\
&= F_{\gamma\gamma}(\boldsymbol{0}) + \frac{1}{2}\sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\,F_{\gamma\gamma}(\boldsymbol{g})\,e^{i\boldsymbol{k}\cdot\boldsymbol{g}} + F_{\gamma\gamma}(\boldsymbol{g})\,e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\
&= F_{\gamma\gamma}(\boldsymbol{0}) + \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} F_{\gamma\gamma}(\boldsymbol{g})\,\Re(e^{i\boldsymbol{k}\cdot\boldsymbol{g}})
\end{aligned} \tag{9.85}$$

Since the Kohn-Sham and the overlap matrices have the same properties (9.82) and (9.83) in the real space, then the previous procedure can be applied as well to the overlap matrix, obtaining the same result. Therefore, since the elements of the Kohn-Sham and the overlap matrices are real in the direct space, the previous calculations show that the diagonal elements of $F_{\gamma\rho}(\boldsymbol{k})$ and $S_{\gamma\rho}(\boldsymbol{k})$ are purely real at all the reciprocal space points $\boldsymbol{k}$.
By differentiating the Kohn-Sham matrix elements with respect to the components of the reciprocal space lattice vectors,

$$\begin{aligned}\frac{\partial F_{\gamma\rho}(\boldsymbol{k})}{\partial k_\mu} \equiv \partial_\mu F_{\gamma\rho}(\boldsymbol{k}) &= \sum_{\{\boldsymbol{g}\}} i\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} \\ &= i\,\mathbf{0}\, F_{\gamma\rho}(\mathbf{0})\, e^{i\boldsymbol{k}\cdot\mathbf{0}} + \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\gamma\rho}(-\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\ &= \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\gamma\rho}(-\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\ &= \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \qquad \mu = x, y, z\end{aligned} \tag{9.86}$$

the following expression is obtained

$$\partial_\mu F_{\gamma\rho}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \qquad \mu = x, y, z \tag{9.87}$$

where the symmetry property (9.82) of the Kohn-Sham matrix has been used. At the same time, the conjugate transpose of each element (9.87) is given by

$$\begin{aligned}\frac{\partial F^*_{\rho\gamma}(\boldsymbol{k})}{\partial k_\mu} \equiv \partial_\mu F^*_{\rho\gamma}(\boldsymbol{k}) &= -\sum_{\{\boldsymbol{g}\}} i\, g_\mu\, F^*_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} = -\sum_{\{\boldsymbol{g}\}} i\, g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} \\ &= -i\,\mathbf{0}\, F_{\rho\gamma}(\mathbf{0})\, e^{-i\boldsymbol{k}\cdot\mathbf{0}} - \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(-\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\ &= \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\rho\gamma}(-\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \\ &= \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] \overset{\text{eq. (9.87)}}{=} \partial_\mu F_{\gamma\rho}(\boldsymbol{k})\end{aligned} \tag{9.88}$$

that is equal to the element of the matrix in (9.87) without the conjugate transpose. It follows that the derivative of the reciprocal space representation of the Kohn-Sham matrix with respect to a reciprocal space point component is an Hermitian matrix. A well-known property of the Hermitian matrices is to have only real elements on the main diagonal. This can be demonstrated very easily in the specific case of the Kohn-Sham matrix derivative, by taking the diagonal elements in equation (9.87)

$$\partial_\mu F_{\gamma\gamma}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} [\, g_\mu\, F_{\gamma\gamma}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\gamma\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,] = i \sum_{\{\boldsymbol{g}\neq\mathbf{0}\}} g_\mu\, F_{\gamma\gamma}(\boldsymbol{g})\, \Im(e^{i\boldsymbol{k}\cdot\boldsymbol{g}}) \quad \in \quad \mathcal{R} \tag{9.89}$$

The same relation can be computed for the diagonal elements of the overlap matrix in the reciprocal space. Thus at real $\boldsymbol{k}$ points, where the complex exponential is real for all $\boldsymbol{g}$, the diagonal elements of the Kohn-Sham and overlap reciprocal space matrices are zero. At complex $\boldsymbol{k}$ points they are purely real, from the product of i with the imaginary part of the complex exponential.
Since the Kohn-Sham and the overlap matrices have the same properties (9.82) and (9.83) in the real space, then the previous procedure can be also applied to the overlap matrix, obtaining the same result. Therefore, it has been demonstrated that the derivatives of the reciprocal space representations of the

Kohn-Sham and the overlap matrices with respect to a reciprocal space point component are Hermitian matrices, that is

$$\begin{aligned} \partial_\mu F_{\gamma\rho}(\boldsymbol{k}) = \partial_\mu F^*_{\rho\gamma}(\boldsymbol{k}) \quad &\rightarrow \quad \partial_\mu \mathbf{F} = \partial_\mu \mathbf{F}^\dagger \\ \partial_\mu S_{\gamma\rho}(\boldsymbol{k}) = \partial_\mu S^*_{\rho\gamma}(\boldsymbol{k}) \quad &\rightarrow \quad \partial_\mu \mathbf{S} = \partial_\mu \mathbf{S}^\dagger \end{aligned} \tag{9.90}$$

Finally, from the expressions previously derived for the derivatives of the Kohn-Sham and overlap reciprocal matrices, that are reported here below

$$\partial_\mu F_{\gamma\rho}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\, g_\mu \, F_{\gamma\rho}(\boldsymbol{g}) \, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu \, F_{\rho\gamma}(\boldsymbol{g}) \, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} \,] \quad \in \quad \mathcal{C} \tag{9.91}$$

$$\partial_\mu S_{\gamma\rho}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\, g_\mu \, S_{\gamma\rho}(\boldsymbol{g}) \, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu \, S_{\rho\gamma}(\boldsymbol{g}) \, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} \,] \quad \in \quad \mathcal{C} \tag{9.92}$$

$$\partial_\mu F_{\gamma\gamma}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\, g_\mu \, F_{\gamma\gamma}(\boldsymbol{g}) \, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu \, F_{\gamma\gamma}(\boldsymbol{g}) \, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} \,] = i \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} g_\mu \, F_{\gamma\gamma}(\boldsymbol{g}) \, \Im(e^{i\boldsymbol{k}\cdot\boldsymbol{g}}) \quad \in \quad \mathcal{R} \tag{9.93}$$

$$\partial_\mu S_{\gamma\gamma}(\boldsymbol{k}) = \frac{i}{2} \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} [\, g_\mu \, S_{\gamma\gamma}(\boldsymbol{g}) \, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu \, S_{\gamma\gamma}(\boldsymbol{g}) \, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} \,] = i \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} g_\mu \, S_{\gamma\gamma}(\boldsymbol{g}) \, \Im(e^{i\boldsymbol{k}\cdot\boldsymbol{g}}) \quad \in \quad \mathcal{R} \tag{9.94}$$

it can be easily noted that real $\boldsymbol{k}$ points, which have an exponential expression that is purely real, have a non-zero contribution only in the off-diagonal elements (9.91) and (9.92). Therefore the band velocities, given by the difference of the Kohn-Sham diagonal elements (9.93) and the overlap diagonal elements (9.94) multiplied by the eigenvalues, are non-zero *only* for complex reciprocal space points $\boldsymbol{k}$, which have a non-zero imaginary part in the exponential expansion. Thus only the complex $\boldsymbol{k}$ points give a contribution in the band velocities calculation (while for real $\boldsymbol{k}$ points the diagonal elements (9.93) and (9.94), i.e. the band velocities, are zero), so that in practical implementations only the complex reciprocal space points in the Brillouin zone can be considered.

9.2.2 The reality of band velocities

Since the band velocities are defined by the equation (9.74), only the diagonal elements of the matrix products in the expression (9.81) have to be considered. From the Hermitian property of the derivatives of the reciprocal space representation of the Kohn-Sham and overlap matrices demonstrated in Section 9.2.1, it follows that

(i) the diagonal elements of the derivatives matrices $\partial_\mu \mathbf{F}(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}(\boldsymbol{k})$ are purely real numbers, so that the diagonal elements of the imaginary part of the reciprocal space Kohn-Sham and overlap derivative matrices are identically equal to zero, that is

$$[\partial_\mu \mathbf{F}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \qquad \text{and} \qquad [\partial_\mu \mathbf{S}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.95}$$

(ii) the real part of reciprocal space Kohn-Sham and overlap derivative matrices is symmetric, while the imaginary part is antisymmetric, namely,

$$\begin{aligned} &\partial_\mu \mathbf{F}_r(\boldsymbol{k}) \quad \text{and} \quad \partial_\mu \mathbf{S}_r(\boldsymbol{k}) \quad \text{are symmetric matrices} \\ &\partial_\mu \mathbf{F}_i(\boldsymbol{k}) \quad \text{and} \quad \partial_\mu \mathbf{S}_i(\boldsymbol{k}) \quad \text{are antisymmetric matrices} \end{aligned} \tag{9.96}$$

For any antisymmetric matrix $\mathbf{A}$ (i.e. for any matrix $\mathbf{A}$ whose elements are so that $A_{\gamma\rho} = -A_{\rho\gamma}$), it can be demonstrated that

$$\mathbf{A} \in \mathcal{R}^{n,n} \text{ is antisymmetric} \quad \rightarrow \quad {}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C} \text{ is antisymmetric} \qquad \forall\ \mathbf{C} \in \mathcal{R}^{n,m} \tag{9.97}$$

The proof is the following:

$$\begin{aligned}({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\beta} &= {}^t(C_{\alpha\gamma})\,A_{\gamma\rho}\,C_{\rho\beta} = C_{\gamma\alpha}\,A_{\gamma\rho}\,C_{\rho\beta} = -C_{\gamma\alpha}\,A_{\rho\gamma}\,C_{\rho\beta} = -C_{\gamma\alpha}\,A_{\rho\gamma}\,{}^t(C_{\beta\rho}) \\ &= -\,{}^t(C_{\beta\rho})\,A_{\rho\gamma}\,C_{\gamma\alpha} = -({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\beta\alpha}\end{aligned} \tag{9.98}$$

where the Einstein notation has been used for the indexes. It is obvious to see that, for the diagonal elements (with $\alpha = \beta$) of the previous antisymmetric matrix product, the following relations are true

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} = -({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \quad \rightarrow \quad ({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} = 0 \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,n} \text{ antisymmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,m}$$

as a consequence of the previous demonstration. Indeed, it is a well-known property of the antisymmetric matrices to have the elements on the diagonal equal to zero.

If $\partial_\mu \mathbf{F}(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}(\boldsymbol{k})$ are Hermitian, their imaginary parts $\partial_\mu \mathbf{F}_i(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}_i(\boldsymbol{k})$ are antisymmetric.
Therefore, as a consequence of the antisymmetry of the purely imaginary parts of the derivative of the Kohn-Sham matrix with respect to a component of a reciprocal space vector, the diagonal elements of the following matrix products, that belongs to the imaginary part in the expression (9.81) are identically equal to zero, that is

$$\partial_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.99}$$

$$\partial_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.100}$$

$$\begin{aligned}\partial_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) = 0\end{aligned} \tag{9.101}$$

$$\begin{aligned}\partial_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) = 0\end{aligned} \tag{9.102}$$

For any symmetric matrix $\mathbf{A}$ (i.e. for any matrix $\mathbf{A}$ whose elements are so that $A_{\gamma\rho} = A_{\rho\gamma}$), it can be demonstrated that

$$\mathbf{A} \in \mathcal{R}^{n,m} \text{ is symmetric} \quad \rightarrow \quad {}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B} \text{ is symmetric} \qquad \forall\ \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\ \mathbf{B} \in \mathcal{R}^{m,q} \tag{9.103}$$

The proof is the following:

$$\begin{aligned}({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B})_{\alpha\beta} &= {}^t(C_{\alpha\gamma})\,A_{\gamma\rho}\,B_{\rho\beta} = C_{\gamma\alpha}\,A_{\gamma\rho}\,B_{\rho\beta} = C_{\gamma\alpha}\,A_{\rho\gamma}\,B_{\rho\beta} = C_{\gamma\alpha}\,A_{\rho\gamma}\,{}^t(B_{\beta\rho}) \\ &= {}^t(B_{\beta\rho})\,A_{\rho\gamma}\,C_{\gamma\alpha} = ({}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C})_{\beta\alpha}\end{aligned} \tag{9.104}$$

where the Einstein notation has been used for the indexes. It is obvious to see that, for the diagonal elements (with $\alpha = \beta$) of the previous symmetric matrix product, the following relations are true

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B})_{\alpha\alpha} = ({}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,m} \text{ symmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\ \mathbf{B} \in \mathcal{R}^{m,q} \tag{9.105}$$

as a consequence of the previous demonstration.

If $\partial_\mu \mathbf{F}(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}(\boldsymbol{k})$ are Hermitian, their real part $\partial_\mu \mathbf{F}_r(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}_r(\boldsymbol{k})$ are symmetric.
Therefore, as a consequence of the symmetry of the purely real parts of the derivative of the Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, each one of the four remaining matrix products that belongs to the imaginary part in the expression (9.81) have diagonal elements identically equal to their opposite sign counterpart, that is

$$\partial_\mu \mathbf{F}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.106}$$

$$\partial_\mu \mathbf{S}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.107}$$

where the hermiticity of the Kohn-Sham and overlap reciprocal matrix derivatives has been used.

In this way, the four remaining elements in the purely imaginary part (last square parenthesis) in equation (9.81) have diagonal terms that cancel each other, namely,

$$[\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} - [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.108}$$

and

$$\begin{aligned} &- [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} + [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ &= -[\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\;\varepsilon_\alpha(\boldsymbol{k}) + [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\;\varepsilon_\alpha(\boldsymbol{k}) \\ &= \{\,-[\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\,\}\;\varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.109}$$

where the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$ is a diagonal matrix whose only non-zero elements are the eigenvalues $\varepsilon_\alpha(\boldsymbol{k})$ on the diagonal (therefore, the product of the diagonal elements of the three-matrices product in (9.109) by the diagonal eigenvalues matrix is separable, and the relation (9.2.2) can be used before multiplying by the diagonal eigenvalue matrix, as done in (9.109)).

Consider now the matrix products associated to the purely real part in (9.81).

For any symmetric matrix $\mathbf{A}$ (i.e. for any matrix $\mathbf{A}$ whose elements are so that $A_{\gamma\rho} = A_{\rho\gamma}$), it can be demonstrated that

$$\mathbf{A} \in \mathcal{R}^{n,n} \text{ is symmetric} \quad \rightarrow \quad {}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C} \text{ is symmetric} \qquad \forall\; \mathbf{C} \in \mathcal{R}^{n,m} \tag{9.110}$$

The proof is the following:

$$\begin{aligned} (\,{}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C}\,)_{\alpha\beta} &= {}^t(C_{\alpha\gamma})\,A_{\gamma\rho}\,C_{\rho\beta} = C_{\gamma\alpha}\,A_{\gamma\rho}\,C_{\rho\beta} = C_{\gamma\alpha}\,A_{\rho\gamma}\,C_{\rho\beta} = C_{\gamma\alpha}\,A_{\rho\gamma}\;{}^t(C_{\beta\rho}) \\ &= {}^t(C_{\beta\rho})\,A_{\rho\gamma}\,C_{\gamma\alpha} = (\,{}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C}\,)_{\beta\alpha} \end{aligned} \tag{9.111}$$

where the Einstein notation has been used for the indexes. It is obvious to see that, for the diagonal elements (with $\alpha = \beta$) of the previous symmetric matrix product, the following relation is true

$$\text{in general} \qquad (\,{}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C}\,)_{\alpha\alpha} \neq 0 \qquad \forall\; \mathbf{A} \in \mathcal{R}^{n,n} \text{ antisymmetric and } \forall\; \mathbf{C} \in \mathcal{R}^{n,m} \tag{9.112}$$

as a consequence of the previous demonstration. Indeed, the symmetric matrices have diagonal elements generally different to zero.

Therefore, as a consequence of the symmetry of the purely real parts of the derivative of the Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, four matrix products that belongs to the purely real part in the expression (9.81) have diagonal elements generally different from zero, that is

$$\partial_\mu\mathbf{F}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} \neq 0 \tag{9.113}$$

$$\partial_\mu\mathbf{F}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \neq 0 \tag{9.114}$$

$$\begin{aligned} \partial_\mu\mathbf{S}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad & [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\;\varepsilon_\alpha(\boldsymbol{k}) \neq 0 \end{aligned} \tag{9.115}$$

$$\begin{aligned} \partial_\mu\mathbf{S}_r(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad & [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\partial_\mu\mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\;\varepsilon_\alpha(\boldsymbol{k}) \neq 0 \end{aligned} \tag{9.116}$$

For any antisymmetric matrix $\mathbf{A}$ (i.e. for any matrix $\mathbf{A}$ whose elements are so that $A_{\gamma\rho} = -A_{\rho\gamma}$), it can be demonstrated that

$$\mathbf{A} \in \mathcal{R}^{n,m} \text{ is antisymmetric} \quad \rightarrow \quad {}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B} \text{ is antisymmetric} \qquad \forall\; \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\; \mathbf{B} \in \mathcal{R}^{m,q} \tag{9.117}$$

The proof is the following:

$$\begin{aligned} (\,{}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B}\,)_{\alpha\beta} &= {}^t(C_{\alpha\gamma})\,A_{\gamma\rho}\,B_{\rho\beta} = C_{\gamma\alpha}\,A_{\gamma\rho}\,B_{\rho\beta} = -C_{\gamma\alpha}\,A_{\rho\gamma}\,B_{\rho\beta} = -C_{\gamma\alpha}\,A_{\rho\gamma}\;{}^t(B_{\beta\rho}) \\ &= -\,{}^t(B_{\beta\rho})\,A_{\rho\gamma}\,C_{\gamma\alpha} = -(\,{}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C}\,)_{\beta\alpha} \end{aligned} \tag{9.118}$$

where the Einstein notation has been used for the indexes. It is obvious to see that, for the diagonal elements (with $\alpha = \beta$) of the previous antisymmetric matrix product, the following relations are true

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B})_{\alpha\alpha} = -({}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,m} \text{ antisymmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\ \mathbf{B} \in \mathcal{R}^{m,q}$$

as a consequence of the previous demonstration.

Therefore, as a consequence of the antisymmetry of the purely imaginary parts of the derivative of the Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, four matrix products that belongs to the purely real part in the expression (9.81) have diagonal elements identically equal to their opposite sign counterpart, that is

$$\partial_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.119}$$

$$\partial_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.120}$$

where the hermiticity of the Kohn-Sham and overlap reciprocal matrix derivatives has been used.
In this way, four elements in the purely real part (first square parenthesis) in equation (9.81) have diagonal terms that cancel each other, namely,

$$-[{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.121}$$

and

$$\begin{aligned} &[{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ &= [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) \\ &= \{\,[{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\,\}\ \varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.122}$$

where the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$ is a diagonal matrix whose only non-zero elements are the eigenvalues $\varepsilon_\alpha(\boldsymbol{k})$ on the diagonal.

Following the results obtained, the previous expression (9.81) is reduced to four purely real terms (matrix products), namely,

$$\begin{aligned} &[\mathbf{C}^\dagger(\boldsymbol{k})\,\partial_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ &= \{\,[{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \\ &\quad - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha}\,\} \end{aligned} \tag{9.123}$$

As a consequence, the band velocities (9.76), given by

$$\begin{aligned} v_{\mu,\alpha}(\boldsymbol{k}) \equiv v_\mu(\alpha,\boldsymbol{k}) = \frac{1}{\hbar}\frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial k_\mu} &= \frac{1}{\hbar}\{[\mathbf{C}^\dagger(\boldsymbol{k})\,\partial_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha}\} \\ &= \frac{1}{\hbar}\{\,[{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \\ &\quad - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_i(\boldsymbol{k})\ \partial_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha}\,\} \end{aligned} \tag{9.124}$$

are purely real numbers.
In the equation (9.124) the diagonal elements of the band velocities are computed, that is

$$v_{\mu,\alpha}(\boldsymbol{k}) \equiv v_\mu(\alpha,\boldsymbol{k}) = \langle \alpha\boldsymbol{k}|\hat{\boldsymbol{v}}_\mu|\alpha\boldsymbol{k}\rangle \tag{9.125}$$

The global character of the band velocities matrix, which includes also the off-diagonal elements,

$$v_{\mu,\alpha\beta}(\boldsymbol{k}) \equiv v_\mu(\alpha\beta,\boldsymbol{k}) = \langle \alpha\boldsymbol{k}|\hat{\boldsymbol{v}}_\mu|\beta\boldsymbol{k}\rangle \tag{9.126}$$

can be written by generalizing (9.76) obtaining

$$v_{\mu,\alpha\beta}(\boldsymbol{k}) \equiv v_\mu(\alpha\beta,\boldsymbol{k}) = \frac{1}{\hbar}\{[\mathbf{C}^\dagger(\boldsymbol{k})\,\partial_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})]_{\alpha\beta} - [\mathbf{C}^\dagger(\boldsymbol{k})\ \partial_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\beta}\} \tag{9.127}$$

recovering the whole matrix

$$\boldsymbol{v}_\mu(\boldsymbol{k}) = \frac{1}{\hbar}\left\{\mathbf{C}^\dagger(\boldsymbol{k})\,\partial_\mu\mathbf{F}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\;\partial_\mu\mathbf{S}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})\right\} \qquad \mu = x, y, z \tag{9.128}$$

Since the matrices $\partial_\mu\mathbf{F}(\boldsymbol{k})$ and $\partial_\mu\mathbf{S}(\boldsymbol{k})$ are Hermitian, as demonstrated in Section 9.2.1, and the eigenvalues $\mathbf{C}(\boldsymbol{k})$ are unitary matrices, the following theorem can be demonstrated and it can be used to study the properties of the band velocities matrix (9.128).

Theorem. A unitary similarity transformation applied to an Hermitian matrix preservers hermiticity.

Proof. If $\mathbf{A}$ is Hermitian (i.e. $\mathbf{A} = \mathbf{A}^\dagger$) and the matrix $\mathbf{C}$ is unitary (i.e. $\mathbf{C}^\dagger\mathbf{C} = \mathbf{C}\mathbf{C}^\dagger = \mathbb{1}$), then the matrix product

$$\mathbf{B} = \mathbf{C}^\dagger\mathbf{A}\mathbf{C} \tag{9.129}$$

has an Hermitian adjoint given by

$$\mathbf{B}^\dagger = (\mathbf{C}^\dagger\mathbf{A}\mathbf{C})^\dagger = \mathbf{C}^\dagger\mathbf{A}^\dagger(\mathbf{C}^\dagger)^\dagger = \mathbf{C}^\dagger\mathbf{A}^\dagger\mathbf{C} = \mathbf{C}^\dagger\mathbf{A}\mathbf{C} = \mathbf{B} \tag{9.130}$$

Therefore, the matrix product $\mathbf{B}$ in equation (9.129) is Hermitian.

As a consequence of this theorem, since the band velocities (9.128) are a sum of similarity transformations applied to the Hermitian matrices $\partial_\mu\mathbf{F}(\boldsymbol{k})$ and $\partial_\mu\mathbf{S}(\boldsymbol{k})$, the band velocities matrix is itself Hermitian. Therefore, the diagonal elements of the band velocities matrix are purely real, as demonstrated before, while the off-diagonal elements are generally complex numbers, with a symmetric purely real part and an anti-symmetric purely imaginary part.

9.2.3 Implementation in the CRYSTAL code

The elements of the Kohn-Sham and overlap reciprocal matrices derivatives with respect to a reciprocal space component, whose mathematical expressions have been computed in (9.86), are given by

$$\partial_\mu F_{\gamma\rho}(\boldsymbol{k}) = \frac{i}{2}\sum_{\{\boldsymbol{g}\}}\left[\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,\right] \tag{9.131}$$

$$\partial_\mu S_{\gamma\rho}(\boldsymbol{k}) = \frac{i}{2}\sum_{\{\boldsymbol{g}\}}\left[\, g_\mu\, S_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, S_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,\right] \tag{9.132}$$

where for simplicity the summation over all $\boldsymbol{g}$ vectors has been extended to the case $\boldsymbol{g} = \mathbf{0}$ as the zero vector components are all zeros (thus giving a null contribution to the summation). In the practical implementation, the code works with the matrix elements

$$\tilde{\partial}_\mu F_{\gamma\rho}(\boldsymbol{k}) = \frac{1}{2}\sum_{\{\boldsymbol{g}\}}\left[\, g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,\right] = -i\,\partial_\mu F_{\gamma\rho}(\boldsymbol{k}) \tag{9.133}$$

$$\tilde{\partial}_\mu S_{\gamma\rho}(\boldsymbol{k}) = \frac{1}{2}\sum_{\{\boldsymbol{g}\}}\left[\, g_\mu\, S_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, S_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}}\,\right] = -i\,\partial_\mu S_{\gamma\rho}(\boldsymbol{k}) \tag{9.134}$$

These new defined matrices will be called in the following *modified* Kohn-Sham and overlap reciprocal matrix derivatives with respect to a reciprocal space point component.
As demonstrated in (9.88), the matrix $\partial_\mu\mathbf{F}(\boldsymbol{k})$ is Hermitian. Therefore, the matrix $\tilde{\partial}_\mu\mathbf{F}(\boldsymbol{k}) = \{\tilde{\partial}_\mu F_{\gamma\rho}(\boldsymbol{k})\}$ whose elements have the expression in equation (9.133) is anti-Hermitian. This can be also demonstrated very easily by computing the expression for the conjugate transpose of the elements in equation (9.133), that is

$$\tilde{\partial}_\mu F^*_{\rho\gamma}(\boldsymbol{k}) = \frac{1}{2}\sum_{\{\boldsymbol{g}\}}\left[\, g_\mu\, F_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, F_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}}\,\right] = -\tilde{\partial}_\mu F_{\gamma\rho}(\boldsymbol{k}) \quad \rightarrow \quad \tilde{\partial}_\mu\mathbf{F}^\dagger = -\tilde{\partial}_\mu\mathbf{F} \tag{9.135}$$

The same reasoning can be applied to the derivative of the reciprocal space overlap matrix,

$$\tilde{\partial}_\mu S^*_{\rho\gamma}(\boldsymbol{k}) = \frac{1}{2}\sum_{\{\boldsymbol{g}\}}[\, g_\mu\, S_{\rho\gamma}(\boldsymbol{g})\, e^{-i\boldsymbol{k}\cdot\boldsymbol{g}} - g_\mu\, S_{\gamma\rho}(\boldsymbol{g})\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}}\,] = -\tilde{\partial}_\mu S_{\gamma\rho}(\boldsymbol{k}) \quad \rightarrow \quad \tilde{\partial}_\mu \mathbf{S}^\dagger = -\tilde{\partial}_\mu \mathbf{S} \tag{9.136}$$

By computing the diagonal elements of the matrices $\tilde{\partial}_\mu \mathbf{F}$ and $\tilde{\partial}_\mu \mathbf{S}$, it can be easily demonstrated, as in (9.89), that

$$\tilde{\partial}_\mu F_{\gamma\gamma}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} g_\mu\, F_{\gamma\gamma}(\boldsymbol{g})\, \Im(e^{i\boldsymbol{k}\cdot\boldsymbol{g}}) \qquad \text{and} \qquad \tilde{\partial}_\mu S_{\gamma\gamma}(\boldsymbol{k}) = \sum_{\{\boldsymbol{g}\neq\boldsymbol{0}\}} g_\mu\, S_{\gamma\gamma}(\boldsymbol{g})\, \Im(e^{i\boldsymbol{k}\cdot\boldsymbol{g}}) \tag{9.137}$$

Thus the diagonal elements of the Kohn-Sham and overlap matrix derivatives are zero at real $\boldsymbol{k}$ points (as already pointed out at the end of Section 9.2.1) and *in the code* these diagonal elements (9.137) are purely imaginary at complex $\boldsymbol{k}$ points.

Furthermore, there is a relation between the real and the imaginary parts of the matrices $\partial_\mu \mathbf{F}$ and $\tilde{\partial}_\mu \mathbf{F}$, as well as between the real and the imaginary parts of the matrices $\partial_\mu \mathbf{S}$ and $\tilde{\partial}_\mu \mathbf{S}$, namely,

$$\begin{aligned} &\tilde{\partial}_\mu \mathbf{F} = \tilde{\partial}_\mu \mathbf{F}_r + i\,\tilde{\partial}_\mu \mathbf{F}_i = -i\,\partial_\mu \mathbf{F} = -i\,[\partial_\mu \mathbf{F}_r + i\,\partial_\mu \mathbf{F}_i] = -i\,\partial_\mu \mathbf{F}_r + \partial_\mu \mathbf{F}_i \\ &\rightarrow \quad \tilde{\partial}_\mu \mathbf{F}_r = \partial_\mu \mathbf{F}_i \qquad \text{and} \qquad \tilde{\partial}_\mu \mathbf{F}_i = -\partial_\mu \mathbf{F}_r \end{aligned} \tag{9.138}$$

$$\begin{aligned} &\tilde{\partial}_\mu \mathbf{S} = \tilde{\partial}_\mu \mathbf{S}_r + i\,\tilde{\partial}_\mu \mathbf{S}_i = -i\,\partial_\mu \mathbf{S} = -i\,[\partial_\mu \mathbf{S}_r + i\,\partial_\mu \mathbf{S}_i] = -i\,\partial_\mu \mathbf{S}_r + \partial_\mu \mathbf{S}_i \\ &\rightarrow \quad \tilde{\partial}_\mu \mathbf{S}_r = \partial_\mu \mathbf{S}_i \qquad \text{and} \qquad \tilde{\partial}_\mu \mathbf{S}_i = -\partial_\mu \mathbf{S}_r \end{aligned} \tag{9.139}$$

By inserting these equivalences into the equation (9.124), the following expression is obtained for the band velocities as functions of the new defined anti-Hermitian matrices (9.133) and (9.134),

$$\begin{aligned} v_{\mu,\alpha}(\boldsymbol{k}) \equiv v_\mu(\alpha,\boldsymbol{k}) = -\frac{1}{\hbar}\{\,&[\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \\ &- [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [\,{}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})]_{\alpha\alpha}\,\} \end{aligned} \tag{9.140}$$

The previous expression for the band velocities can be demonstrated starting from the equation

$$\begin{aligned} &\mathbf{C}^\dagger(\boldsymbol{k})\,\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})\;\mathbf{C}(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\ &= [\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) + {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) - {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) + {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) \\ &\quad - {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) - {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) + {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\ &\quad - {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})\,] \\ &+ i\,[\,{}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) - {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k}) + {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) + {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k}) \\ &\quad - {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) + {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_r(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) - {}^t\mathbf{C}_r(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k}) \\ &\quad - {}^t\mathbf{C}_i(\boldsymbol{k})\;\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\;\mathbf{C}_i(\boldsymbol{k})\;\mathbf{E}(\boldsymbol{k})\,] \end{aligned} \tag{9.141}$$

which is a modification of equation (9.81), where instead of using the Hermitian matrices $\partial_\mu \mathbf{F}(\boldsymbol{k})$ and $\partial_\mu \mathbf{S}(\boldsymbol{k})$ as in equation (9.81), their anti-Hermitian counterparts $\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})$ have been used. Equation (9.141) is the expression practically adopted in the CRYSTAL code to compute the band velocities.

From the anti-Hermitian property of the modified Kohn-Sham and overlap reciprocal space matrices derivatives previously defined, it follows that

(i) the diagonal elements of the derivatives matrices $\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})$ are purely imaginary numbers, so that the diagonal elements of the real part of the reciprocal space Kohn-Sham and overlap derivative matrices are identically equal to zero, that is

$$[\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})]_{\alpha\alpha} = 0 \qquad \text{and} \qquad [\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.142}$$

(ii) the imaginary part of the modified Kohn-Sham and overlap reciprocal space derivative matrices is symmetric, while the real part is antisymmetric, namely,

$$\begin{array}{llll} \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k}) & \text{and} & \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k}) & \text{are symmetric matrices} \\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k}) & \text{and} & \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k}) & \text{are antisymmetric matrices} \end{array} \tag{9.143}$$

Using the reality and the symmetry properties of the matrices, it can be demonstrated (see equations (9.98) for the proof) that

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} = -({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \quad \rightarrow \quad ({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} = 0 \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,n} \text{ antisymmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,m}$$

If $\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})$ are anti-Hermitian, their real parts $\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})$ are antisymmetric. Therefore, as a consequence of the antisymmetry of the purely real parts of the modified derivative of the Kohn-Sham matrix with respect to a component of a reciprocal space vector, the diagonal elements of the following matrix products, that belongs to the real part in the expression (9.141) are identically equal to zero, that is

$$\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.144}$$

$$\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.145}$$

$$\begin{aligned} \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.146}$$

$$\begin{aligned} \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.147}$$

Furthermore, as demonstrated in equations (9.104),

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B})_{\alpha\alpha} = ({}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,m} \text{ symmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\ \mathbf{B} \in \mathcal{R}^{m,q} \tag{9.148}$$

If $\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})$ are anti-Hermitian, their imaginary parts $\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})$ and $\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})$ are symmetric. Therefore, as a consequence of the symmetry of the purely imaginary parts of the modified derivative of the Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, each one of the four remaining matrix products that belongs to the real part in the expression (9.141) have diagonal elements identically equal to their opposite sign counterpart, that is

$$\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.149}$$

$$\tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.150}$$

where the anti-hermiticity of the modified Kohn-Sham and overlap reciprocal matrix derivatives has been used. In this way, the four remaining elements in the purely real part (first square parenthesis) in equation (9.141) have diagonal terms that cancel each other, namely,

$$[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.151}$$

and

$$\begin{aligned} & -[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = -[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) + [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) \\ & = \{ -[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \}\ \varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.152}$$

where the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$ is a diagonal matrix whose only non-zero elements are the eigenvalues $\varepsilon_\alpha(\boldsymbol{k})$ on the diagonal (therefore, the product of the diagonal elements of the three-matrices product in (9.109) by the diagonal eigenvalues matrix is separable, and the relation (9.2.2) can be used before multiplying by the diagonal eigenvalue matrix, as done in (9.152)).

Consider now the matrix products associated to the purely imaginary part in equation (9.141).

As demonstrated in equations (9.111),

$$\text{in general} \qquad ({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \neq 0 \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,n} \text{ antisymmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,m} \tag{9.153}$$

Therefore, as a consequence of the symmetry of the purely imaginary parts of the derivative of the modified Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, four matrix products that belongs to the purely imaginary part in the expression (9.141) have diagonal elements generally different from zero, that is

$$\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} \neq 0 \tag{9.154}$$

$$\tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \neq 0 \tag{9.155}$$

$$\begin{aligned} \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) \neq 0 \end{aligned} \tag{9.156}$$

$$\begin{aligned} \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k}) \text{ symmetric} \quad \rightarrow \quad & [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) \neq 0 \end{aligned} \tag{9.157}$$

Finally, as demonstrated in equations (9.111),

$$({}^t\mathbf{C}\,\mathbf{A}\,\mathbf{B})_{\alpha\alpha} = -({}^t\mathbf{B}\,\mathbf{A}\,\mathbf{C})_{\alpha\alpha} \qquad \forall\ \mathbf{A} \in \mathcal{R}^{n,m} \text{ antisymmetric and } \forall\ \mathbf{C} \in \mathcal{R}^{n,p} \text{ and } \forall\ \mathbf{B} \in \mathcal{R}^{m,q}$$

Therefore, as a consequence of the antisymmetry of the purely real parts of the modified derivative of the Kohn-Sham reciprocal matrix with respect to a component of a reciprocal space vector, four matrix products that belongs to the purely imaginary part in the expression (9.141) have diagonal elements identically equal to their opposite sign counterpart, that is

$$\tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.158}$$

$$\tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k}) \text{ antisymmetric} \quad \rightarrow \quad [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} = [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \tag{9.159}$$

where the anti-hermiticity of the modified Kohn-Sham and overlap reciprocal matrix derivatives has been used.
In this way, four elements in the purely imaginary part (last square parenthesis) in equation (9.141) have diagonal terms that cancel each other, namely,

$$-[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} = 0 \tag{9.160}$$

and

$$\begin{aligned} & [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\ \varepsilon_\alpha(\boldsymbol{k}) \\ & = \{\,[{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_r(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha}\,\}\ \varepsilon_\alpha(\boldsymbol{k}) = 0 \end{aligned} \tag{9.161}$$

where the eigenvalues matrix $\mathbf{E}(\boldsymbol{k})$ is a diagonal matrix whose only non-zero elements are the eigenvalues $\varepsilon_\alpha(\boldsymbol{k})$ on the diagonal.

Following the results obtained, the previous expression (9.81) is reduced to four purely imaginary terms (matrix products), namely,

$$\begin{aligned} & [\mathbf{C}^\dagger(\boldsymbol{k})\,\tilde{\partial}_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} \\ & = i\,\{\,[{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})]_{\alpha\alpha} + [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{F}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})]_{\alpha\alpha} \\ & \qquad - [{}^t\mathbf{C}_r(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_r(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha} - [{}^t\mathbf{C}_i(\boldsymbol{k})\ \tilde{\partial}_\mu \mathbf{S}_i(\boldsymbol{k})\ \mathbf{C}_i(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k})]_{\alpha\alpha}\,\} \end{aligned} \tag{9.162}$$

Once the previous expression has been computed, it has to be multiplied by $i/\hbar$ in order to recover the correct band velocities expression (9.140).

9.2.4 Orbital rotations and transport properties

The normalized eigenvectors $\mathbf{C}(\boldsymbol{k})$ that results from the diagonalization problem are not unique. Indeed,

(i) for each and every eigenvector there is a random phase with magnitude unity $e^{i\phi}$

(ii) within a degenerate set of eigenvectors any post-applied unitary transformation (that corresponds to a rotation) to the set is also a set of orthonormalized eigenvectors with the same eigenvalues

An element of the matrix $\boldsymbol{v}_\mu$ which contains the band velocities can be computed starting from equation (9.76), that is reported below

$$v_{\mu,\alpha}(\boldsymbol{k}) = \frac{1}{\hbar}\frac{\partial \varepsilon_\alpha(\boldsymbol{k})}{\partial k_\mu} = \frac{1}{\hbar}\left\{ \left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k}) \right]_{\alpha\alpha} - \left[\mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \right]_{\alpha\alpha} \right\} \tag{9.163}$$

where $\mu = x, y, z$ is the direction of the derivative in the reciprocal space. Taking into account also the off-diagonal elements, the previous expression becomes

$$\boldsymbol{v}_\mu(\boldsymbol{k}) = \frac{1}{\hbar}\left\{ \mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\ \frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu}\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \right\} \tag{9.164}$$

Using the following notation

$$\eth_\mu \mathbf{F}(\boldsymbol{k}) \equiv \frac{1}{\hbar}\frac{\partial \mathbf{F}(\boldsymbol{k})}{\partial k_\mu} \qquad \text{and} \qquad \eth_\mu \mathbf{S}(\boldsymbol{k}) \equiv \frac{1}{\hbar}\frac{\partial \mathbf{S}(\boldsymbol{k})}{\partial k_\mu} \tag{9.165}$$

equation (9.164) can be rewritten as

$$\boldsymbol{v}_\mu(\boldsymbol{k}) = \mathbf{C}^\dagger(\boldsymbol{k})\ \eth_\mu \mathbf{F}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k}) - \mathbf{C}^\dagger(\boldsymbol{k})\ \eth_\mu \mathbf{S}(\boldsymbol{k})\ \mathbf{C}(\boldsymbol{k})\ \mathbf{E}(\boldsymbol{k}) \tag{9.166}$$

Therefore, an element $v_{\mu,\alpha\beta}(\boldsymbol{k})$ of the matrix $\boldsymbol{v}_\mu(\boldsymbol{k})$ defined in (9.166) is given by

$$\begin{aligned} v_{\mu,\alpha\beta}(\boldsymbol{k}) &= \sum_{\gamma=1}^{m}\sum_{\rho=1}^{m} C^\dagger_{\alpha\gamma}(\boldsymbol{k})\ \eth_\mu F_{\gamma\rho}(\boldsymbol{k})\ C_{\rho\beta}(\boldsymbol{k}) - \sum_{\gamma=1}^{m}\sum_{\rho=1}^{m} C^\dagger_{\alpha\gamma}(\boldsymbol{k})\ \eth_\mu S_{\gamma\rho}(\boldsymbol{k})\ C_{\rho\beta}(\boldsymbol{k})\ \varepsilon_\beta(\boldsymbol{k}) \\ &= \sum_{\gamma=1}^{m}\sum_{\rho=1}^{m} C^\dagger_{\alpha\gamma}(\boldsymbol{k})\left[\ \eth_\mu F_{\gamma\rho}(\boldsymbol{k}) - \eth_\mu S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon_\beta(\boldsymbol{k})\ \right] C_{\rho\beta}(\boldsymbol{k}) \\ &= \sum_{\gamma=1}^{m} C^\dagger_{\alpha\gamma}(\boldsymbol{k})\ \sum_{\rho=1}^{m}\left[\ \eth_\mu F_{\gamma\rho}(\boldsymbol{k}) - \eth_\mu S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon_\beta(\boldsymbol{k})\ \right] C_{\rho\beta}(\boldsymbol{k}) \end{aligned} \tag{9.167}$$

where m are the number of the basis functions (atomic orbitals) and $\varepsilon_\beta(\boldsymbol{k})$ are the diagonal elements of the diagonal eigenvalue matrix $\mathbf{E}(\boldsymbol{k}) = \{E_{\alpha\beta}\,\delta_{\alpha\beta} \equiv \varepsilon_\beta\}$.

9.2.4.1 Off-diagonal elements of band velocities and random phases of the eigenvectors

First consider the effect of the random phase on the band velocity matrix $\boldsymbol{v}_\mu$. The expression for each element of $\boldsymbol{v}_\mu$ has been computed in equation (9.167) and it is given by

$$v_{\mu,\alpha\beta}(\boldsymbol{k}) = \sum_{\gamma=1}^{m} C^\dagger_{\alpha\gamma}(\boldsymbol{k})\ \sum_{\rho=1}^{m}\left[\ \eth_\mu F_{\gamma\rho}(\boldsymbol{k}) - \eth_\mu S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon_\beta(\boldsymbol{k})\ \right] C_{\rho\beta}(\boldsymbol{k}) \tag{9.168}$$

where m are the number of atomic orbitals. Consider a related matrix $\boldsymbol{u}_\mu$ which differs solely by each eigenvectors having a different random phase, that is

$$u_{\mu,\alpha\beta}(\boldsymbol{k}) = \sum_{\gamma=1}^{m} e^{-i\phi_\alpha}\, C^\dagger_{\alpha\gamma}(\boldsymbol{k})\ \sum_{\rho=1}^{m}\left[\ \eth_\mu F_{\gamma\rho}(\boldsymbol{k}) - \eth_\mu S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon_\beta(\boldsymbol{k})\ \right] e^{i\phi_\beta}\, C_{\rho\beta}(\boldsymbol{k}) = e^{i(\phi_\beta - \phi_\alpha)}\, v_{\mu,\alpha\beta}(\boldsymbol{k}) \tag{9.169}$$

Thus it is clear that, due to the random phase, the off-diagonal elements of $\boldsymbol{v}_\mu(\boldsymbol{k})$ may vary when calculated using eigenvectors from different diagonalizers, even for pairs of non-degenerate eigenvalues. However, the random phase does not affect the diagonal terms. This is not a big worry as only the diagonal terms have a physical meaning in the band velocities calculation using the Boltzmann transport equations theory.

9.2.4.2 Diagonal elements of band velocities and rotations within degenerate sets of orbitals

The i-th diagonal element of the band velocity is given by

$$v_{\mu,\alpha\alpha}(\boldsymbol{k}) = \sum_{\gamma=1}^{m} C^{\dagger}_{\alpha\gamma}(\boldsymbol{k}) \sum_{\rho=1}^{m} [\ \eth_{\mu} F_{\gamma\rho}(\boldsymbol{k}) - \eth_{\mu} S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon_{\alpha}(\boldsymbol{k})\ ]\, C_{\rho\alpha}(\boldsymbol{k}) \tag{9.170}$$

where m are the number of atomic orbitals. For non-degenerate eigenvalues $\varepsilon_{\alpha}(\boldsymbol{k})$ each diagonal element is unique. However, if $\varepsilon_{\alpha}(\boldsymbol{k})$ is part of a degenerate set, the correspondent eigenvectors are no longer unique, and the effect of taking different linear combinations of the eigenvectors within the degenerate states has to be considered. As such consider a d_g degenerate state with energy $\varepsilon(\boldsymbol{k})$ in a system with M basis functions. The eigenvectors within the degenerate set can be expressed as

$$\mathbf{D}(\boldsymbol{k}) = \mathbf{C}(\boldsymbol{k})\, \mathbf{Q}(\boldsymbol{k}) \tag{9.171}$$

where $\mathbf{C}$ is the $M \times d_g$ matrix containing the original eigenvectors, $\mathbf{D}$ is a $M \times d_g$ matrix containing the rotated eigenvectors, and $\mathbf{Q}$ is a $d_g \times d_g$ unitary but otherwise arbitrary matrix.
Thus a new band velocities matrix $\boldsymbol{u}_{\mu}(\boldsymbol{k})$ for the set of degenerate states with the rotates eigenvectors can be defined using (9.171) as

$$\begin{aligned}
u_{\mu,\alpha\alpha}(\boldsymbol{k}) &= \sum_{\gamma=1}^{m} D^{\dagger}_{\alpha\gamma}(\boldsymbol{k}) \sum_{\rho=1}^{m} [\ \eth_{\mu} F_{\gamma\rho}(\boldsymbol{k}) - \eth_{\mu} S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon(\boldsymbol{k})\ ]\, D_{\rho\alpha}(\boldsymbol{k}) \\
&= \sum_{\gamma=1}^{m} \sum_{\lambda=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\, C^{\dagger}_{\lambda\gamma}(\boldsymbol{k}) \sum_{\rho=1}^{m} [\ \eth_{\mu} F_{\gamma\rho}(\boldsymbol{k}) - \eth_{\mu} S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon(\boldsymbol{k})\ ] \sum_{\eta=1}^{d_g} C_{\rho\eta}(\boldsymbol{k})\, Q_{\eta\alpha}(\boldsymbol{k}) \\
&= \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k}) \underbrace{\sum_{\rho=1}^{m} C^{\dagger}_{\lambda\rho}(\boldsymbol{k}) \sum_{n} [\ \eth_{\mu} F_{\gamma\rho}(\boldsymbol{k}) - \eth_{\mu} S_{\gamma\rho}(\boldsymbol{k})\ \varepsilon(\boldsymbol{k})\ ]\, C_{\rho\eta}(\boldsymbol{k})}_{= v_{\mu,\lambda\eta}(\boldsymbol{k}) \ \text{[see eq. (9.168)]}}\ Q_{\eta\alpha}(\boldsymbol{k}) \\
&= \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\ v_{\mu,\lambda\eta}(\boldsymbol{k})\ Q_{\eta\alpha}(\boldsymbol{k})
\end{aligned} \tag{9.172}$$

Therefore, the result is that the rotated velocities (9.172) are clearly different from the original ones given by equation (9.170). However, since $\mathbf{Q}(\boldsymbol{k})$ is unitary, it satisfies the relation

$$\sum_{\alpha=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\, Q_{\eta\alpha}(\boldsymbol{k}) = \sum_{\alpha=1}^{d_g} Q_{\eta\alpha}(\boldsymbol{k})\, Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k}) = \delta_{\lambda\eta} \tag{9.173}$$

It follows that

$$\begin{aligned}
\mathrm{Tr}[\boldsymbol{u}_{\mu}(\boldsymbol{k})] &= \sum_{\alpha=1}^{d_g} u_{\mu,\alpha\alpha}(\boldsymbol{k}) = \sum_{\alpha=1}^{d_g} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\ v_{\mu,\lambda\eta}(\boldsymbol{k})\ Q_{\eta\alpha}(\boldsymbol{k}) \\
&= \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} v_{\mu,\lambda\eta}(\boldsymbol{k}) \sum_{\alpha=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\ Q_{\eta\alpha}(\boldsymbol{k}) = \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} v_{\mu,\lambda\eta}(\boldsymbol{k})\, \delta_{\lambda\beta'} \\
&= \sum_{\lambda=1}^{d_g} v_{\mu,\lambda\lambda}(\boldsymbol{k}) = \sum_{\alpha=1}^{d_g} v_{\mu,\alpha\alpha}(\boldsymbol{k}) = \mathrm{Tr}[\boldsymbol{v}_{\mu}(\boldsymbol{k})]
\end{aligned} \tag{9.174}$$

where in the last identity the dummy index r has been changed to i so that the comparison between the first and the last members of the previous equation (9.174) can be seen more explicitly.
Labeling the diagonal elements of the matrices $\boldsymbol{u}_{\mu}$ and $\boldsymbol{v}_{\mu}$ with only one index, as will be done also in the following treatment, the previous findings can be written as

$$\mathrm{Tr}[\boldsymbol{u}_{\mu}(\boldsymbol{k})] = \sum_{\alpha=1}^{d_g} u_{\mu,\alpha}(\boldsymbol{k}) \equiv \sum_{\alpha=1}^{d_g} u_{\mu,\alpha\alpha}(\boldsymbol{k}) = \sum_{\alpha=1}^{d_g} v_{\mu,\alpha\alpha}(\boldsymbol{k}) \equiv \sum_{\alpha=1}^{d_g} v_{\mu,\alpha}(\boldsymbol{k}) = \mathrm{Tr}[\boldsymbol{v}_{\mu}(\boldsymbol{k})] \tag{9.175}$$

where $\{u_{\mu,\alpha\alpha} \equiv u_{\mu,\alpha}\}$ and $\{v_{\mu,\alpha\alpha} \equiv v_{\mu,\alpha}\}$ are the diagonal elements of the band velocities matrix. Therefore, while within a degenerate set of states the *individual band velocities are not invariant* with respect to the eigenvector rotations within the degenerate set, *the sum (i.e. the trace) of the velocities is invariant*. This is not entirely surprising, but it does have consequences for the next stage of the calculation.

Consider equation (7) from G. Sansone *et al.*[22], reported here below

$$\Xi_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, v_{\lambda,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \tag{9.176}$$

where the delta function means that the sum is over the degenerate set of states at energy $\varepsilon = \varepsilon_{\alpha}(\boldsymbol{k})$. With no degeneracy there is no problem. However, when there is a degeneracy it seems from the above discussion that the transport distribution function Ξ is not invariant with respect to rotations within degenerate sets of eigenvectors. Indeed, considering the band velocities matrix $\boldsymbol{u}_{\mu}(\boldsymbol{k})$ for a set of d_g degenerate states, whose elements are given by (9.172), the transport distribution function computed for those velocities is given by

$$\begin{aligned}
\tilde{\Xi}_{\mu\lambda}(\varepsilon) &= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} u_{\mu,\alpha}(\boldsymbol{k})\, u_{\lambda,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \qquad (9.177)\\
&= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\; v_{\mu,\lambda\eta}(\boldsymbol{k})\; Q_{\eta\alpha}(\boldsymbol{k}) \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} Q^{\dagger}_{\alpha\kappa}(\boldsymbol{k})\; v_{\lambda,\kappa\zeta}(\boldsymbol{k})\; Q_{\zeta\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k}))\\
&= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\, Q^{\dagger}_{\alpha\kappa}(\boldsymbol{k})\, v_{\mu,\lambda\eta}(\boldsymbol{k})\, v_{\lambda,\kappa\zeta}(\boldsymbol{k})\, Q_{\eta\alpha}(\boldsymbol{k})\; Q_{\zeta\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k}))\\
&\neq \Xi_{\mu\lambda}(\varepsilon)
\end{aligned}$$

that is not invariant with respect to rotations within degenerate sets of eigenvectors. This could lead to an incorrect result for the calculation of the transport distribution function in the case of energy degeneracy. Looking at the previous equation (9.177) and comparing the final expression with the transport distribution formula (9.176), it can be noted that the first step towards an orbital rotation invariant formula for the transport distribution function is to disentangle the indexes α associated to the bands quantum number in the velocities $u_{\mu,\alpha}(\boldsymbol{k})$ and $u_{\lambda,\alpha}(\boldsymbol{k})$, so that also the indexes of the related unitary matrices $\mathbf{Q}(\boldsymbol{k})$ can be separated and the property (9.173) can be used, recovering the same expression for the transport distribution function but with the velocities $v_{\mu,\alpha}(\boldsymbol{k})$ and $v_{\lambda,\alpha}(\boldsymbol{k})$ instead of those associated to a degenerate set. Indeed, if the calculation is performed following the equation

$$\Xi_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\beta=1}^{m} v_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k})) \tag{9.178}$$

then the form of the transport distribution function results to be invariant with respect to rotations within a degenerate set. This can be demonstrated considering the band velocities matrix $\boldsymbol{u}_{\mu}(\boldsymbol{k})$ for a set of d_g degenerate states, whose elements are given by (9.172), and computing the previous expression (9.178) for this degenerate set,

$$\begin{aligned}
\tilde{\Xi}_{\mu\lambda}(\varepsilon) &= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} u_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\beta=1}^{m} u_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k})) \qquad (9.179)\\
&= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} Q^{\dagger}_{\alpha r}(\boldsymbol{k})\, v_{\mu,\lambda\eta}(\boldsymbol{k})\, Q_{\eta\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\beta=1}^{m} \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} Q^{\dagger}_{\beta\kappa}(\boldsymbol{k})\, v_{\lambda,\kappa\zeta}(\boldsymbol{k})\, Q_{\zeta\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k}))\\
&= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} v_{\mu,\lambda\eta}(\boldsymbol{k}) \sum_{\alpha=1}^{m} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\, Q_{\eta\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} v_{\lambda,\kappa\zeta}(\boldsymbol{k}) \sum_{\beta=1}^{m} Q^{\dagger}_{\beta\kappa}(\boldsymbol{k})\, Q_{\zeta\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k}))\\
&= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} v_{\mu,\lambda\eta}(\boldsymbol{k}) \sum_{\alpha=1}^{d_g} Q^{\dagger}_{\alpha\lambda}(\boldsymbol{k})\, Q_{s\alpha}(\boldsymbol{k}) \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} v_{\lambda,\kappa\zeta}(\boldsymbol{k}) \sum_{\beta=1}^{d_g} Q^{\dagger}_{\beta\kappa}(\boldsymbol{k})\, Q_{\zeta\beta}(\boldsymbol{k})
\end{aligned}$$

$$= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\lambda=1}^{d_g} \sum_{\eta=1}^{d_g} v_{\mu,\lambda\eta}(\boldsymbol{k})\, \delta_{\lambda\eta} \sum_{\kappa=1}^{d_g} \sum_{\zeta=1}^{d_g} v_{\lambda,\kappa\zeta}(\boldsymbol{k})\, \delta_{\kappa\zeta}$$

$$= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\lambda=1}^{d_g} v_{\mu,\lambda\lambda}(\boldsymbol{k}) \sum_{\kappa=1}^{d_g} v_{\lambda,\kappa\kappa}(\boldsymbol{k}) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{d_g} v_{\mu,\alpha}(\boldsymbol{k}) \sum_{\beta=1}^{d_g} v_{\lambda,\beta}(\boldsymbol{k})$$

$$= \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\beta=1}^{m} v_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k})) = \Xi_{\mu\lambda}(\varepsilon)$$

so that equation (9.178) is recovered also in the case of a degenerate set. Finally, it is worth noting that in the non-degenerate case, equation (9.178) reduces to the form (9.176).

9.3 Massively Parallel Processing implementation

The main step in the calculation of the electronic transport properties is the definition of the transport distribution function

$$\tilde{\Xi}_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} u_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\beta=1}^{m} u_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k})) \tag{9.180}$$

In the CRYSTAL code, the Massively Parallel Processing is exploited in the calculation of the band velocities $u_{\mu,\alpha}(\boldsymbol{k})$, $\mu = x, y, z$. Indeed, the band velocity defined in a reciprocal space point $\boldsymbol{k}$ is independent from all the other band velocities defined in any one of the other reciprocal space points. Therefore, the calculation of the band velocities can be easily divided among processors, so that each process computes the band velocity contribution for a given reciprocal space point $\boldsymbol{k}$ and a given spin $\sigma = \uparrow, \downarrow$. In the following, the reciprocal space point for a given spin will be labeled by the couple $(\boldsymbol{k}, \sigma)$.

Two different cases can be considered:

(i) the number of reciprocal space points $(\boldsymbol{k}, \sigma)$ is equal or greater than the number of processors used for the calculation. In this case, the entire information of the band velocities for one or more $(\boldsymbol{k}, \sigma)$ points is stored on one single processor, i.e. each processor computes and own the memory of one or more band velocities $u_{\mu,\alpha}(\boldsymbol{k})$ ($\mu = x, y, z$ and $\alpha = 1, ..., m$) associated to one or more couples $(\boldsymbol{k}, \sigma)$. The reciprocal space points $(\boldsymbol{k}, \sigma)$ are distributed as uniformly as possible among processors. The order of distribution is the following: for a given spin σ, the reciprocal space points are assigned alternatively one after another to one processor in an ordered manner until the list of processors terminates, then this list is started again and the assignation continues in the same ordered manner, and so on.

(ii) the number of reciprocal space points $(\boldsymbol{k}, \sigma)$ is less than the number of processors used for the calculation. In this case, the computation and the memory of a single band velocity $u_{\mu,\alpha}(\boldsymbol{k})$ ($\mu = x, y, z$ and $\alpha = 1, ..., m$) associated to a couple $(\boldsymbol{k}, \sigma)$ is divided among a group of processes, i.e. the entire information of the band velocity for a given $(\boldsymbol{k}, \sigma)$ point is stored on multiple processors belonging to a given group. Each reciprocal space point $(\boldsymbol{k}, \sigma)$ is assigned to a group of processors using the hypothesis that the memory and timings needed for the computation of the band velocity for a complex $\boldsymbol{k}$ point is twice the amount needed for a real $\boldsymbol{k}$ point. With this splitting method some processors can remain idle during the calculation.

The example of the $KMnF_3$ crystalline open shell system reported in Table 9.1 is explicative of both the splitting cases (i) and (ii).
After the parallel calculation of the band velocities for each $(\boldsymbol{k}, \sigma)$ point, performed using the massive parallelization method described above, the calculation of the transport distribution function (9.180) is implemented through the following procedure:

1. Global sum of the generated band velocities by the current group of processors, using the communicator for that group.

2. Parallelization over the processors in that group the adding in of the contributions from the band velocities into the transport distribution function, so that each n-th processor Pn contributes in the transport distribution function by the following quantity:

$$\tilde{\Xi}^{\mathrm{P}(n)}_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\substack{\alpha = c_r(n)+1 \\ c_s(n)}}^{m} u_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\alpha}(\boldsymbol{k})) \sum_{\substack{\beta = c_r(n)+1 \\ c_s(n)}}^{m} u_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_{\beta}(\boldsymbol{k})) \tag{9.181}$$

where $c_r(n)$ is the rank of the processor Pn $\equiv$ P(n) in its own communicator, $c_s(n)$ is the size of the communicator to which the processor Pn belongs and the second line in the summation is the step assigned to the integer i, so that the sum is over $\alpha = c_r(n) + 1, c_r(n) + 1 + c_s(n), c_r(n) + 1 + 2c_s(n), ..., M$ and the calculation of the transport distribution function is distributed among

System : $KMnF_3$ - SHRINK 4 4 $\rightarrow$ 10 $\boldsymbol{k}$ points (20 $(\boldsymbol{k},\sigma)$ couples) in the IBZ				
Spin σ	$\boldsymbol{k}$ point		Ranks case(i) : 3 processors	Ranks case (ii) : 66 processors
1	(0, 0, 0)	$\mathcal{R}$	0	0 - 1
1	(1, 0, 0)	$\mathcal{C}$	1	2 - 3 - 4 - 5
1	(2, 0, 0)	$\mathcal{R}$	2	6 - 7
1	(1, 1, 0)	$\mathcal{C}$	0	8 - 9 - 10 - 11
1	(2, 1, 0)	$\mathcal{C}$	1	12 - 13 - 14 - 15
1	(2, 2, 0)	$\mathcal{R}$	2	16 - 17
1	(1, 1, 1)	$\mathcal{C}$	0	18 - 19 - 20 - 21
1	(2, 1, 1)	$\mathcal{C}$	1	22 - 23 - 24 - 25
1	(2, 2, 1)	$\mathcal{C}$	2	26 - 27 - 28 - 29
1	(2, 2, 2)	$\mathcal{R}$	0	30 - 31
2	(0, 0, 0)	$\mathcal{R}$	1	32 - 33
2	(1, 0, 0)	$\mathcal{C}$	2	34 - 35 - 36 - 37
2	(2, 0, 0)	$\mathcal{R}$	0	38 - 39
2	(1, 1, 0)	$\mathcal{C}$	1	40 - 41 - 42 - 43
2	(2, 1, 0)	$\mathcal{C}$	2	44 - 45 - 46 - 47
2	(2, 2, 0)	$\mathcal{R}$	0	48 - 49
2	(1, 1, 1)	$\mathcal{C}$	1	50 - 51 - 52 - 53
2	(2, 1, 1)	$\mathcal{C}$	2	54 - 55 - 56 - 57
2	(2, 2, 1)	$\mathcal{C}$	0	58 - 59 - 60 - 61
2	(2, 2, 2)	$\mathcal{R}$	1	62 - 63
Idle processors :			0	2

Table 9.1: Distribution of the reciprocal space points $(\boldsymbol{k},\sigma)$ among processors within Massive Parallel Processing method for the case of crystalline $KMnF_3$ (open shell) system. The ranks within the parent communicator map onto the $(\boldsymbol{k},\sigma)$ points as described by the last two columns of the table, for the two cases with the number of processors less (i) and greater (ii) than the number of $(\boldsymbol{k},\sigma)$ couples used to perform the electronic transport properties calculation.

processors according to the rank of the processor in the communicator (each processor add a contribution in the transport distribution function from a given set of bands ruled by the index α, that is, ruled by the rank of the processor in its communicator).

3. Summation of all the values from all processors and distribution of the result back to all processors through a Gsum (Mpi_Allreduce) at the end of the calculation for all the reciprocal space points, as there are no replicated contributions to the transport distribution function.

The parallelization strategy is illustrated in Figure 9.1 for the more general case (ii), with a number of processors greater than the number of irreducible $\boldsymbol{k}$ points.

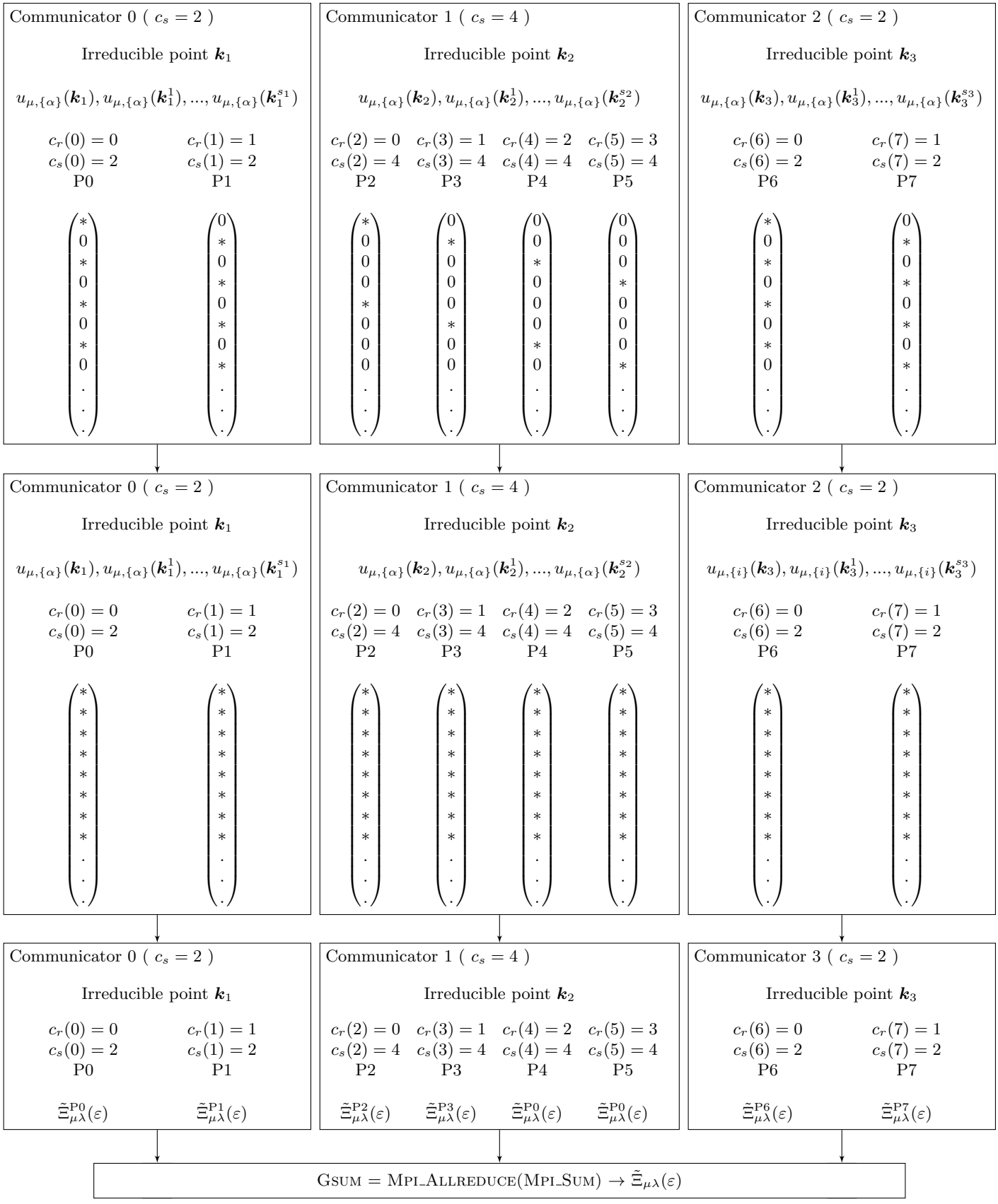

Figure 9.1: Massive parallelization strategy for the calculation of the transport distribution function in the Crystal code, for the more general case with a number of processors $n_p = 8$ greater than the number of irreducible $\boldsymbol{k}$ points $n_{\boldsymbol{k}} = 3$. The distribution among processors in the band velocities and transport distribution function calculations has been performed on the base of the band index, namely, each processor owns the values for a certain set of bands $\{\alpha\}$ (* indicates double precision real values). For this subset of bands $\{\alpha\}$, each processor owns the values of $u_{\mu,\{\alpha\}}(\boldsymbol{k}_\beta), u_{\mu,\{\alpha\}}(\boldsymbol{k}_\beta^1), ..., u_{\mu,\{\alpha\}}(\boldsymbol{k}_\beta^{s_\beta})$, where $\mu = x, y, z$ is the reciprocal space direction of the band velocities and s_j $(j = 1, ..., n_{\boldsymbol{k}})$ are the number of reducible $\boldsymbol{k}$ points generated from the j-th irreducible one. The transport distribution function is finally distributed among processors as in equation (9.180), then each contribution from each processor is summed up and the result is redistributed back to all processes.

9.4 Results and Discussion: the famous case of silicon

9.4.1 Band structure

Figure 9.2 reports the band structure of crystalline silicon computed with DFT (PBE functional), a basis set of 26 atomic orbitals (basis set 6-21 modified, with one *s* shell and three *sp* shells for each silicon atom), 47 $\boldsymbol{k}$ points sampling in the irreducible Brillouin zone, a convergence on energy in SCF procedure equal to 10^{-8} Ha and thresholds for the calculation of Coulomb and exchange integrals given by $(6, 6, 20, 20, 20)$.
The final energy gap obtained with these computational parameters is equal to 0.7659 eV.
The top of the valence band is in Γ point with an energy of -3.07103 eV, while the bottom of the conduction band in $\boldsymbol{k} = (2/5, 2/5, 0)$ has a energy equal to -2.30517 eV.

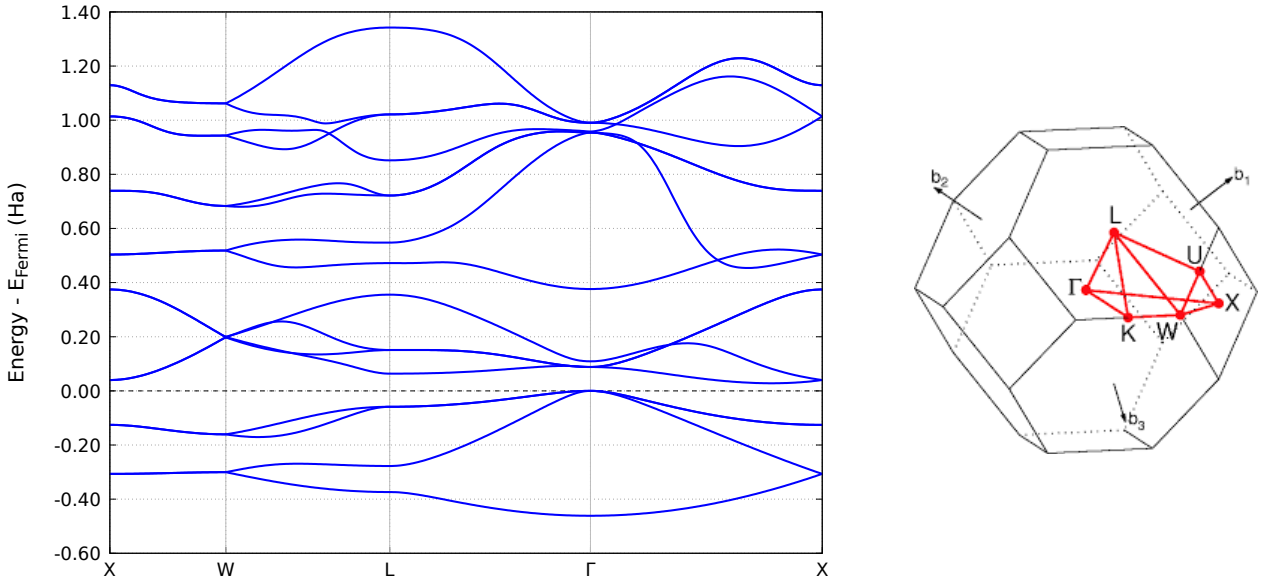

Figure 9.2: Band structure of crystalline silicon computed with DFT (PBE functional), 47 $\boldsymbol{k}$ points sampling in the irreducible Brillouin zone and a convergence on energy in SCF calculation equal to 10^{-8} Ha. The form of the Brillouin zone is reported in the right panel.

9.4.2 Comparison of P and MPP band velocities

The calculation of the band velocities and of the electronic transport properties are performed using a value of NEWK equal to 20 20, that corresponds to 256 reciprocal $\boldsymbol{k}$ points in the irreducible Brillouin zone and to a number of reciprocal $\boldsymbol{k}$ points for the sampling of the transport distribution function equal to $20^3 = 8000$ $\boldsymbol{k}$. The chemical potential range and the energy range of distribution function for transport properties calculation are both chosen to be equal to $(-14.0, 8.0)$ eV, with a step of 0.01 eV. This energy range allows to include in the calculation of the electronic transport properties the contribution from the four valence and the first four non occupied (virtual) electronic bands. The value of the relaxation constant τ within the relation time approximation is taken equal to its default value of 10 fs.
Comparisons are performed between the parallel (P) and the massive parallel (MPP) versions of the CRYSTAL code, by comparing the values of diagonal elements

$$\mathbf{f}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} \qquad \mu = x, y, z \tag{9.182}$$

$$\mathbf{s}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} \qquad \mu = x, y, z \tag{9.183}$$

and of band velocities values

$$\boldsymbol{v}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}\, \boldsymbol{\varepsilon}_\alpha \qquad \mu = x, y, z \tag{9.184}$$

between the two versions, where $\mathbf{C}(\boldsymbol{k})$ are the eigenvectors, $\mathbf{F}(\boldsymbol{k})$ and $\mathbf{S}(\boldsymbol{k})$ are respectively the Kohn-Sham and overlap matrices related to a given $\boldsymbol{k}$ point, $\boldsymbol{\varepsilon}_\alpha$ are the correspondent eigenvalues and the

symbol ∂_μ represents the derivative with respect to the reciprocal lattice vector component k_μ ($\mu = x, y, z$). The absolute values of the difference between the P and MPP diagonal elements in (9.182) and (9.183) and in the band velocities (9.184) are indicated in the following and given by

$$\mathbf{d}_{F,\mu}(\boldsymbol{k}) = \mathrm{abs}\{\,[\mathbf{f}_\mu(\boldsymbol{k})]_{\mathrm{P}} - [\mathbf{f}_\mu(\boldsymbol{k})]_{\mathrm{MPP}}\,\} \tag{9.185}$$

$$\mathbf{d}_{S,\mu}(\boldsymbol{k}) = \mathrm{abs}\{\,[\mathbf{s}_\mu(\boldsymbol{k})]_{\mathrm{P}} - [\mathbf{s}_\mu(\boldsymbol{k})]_{\mathrm{MPP}}\,\} \tag{9.186}$$

$$\mathbf{d}_{v,\mu}(\boldsymbol{k}) = \mathrm{abs}\{\,[\boldsymbol{v}_\mu(\boldsymbol{k})]_{\mathrm{P}} - [\boldsymbol{v}_\mu(\boldsymbol{k})]_{\mathrm{MPP}}\,\} \tag{9.187}$$

All the calculations (P and MPP) are performed using the same fort.9 file (generated through a Pcrystal calculation) containing the information on the electronic ground state wavefunction. An example of the values of the quantities (9.182), (9.183), (9.184) for a particular $\boldsymbol{k}$ point in a particular Cartesian direction ($\mu = x$), in the case of both P and MPP calculations, are reported in Table 9.2.

Silicon Bulk - $\boldsymbol{k}$ point with indexes $(\mathrm{is}_1, \mathrm{is}_2, \mathrm{is}_3) = (0, 0, 1)$ [SHRINK 20 20] - Cartesian direction $\mu = x$

P			MPP			
$\mathfrak{I}(\mathbf{f}_\mu(\boldsymbol{k}))$ [Ha·Bohr]	$\mathfrak{I}(\mathbf{s}_\mu(\boldsymbol{k}))$ [Bohr]	$\mathfrak{I}(\boldsymbol{v}_\mu(\boldsymbol{k}))$ [Ha·Bohr]	$\mathfrak{I}(\mathbf{f}_\mu(\boldsymbol{k}))$ [Ha·Bohr]	$\mathfrak{I}(\mathbf{s}_\mu(\boldsymbol{k}))$ [Bohr]	$\mathfrak{I}(\boldsymbol{v}_\mu(\boldsymbol{k}))$ [Ha·Bohr]	ε Ha
0.0017551002	-0.0000267834	0.0000038400	0.0017551097	-0.0000267835	0.0000038401	-65.3860984455
-0.0030914781	0.0000477327	0.0000295726	-0.0030914959	0.0000477330	0.0000295725	-65.3860304417
0.0096770572	-0.0019195593	0.0000196901	0.0096769942	-0.0019195467	0.0000196902	-5.0310335746
-0.0047061348	0.0009424140	0.0000349847	-0.0047061386	0.0009424148	0.0000349849	-5.0308246709
-0.0029258608	**0.0008619852**	**0.0000411614**	**-0.0023344966**	**0.0006917704**	**0.0000466329**	**-3.4420803620**
-0.0023344962	**0.0006917703**	**0.0000466329**	**-0.0029258603**	**0.0008619850**	**0.0000411614**	**-3.4420803620**
0.0042228743	-0.0012309345	-0.0000140980	0.0042229355	-0.0012309620	-0.0000141315	-3.4420776812
0.0001437066	-0.0000457736	-0.0000138213	0.0001436448	-0.0000457460	-0.0000137879	-3.4414543683
0.0022759250	**-0.0006729136**	**-0.0000398761**	**0.0022759325**	**-0.0006729158**	**-0.0000398760**	**-3.4414536770**
0.0014916679	**-0.0004397002**	**-0.0000215399**	**0.0014916660**	**-0.0004396996**	**-0.0000215399**	**-3.4414536770**
-0.0962409761	0.1220341696	-0.0263027961	-0.0962408998	0.1220340398	-0.0263027942	-0.5731032566
0.4833324139	-1.3968158290	0.3033795644	0.4833324192	-1.3968158480	0.3033795672	-0.1288307634
0.0298856400	**-0.0962859882**	**0.0187342614**	**0.1633076821**	**-0.5178094186**	**0.1033374933**	**-0.1158151757**
0.1633076821	**-0.5178094186**	**0.1033374933**	**0.0298856400**	**-0.0962859882**	**0.0187342614**	**-0.1158151757**
-0.0282754964	0.1309942294	-0.0255218501	-0.0282754958	0.1309942263	-0.0255218496	-0.0210211271
-0.0323812838	**0.2008055355**	**-0.0281842969**	**-0.0323812838**	**0.2008055355**	**-0.0281842969**	**-0.0209007527**
-0.1451161250	**0.7082922249**	**-0.1303122844**	**-0.1451161250**	**0.7082922249**	**-0.1303122844**	**-0.0209007527**
-0.2381474744	0.8708491389	-0.2442567491	-0.2381474469	0.8708491798	-0.2442567219	0.0070153077
-0.0373795303	0.1520528576	-0.0779715862	-0.0373795309	0.1520528547	-0.0779715860	0.2669601644
-0.6230940960	-1.1887758304	0.3519055382	-0.6230940859	-1.1887758296	0.3519055476	0.8201711452
-0.8703705908	**-0.7715183957**	**-0.2187750830**	**-1.1289501620**	**-1.2425370854**	**-0.0795499303**	**0.8445625037**
-1.5255750597	**-1.9650138797**	**0.1340019823**	**-1.2669954885**	**-1.4939951900**	**-0.0052231704**	**0.8445625037**
0.5724374921	0.7211093633	-0.0404352875	0.5724374487	0.7211094870	-0.0404354361	0.8499026789
1.5225730636	**1.9999557347**	**-0.2431263144**	**1.5225730564**	**1.9999557367**	**-0.2431263235**	**0.8828692292**
0.6521235567	**0.7518346128**	**-0.0116480885**	**0.6521235582**	**0.7518346124**	**-0.0116480866**	**0.8828692292**
0.2588799979	0.6470054955	-0.3200602372	0.2588799979	0.6470054709	-0.3200602153	0.8947995638
Sum over the degenerate states:						
-0.0052603570	0.0015537555	0.0000877943	-0.0052603569	0.0015537554	0.0000877943	-3.4420803620
0.0037675929	-0.0011126138	-0.0000614160	0.0037675985	-0.0011126154	-0.0000614159	-3.4414536770
0.1931933221	-0.6140954068	0.1220717547	0.1931933221	-0.6140954068	0.1220717547	-0.1158151757
-0.1774974088	0.9090977604	-0.1584965813	-0.1774974088	0.9090977604	-0.1584965813	-0.0209007527
-2.3959456505	-2.7365322754	-0.0847731007	-2.3959456505	-2.7365322754	-0.0847731007	0.8445625037
2.1746966203	2.7517903475	-0.2547744029	2.1746966146	2.7517903491	-0.2547744101	0.8828692292

Table 9.2: Imaginary parts $\mathfrak{I}(\mathbf{f}_\mu(\boldsymbol{k}))$, $\mathfrak{I}(\mathbf{s}_\mu(\boldsymbol{k}))$ and $\mathfrak{I}(\boldsymbol{v}_\mu(\boldsymbol{k}))$ of the quantities in equations (9.182), (9.183), (9.184) computed along the Cartesian direction ($\mu = x$) in the reciprocal space, for the particular $\boldsymbol{k}$ point with indexes $(0, 0, 1)$ (Cartesian coordinates $(0, 0, 1/20)$ in the reciprocal space), obtained using the Parallel version (P) and the Massive Parallel (MPP) method. The real parts of the reported quantities are all equal to zero. In the last column, the correspondent eigenvectors ($n_{ao} = 26$) are also reported.

As highlighted in Table 9.2, some values of the imaginary part of the quantities (9.182), (9.183) and (9.184) computed using the P version, that are all associated to degenerate eigenvalues, differ from the corresponding values obtained using the MPP code. However, as reported in the last part of the table, by summing up over degenerate states the imaginary parts of the quantities (9.182), (9.183) and (9.184) that differs among each other leads to equal results between the P and the MPP version.
Considering this single example, the necessity of taking the trace for the correct description of the degenerate states clearly emerges. At the same time, taking into account the whole calculation of the electronic transport properties for the case of Silicon bulk crystal, the maximum values of the differences (9.185), (9.186) and (9.187) obtained using the trace for degenerate states in the evaluation of the transport distribution function, as in equation (9.178), and without the trace as in equation (9.176) are reported in Table 9.3. As already noted, significant differences appear when the standard formula (9.176) is applied, while using equation (9.178) decreases considerably these discrepancies.
In the next section, the importance in taking into account the invariant formulation (9.178) when dealing with degenerate states is pointed out, by applying electronic transport properties calculation on a unit cell of Silicon bulk, and comparing the final thermoelectric properties when the non invariant equation (9.176) or the orbital invariant formula (9.178) is used in the calculation of the transport distribution function. Since in this case the aim is to analyze the role of the form of the equations involved in the calculation, without considering effects possibly due to different kinds of implementations, both the formulations (orbital invariant and not invariant) are applied in the parallel (P) version of the CRYSTAL code.

	$\max\{\mathbf{d}_{F,\mu}(\boldsymbol{k})\}$ [Ha·Bohr]	$\max\{\mathbf{d}_{S,\mu}(\boldsymbol{k})]\}$ [Bohr]	$\max\{\mathbf{d}_{v,\mu}(\boldsymbol{k})\}$ [Ha·Bohr]
Without trace	1.80216794	3.00830516	$8.66729501 \cdot 10^{-1}$
With trace	$3.97392868 \cdot 10^{-5}$	$1.15729535 \cdot 10^{-5}$	$4.06469000 \cdot 10^{-7}$

Table 9.3: Maximum values found for the quantities in equations (9.185), (9.186) and (9.187) (representing the differences in the main quantities involved in the transport electronic properties calculation between P and MPP codes) with and without the trace computed on degenerate states. The number of values compared per each quantity is equal to $20^3 \cdot 3 \cdot 26 = 624000$, representing the $n_{ao} = 26$ band velocities along the three Cartesian directions at each $\boldsymbol{k}$ point ($n_{\boldsymbol{k}} = 20^3$) in the transport distribution function reciprocal space sampling net. The eigenvalues at each $\boldsymbol{k}$ point are considered degenerate within a threshold of 10^{-10} Ha.

9.4.3 Effect of band degeneracy on the electronic transport properties

In this section, a comparison of the transport distribution function computed with the equation (7) from G. Sansone et al.[22], given by

$$\Xi_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, v_{\lambda,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_i(\boldsymbol{k})) \tag{9.188}$$

that is not invariant under rotations of the eigenvectors, with the orbital rotation invariant formulation of the transport distribution function, given by

$$\Xi_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_\alpha(\boldsymbol{k})) \sum_{\beta=1}^{m} v_{\lambda,\beta}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_\beta(\boldsymbol{k})) \tag{9.189}$$

is analyzed. These two formulations have been both implemented in the parallel (P) version of the CRYSTAL code, so that the effects possibly coming from different kinds of implementations can be excluded, and the role of the two formulation on the thermoelectric properties can be evaluated consistently. It is worth noting that the two indexes of the band velocities $\{v_{\mu,\alpha}(\boldsymbol{k})\}$ in equations (9.188) and (9.189) are associated to the direction in the reciprocal space $\mu = x, y, z$ and to the atomic orbital $\alpha = 1, ..., m = n_{ao}$, where n_{ao} are the number of atomic orbitals included in the basis set.
Starting from the transport distribution function, the expressions of the three so-called transport coefficients, namely, the electrical conductivity $\boldsymbol{\sigma}$, the Seebeck coefficient $\boldsymbol{S}$, and the electron contribution to

Units		PPROPERTIES : trace vs no trace
[eV· fs/($\hbar^2$Å)]	max{$\Delta\boldsymbol{\Xi}$}	**6.98669**
	Ξ trace [eq. (9.189)]	13.97338
	Ξ no trace [eq. (9.188)]	6.98669
	energy ε [eV]	3.70
	n. degeneracy	1
	degeneracy order	2
[1/(Ω· m)]	max{$\Delta\boldsymbol{\sigma}$}	$\mathbf{2.49860 \cdot 10^6}$
	σ trace	$1.95473 \cdot 10^7$
	σ no trace	$1.70487 \cdot 10^7$
	energy ε [eV]	3.70
	n. degeneracy	1
	degeneracy order	2

Table 9.4: Maximum differences found in the transport electronic properties between two PPROPERTIES calculations with and without the trace computed on degenerate states. $\boldsymbol{\Xi}$ is the transport distribution function (TDF) and $\boldsymbol{\sigma}$ is the electrical conductivity. The eigenvalues at each $\boldsymbol{k}$ point are considered degenerate within a threshold of 10^{-12} Ha. The energy ε correspondent to the point on the TDF energy grid where the maximum difference is found is also reported, together with the degeneracy order found at that point.

the thermal conductivity $\boldsymbol{\kappa}_{el}$, can be derived from Boltzmann semiclassical transport theory (see Section 9.1). The components of the correspondent matrices are given by the following expressions

$$\sigma_{\mu\lambda}(\xi;T) = \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial\varepsilon}\right) \Xi_{\mu\lambda}(\varepsilon) \tag{9.190}$$

$$[\boldsymbol{\sigma S}]_{\mu\lambda}(\xi;T) = \frac{1}{T} \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial\varepsilon}\right) (\varepsilon-\xi)\ \Xi_{\mu\lambda}(\varepsilon) \tag{9.191}$$

$$(\kappa_{el})_{\mu\lambda}(\xi;T) = \frac{1}{T} \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial\varepsilon}\right) (\varepsilon-\xi)^2\ \Xi_{\mu\lambda}(\varepsilon) \tag{9.192}$$

where ξ is the chemical potential, T is the temperature, $\mu = x, y, z$ and $\nu = x, y, z$ are the Cartesian directions in the reciprocal space, f_{eq} is the Fermi-Dirac distribution function and $\Xi_{\mu\nu}(\varepsilon)$ is the transport distribution function.

As test case, the unit cell of Silicon bulk system (see computational details in Section 9.4.1) is considered. The maximum difference found in the values of the transport distribution function $\boldsymbol{\Xi}$ and the electronic conductivity $\boldsymbol{\sigma}$ are reported in Table 9.4, together with the correspondent values involved in the maximum difference, the associated eigenvalue and its degeneracy order. As shown in the Table, significant differences are found between the PPROPERTIES calculations using the different formulations (9.188) and (9.189), all in correspondence with a degenerate eigenvalue. In order to evaluate the contribution of these differences in the thermoelectric properties, the electron conductivity components σ_{xx}, σ_{yy} and σ_{zz} and the Seebeck coefficient components S_{xx}, S_{yy} and S_{zz} of the unit cell of Silicon bulk are computed with PPROPERTIES executable using the orbital rotational invariant formula (9.189) with trace as well as the non invariant equation (9.188) without trace, and they are reported in Figures 9.3, 9.4 and 9.5, respectively. As shown by the figure, very few differences can be noted in the thermoelectric features, as well as in the overall trend. However, two important comments must be done. First of all, looking at the values of the maximum differences found in the corresponding quantities for the two calculations, reported in Table 9.5, significant differences can be noted, especially in the quantity $\boldsymbol{\sigma S}$ and in the electronic contribution to the thermal conductivity $\boldsymbol{\kappa}_{el}$, and the association between the maximum difference and a degenerate energy value is lost (due to the presence of powers of the difference $(\varepsilon-\xi)$ in the definitions of the quantities (9.191) and (9.192), which shift the eigenvalues energies by the chemical potential). Secondly, looking at the three equivalent components xx, yy and zz of the Seebeck coefficient, reported in Figure 9.6, it can be noted that the isotropy of the Silicon cubic system is perfectly reproduced for the three components only if the invariant formulation (9.189) is used. All these findings highlight the importance to consider the correct orbital rotational invariant formulation when degenerate states are involved.

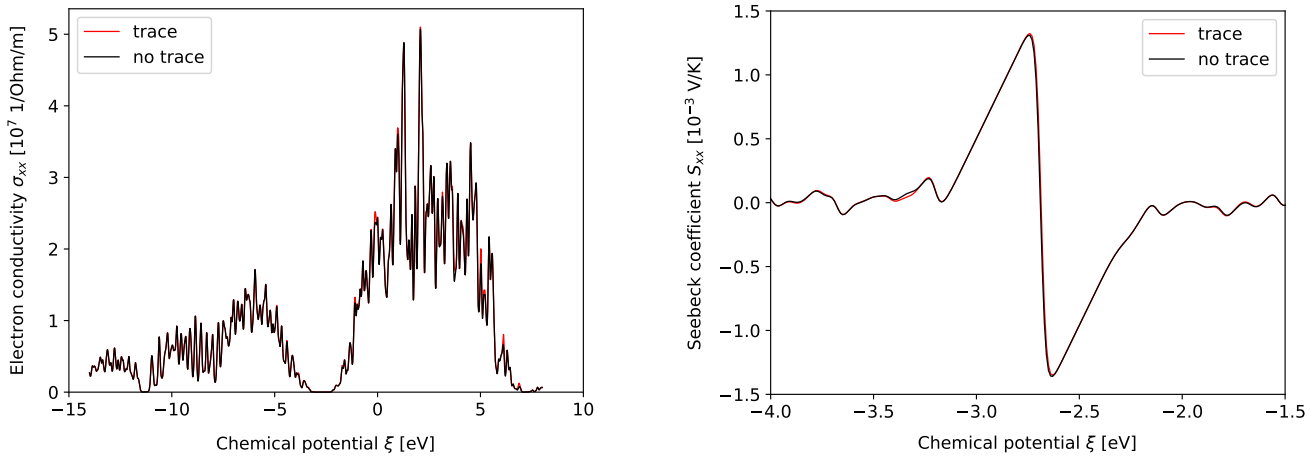

Figure 9.3: Electron conductivity component σ_{xx} (left panel) and Seebeck coefficient component S_{xx} (right panel) computed with PPROPERTIES executable using formula (9.189) with trace (red lines) and formula (9.188) without trace (black lines).

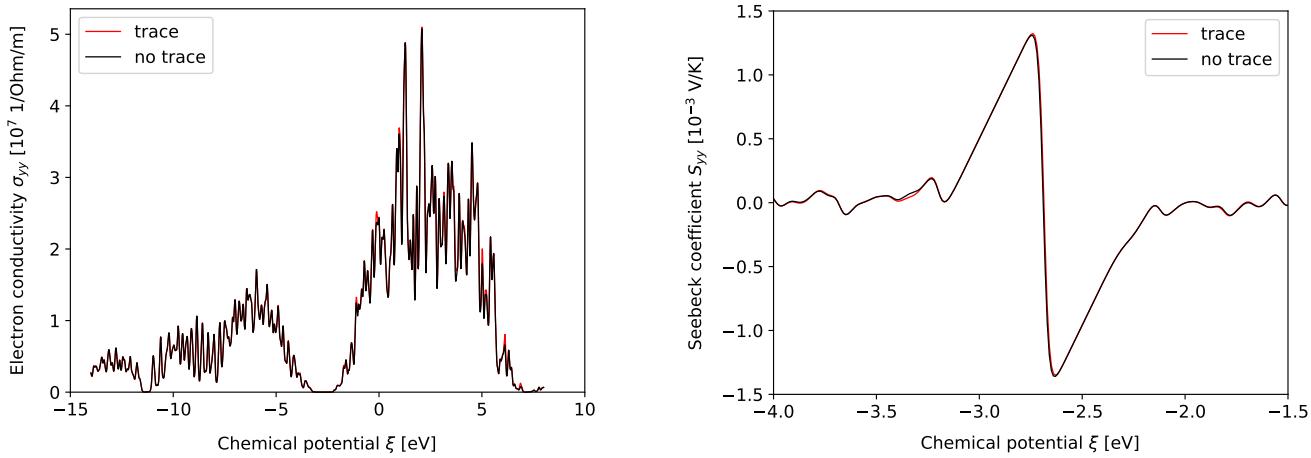

Figure 9.4: Electron conductivity component σ_{yy} (left panel) and Seebeck coefficient component S_{yy} (right panel) computed with PPROPERTIES executable using formula (9.189) with trace (red lines) and formula (9.188) without trace (black lines).

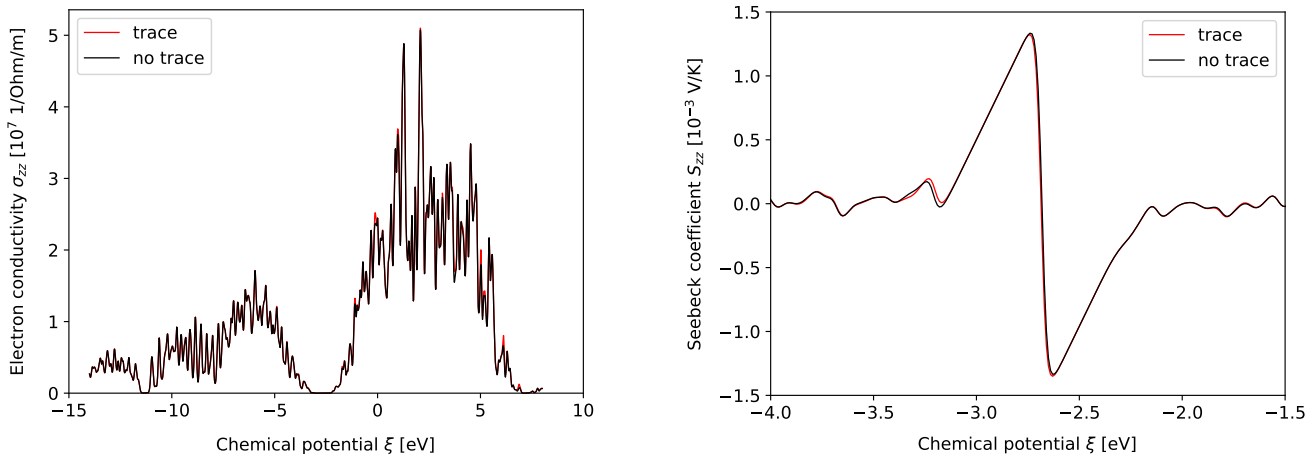

Figure 9.5: Electron conductivity component σ_{zz} (left panel) and Seebeck coefficient component S_{zz} (right panel) computed with PPROPERTIES executable using formula (9.189) with trace (red lines) and formula (9.188) without trace (black lines).

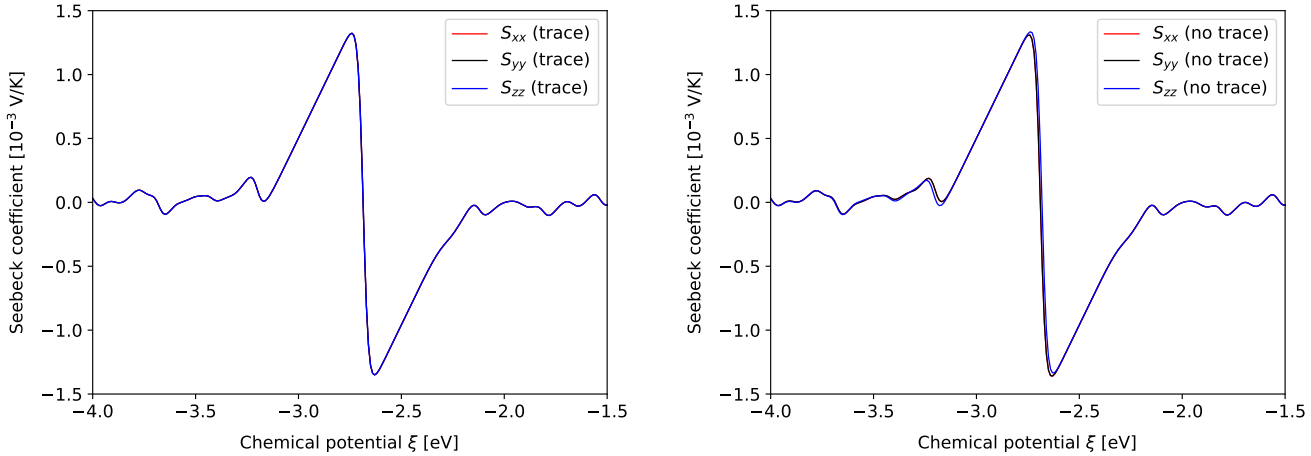

Figure 9.6: Seebeck conductivity components S_{xx}, S_{yy}, S_{zz} computed with PPROPERTIES executable using formula (9.189) with trace (left panel) and formula (9.188) without trace (right panel). The right panel shows that, using formula (9.188) without trace, the isotropy of the Silicon bulk cubic system is not perfectly reproduced.

Units		PPROPERTIES : trace vs no trace
[A/(m·K)]	$\max\{\Delta(\boldsymbol{\sigma S})\}$	$\mathbf{1.92808 \cdot 10^{2}}$
	(σS) trace	$-9.41794 \cdot 10^{2}$
	(σS) no trace	$-1.13460 \cdot 10^{3}$
	energy ε [eV]	3.66
	n. degeneracy	0
	degeneracy order	0
[V/K]	$\max\{\Delta \boldsymbol{S}\}$	$\mathbf{2.45126 \cdot 10^{-4}}$
	S trace	$-3.36452 \cdot 10^{-4}$
	S no trace	$-9.13265 \cdot 10^{-5}$
	energy ε [eV]	-2.68
	n. degeneracy	0
	degeneracy order	0
[W/(m·K)]	$\max\{\Delta \boldsymbol{\kappa}_{el}\}$	**9.77808**
	κ_{el} trace	$1.35113 \cdot 10^{2}$
	κ_{el} no trace	$1.25335 \cdot 10^{2}$
	energy ε [eV]	5.09
	n. degeneracy	0
	degeneracy order	0

Table 9.5: Maximum differences found in the transport electronic properties between two PPROPERTIES calculations with and without the trace computed on degenerate states. $\boldsymbol{\sigma}$ is the electrical conductivity, $\boldsymbol{S}$ is the Seebeck coefficient, and $\boldsymbol{\kappa}_{el}$ is the electron contribution to the thermal conductivity. The eigenvalues at each $\boldsymbol{k}$ point are considered degenerate within a threshold of 10^{-12} Ha. The energy ε correspondent to the point on the TDF energy grid where the maximum difference is found is also reported for each transport property, together with the degeneracy order found at that point.

9.5 Parallel implementation: problems and solutions

The parallel version of CRYSTAL code, in its public version, writes on units 70 (fort.70 files) the eigenvectors $\mathbf{C}_{red}(\boldsymbol{k})$ associated to reducible reciprocal $\boldsymbol{k}$ points generated, through the symmetry of the space group, starting from the reducible $\boldsymbol{k}$ points and the associated eigenvectors $\mathbf{C}_{irr}(\boldsymbol{k})$.
In a typical electronic transport properties calculation, where a dense grid in the reciprocal space has to be used to obtain accurate results, the generation of these fort.70 files can become problematic in terms of disk space usage. Indeed, each file fort.70 contains the eigenvectors, stored in the matrix $\mathbf{C}_{red}(\boldsymbol{k})$ with dimension n_{ao}^2, where n_{ao} is the number of orbitals) of each reducible $\boldsymbol{k}$ point, for each reducible $\boldsymbol{k}$ points in the dense reciprocal space mesh.
In CRYSTAL, each floating point number is a kind selected_real_kind(13, 100) which corresponds to 8 bytes of storage for each floating point number. Therefore, if $n_{\boldsymbol{k}}$ is the total number of reducible $\boldsymbol{k}$ points in the dense reciprocal space mesh, each processor generates a fort.70 file with dimension

$$\text{disk usage for each fort.70 file (same for each processor)} : \qquad 8\, n_{\boldsymbol{k}} \cdot n_{ao} \cdot n_{ao} \text{ bytes} \tag{9.193}$$

In the calculation of electronic transport properties for large defective systems, as for example in the case of half-Heusler ZrNiSn of TiNiSn thermoelectric material with interstitial or substitutional defects, which requires a supercell approach to model a reasonable concentration of defects, the generation of fort.70 files can be as large as or can even easily exceed the total storage capacity of a single node of a modern supercomputer, thus preventing the possibility to conclude the calculation of electronic transport properties. A solution to this problem has been found in the rewriting of the part of the code where the fort.70 files were generated, substituting this part with a version that does not use files to store the information required about the eigenvector related to each reducible reciprocal space point, but instead computes on the fly the eigenvector for each reciprocal space point in the dense mesh used for the calculation.
The calculation of the eigenvectors for reducible $\boldsymbol{k}$ points in the public version of CRYSTAL is performed in the subroutine SMAT, where the computed eigenvectors $\mathbf{C}(\boldsymbol{k}_{red})$ are also written in fort.70 files. In particular, the eigenvectors for reducible $\boldsymbol{k}$ points are computed in the subroutines ESTROF, ESTROE (for complex $\boldsymbol{k}$ points, see Appendix F, Sections F.13 and F.14) and ESTROG (for real $\boldsymbol{k}$ points, see Appendix F, Section F.15), starting from the eigenvectors $\mathbf{C}(\boldsymbol{k}_{irr})$ of the irreducible $\boldsymbol{k}$ points (read from fort.8 files) and the symmetry operators of the lattice space group. Each irreducible $\boldsymbol{k}$ point generates a pool of reducible $\boldsymbol{k}$ points on the base of space group symmetries, so that by applying the symmetry operators on the eigenvector $\mathbf{C}(\boldsymbol{k}_{irr})$ of the reducible $\boldsymbol{k}$ point all the other eigenvectors will be generated by rotation. For example, in the case of silicon bulk using a mesh of $4 \times 4 \times 4 = 64$ reciprocal space points, the first real irreducible $\boldsymbol{k}$ point $\boldsymbol{\Gamma} = (0,0,0) \in \mathcal{R}$ generates only the same point $(0,0,0)$ through the identity symmetry operator, the real irreducible point $\boldsymbol{k} = (2,0,0)$ generates the same point through the identity symmetry operator and the points $(0,2,0)$, $(2,2,2)$ and $(0,0,2)$ through the symmetry operators with indexes $2, 3$ and 4, respectively, while the real irreducible point $\boldsymbol{k} = (2,2,0)$ generates the same point itself through the identity symmetry operator, the points $(0,2,2)$ and $(2,0,2)$ through the symmetry operators 5 and 6, respectively (see Appendix F, Table F.32). At the same time, the complex irreducible $\boldsymbol{k}$ point $(1,0,0)$ generates the same point through the identity symmetry operator, and the points $(0,1,0)$, $(3,3,3)$, $(0,0,1)$, $(0,3,0)$, $(3,0,0)$, $(1,1,1)$ and $(0,0,3)$ by means of the symmetry operators with indexes 2, 3, 4, 13, 14, 15 and 16, respectively (see Appendix F, Table F.30, where also all the other reciprocal space points $\boldsymbol{k}_{red}$ generated by the $(i_{irr,\boldsymbol{k}})$-th point with symmetry operator of index i_{sym} are reported for the case of silicon bulk using a mesh of 64 $\boldsymbol{k}$ points). For all these reducible reciprocal space points generated, the correspondent eigenvector is computed and written on fort.70 files. After that, in subroutine TDF_CALC of boltzatorb.f90 module a loop over the mesh of reducible $\boldsymbol{k}$ points is performed, the eigenvector for each reciprocal space point is read from fort.70 files and the band velocities are computed (see Workflow 1).
As a first attempt to rewrite the code without the creation of input/output fort.70 units, a new subroutine SMAT_SINGLEK has been written (see Workflow 2), following the pattern of the SMAT subroutine, but computing on the fly the eigenvector $\mathbf{C}(\boldsymbol{k}_{red})$ associated to each reducible $\boldsymbol{k}$ point, and using it immediatly in the loop over the reducible reciprocal space points mesh in the subroutine TDF_CALC (boltzatorb.f90 module), as described in Workflow 2. This solution represents only a first attempt to rewrite the code

without the usage of fort.70 input/output files, but it is easily to see that the new code can slow down the calculation with respect to the old implementation. Indeed, the new version is based on the loop over reducible reciprocal space points: for each reducible $\boldsymbol{k}$ point $\boldsymbol{k}_{red}$, the subroutine SMAT_SINGLEK is called, where a loop over the irreducible $\boldsymbol{k}$ points is performed, within which a loop over the symmetry operators is executed until the $(i_{irr,\boldsymbol{k}})$-th irreducible reciprocal space point and the (i_{sym})-th symmetry operator that generates the input $\boldsymbol{k}_{red}$ point required at that point of the TDF_CALC loop is found, so that, using the correspondent symmetry operator, the eigenvector $\mathbf{C}(\boldsymbol{k}_{red})$ can be computed by means of subroutines ESTROF, ESTROE or ESTROG and subsequently used to compute the band velocities in that reciprocal space point $\boldsymbol{k}_{red}$ (see Workflow 2). This procedure, differently from the old one, does not generates a pool of reducible $\boldsymbol{k}$ points looping on the symmetry operators for a given irreducible $\boldsymbol{k}$ point, but instead it searches, for each reducible $\boldsymbol{k}$ point, the irreducible $\boldsymbol{k}$ point and the symmetry operator which generate it, by means of a loop over the symmetry operators for a given irreducible $\boldsymbol{k}$ point, so that the symmetry operator found can be used to compute the eigenvector associated to that reducible $\boldsymbol{k}$ point. This can, obviously, slow down the calculation with respect to the old version, but this difference can be in part reduced by exploiting the parallelization over processors in performing the loop over the reducible reciprocal space points mesh, also implemented in the old version of the code. The tests performed in the following section in order to compare the results and the timings obtained for the electronic transport properties will highlight these differences in computational timings between the old and the new version of the code.
Moreover, in the new version of the code, the creation of fort.70 files for input/output has been disabled by default, but the users can also recover the old version by using the keyword IOACTIVE in the BOLTZTRA section.

9.5.1 Test cases: results and discussion

In order to test the new implementation, a comparison of the quantities

$$\Xi_{\mu\lambda}(\varepsilon) = \tau \sum_{\boldsymbol{k}} \frac{1}{n_{\boldsymbol{k}}} \sum_{\alpha=1}^{m} v_{\mu,\alpha}(\boldsymbol{k})\, v_{\lambda,\alpha}(\boldsymbol{k})\, \delta(\varepsilon - \varepsilon_i(\boldsymbol{k})) \tag{9.194}$$

$$\sigma_{\mu\lambda}(\xi,T) = \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial \varepsilon}\right) \Xi_{\mu\lambda}(\varepsilon) \tag{9.195}$$

$$[\boldsymbol{\sigma S}]_{\mu\lambda}(\xi,T) = \frac{1}{T} \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial \varepsilon}\right) (\varepsilon - \xi)\, \Xi_{\mu\lambda}(\varepsilon) \tag{9.196}$$

$$(\kappa_{el})_{\mu\lambda}(\xi,T) = \frac{1}{T} \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\xi,T)}{\partial \varepsilon}\right) (\varepsilon - \xi)^2\, \Xi_{\mu\lambda}(\varepsilon) \tag{9.197}$$

$$P_{\mu\lambda}(\xi,T) = \sigma_{\mu\lambda}(\xi,T)\, [S_{\mu\lambda}(\xi,T)]^2 \tag{9.198}$$

between the parallel version (P-OLD SVN_1335 public) with input/output in fort.70 files and the parallel version (P-NEW SVN_1286 develop) of the CRYSTAL code with the new implementation is performed, where $\boldsymbol{\Xi}(\varepsilon)$ is the transport distribution function, $\boldsymbol{\sigma}(\xi,T)$ is the electronic conductivity, $\boldsymbol{S}(\xi,T)$ is the Seebeck coefficient, $\boldsymbol{\kappa}_{el}(\xi,T)$ is the electronic contribution to the thermal conductivity and $\boldsymbol{P}(\xi,T)$ is the power factor for a given chemical potential ξ and a given temperature T, while $f_{eq}(\varepsilon,\xi,T)$ is the Fermi distribution and $\boldsymbol{\varepsilon}_i$ are the Hartree-Fock or Kohn-Sham eigenvalues. The absolute values of the difference between all these quantities, as listed in the following, are considered, in order to compare the two versions of the code.

$$\mathbf{d}[\boldsymbol{\sigma}(\xi,T)] = \text{abs}\{\, [\boldsymbol{\sigma}(\xi,T)]_{\text{P-NEW}} - [\boldsymbol{\sigma}(\xi,T)]_{\text{P-OLD}}\,\} \quad \text{File: SIGMA.DAT} \tag{9.199}$$

$$\mathbf{d}[\boldsymbol{\sigma S}(\xi,T)] = \text{abs}\{\, [\boldsymbol{\sigma S}(\xi,T)]_{\text{P-NEW}} - [\boldsymbol{\sigma S}(\xi,T)]_{\text{P-OLD}}\,\} \quad \text{File: SIGMAS.DAT} \tag{9.200}$$

$$\mathbf{d}[\boldsymbol{S}(\xi,T)] = \text{abs}\{\, [\boldsymbol{S}(\xi,T)]_{\text{P-NEW}} - [\boldsymbol{S}(\xi,T)]_{\text{P-OLD}}\,\} \quad \text{File: SEEBECK.DAT} \tag{9.201}$$

$$\mathbf{d}[\boldsymbol{\kappa}_{el}(\xi,T)] = \text{abs}\{\, [\boldsymbol{\kappa}_{el}(\xi,T)]_{\text{P-NEW}} - [\boldsymbol{\kappa}_{el}(\xi,T)]_{\text{P-OLD}}\,\} \quad \text{File: KAPPA.DAT} \tag{9.202}$$

$$\mathbf{d}[\boldsymbol{P}(\xi,T)] = \text{abs}\{\, [\boldsymbol{P}(\xi,T)]_{\text{P-NEW}} - [\boldsymbol{P}(\xi,T)]_{\text{P-OLD}}\,\} \quad \text{File: POWER.DAT} \tag{9.203}$$

All the atomic systems considered as test cases are reported in Table 9.6, together with the computational parameters kept constant during all the simulations (number of symmetry operators, number of orbitals, thresholds for Coulomb and exchange integrals and theoretical methods used).

System name	System	n_{sym}	n_{ao}	TOLINTEG	Method
S1	Silicon (3D)	48	18	6 6 6 6 12	HF
S2	Silicon (3D)	48	18	6 6 20 20 20	DFT (BECKE-PZ)
S3	Lithium (3D)	48	9	6 6 20 20 20	DFT (PWGGA)
S4	$KMnF_3$ (3D)	48	83	7 7 7 7 14	UHF
S5	Berillium (2D)	12	40	6 6 6 6 12	HF
S6	Formamide (1D)	2	114	6 6 20 20 20	DFT (LDA-PZ)

Table 9.6: Name of the atomic systems used as test cases, number of symmetry operators n_{sym}, number of atomic orbitals n_{ao} per unit cell, values of the thresholds TOLINTEG for Coulomb and exchange integrals evaluations and theoretical method used.

The results of the tests, all performed on 8 processors with Intel(R) Xeon(R) CPU E5420@2.50GHz, are reported in Table 9.7.
As already envisaged, the new version P-NEW of the code is slower than the public one, for all the test cases considered and all the calculations using different reciprocal space grids. In particular, the computational time difference is significant for the case of Silicon bulk (systems S1 and S2).
This difference in execution time has to be searched in the structure of the two algorithms. Indeed, in the public version, the main loop which computes all the eigenvectors associated to the reducible reciprocal space point $\boldsymbol{k}_{red}$ is a loop over the irreducible reciprocal space points $\boldsymbol{k}_{irr}$, so that for each irreducible point $\boldsymbol{k}_{irr}$ a pool of reducible reciprocal vectors $\boldsymbol{k}_{red}$ is generated. The parallelization is applied in the SMAT subroutine, for the reading of the eigenvectors related to the irreducible reciprocal space point from fort.8 files and the calculation of the associated pool of eigenvectors for the reducible reciprocal points. After that, the parallelization among processors is applied in the loop over reducible reciprocal points $\boldsymbol{k}_{red}$, where the associated eigenvectors are read from files fort.70 and the band velocities are computed.
On the contrary, in the new version the main loop is the one over the reducible reciprocal points $\boldsymbol{k}_{red}$. At the beginning of the loop, a parallelization over processors is applied in the SMAT_SINGLEK subroutine, in the same way as in the original SMAT routine. In the SMAT_SINGLEK subroutine a loop over the irreducible reciprocal space points $\boldsymbol{k}_{irr}$ is performed in a parallel way (each processor deals with a pool of reciprocal space points) until the irreducible $\boldsymbol{k}_{irr}$ vector that generates the required reducible $\boldsymbol{k}_{red}$ point is found, and the associated eigenvector computed. Once found, the main body of the loop over the reducible $\boldsymbol{k}_{red}$ points is executed in a parallel way, with a parallelization strategy equal to the one adopted in the public version P-PUB.
In this framework, even if the computational cost required by the new version P-NEW is greater than the one implied by the public version P-PUB, the efficiency of the new implemented algorithm should increase by increasing the number of processors involved in the calculation more than the correspondent efficiency of the public version.
On the other hand, this novel implementation permits to perform electronic transport properties calculations on a large mesh of reciprocal space points, without limiting the calculation due to the out of space memory error caused by the generation of fort.70 input/output files.
Moreover, the new implementation paves the way to further improvement in the organization of the algorithm for the calculation of rotated eigenvectors associated to the reducible reciprocal space points, thus leading to an increased efficiency of the overall computational timings with respect to the actual public version.

System: S1 Silicon ($n_{ao} = 18$)								
$n_{\boldsymbol{k}}$	512	4096	32768	64000	110592	175616	262144	373248
T_{cpu} P-NEW [s/proc]	1.12	4.05	49.43	138.06	341.18	780.14	1733.86	3621.57
T_{cpu} P-PUB [s/proc]	0.80	1.10	3.43	6.05	26.76	75.30	161.06	538.28
fort.70 P-PUB [bytes]	1327104	10616832	84934656	165888000	286654464	455196672	679477248	967458816

System: S2 Silicon ($n_{ao} = 18$)								
$n_{\boldsymbol{k}}$	512	4096	32768	64000	110592	175616	262144	373248
T_{cpu} P-NEW [s/proc]	1.12	4.20	49.92	138.80	349.51	779.71	1745.10	3624.56
T_{cpu} P-PUB [s/proc]	0.81	1.09	3.44	6.07	31.69	76.46	165.92	421.14
fort.70 P-PUB [bytes]	1327104	10616832	84934656	165888000	286654464	455196672	679477248	967458816

System: S3 Lithium ($n_{ao} = 9$)								
$n_{\boldsymbol{k}}$	512	4096	13824	32768	64000	110592	140608	175616
T_{cpu} P-NEW [s/proc]	0.97	2.58	9.60	34.55	109.10	266.99	412.31	728.83
T_{cpu} P-PUB [s/proc]	0.79	0.94	1.70	4.39	12.14	31.26	48.16	963.27
fort.70 P-PUB [bytes]	331776	2654208	8957952	21233664	41472000	71663616	91113984	113799168

System: S4 $KMnF_3$ ($n_{ao} = 83$)								
$n_{\boldsymbol{k}}$	128	1024	8192	16000	27648	43904	65536	93312
T_{cpu} P-NEW [s/proc]	3.85	26.42	240.84	502.62	937.78	1614.15	2642.08	4186.83
T_{cpu} P-PUB [s/proc]	1.87	5.04	54.48	126.61	229.33	420.56	784.41	1079.32
fort.70 P-PUB [bytes]	7054336	56434688	451477504	881792000	1523736576	2419637248	3611820032	5142610944

System: S5 Berillium ($n_{ao} = 40$)								
$n_{\boldsymbol{k}}$	256	576	2304	4096	6724	15376	26896	33124
T_{cpu} P-NEW [s/proc]	1.45	2.39	7.76	13.92	23.78	69.97	184.26	274.05
T_{cpu} P-PUB [s/proc]	0.84	0.98	1.77	2.67	4.09	14.86	36.44	49.59
fort.70 P-PUB [bytes]	3276800	7372800	29491200	52428800	86118400 86016000	196812800	344268800	424038400 423936000

System: S6 Formamide ($n_{ao} = 114$)								
$n_{\boldsymbol{k}}$	10	120	180	240	360	420	500	1000
T_{cpu} P-NEW [s/proc]	0.96	3.48	4.89	6.39	9.51	11.12	13.27	28.50
T_{cpu} P-PUB [s/proc]	0.83	1.38	1.69	2.00	2.60	2.90	3.30	6.52
fort.70 P-PUB [bytes]	1663488 831744	12476160	19130112 18298368	24952320	37428480	44082432 43250688	52399872 51568128	103968000

Table 9.7: Computational time (seconds per processor) required by electronic transport properties calculation using the public version of the code (P-PUB) and the new version of the code (P-NEW) without input/output in file fort.70 units. For the public version of the code, the weight in bytes of the fort.70 files per each processor is also reported. The cases where two values of disk usage are listed for a given simulation regard two different weights of fort.70 files generated by different groups of processors.

Workflow 1. Workflow of the code with output writing to fort.70 files

boltzatorb.f90 (module boltzao)
subroutine TDF_CALC

open fort.70 files

call allocate_smat_servi [allocate vectors $\mathbf{C}_{irr}(\boldsymbol{k})$ and $\mathbf{C}_{red}(\boldsymbol{k})$ (dimension: $2n_{ao}^2$)]

call smat(.true.)

↓

libxj.f
subroutine SMAT

Loop 1 (for closed shell systems) or 2 (for open shell systems) times :

- define number of occupied states n_{occ}

Loop over all irreducible $\boldsymbol{k}$ points (index $i_{irr,\boldsymbol{k}}$) :

- separate the case of complex and real $\boldsymbol{k}$ points
- read from fort.8 the eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$ of the irreducible $\boldsymbol{k}$ point (each $\boldsymbol{k}$ point is read by only one processor)
- broadcast the eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$ of the irreducible $\boldsymbol{k}$ point from the processor that read it to all processors

Loop over all symmetry operators (index i_{sym}) :

- generate reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red}$ from irreducible ones $\boldsymbol{k}_{irr}$ using the symmetry operator i_{sym}
- reconstuct the eigenvector $\mathbf{C}_{red}(\boldsymbol{k})$ of reducible $\boldsymbol{k}$ points by applying symmetry operator i_{sym} on irreducible eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$ [subroutines ESTROF ESTROE and ESTROG]
- write the eigenvectors $\mathbf{C}_{red}(\boldsymbol{k})$ of reducible $\boldsymbol{k}$ points on fort.70 files (parallelized over processors if boltzao_active)

↓

boltzatorb.f90 (module boltzao)
subroutine TDF_CALC

call free_smat_servi [deallocate vectors $\mathbf{C}_{irr}(\boldsymbol{k})$ and $\mathbf{C}_{red}(\boldsymbol{k})$ ($2n_{ao}^2$) containing the HF/KS eigenvectors]

call sdig [compute $\mathbf{S}_{irr}(\boldsymbol{g})$ and trasform $\mathbf{S}_{irr}(\boldsymbol{g}) \rightarrow \mathbf{S}_{red}(\boldsymbol{g})$]

call frota(fg_irr, fg_red, idimf, isize_fg_red, uhf, 'F') [transform $\mathbf{F}_{irr}(\boldsymbol{g}) \rightarrow \mathbf{F}_{red}(\boldsymbol{g})$]

Loop 1 (for closed shell systems) or 2 (for open shell systems) times (parallelized over processors) :

Loop over all reducible $\boldsymbol{k}$ points (index $i_{red,\boldsymbol{k}}$) (parallelized over processors) :

- read eigenvectors $\mathbf{C}_{red}(\boldsymbol{k})$ from files fort.70
- call velocity_effmass_calc [$\mathbf{C}_{red}(\boldsymbol{k}) \equiv \mathbf{C}(\boldsymbol{k})$] :
  - Fourier transform $\mathbf{F}(\boldsymbol{g}) \rightarrow \partial_\mu \mathbf{F}(\boldsymbol{k})$ $\quad \mu = x, y, z$
  - compute $\mathbf{f}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}$ $\quad \mu = x, y, z$
  - Fourier transform $\mathbf{S}(\boldsymbol{g}) \rightarrow \partial_\mu \mathbf{S}(\boldsymbol{k})$ $\quad \mu = x, y, z$
  - compute $\mathbf{s}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}$ $\quad \mu = x, y, z$
- compute the band velocities :

  $\boldsymbol{v}_\mu(\boldsymbol{k}) \equiv \mathbf{f}_\mu(\boldsymbol{k}) - \mathbf{s}_\mu(\boldsymbol{k}) = [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}\, \boldsymbol{\varepsilon}_\alpha \qquad \mu = x, y, z$
- compute number of carriers $n_c(\mu, T)$
- compute Transport Distribution Function $\Xi_{\mu\nu}(\varepsilon)$ [see equations (9.176) and (9.178)]

call gsum($n_c(\mu, T)$) and call gsum($\Xi_{\mu\nu}(\varepsilon)$) $\rightarrow$ collect data from all processors

Workflow 2. Workflow of the code without output writing to fort.70 files

boltzatorb.f90 (module boltzao)
subroutine TDF_CALC

call sdig [compute $\mathbf{S}_{irr}(\boldsymbol{g})$ and trasform $\mathbf{S}_{irr}(\boldsymbol{g}) \rightarrow \mathbf{S}_{red}(\boldsymbol{g})$]

call frota(fg_irr, fg_red, idimf, isize_fg_red, uhf, 'F') [transform $\mathbf{F}_{irr}(\boldsymbol{g}) \rightarrow \mathbf{F}_{red}(\boldsymbol{g})$]

Loop 1 (for closed shell systems) or 2 (for open shell systems) times :

Loop over all reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red}$ (index $i_{red,\boldsymbol{k}}$) :

- call **SMAT_SINGLEK** :
  - find the irreducible $\boldsymbol{k}$ point and the symmetry operator i_{sym} that generate the reducible $\boldsymbol{k}$ point $\boldsymbol{k}_{red}$
  - read from fort.8 the eigenvector of the irreducible $\boldsymbol{k}$ point (each $\boldsymbol{k}$ point is read by only one processor)
  - broadcast the eigenvector of the irreducible $\boldsymbol{k}$ point from the processor that read it to all processors
  - reconstuct the eigenvector $\mathbf{C}_{red}(\boldsymbol{k})$ of reducible $\boldsymbol{k}$ points by applying symmetry operator i_{sym} on irreducible eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$ [subroutines ESTROF ESTROE and ESTROG]
- call velocity_effmass_calc (parallelized over processors) $[\mathbf{C}_{red}(\boldsymbol{k}) \equiv \mathbf{C}(\boldsymbol{k})]$:
  - Fourier transform $\mathbf{F}(\boldsymbol{g}) \rightarrow \partial_\mu \mathbf{F}(\boldsymbol{k})$ $\quad \mu = x, y, z$
  - compute $\mathbf{f}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}$ $\quad \mu = x, y, z$
  - Fourier transform $\mathbf{S}(\boldsymbol{g}) \rightarrow \partial_\mu \mathbf{S}(\boldsymbol{k})$ $\quad \mu = x, y, z$
  - compute $\mathbf{s}_\mu(\boldsymbol{k}) \equiv [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}$ $\quad \mu = x, y, z$
- compute the band velocities (parallelized over processors) $[\mathbf{C}_{red}(\boldsymbol{k}) \equiv \mathbf{C}(\boldsymbol{k})]$:

  $\boldsymbol{v}_\mu(\boldsymbol{k}) \equiv \mathbf{f}_\mu(\boldsymbol{k}) - \mathbf{s}_\mu(\boldsymbol{k}) = [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{F}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha} - [\mathbf{C}^\dagger(\boldsymbol{k})\, \partial_\mu \mathbf{S}(\boldsymbol{k})\, \mathbf{C}(\boldsymbol{k})]_{\alpha\alpha}\, \boldsymbol{\varepsilon}_\alpha \qquad \mu = x, y, z$
- compute number of carriers $n_c(\mu, T)$ (parallelized over processors)
- compute Transport Distribution Function $\Xi_{\mu\nu}(\varepsilon)$ (parallelized over processors)

call gsum($n_c(\mu, T)$) and call gsum($\Xi_{\mu\nu}(\varepsilon)$) $\rightarrow$ collect data from all processors

libxj.f
subroutine SMAT_SINGLEK

Loop over all irreducible $\boldsymbol{k}$ points (index $i_{irr,\boldsymbol{k}}$) :

- separate the case of complex and real $\boldsymbol{k}$ points

Loop over all symmetry operators (index i_{sym}) :

- find the index of the irreducible $\boldsymbol{k}$ point $\boldsymbol{k}_{irr}$ and the symmetry operator i_{sym} that generate the reducible $\boldsymbol{k}$ point $\boldsymbol{k}_{red}$

Read from fort.8 the eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$ of the irreducible $\boldsymbol{k}$ point (each $\boldsymbol{k}$ point is read by only one processor)

Broadcast the eigenvector of the irreducible $\boldsymbol{k}$ point from the processor that read it to all processors

Separate the case of complex and real $\boldsymbol{k}$ points and Reconstuct the eigenvector $\mathbf{C}_{red}(\boldsymbol{k})$ of the reducible $\boldsymbol{k}$ points by applying symmetry operator i_{sym} on irreducible eigenvector $\mathbf{C}_{irr}(\boldsymbol{k})$
[subroutines ESTROF ESTROE and ESTROG]

Chapter 10

Conclusions and future perspectives

The theoretical foundation for the formalization and the definition of different ensembles in molecular dynamics simulations has been made uniform and general. Starting from a Lagrangian and Hamiltonian formulation, the equations of motion have been derived and integrated in a symplectic way on the base of the Suzuki-Trotter decomposition method.[34, 35] This theoretical procedure is general, and can be used to find the equations of motion and the correspondent time propagation algorithm for all the ensembles of interest. In particular, the microcanonical, canonical and isothermal-isobaric ensembles have been formalized and successively implemented in the CRYSTAL code. For the purpose of demonstrating the capabilities of molecular dynamics module in the CRYSTAL code, calculations on the proton-ordered ice crystalline structure and the liquid-like $(H_2O)_{32}$ periodic cubic system have been performed in microcanonical and canonical ensembles. These two models are representative of hydrogen bonded environments, that are known to be very difficult to characterize, thus being the starting test cases to validate computational methods and the corresponding practical implementation in quantum mechanical packages. At the same time, these systems are very well known and extensively studied, thus permitting a strict comparison with the results find in existing literature. First of all, the accuracy in terms of the drift in the conserved quantity and the related standard deviation has been evaluated, for all the molecular dynamics simulations in both the microcanonical (NVE) and the canonical (NVT) ensembles. The data demonstrate the presence of a very small drift in the conserved quantity, comparable with that obtained using other ab initio molecular dynamics codes,[148, 149, 150] and a good temperature control reproduced by the NVT ensemble. The calculation of dynamical properties from the post processing of molecular dynamics trajectory has been also very useful to evaluate the accuracy and capabilities of the Born-Oppenheimer Molecular Dynamics implementation. The results, expecially using B3LYP functional, are in good agreement with both experimental findings and theoretical calculations found in literature using other quantum mechanical codes. These results demonstrate that also the dynamical properties can be accurately determined from both the microcanonical and the canonical (Nosé thermostat) ensembles, as implemented in the CRYSTAL code. All these theoretical novelties and the results obtained by means of Born-Oppenheimer molecular dynamics with the algorithms implemented in the CRYSTAL code lead to the publication of the article:
C. Ribaldone and S. Casassa, *Born-Oppenheimer Molecular Dynamics with Linear Combination of Atomic Orbitals and Hybrid Functionals for Condensed Matter Simulations Made Possible. Theory and Performance for the Microcanonical and Canonical Ensembles*, Journal of Chemical Theory and Computation, accepted manuscript (January 2024), DOI: https://doi.org/10.1021/acs.jctc.3c01231.

As regards the Fast Inertial Relaxation Engine structural optimization method, its implementation, in the basic and improved versions, has been described and its efficiency and reliability in the framework of the CRYSTAL code has been assessed. A screening of the four FIRE adjustable parameters has been realized through structural optimizations of atomic systems with different dimensionality, identifying a set of robust default values. Then, structural optimizations of atomic systems with different number of atoms, dimensionality and kind of bonds have been performed with FIRE algorithm. The accuracy of FIRE in finding energy minima of the potential energy surface (PES) for these systems has been demonstrated comparing the total energy and geometry of FIRE final structures with those obtained with quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme, already implemented in the CRYSTAL

code. The reliability of FIRE in minimizing the PES shaped by different functionals, such as PBE, B3LYP, HSE06 and PBE0 has been proven for the case of urea molecular crystal. Finally, as regards the computational time, the conclusion it that the FIRE structural optimization method has generally a greater computational cost than the correspondent BFGS one, due to the fact that it employs more iterations to reach convergence. Nevertheless, a single FIRE step has a less computational cost than a BFGS one, so that the overall FIRE minimization becomes the most efficient one when the increasing of the computational cost due to a greater number of iterations is compensated for by a reduction of timings in performing each single step. This study paves the way for other important implementations, namely, a finite temperature structural optimization algorithm and the Nudged Elastic Band method for transition states calculations. These results lead to the publication of the article:
C. Ribaldone and S. Casassa, *Fast Inertial Relaxation Engine in the* CRYSTAL *code*, AIP Advances 12, 015323 (2022), DOI: https://doi.org/10.1063/5.0082185.

As regards the electronic transport properties, the Massive Parallel Processing (MPP) approach has been implemented in the CRYSTAL code, exploiting the independence of the points in the reciprocal space in order to parallelize the calculation of the band velocities, thus improving the capability of the code to compute transport properties on dense reciprocal grids. In the route through the implementation of MPP transport properties calculations, the influence of the presence of degenerate electronic states in the band structure has been addressed. In particular, it has to be underlined that the formula reported in the article by G. Sansone et al.[22] and currently used in the CRYSTAL code for the calculation of the transport distribution function is not invariant under rotation of atomic orbitals in the case of degenerate electronic states. For this reason, an atomic orbitals rotational invariant formula has been suggested for the calculation of the transport distribution function, which reduces itself to the standard formulation when the electronic states are not degenerate. The necessity of this correction in the formula for the transport distribution function has been also confirmed by other recent theoretical studies.[146] Since the transport distribution function is a key quantity used to compute all the electronic transport properties, the effect of the correct formula in degenerate cases has to be assessed in the calculation of the transport coefficients. For this reason, a series of crystalline materials with a certain level of degeneracy in the band structures have been analyzed, from the basic crystalline silicon case to the $SrTiO_3$ cubic perovskite crystal. Very little changes in the final results obtained using the invariant formulation have been found for the electronic conductivity and the Seebeck coefficient with respect to the non invariant formulation. More interestingly, the isotropy of the Seebeck coefficient is perfectly reproduced for isotropic crystalline systems using the invariant formulation, while anisotropic effects appear in the Seebeck coefficients along the three spatial directions when using the non invariant formulation. A further analysis of these effects has to be performed, exploring different materials with high degeneracy in the band structures, where the effect of degenerate states could be important and could affect in a significant way the electronic transport properties.

Appendices

Appendix A

Notation stuffs and demonstrations

A.1 Maxwell-Boltzmann distribution

In 1860, James Clerk Maxwell published a derivation[151] of what it is now called the Maxwellian velocity distribution, the distribution of molecular speeds in an ideal gas in thermal equilibrium. The Maxwell-Boltzmann function describes the distribution of velocities in a non-interacting gas of particles. Remarkably, the Maxwell distribution also holds in the presence of any interactions. In fact, Maxwell original derivation[151] of the distribution makes no reference to any properties of the gas.
Let N be the whole number of particles. Let v_x, v_y, v_z be the components of the velocity of each particle in three spatial directions, and let the number of particles for which v_x lies in the interval $(v_x, v_x + dv_x)$ be $Nf(v_x)dv_x$, where $f(v_x)$ is a function of v_x to be determined. The number of particles for which v_y lies in the interval $(v_y, v_y + dv_y)$ will be $Nf(v_y)dv_y$; and the number for which v_z lies in the interval $(v_z, v_z + dv_z)$ will be $Nf(v_z)dv_z$, where f always stands for the same function.
The essential ingredient of the demonstration was an assumption, motivated by symmetry and mathematical considerations, that the velocity-space number density of atoms as a function of speed must factor into separate, identical functions of the Cartesian velocity components. Indeed, the existence of the velocity v_x does not in any way affect that of the velocities v_y or v_z, since these are all at right angles to each other and independent, so that the number of particles whose velocity lies between v_x and $v_x + dv_x$, and also between v_y and $v_y + dv_y$, and also between v_z and $v_z + dv_z$, is

$$Nf(v_x)f(v_y)f(v_z)\ dv_x\, dv_y\, dv_z \tag{A.1}$$

Suppose that the N particles start from the origin at the same instant, then this will be the number in the element of volume $(dv_x\, dv_y\, dv_z)$ after unit of time, and the number referred to unit of volume will be

$$Nf(v_x)f(v_y)f(v_z) \tag{A.2}$$

But the orientation of the coordinates frame of reference is perfectly arbitrary, and therefore this number must depend on the distance from the origin alone. Indeed, rotational symmetry means that the form of the velocities components distributions in the x, y, z directions has to be the same. However, rotational invariance also requires that the full distribution can not depend on the direction of the velocity, but it can only depend on the speed

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{A.3}$$

This means that the form of two functions have to be found, $\tilde{F}(v)$ and $f(v_j)$ with $j = x, y, z$, such that

$$N\tilde{F}(v)\, dv_x\, dv_y\, dv_z = Nf(v_x)f(v_y)f(v_z)\, dv_x\, dv_y\, dv_z \tag{A.4}$$

Therefore, the distribution of velocities in the atomic system can be written as

$$\tilde{F}(v) = f(v_x)f(v_y)f(v_z) \tag{A.5}$$

It does not look as if there is enough information to solve this equation for both $\tilde{F}(v)$ and $f(v_j)$, with $j = x, y, z$. But, remarkably, there is only one solution. By differentiation of equation (A.5), the following

expression is obtained

$$\frac{\partial \tilde{F}(v)}{\partial v_x} = \frac{d\tilde{F}(v)}{dv}\frac{\partial v}{\partial v_x} = \frac{d\tilde{F}(v)}{dv}\frac{v_x}{v} = \frac{df(v_x)}{dv_x} f(v_y) f(v_z) \tag{A.6}$$

Rewriting the last equivalence of the previous equation leads to

$$\frac{d\tilde{F}(v)}{dv}\frac{v_x}{v} = \frac{df(v_x)}{dv_x} f(v_y) f(v_z) \tag{A.7}$$

Dividing this expression by the factor $v_x \tilde{F}(v) = v_x f(v_x) f(v_y) f(v_z)$ gives

$$\frac{1}{v\tilde{F}(v)}\frac{d\tilde{F}(v)}{dv} = \frac{1}{v_x f(v_x)}\frac{df(v_x)}{dv_x} \tag{A.8}$$

Following the same calculations for v_y and v_z components, two other equations can be found,

$$\frac{1}{v\tilde{F}(v)}\frac{d\tilde{F}(v)}{dv} = \frac{1}{v_y f(v_y)}\frac{df(v_y)}{dv_y} \qquad\qquad \frac{1}{v\tilde{F}(v)}\frac{d\tilde{F}(v)}{dv} = \frac{1}{v_z f(v_z)}\frac{df(v_z)}{dv_z} \tag{A.9}$$

which are consequences of the symmetry between velocity components. Given the mathematical independence of the velocity components, each of the equal terms in (A.8) and (A.9) must in fact be constant,

$$\frac{1}{v\tilde{F}(v)}\frac{d\tilde{F}(v)}{dv} = \text{constant} \tag{A.10}$$

Upon integration, one finds

$$\tilde{F}(v) = A\, e^{\pm Bv^2} \tag{A.11}$$

where $A > 0$, $B > 0$ and the positive or negative sign in the exponent depends on the sign of the constant chosen in equation (A.10). If the positive sign is chosen in the exponent, then the number of particles will increase with the velocity, so that the whole number of particles is find to be infinite. For this reason, the negative sign has to be chosen in equation (A.11), leading to the solution

$$\tilde{F}(v) = A\, e^{-Bv^2} \tag{A.12}$$

where A and B are positive constants. The insertion of the solution (A.12) in equations (A.8) and (A.9) leads to

$$\frac{1}{v_j f(v_j)}\frac{df(v_j)}{dv_j} = -2B \qquad\qquad j = x, y, z \tag{A.13}$$

and the solution of the previous differential equation is given by

$$f(v_j) = C\, e^{-Bv_j^2} \qquad\qquad j = x, y, z \tag{A.14}$$

where the constant C can be determined inserting the solutions (A.12) and (A.14) in the relation (A.5),

$$\tilde{F}(v) = A\, e^{-Bv^2} = f(v_x) f(v_y) f(v_z) = C^3\, e^{-Bv_x^2}\, e^{-Bv_y^2}\, e^{-Bv_z^2} = C^3\, e^{-B(v_x^2+v_y^2+v_z^2)} = C^3\, e^{-Bv^2}$$

$$\rightarrow \qquad C = A^{1/3}$$

Finally, inserting this expression for the constant in the solution (A.14) gives the following distribution for each velocity component

$$f(v_j) = A^{1/3}\, e^{-Bv_j^2} \qquad\qquad j = x, y, z \tag{A.15}$$

This procedure shows that the functional form of the velocity distributions in an atomic system arises from the rotational invariance condition (A.5) alone. Using (A.12) and (A.4), the probability that one atom has a velocity component along x that lies between v_x and $v_x + dv_x$, a velocity component along

y that lies between the values v_y and $v_y + dv_y$, and a velocity component along z that lies in the range of values v_z and $v_z + dv_z$, is given by

$$\tilde{F}(v)\, dv_x\, dv_y\, dv_z = A\, e^{-Bv^2}\, dv_x\, dv_y\, dv_z \tag{A.16}$$

The positive normalization factor A as a function of the positive constant B can be determined by insisting that the probabilities sum to one, namely,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{F}(v)\, dv_x\, dv_y\, dv_z = 1 \tag{A.17}$$

Inserting the form (A.12) of the function $\tilde{F}(v)$ in the previous integrals and solving them leads to

$$\begin{aligned}&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{F}(v)\, dv_x\, dv_y\, dv_z = A\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-Bv^2}\, dv_x\, dv_y\, dv_z \\ &= A\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-B(v_x^2+v_y^2+v_z^2)}\, dv_x\, dv_y\, dv_z = A\underbrace{\int_{-\infty}^{\infty} e^{-Bv_x^2}\, dv_x}_{=\sqrt{\pi/B}}\underbrace{\int_{-\infty}^{\infty} e^{-Bv_y^2}\, dv_y}_{=\sqrt{\pi/B}}\underbrace{\int_{-\infty}^{\infty} e^{-Bv_z^2}\, dv_z}_{=\sqrt{\pi/B}}\end{aligned}$$

Therefore, the positive normalization constant A is given by

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{F}(v)\, dv_x\, dv_y\, dv_z = A\left(\frac{\pi}{B}\right)^{3/2} = 1 \qquad \rightarrow \qquad A = \left(\frac{B}{\pi}\right)^{3/2} \tag{A.18}$$

At the same time, from the theorem of the equipartition of energy the average kinetic energy of the atomic system can be written as

$$\frac{1}{2}m\,\langle v^2\rangle = \frac{3}{2}k_b T_a \tag{A.19}$$

where T_a is the actual temperature of the system, given by the instantaneous velocity distribution. The average square speed of the particles can thus the be expressed as

$$\langle v^2\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v^2\, \tilde{F}(v)\, dv_x\, dv_y\, dv_z = \frac{3k_b T_a}{m} \tag{A.20}$$

where the first identity above has been written following the definition of average quantity in a system with probability density function equal to $\tilde{F}(v)$, and the last identity above has been written following the equipartition theorem (A.19). From the condition (A.20), the positive constant B can be derived, so that the value of the positive normalization factor A in (A.18) will be found, and the velocity distribution (A.12) will be completely defined. Indeed, the solution of the integrals in (A.20) is

$$\begin{aligned}&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v^2\, \tilde{F}(v)\, dv_x\, dv_y\, dv_z = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (v_x^2+v_y^2+v_z^2)\, \tilde{F}(v)\, dv_x\, dv_y\, dv_z \qquad (A.21)\\ &= \sum_{j=x,y,z}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_j^2\, \tilde{F}(v)\, dv_x\, dv_y\, dv_z = A\sum_{j=x,y,z}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_j^2\, e^{-B(v_x^2+v_y^2+v_z^2)}\, dv_x\, dv_y\, dv_z\end{aligned}$$

Since the three integrals $j = x, y, z$ have the same form, the solution for the case $j = x$ is derived in the following, while for the other two integrals the solution can be extended with minor differences.

$$\begin{aligned}j = x: \qquad &\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_x^2\, e^{-B(v_x^2+v_y^2+v_z^2)}\, dv_x\, dv_y\, dv_z = \underbrace{\int_{-\infty}^{\infty} e^{-Bv_z^2}\, dv_z}_{=\sqrt{\pi/B}}\underbrace{\int_{-\infty}^{\infty} e^{-Bv_z^2}\, dv_z}_{=\sqrt{\pi/B}}\int_{-\infty}^{\infty} v_x^2\, e^{-Bv_x^2}\, dv_x \\ &= \sqrt{\frac{\pi}{B}}\sqrt{\frac{\pi}{B}}\int_{-\infty}^{\infty} v_x^2\, e^{-Bv_x^2}\, dv_x = \frac{\pi}{B}\left(-\frac{v_x}{2B}e^{-Bv_x^2}\Big|_{-\infty}^{\infty} + \frac{1}{2B}\underbrace{\int_{-\infty}^{\infty} e^{-Bv_x^2}\, dv_x}_{=\sqrt{\pi/B}}\right) = \frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2}\end{aligned}$$

where the first integral in the second line with integration variable dv_x has been solved by parts.

The same result is obtained for $j = y, z$, namely,

$$j = y: \qquad \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_y^2\, e^{-B(v_x^2+v_y^2+v_z^2)}\, dv_x\, dv_y\, dv_z = \frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2}$$

and

$$j = z: \qquad \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} v_z^2\, e^{-B(v_x^2+v_y^2+v_z^2)}\, dv_x\, dv_y\, dv_z = \frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2}$$

Finally, substituting these results in (A.21), the solution of the integrals can be written as

$$\langle v^2\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{v}^2\, F(v)\, dv_x\, dv_y\, dv_z = A\left[\frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2} + \frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2} + \frac{1}{2B}\left(\frac{\pi}{B}\right)^{3/2}\right] = \frac{3A}{2B}\left(\frac{\pi}{B}\right)^{3/2}$$

Using equation (A.20) derived from the theorem of energy equipartition, the previous product of constant will be equal to

$$\frac{3A}{2B}\left(\frac{\pi}{B}\right)^{3/2} = \langle v^2\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{v}^2\, F(v)\, dv_x\, dv_y\, dv_z = \frac{3k_bT_a}{m} \tag{A.22}$$

The constant B can be finally derived substituting the expression for the normalization constant A as a function of B, given by expression (A.18), in the previous equation

$$\frac{3A}{2B}\left(\frac{\pi}{B}\right)^{3/2} = \frac{3}{2B}\left(\frac{B}{\pi}\right)^{3/2}\left(\frac{\pi}{B}\right)^{3/2} = \frac{3}{2B} = \frac{3k_bT_a}{m} \qquad \rightarrow \qquad B = \frac{m}{2k_bT_a} \tag{A.23}$$

Recollecting the results obtained, the probability density function for the velocity speed in an atomic system (in a Cartesian coordinates reference frame) is

$$\tilde{F}(v) = A\, e^{-Bv^2} \tag{A.24}$$

where the normalization constant A and the exponential factor B are both positive constants, that are derived from the probability density function normalization condition and the theorem of energy equipartition to be equal to, respectively,

$$B = \frac{m}{2k_bT_a} \qquad\qquad A = \left(\frac{B}{\pi}\right)^{3/2} = \left(\frac{m}{2\pi k_bT_a}\right)^{3/2} \tag{A.25}$$

Using these positive constants, the probability density function for the velocity speed in an atomic system (in a Cartesian coordinates reference frame) assumes the form

$$\tilde{F}(v) = \left(\frac{m}{2\pi k_bT_a}\right)^{3/2} e^{-mv^2/(2k_bT_a)} = \left(\frac{m}{2\pi k_bT_a}\right)^{3/2} e^{-m(v_x^2+v_y^2+v_z^2)/(2k_bT_a)} \tag{A.26}$$

The probability (A.16) is expressed in Cartesian coordinates in the space of the velocity. However, the variable v defined by (A.3) has the form of a typical radial variable, and it is therefore more convenient to rewrite the probability (A.16) using spherical coordinates. The change of variable from Cartesian to spherical coordinates can lead to a probability function $F(v)\, dv$ that depends only on the velocity variable (A.3), and not on the velocity vectors components, thus representing and satisfying the space rotational invariance required. Starting from the probability density function (A.26) whose form has been just derived, and proceeding with the change of variables in the same way as in Section 3.1 (see equation (3.37) and the derivation below it), the probability density function associated to the distribution of velocity modulus in a classical atomic system can be written as

$$F(v) = 4\pi\left(\frac{m}{2\pi k_bT_a}\right)^{3/2} v^2\, e^{-mv^2/(2k_bT_a)} \tag{A.27}$$

and the probability that an atom in the system has a velocity modulus whose value lies in the range $(v, v + dv)$ will be given, using spherical coordinates, by $F(v)dv$ (see equation (3.38)).

A.1.1 Initial nuclear velocities distribution: extension and examples

As demonstrated in Section 3.1, by using the Box-Muller algorithm the probability density function for the collection of $3N$ random numbers u_i (where N is the number of atoms in the system) is a standard normal distribution with the form (3.23). In order to obtain the correspondent distribution for physical meaningful nuclear velocities variables v_i, the transformation (3.25) has to be performed, finally obtaining a set of initial random nuclear velocities components distributed following the normalized Gaussian probability density functions (3.28). Each one of the three components of the nuclear velocities for the collection of atoms follows a own Gaussian distribution, given by the three probability density functions in (3.30)-(3.32). Therefore, it can be stated that the three components of the velocity vectors follow, respectively, the three probability density functions

$$f(v_j) = \sqrt{\frac{m}{2\pi k_b T_a}}\; e^{-mv_j^2/(2k_bT_a)} \qquad\qquad j = x, y, z \tag{A.28}$$

The three probability density functions (A.28) for the nuclear velocities components have a Gaussian form with standard deviation given by

$$\sigma = \sqrt{\frac{2k_bT_a}{m}} \tag{A.29}$$

Therefore, their standard deviations depend on the atomic mass m and on the temperature T_a of the system. The greater the atomic mass, the less the standard deviation, the stricter the dispersion of the Gaussian distribution. Conversely, the greater the temperature of the system, the greater the standard deviation, the broader the dispersion of the Gaussian. The value of the system temperature T_a has an effect also on the intensity (on the maximum point) of the Gaussian. The grater the temperature T_a, the less the value of the standard deviation σ, the greater the multiplicative coefficient associated to the Gaussian distributions (A.28), the greater the maximum intensity of the Gaussian distributions.
The product of the three probability density functions (A.28), transformed from Cartesian to spherical coordinate reference frame, gives the probability density function followed by the nuclear velocities vectors modulus, described by the form (3.39), reported here below for completeness

$$F(v) = 4\pi \left(\frac{m}{2\pi k_b T_a}\right)^{3/2} v^2\, e^{-mv^2/(2k_bT_a)} \tag{A.30}$$

where v is the modulus of the velocity vector associated to the nuclei, given by

$$v = \|\boldsymbol{v}\| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{A.31}$$

Figure A.1 reports the histogram for the three components of nuclear velocities and for the modulus of the nuclear velocities initialized for a $20\times10\times10$ supercell ($N = 4000$ atoms) of Silicon crystalline system with an initial temperature equal to $T_a = 600$ K (upper panel) and $T_a = 1200$ K (lower panel). The black curves in the first three left panels (upper and lower) are the functions obtained by plotting the normalized Gaussian distributions with probability density functions (A.28), and the black curves in the last panel (upper and lower) are the Maxwell-Boltzmann distributions with probability density functions of the form (A.30). The input values for the nuclear velocities in all the plotted functions are a number of N points equally spaced in the interval [-0.02,0.02] Å/fs for the velocity components distribution, and in the interval [0.0,0.025] Å/fs for the Maxwell-Boltzmann distributions, while the input values for the masses m are all equal to the mass of Silicon atom.
The case of silicon supercell system is simple, since only one atomic mass population is involved, so that the distributions with probability density functions of the form (A.28) and (A.30) can be used directly, imposing the mass m to be equal to the Silicon mass. A more complicated case is a crystalline system with 12 water molecules, where two different atomic masses are considered: the hydrogen mass m_h and the oxygen mass m_o, where $m_h < m_o$. In this case, since two different masses are involved, the distributions (A.28) have to be divided in two Gaussian functions, one for the hydrogen subsystem and the other one for the oxygen subsystem, namely,

$$f(v_x) = f(v_{x,h}) + f(v_{x,o}) = \sqrt{\frac{m_h}{2\pi k_b T_{a,h}}}\; e^{-m_h v_{x,h}^2/(2k_bT_{a,h})} + \sqrt{\frac{m_o}{2\pi k_b T_{a,o}}}\; e^{-m_o v_{x,o}^2/(2k_bT_{a,o})} \tag{A.32}$$

$$f(v_y) = f(v_{y,h}) + f(v_{y,o}) = \sqrt{\frac{m_h}{2\pi k_b T_{a,h}}}\, e^{-m_h v_{y,h}^2/(2k_b T_{a,h})} + \sqrt{\frac{m_o}{2\pi k_b T_{a,o}}}\, e^{-m_o v_{y,o}^2/(2k_b T_{a,o})} \tag{A.33}$$

$$f(v_z) = f(v_{z,h}) + f(v_{z,o}) = \sqrt{\frac{m_h}{2\pi k_b T_{a,h}}}\, e^{-m_h v_{z,h}^2/(2k_b T_{a,h})} + \sqrt{\frac{m_o}{2\pi k_b T_{a,o}}}\, e^{-m_o v_{z,o}^2/(2k_b T_{a,o})} \tag{A.34}$$

where $(v_{x,h}, v_{y,h}, v_{z,h})$ and $(v_{x,o}, v_{y,o}, v_{z,o})$ are respectively the velocity components of the hydrogen and oxygen atoms, $T_{a,h}$ is the temperature related to the hydrogen subsystem, while $T_{a,o}$ is the temperature of the oxygen atoms subsystem. In the same way, the Maxwell-Boltzmann distribution with probability density function (A.30) becomes the sum of two Maxwell-Boltzmann distributions (i.e. the sum of two probability density functions with form given by (A.30)), with two different associated temperatures ($T_{a,h}$ and $T_{a,o}$) and two different atomic masses, that is

$$F(v) = 4\pi \left(\frac{m_h}{2\pi k_b T_{a,h}}\right)^{3/2} v_h^2\, e^{-m_h v_h^2/(2k_b T_{a,h})} + 4\pi \left(\frac{m_o}{2\pi k_b T_{a,o}}\right)^{3/2} v_o^2\, e^{-m_o v_o^2/(2k_b T_{a,o})} \tag{A.35}$$

where v_h and v_o are the modulus of the velocities for the hydrogen and the oxygen atoms, respectively,

$$v_h = \|\boldsymbol{v}_h\| = \sqrt{v_{x,h}^2 + v_{y,h}^2 + v_{z,h}^2} \qquad\qquad v_o = \|\boldsymbol{v}_o\| = \sqrt{v_{x,o}^2 + v_{y,o}^2 + v_{z,o}^2} \tag{A.36}$$

The generalization of the previous equations to a number n_s of atomic species in the system is straightforward. The Gaussian distributions associated to the three components v_j, $j = x, y, z$ of the nuclear velocities have, in the general case of n_s number of atomic species in the system, the following three probability density functions

$$f(v_j) = \sum_{\alpha=1}^{n_s} f(v_{j,\alpha}) = \sum_{\alpha=1}^{n_s} \sqrt{\frac{m_\alpha}{2\pi k_b T_{a,\alpha}}}\, e^{-m_\alpha v_{j,\alpha}^2/(2k_b T_{a,\alpha})} \qquad\qquad j = x, y, z \tag{A.37}$$

where the temperature $T_{a,\alpha}$ is the actual temperature related to the α-th atomic element subsystem.
In the same way, the Maxwell-Boltzmann distribution for the modulus of the nuclear velocities, in the general case of n_s number of atomic species in the system, is associated to the following probability density function

$$F(v) = 4\pi \sum_{\alpha=1}^{n_s} \left(\frac{m_\alpha}{2\pi k_b T_{a,\alpha}}\right)^{3/2} v_\alpha^2\, e^{-m_\alpha v_\alpha^2/(2k_b T_{a,\alpha})} \tag{A.38}$$

where v_α are the modulus of the velocities for all the atoms of the α-th species (element).
This separation of the probability density function on the base of the atomic masses values which form the system is well reproduced in Figure A.2, where the histograms for the three components of nuclear velocities and for the modulus of the nuclear velocities initialized for a $4 \times 4 \times 4$ supercell ($N = 2304$ atoms) of the Bernal-Folwer ice (768 oxygen atoms and 1536 hydrogen atoms) with an initial temperature equal to $T_a = 600$ K (upper panel) and $T_a = 1200$ K (lower panel) are reported. The blue and black curves in the first three panels of Figure A.2 are the functions obtained by plotting, respectively, the first and the second terms in (A.32)-(A.34). Since the mass of the oxygen is greater than the mass of the hydrogen atom ($m_o > m_h$), then the Gaussian distributions associated to the hydrogen atoms have a probability density function with a greater standard deviation (blue curves) with respect to the oxygen atoms (black curves), i.e. $\sigma_o < \sigma_h$. In the last panel of Figure A.2, the blue and black curves are, respectively, the first and the second term of the Maxwell-Boltzmann distribution in (A.35). The input values for the nuclear velocities in all the plotted functions are a number of N points equally spaced in the interval [-0.10,0.10] Å/fs for the velocity component distributions and in the interval [0.00,0.14] Å/fs for the Maxwell-Boltzmann distribution, and the input masses are the mass m_h of the hydrogen atom for the blue curves and the mass m_o of the oxygen atom for the black curves.
The effect of the temperature is visible both in Figure A.2 and Figure A.1, moving from the upper to the lower panel. Indeed, increasing the temperature, the standard deviations of the distributions increase, so that the probability density functions are broader for a larger temperature (lower panels). At the same time, increasing the temperature, the maximum value of the distributions decreases.

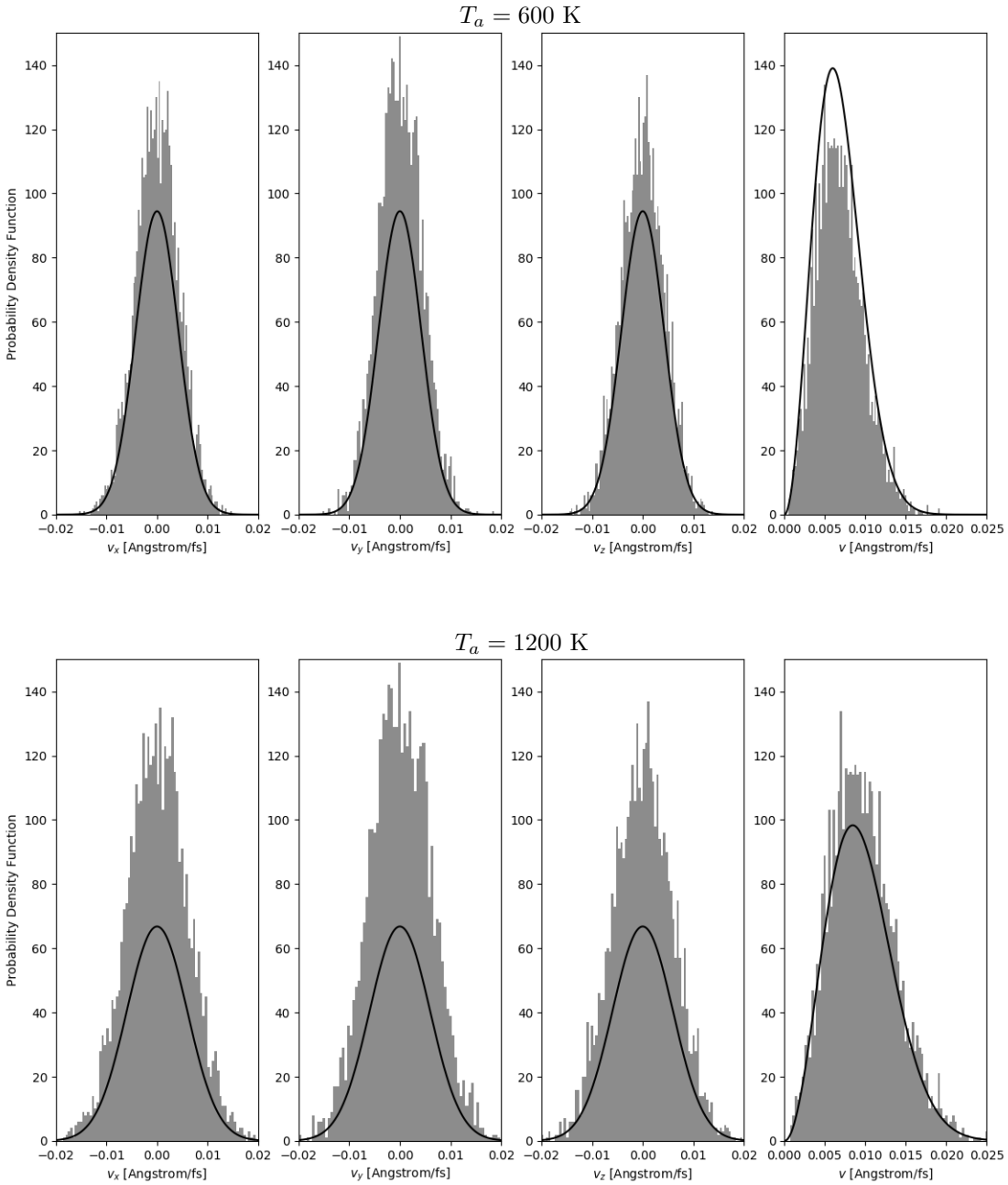

Figure A.1: System: $20 \times 10 \times 10$ silicon supercell ($N = 4000$ atoms), temperature $T_a = 600$ K (upper panel) and $T_a = 1200$ K (lower panel). In the first three panels from the left, histograms for the velocity components distributions $f(v_x)$, $f(v_y)$ and $f(v_z)$ are reported. Black curves in the first three left panels (upper and lower) are the functions (A.28) sampled at equal points in the interval [-0.02,0.02] Å/fs. In the last right panel (upper and lower), the distribution for the modulus of the nuclear velocities $F(v)$ is reported, together with the Maxwell-Boltzmann function (A.30) (black curve) sampled at equal points in the interval [0.0,0.025] Å/fs. Histograms are plotted using the object hist in python (matplotlib.pyplot library), with 90 number of bins and relative width of the bars as a fraction of the bin width equal to 1.0.

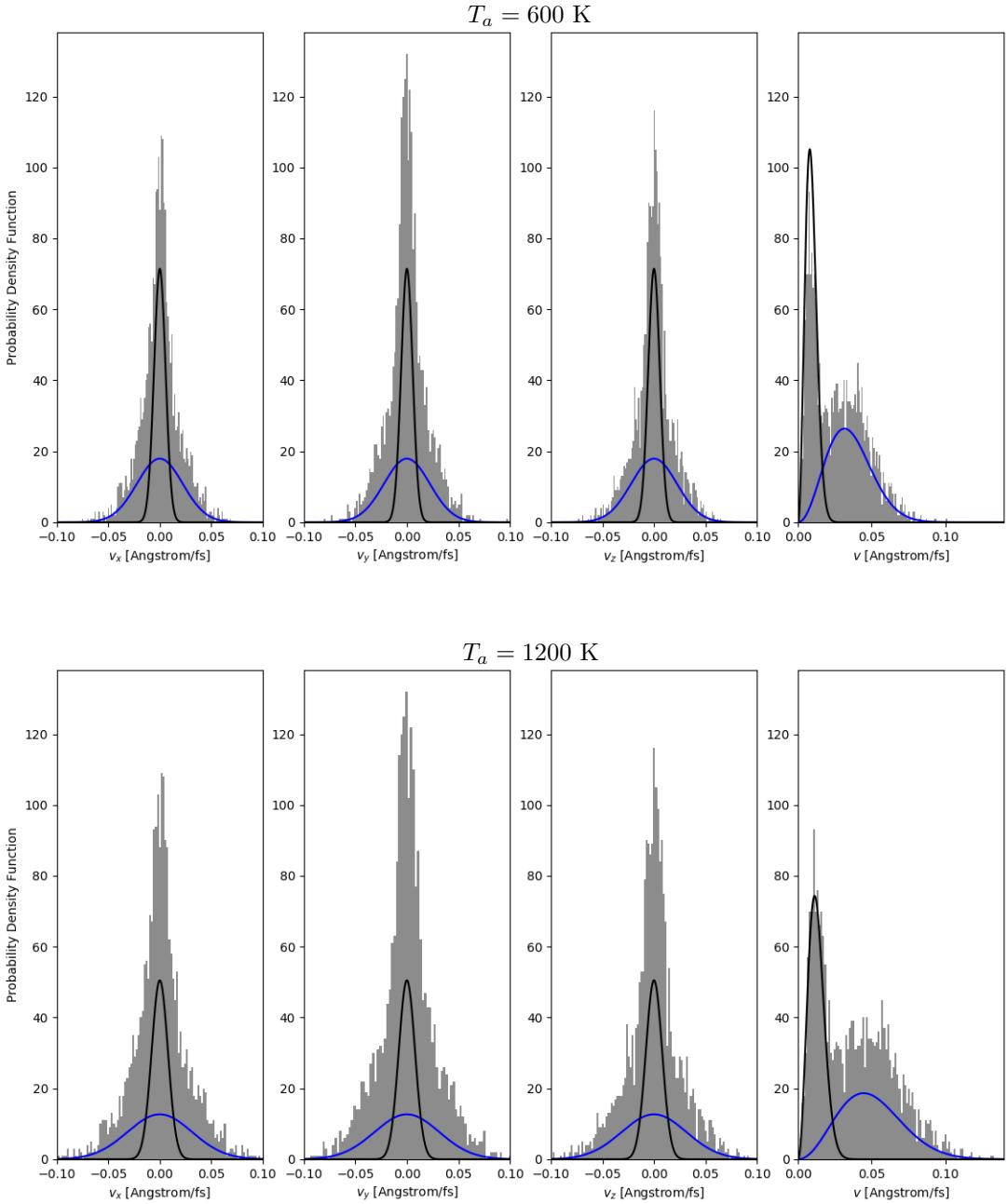

Figure A.2: System: $4 \times 4 \times 4$ supercell of Bernal-Folwer crystalline ice (12 water molecules crystalline system, $N = 2304$ atoms, 768 oxygen and 1536 hydrogen atoms), temperature $T_a = 600$ K (upper panel) and $T_a = 1200$ K (lower panel). In the first three panels from the left, histograms for the velocity components distributions $f(v_x)$, $f(v_y)$ and $f(v_z)$ are reported. Blue and black curves in the first three left panels (upper and lower) are respectively the first and second terms in functions (A.32)-(A.34), sampled at equal points in the interval [-0.10,0.10] Å/fs. In the last right panel (upper and lower), the distribution for the modulus of the nuclear velocities $F(v)$ is reported, together with the Maxwell-Boltzmann function (A.30) for the hydrogen atoms (blue curve) and the oxygen atoms (black curve) sampled at equal points in the interval [0.0,0.14] Å/fs. Histograms are plotted using the object hist in python (matplotlib.pyplot library), with 110 number of bins and relative width of the bars as a fraction of the bin width equal to 1.0.

A.1.2 Initial nuclear velocities rescaling

As explained in Chapter 5.5, Section 3.1.2, in order to match the initial target temperature T_0 required by the molecular dynamics simulation, the nuclear velocities $\{v_{ij}\}$, given by equation (3.24) and obtained through the Box-Muller algorithm, after the subtraction of the center of mass velocity, have to be rescaled. The calculation of the rescaling factor proceeds as described in Section 3.1.2. Here, it will be demonstrated that the multiplicative factor $\sqrt{k_b T_a}$ in equation (3.24) is a fictitious factor, that can be completely ignored because it does not change the values of the initial nuclear velocities, thanks to the rescaling procedure. To this end, it is defined a set of numbers $\{\tilde{u}_i\}$ simply given by the standard normal distributed random numbers $\{u_i\}$ normalized with respect to the square root of the atomic masses, so that

$$\tilde{u}_i = \frac{u_i}{\sqrt{m_i}} \qquad\qquad i = 1, ..., 3N \tag{A.39}$$

or, alternatively, changes the indexes as in equation (3.24), so that

$$\tilde{u}_{3(i-1)+j} = \frac{u_{3(i-1)+j}}{\sqrt{m_i}} \qquad i = 1, ..., N \quad \text{and} \quad j = 1, 2, 3 \tag{A.40}$$

In this way, two sets of initial velocities can be defined, which differs only for the multiplicative factor proportional to the actual temperature, in the following way,

$$v_{ij} = \tilde{u}_{3(i-1)+j} \qquad \text{and} \qquad v_{ij} = \tilde{u}_{3(i-1)+j}\sqrt{k_b T_a} \qquad (i = 1, ..., N \quad \text{and} \quad j = 1, 2, 3) \tag{A.41}$$

where T_a is the actual temperature given by the initial normal Gaussian distribution of random numbers $\{u_i\}$. In the following, it will be demonstrated that these two initialization ways for the nuclear velocities are completely equivalent, that is, by (i) subtracting the center of mass velocity and by (ii) rescaling with respect to the target temperature the initial velocities in (A.41) leads to the same set of numbers for the nuclear velocities.

Proof.

Consider the first set of velocities in (A.41). Following the procedure described in Sections 3.1.1 and 3.1.2, the calculations performed on the first set of nuclear velocities in (A.41) are reported here below.

$$\begin{aligned}
&\text{Box-Muller :} && u_{3(i-1)+j} && i = 1, ..., N && j = 1, 2, 3 \\
&\text{Initial velocities :} && v_{ij}(t_0) = u_{3(i-1)+j}\sqrt{\frac{1}{m_i}} = \tilde{u}_{3(i-1)+j} && i = 1, ..., N && j = 1, 2, 3 \\
&\text{COM velocity :} && v^c_{ij}(t_0) = v_{ij}(t_0) - \frac{1}{m}\sum_{i=1}^{N} m_i v_{ij}(t_0) = \tilde{u}_{3(i-1)+j} - \frac{1}{m}\sum_{i=1}^{N} m_i \tilde{u}_{3(i-1)+j} \equiv \tilde{v}^c_{ij}(t_0) \\
&\text{Kinetic energy :} && E_k(t_0) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i \left[\tilde{v}^c_{ij}(t_0)\right]^2 \\
&\text{Actual temperature :} && T(t_0) = \frac{2}{g k_b} E_k(t_0) = \frac{1}{g k_b}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i \left[\tilde{v}^c_{ij}(t_0)\right]^2 \equiv \tilde{T}(t_0) \\
&\text{Temperature scaling :} && v^{in}_{ij}(t_0) = \tilde{v}^c_{ij}(t_0)\sqrt{\frac{T_0}{\tilde{T}(t_0)}} && i = 1, ..., N && j = 1, 2, 3
\end{aligned}$$

Consider now the second set of velocities in (A.41). Following the procedure described in Sections 3.1.1 and 3.1.2, the calculations performed on the second set of nuclear velocities in (A.41) are reported here below, to be compared with the previous derivation.

$$\text{Box-Muller :} \qquad u_{3(i-1)+j} \qquad i = 1, ..., N \qquad j = 1, 2, 3$$

$$\text{Kinetic energy :} \qquad E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, u^2_{3(i-1)+j}$$

$$\text{Actual temperature :} \qquad T_a = \frac{2}{gk_b}\,E_k(t_0) = \frac{1}{gk_b}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, u^2_{3(i-1)+j}$$

$$\text{Initial velocities :} \qquad v_{ij}(t_0) = u_{3(i-1)+j}\sqrt{\frac{k_bT_a}{m_i}} = \sqrt{k_bT_a}\;\tilde{u}_{3(i-1)+j} \qquad i = 1, ..., N \qquad j = 1, 2, 3$$

$$\text{COM velocity :} \qquad v^c_{ij}(t_0) = v_{ij}(t_0) - \frac{1}{m}\sum_{i=1}^{N} m_i v_{ij}(t_0) = \sqrt{k_bT_a}\underbrace{\left[\tilde{u}_{3(i-1)+j} - \frac{1}{m}\sum_{i=1}^{N} m_i\tilde{u}_{3(i-1)+j}\right]}_{=\,\tilde{v}^c_{ij}(t_0)}$$

$$\text{Kinetic energy :} \qquad E_k(t_0) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\,[v^c_{ij}(t_0)]^2 = \frac{k_bT_a}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\,[\tilde{v}^c_{ij}(t_0)]^2$$

$$\text{Actual temperature :} \qquad T(t_0) = \frac{2}{gk_b}\,E_k(t_0) = \frac{T_a}{g}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\,[\tilde{v}^c_{ij}(t_0)]^2 = k_bT_a\;\tilde{T}(t_0)$$

$$\text{Temperature scaling :} \qquad v^{in}_{ij}(t_0) = v^c_{ij}(t_0)\sqrt{\frac{T_0}{T(t_0)}} = \sqrt{k_bT_a}\;\tilde{v}^c_{ij}(t_0)\sqrt{\frac{T_0}{k_bT_a\;\tilde{T}(t_0)}} = \tilde{v}^c_{ij}(t_0)\sqrt{\frac{T_0}{\tilde{T}(t_0)}}$$

c.v.d.

where $\tilde{v}^c_{ij}(t_0)$ and $\tilde{T}(t_0)$ are same the nuclear velocities scaled with respect to the center of mass velocity and the same actual temperature at time t_0 defined for the first set of initial velocities in (A.41) and introduced in the first block of equations written in this proof. Note that in this last block of equations, two additional calculations are performed with respect to the first block, that are required to compute the actual temperature T_a (associated to the normal Gaussian distributed random numbers $\{u_{3(i-1)+j}\}$) that is used for the definition of the second set of nuclear velocities $v_{ij}(t_0)$ in (A.41).
Finally, it can be stated that using the two sets of nuclear velocities (A.41) is equivalent, because they both lead to the same expression for the initial nuclear velocities $v^{in}_{ij}(t_0)$ to be used as initial condition in a molecular dynamics simulation.

A.2 Position and Velocity Verlet algorithms

Both position and velocity Verlet algorithms are finite-difference methods, that have been introduced with the aim of solving second order differential equations.
Consider the case of a two-dimensional system described by the coordinates (x, y). Second order differential equations of the form

$$\ddot{y} = f(x, y) \tag{A.42}$$

with initial conditions

$$y(a) = \alpha \qquad \dot{y}(a) = \gamma \tag{A.43}$$

are very often encountered in mathematical physics. The equation (A.42) can be reduced to a set of two simultaneous first-order differential equations using special methods. One can simply replace the derivatives in the differential equation and initial conditions by symmetric difference approximations

$$\dot{y}(x_n) \approx \frac{y_{n+1} - y_{n-1}}{2h} \tag{A.44}$$

$$\ddot{y}(x_n) \approx \frac{1}{h}(\dot{y}_{n+1/2} - \dot{y}_{n-1/2}) = \frac{1}{h}\left(\frac{y_{n+1} - y_n}{h} - \frac{y_n - y_{n-1}}{h}\right) = \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} \tag{A.45}$$

If the notation $f_n = f(x_n, y_n)$ is used, a finite-difference solution for the second order differential equation (A.42) can be written as

$$y_{n+1} - 2y_n + y_{n-1} = h^2\, \ddot{y}(x_n) = h^2\, f_n \tag{A.46}$$

$$y_0 = \alpha \qquad y_1 - y_{-1} = 2h\gamma \tag{A.47}$$

where y_{-1} can be eliminated by means of equation (A.46) with $n = 0$, that is by means of the equation

$$y_1 - 2y_0 + y_{-1} = h^2\, f_0 \quad \rightarrow \quad y_{-1} = h^2\, f_0 - y_1 + 2y_0 \tag{A.48}$$

so that

$$y_1 - y_{-1} = 2h\gamma \quad \rightarrow \quad y_1 - (h^2\, f_0 - y_1 + 2y_0) = 2h\gamma \quad \rightarrow \quad 2y_1 - 2y_0 = 2h\gamma + h^2\, f_0 \tag{A.49}$$

The method that leads to equation (A.46) is called *explicit central difference method*, and it is the simplest member of the Störmer family of methods. The application of this method for solving the Newton's equations of motion of the form $\ddot{q} = f(q)$ for the motion of the nuclei in a condensed matter system it is called position Verlet algorithm, since it has been introduced by L. Verlet[40] in 1967.[1] The starting procedure for this method is therefore

$$y_0 = \alpha \qquad y_1 - y_0 = h\gamma + \frac{1}{2}h^2\, f_0 \tag{A.50}$$

and then y_2, y_3, y_4, etc. can be computed successively by means of equation (A.46), i.e.

$$y_{n+1} = 2y_n - y_{n-1} + h^2\, f_n \tag{A.51}$$

Note that at each step there is an addition of the form $\mathcal{O}(1) + \mathcal{O}(h^2)$, this gives unfavorable rounding errors when h is small. Furthermore, the algorithm defined by equation (A.51) is a two-step method, since y_{n+1} is computed using y_n and y_{n-1}.
One should instead use another finite-difference method, called the *summed form*, which consists in introducing the quantity

$$z_n = \frac{y_{n+1} - y_n}{h} \tag{A.52}$$

[1] A curious fact is that Professor Loup Verlet, who later became interested in the history of science, discovered precisely his method in several places in the classical literature, for example, in the calculations of logarithms and astronomical tables by J. B. Delambre in 1792. Even more spectacular is the finding that the Verlet method was used in Newton's Principia from 1687 to prove Kepler's second law. An especially clear account can be found in Feynman's Messenger Lecture from 1964.

so that, using equation (A.46), the following expression can be derived

$$z_n - z_{n-1} = \frac{y_{n+1} - y_n}{h} - \frac{y_n - y_{n-1}}{h} = \frac{y_{n+1} - 2y_n + y_{n-1}}{h} = h\, f_n \tag{A.53}$$

Then the summed form is defined by the formulae

$$y_{n+1} = y_n + h z_n \tag{A.54}$$

$$z_n = z_{n-1} + h\, f_n \tag{A.55}$$

$$y_0 = \alpha \qquad z_0 = \gamma + \frac{1}{2} h\, f_0 \tag{A.56}$$

This summed form is mathematically equivalent, but not numerically equivalent to the explicit central difference method, since the summed form is superior on a computer finite precision computation.[115] The application of the summed form method for solving the Newton's equations of motion $\ddot{q} = f(q)$ for the motion of the nuclei in a condensed matter system has been introduced by W. Swope *et al.*[39] in 1982, and it is called velocity Verlet algorithm, since it is a modification of the position Verlet algorithm in which the nuclear velocity $v_n = \dot{q}_n$ appears directly in the equations to be iterated.

The following example clarify the reason for the superiority of the summed form on the explicit central difference method as regards the numerical precision.
Take the differential equation

$$\ddot{y} = k \qquad y(0) = 0.1 \qquad \dot{y}(0) = 0$$

where $k = 0.654321$, $h = 0.01$, $n \leq 100$. Then one can easily show that $0.1 \leq y_n < 1$. Suppose that six-digit floating decimal arithmetic is used. For the explicit central difference method each operation will be of the form:

$$\begin{aligned} 2y_n - y_{n-1} &= 0.\text{xxxxxx} \\ + \qquad h^2\, f_n &= 0.000065\,4321 \\ \hline \\ y_{n+1} &= 0.\text{yyyyyy} \end{aligned}$$

The last four digits have no influence. In fact, the equation $\ddot{y} = 0.65$ is treated instead of $\ddot{y} = 0.654321$. For the summed form, the inequality $0.1 < z_n < 1$ holds for $n > 15$. Hence most operations will be of the form
In the calculation of z the last two digits of f are lost. This means roughly that for $x > 0.15$ the equation $\ddot{y} = 0.6543$ is treated. The resultant effect is roughly that the last two digits of z become unreliable, hence the shift of z in the addition for y is of minor importance. A clue to an explanation of the difference between the two algorithms is the role of z as a carrier of information from one step to the next. There is still a loss of accuracy in the summed form, but one has the same kind of loss, for instance, in numerical quadrature or with other methods for solving differential equations. One could store z and y in double precision, while single precision is used in the computation of f, which is usually the most time consuming part of the work. If such partial double precision is used, then the advantage of the summed form is reduced.
In order to recover the explicit expression of the position and velocity Verlet algorithms, consider the motion of one particle in a one dimensional space, described by the position q, with $f(q)$ is the force acting on the particle divided by its mass. The second order differential equation describing the classical motion of the particle is a Newtonian equation of motion given by

$$\ddot{q} = f(q) \tag{A.57}$$

where the right-hand side $f(q)$ does not depend on $\dot{q}$. Equation (A.57) is usually given together with some initial conditions, assigning a value to the particle position q_0 and to its velocity $\dot{q}_0 = v_0$ at the initial time t_0. Using the explicit central difference method, the second order differential equation (A.57) can be resolved by an iterative procedure

$$q_{n+1} = 2q_n - q_{n-1} + h^2\, f_n \tag{A.58}$$

which results in the so-called position Verlet algorithm, that is a two-step method where q_{n+1} is determined whenever q_{n-1} and q_n are known (geometrically, this amounts to determining an interpolating parabola which, in the mid-point, assumes the second derivative prescribed by equation (A.57), see Figure A.3, left panel). This formulation is not very interesting. The second order ordinary differential equation (A.57) can be transformed into a first order system of differential equations of dimension two, by introducing the unknown function $v = \dot{q}$, so that the system (A.57) becomes

$$\dot{q} = v \qquad \dot{v} = f(q) \tag{A.59}$$

that represents an equation in the so-called phase space. In physics often this type of system is closely related to the motion of a physical body. In this sense, the variable v can be seen as the velocity, and the variable q as the displacement (as functions of the time) and $f(q)$ describes some kind of force acting on the physical body. The discrete approximation of the velocity v, indicated as v_n, can be defined as a function of the quantity z_n (where z_n has been defined in equation (A.52)) using equation (A.44) as

$$v_n = \dot{q}_n = \frac{q_{n+1} - q_{n-1}}{2h} = \frac{z_n + z_{n-1}}{2} \overset{\text{eq. (A.55)}}{=} \frac{2z_n - h\, f_n}{2} = z_n - \frac{h}{2} f_n \tag{A.60}$$

so that, on the contrary, the quantity z_n can be written as a function of the velocity v_n as

$$z_n = v_n + \frac{h}{2} f_n \tag{A.61}$$

Applying equation (A.54), the solution of the second order differential equation (A.57) using the summed form method can be written as

$$q_{n+1} = q_n + h z_n \overset{\text{eq. (A.61)}}{=} q_n + h v_n + \frac{h^2}{2} f_n \tag{A.62}$$

The explicit expression for the velocity v_{n+1} without the dependence on the z_{n+1} variable can be obtained considering the two equations (written starting from (A.60))

$$v_n = z_n - \frac{h}{2} f_n \tag{A.63}$$

$$v_{n+1} = z_{n+1} - \frac{h}{2} f_{n+1} \tag{A.64}$$

and subtracting the first one to the second one

$$v_{n+1} - v_n = z_{n+1} - z_n - \frac{h}{2} f_{n+1} + \frac{h}{2} f_n \tag{A.65}$$

The difference $(z_{n+1} - z_n)$ can be expressed as a function of f_n and f_{n+1} using equation (A.55) with index substitution $n \to (n+1)$, so that

$$z_{n+1} = z_n + h\, f_{n+1} \quad \to \quad z_{n+1} - z_n = h\, f_{n+1} \tag{A.66}$$

Finally, the substitution of the difference (A.66) in (A.65) leads to the equation

$$v_{n+1} - v_n = h\, f_{n+1} - \frac{h}{2} f_{n+1} + \frac{h}{2} f_n = \frac{h}{2} (f_{n+1} + f_n) \tag{A.67}$$

and the velocity at step $n+1$ can be recovered from the velocity at the step n and the forces at n and $n+1$ with the formula

$$v_{n+1} = v_n + \frac{h}{2} (f_{n+1} + f_n) \tag{A.68}$$

Thus, it has been demonstrated that the solution of the second order differential equation (A.57) using the summed form method leads to the so-called velocity Verlet algorithm, a one-step method (see Figure A.3, right panel) that can be resumed by the following two equations

$$q_{n+1} = q_n + h v_n + \frac{h^2}{2} f_n \tag{A.69}$$

$$v_{n+1} = v_n + \frac{h}{2} (f_{n+1} + f_n) \tag{A.70}$$

These equations permit to evolve the equation of motion (A.57) in a discrete way, with a time step represented by h, computing at each n-th step the position q_{n+1} and the velocity v_{n+1} of the particle for the next $(n+1)$-th step, using the forces f_n and f_{n+1} acting on the same particle, respectively at the n-th and at the $(n+1)$-th step.

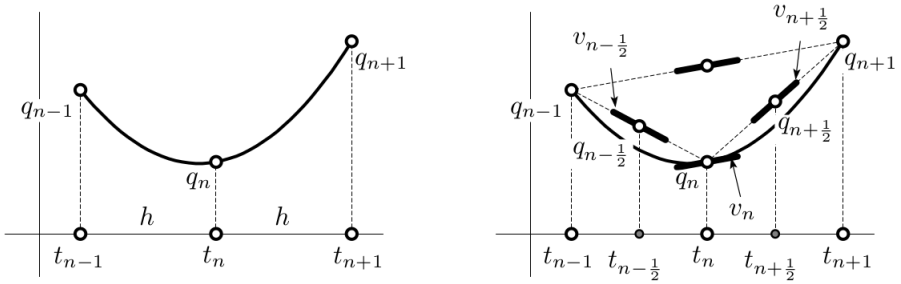

Figure A.3: Two-step formulation (left) and one-step formulation (right) for the solution of the second order differential equation (A.57), which gives rise to the so-called position and velocity Verlet algorithm, respectively.

A.3 Phase space notation

A point in phase space $\mathbb{R}^{2Nd}$, where Nd is the number of degrees of freedom, is given by the column vector

$$\boldsymbol{x}(t) \equiv {}^t(q_1(t), ..., q_{Nd}(t), p_1(t), ..., p_{Nd}(t)) \tag{A.71}$$

where the superscript t above indicates the transpose function. Using the Hamilton equations of motion, the time derivative of (A.71) can be obtained using the equation

$$\dot{\boldsymbol{x}} = \mathbf{M}\,\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} = \begin{pmatrix} \mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0} \end{pmatrix} \frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} \tag{A.72}$$

where $\dot{\boldsymbol{x}}$ and $(\partial\mathcal{H}/\partial\boldsymbol{x})$ are $2Nd\times 1$ column matrices, while $\mathbf{M}$ is a $2Nd\times 2Nd$ skew-symmetric, orthogonal block matrix, with $\mathbb{0}$ and $\mathbb{1}$ the $d\times d$ zero and identity matrices, respectively. Using equation (A.72), the evolution of time of a the point $\boldsymbol{x}$ in the phase space can be expressed as

$${}^t\dot{\boldsymbol{x}}(t) = \left(\frac{\partial \mathcal{H}}{\partial p_1}, ..., \frac{\partial \mathcal{H}}{\partial p_{Nd}}, -\frac{\partial \mathcal{H}}{\partial q_1}, ..., -\frac{\partial \mathcal{H}}{\partial q_{Nd}}\right) \tag{A.73}$$

The number of degrees of freedom Nd depends on both the number of particle involved in the system and the dimension of the space in which the particles move. For example, for $N = 2$ particles in $d = 3$ dimensions, the number of degrees of freedom are $Nd = 3N = 6$ and the matrix equation (A.72) on the phase space $\mathbb{R}^8$ can be written explicitly as follows

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \\ \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \\ \dot{p}_4 \\ \dot{p}_5 \\ \dot{p}_6 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \partial\mathcal{H}/\partial q_1 \\ \partial\mathcal{H}/\partial q_2 \\ \partial\mathcal{H}/\partial q_3 \\ \partial\mathcal{H}/\partial q_4 \\ \partial\mathcal{H}/\partial q_5 \\ \partial\mathcal{H}/\partial q_6 \\ \partial\mathcal{H}/\partial p_1 \\ \partial\mathcal{H}/\partial p_2 \\ \partial\mathcal{H}/\partial p_3 \\ \partial\mathcal{H}/\partial p_4 \\ \partial\mathcal{H}/\partial p_5 \\ \partial\mathcal{H}/\partial p_6 \end{pmatrix} \tag{A.74}$$

so that the Hamilton equations of motion can be recovered

$$\dot{q}_\alpha = \frac{\partial \mathcal{H}}{\partial p_\alpha} \qquad \dot{p}_\alpha = -\frac{\partial \mathcal{H}}{\partial q_\alpha} \qquad \alpha = 1, ..., 6 \tag{A.75}$$

However, it is convenient to collect the coordinates in vectors on the base of particle indexes, i.e. rewriting the point $\boldsymbol{x}(t)$ in phase space as

$$\tilde{\boldsymbol{x}}(t) \equiv {}^t(\boldsymbol{q}_1(t), ..., \boldsymbol{q}_N(t), \boldsymbol{p}_1(t), ..., \boldsymbol{p}_N(t)) \tag{A.76}$$

where

$$\begin{aligned}\boldsymbol{q}_i(t) &= (q_{1+d(i-1)}(t), ..., q_{d+d(i-1)}(t)) \\ \boldsymbol{p}_i(t) &= (p_{1+d(i-1)}(t), ..., p_{d+d(i-1)}(t))\end{aligned} \qquad i = 1, ..., N \quad \text{and} \quad d = \text{space dimension} \tag{A.77}$$

Using this new simplified notation, equation (A.74) can be rewritten as follows

$$\begin{pmatrix}\dot{\boldsymbol{q}}_1 \\ \dot{\boldsymbol{q}}_2 \\ \dot{\boldsymbol{p}}_1 \\ \dot{\boldsymbol{p}}_2\end{pmatrix} = \begin{pmatrix}0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0\end{pmatrix} \begin{pmatrix}\partial\mathcal{H}/\partial\boldsymbol{q}_1 \\ \partial\mathcal{H}/\partial\boldsymbol{q}_2 \\ \partial\mathcal{H}/\partial\boldsymbol{p}_1 \\ \partial\mathcal{H}/\partial\boldsymbol{p}_2\end{pmatrix} \tag{A.78}$$

with $\boldsymbol{q}_1 = (q_1, q_2, q_3)$, $\boldsymbol{q}_2 = (q_4, q_5, q_6)$, $\boldsymbol{p}_1 = (p_1, p_2, p_3)$ and $\boldsymbol{p}_2 = (p_4, p_5, p_6)$. In this way, the Hamilton equations of motion can be recovered

$$\dot{\boldsymbol{q}}_i = \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_i} \qquad \dot{\boldsymbol{p}}_i = -\frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_i} \qquad i = 1, ..., N \tag{A.79}$$

so that equation (A.72) becomes

$$\dot{\tilde{\boldsymbol{x}}}(t) = \tilde{\mathbf{M}}\,\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}} = \begin{pmatrix}\mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0}\end{pmatrix}\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}} \tag{A.80}$$

where the point $\tilde{\boldsymbol{x}}$ in phase space is expressed by (A.76) and $\tilde{\mathbf{M}}$ is a $2N \times 2N$ skew-symmetric, orthogonal block matrix, with $\mathbb{0}$ and $\mathbb{1}$ the $N \times N$ zero and identity matrices, respectively. Moreover, if the expression for a point in phase space would be written as a row vector instead of a column vector, that is

$$\bar{\boldsymbol{x}}(t) \equiv (\boldsymbol{q}_1(t), ..., \boldsymbol{q}_N(t), \boldsymbol{p}_1(t), ..., \boldsymbol{p}_N(t)) \tag{A.81}$$

then the transpose of equation (A.80) has to be taken, leading to

$$\bar{\boldsymbol{x}}(t) = {}^t\dot{\tilde{\boldsymbol{x}}}(t) = {}^t\left(\tilde{\mathbf{M}}\,\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right) = {}^t\left[\begin{pmatrix}\mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0}\end{pmatrix}\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right] = {}^t\left(\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right)\begin{pmatrix}\mathbb{0} & -\mathbb{1} \\ \mathbb{1} & \mathbb{0}\end{pmatrix} \equiv \frac{\partial\mathcal{H}}{\partial\bar{\boldsymbol{x}}}\,\bar{\mathbf{M}} \tag{A.82}$$

$$\text{where} \qquad \frac{\partial\mathcal{H}}{\partial\bar{\boldsymbol{x}}} = \left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_1}, ..., \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_N}, \frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_1}, ..., \frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_N}\right) \qquad \text{and} \qquad \bar{\mathbf{M}} = \begin{pmatrix}\mathbb{0} & -\mathbb{1} \\ \mathbb{1} & \mathbb{0}\end{pmatrix} \tag{A.83}$$

2

[2]Note that if in the vector $\tilde{\boldsymbol{x}}(t)$ given by (A.76) the two sets of generalized coordinates invert their order, that is, if the general expression of a phase space point is given by

$$\tilde{\boldsymbol{x}}(t) \equiv {}^t(\boldsymbol{p}_1(t), ..., \boldsymbol{p}_N(t), \boldsymbol{q}_1(t), ..., \boldsymbol{q}_N(t)) \tag{A.84}$$

then the derivative of the first coordinates (momenta) have a minus sign and the derivative of the last coordinates (positions) have a plus sign (according to Hamilton equations of motion), so that the transpose of matrix $\tilde{\mathbf{M}}$ in equation (A.80) has to be considered. Therefore, in this case equation (A.80) becomes

$$\dot{\tilde{\boldsymbol{x}}}(t) = \tilde{\mathbf{M}}\,\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}} = \begin{pmatrix}\mathbb{0} & -\mathbb{1} \\ \mathbb{1} & \mathbb{0}\end{pmatrix}\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}} \tag{A.85}$$

Furthermore, if the general expression of a phase space point is given by the row vector

$$\bar{\boldsymbol{x}}(t) \equiv (\boldsymbol{p}_1(t), ..., \boldsymbol{p}_N(t), \boldsymbol{q}_1(t), ..., \boldsymbol{q}_N(t)) \tag{A.86}$$

that corresponds to the phase space point with expression (A.81) but with the two sets of generalized coordinates inverted in the order, then to obtain the correct formula for the Hamilton equation, the transpose of the matrix expression (A.85) has to be taken, leading to

$$\bar{\boldsymbol{x}}(t) = {}^t\dot{\tilde{\boldsymbol{x}}}(t) = {}^t\left(\tilde{\mathbf{M}}\,\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right) = {}^t\left[\begin{pmatrix}\mathbb{0} & -\mathbb{1} \\ \mathbb{1} & \mathbb{0}\end{pmatrix}\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right] = {}^t\left(\frac{\partial\mathcal{H}}{\partial\tilde{\boldsymbol{x}}}\right)\begin{pmatrix}\mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0}\end{pmatrix} \equiv \frac{\partial\mathcal{H}}{\partial\bar{\boldsymbol{x}}}\,\bar{\mathbf{M}} \tag{A.87}$$

$$\text{where} \qquad \frac{\partial\mathcal{H}}{\partial\bar{\boldsymbol{x}}} = \left(\frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_1}, ..., \frac{\partial\mathcal{H}}{\partial\boldsymbol{p}_N}, \frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_1}, ..., \frac{\partial\mathcal{H}}{\partial\boldsymbol{q}_N}\right) \qquad \text{and} \qquad \bar{\mathbf{M}} = \begin{pmatrix}\mathbb{0} & \mathbb{1} \\ -\mathbb{1} & \mathbb{0}\end{pmatrix} \tag{A.88}$$

A.4 Fluctuation-Dissipation Theorem

A reasonable physical assumption about the intensity of the random force R_α acting on the α-th nucleus in a d dimensional system of N atoms can be made on the basis of the fluctuation-dissipation theorem, that gives the relationship between a fluctuating force on some degree of freedom and the damping coefficient that determines dissipation in this degree of freedom:

$$\langle R_\alpha(t) R_\beta(t+\tau)\rangle = 2 m_\alpha \gamma_\alpha \, k_b \, T_0 \, \delta(\tau) \delta_{\alpha\beta} \tag{A.89}$$

where T_0 is the equilibrium temperature of the system in contact with a thermal bath. This theorem can be proven by utilizing the Langevin equation

$$m_\alpha \dot{v}_\alpha(t) = F_\alpha(t) - m_\alpha \gamma_\alpha v_\alpha(t) + R_\alpha(t) \qquad\qquad \alpha = 1, ..., Nd \tag{A.90}$$

and omitting the interatomic force F_α as not participating in the dissipation process, so that the previous equation becomes

$$m_\alpha \dot{v}_\alpha(t) = -m_\alpha \gamma_\alpha v_\alpha(t) + R_\alpha(t) \qquad\qquad \alpha = 1, ..., Nd \tag{A.91}$$

The general solution of this equation reads

$$v_\alpha(t) = v_\alpha(0) e^{-\gamma_\alpha t} + \frac{1}{m_\alpha} \int_0^t R_\alpha(t') e^{-\gamma_\alpha (t-t')} \, dt' \qquad\qquad \alpha = 1, ..., Nd \tag{A.92}$$

For a system in equilibrium conditions, the mean square value of this function is equal to

$$\langle v_\alpha^2(t)\rangle = \frac{k_b T_0}{m_\alpha} \tag{A.93}$$

According to the ergodic hypothesis, the time-averaged quantities for a system in thermodynamic equilibrium are equal to the corresponding ensemble average quantities, so that the ensemble mean value of the velocity v_α (A.93) can be also interpreted as the time-averaged nuclear velocity. Substituting equation (A.92) into equation (A.93) where the ensemble average is utilized, yields

$$\begin{aligned} \langle v_\alpha^2(t)\rangle = \langle v_\alpha^2(0)\rangle \, e^{-2\gamma_\alpha t} &+ \frac{2}{m_\alpha} \int_0^t \langle v_\alpha(0) R_\alpha(t')\rangle \, e^{-\gamma_\alpha (2t-t')} \, dt' \\ &+ \frac{1}{m_\alpha m_\beta} \int_0^t \int_0^t \langle R_\alpha(t') R_\beta(t'')\rangle \, e^{-\gamma_\alpha (t-t')} \, e^{-\gamma_\beta (t-t'')} \, dt' \, dt'' \end{aligned} \qquad \alpha = 1, ..., Nd \tag{A.94}$$

In the previous expression

$$\langle v_\alpha^2(t)\rangle = \langle v_\alpha^2(0)\rangle = k_b T_0 / m_\alpha \tag{A.95}$$

$$\langle v_\alpha(0) R_\alpha(t')\rangle = 0 \tag{A.96}$$

$$\langle R_\alpha(t') R_\beta(t'')\rangle = a \, \delta(t'-t'') \delta_{\alpha\beta} \tag{A.97}$$

Therefore, the relationship (A.94) can be reduced to

$$\begin{aligned} \frac{k_b T_0}{m_\alpha}(1 - e^{-2\gamma_\alpha t}) &= \frac{1}{m_\alpha m_\beta} \int_0^t \int_0^t a \, \delta(t'-t'') \delta_{\alpha\beta} \, e^{-\gamma_\alpha (t-t')} \, e^{-\gamma_\beta (t-t'')} \, dt' \, dt'' \\ \frac{k_b T_0}{m_\alpha}(1 - e^{-2\gamma_\alpha t}) &= \frac{1}{m_\alpha^2} \int_0^t a \, e^{-2\gamma_\alpha (t-t')} \, dt' = \frac{a}{m_\alpha^2} \, e^{-2\gamma_\alpha t} \int_0^t e^{2\gamma_\alpha t'} \, dt' \\ \frac{k_b T_0}{m_\alpha}(1 - e^{-2\gamma_\alpha t}) &= \frac{a}{m_\alpha^2} \frac{1}{2\gamma_\alpha} \, e^{-2\gamma_\alpha t}(e^{2\gamma_\alpha t} - 1) = \frac{a}{m_\alpha^2} \frac{1}{2\gamma_\alpha}(1 - e^{-2\gamma_\alpha t}) \end{aligned} \tag{A.98}$$

so that the constant a takes the value

$$a = 2 m_\alpha \gamma_\alpha \, k_b \, T_0 \tag{A.99}$$

Finally, inserting this value for the constant a in equation (A.97), the fluctuation-dissipation theorem (A.89) is recovered in the form

$$\langle R_\alpha(t') R_\beta(t'')\rangle = 2 m_\alpha \gamma_\alpha \, k_b \, T_0 \, \delta(t'-t'') \delta_{\alpha\beta} \tag{A.100}$$

As follows from previous equations, dynamic properties of the random force R_α in Langevin equation (A.91) can be summarized as

$$\langle R_\alpha(t)\rangle = 0 \qquad \langle R_\alpha(t)R_\beta(t')\rangle = 2m_\alpha\gamma_\alpha\, k_b\, T_0\, \delta(t-t')\delta_{\alpha\beta} \tag{A.101}$$

where T_0 is the target system temperature. These relationships can be physically interpreted as follows: (i) the function $R_\alpha(t)$ has no directional preference, (ii) the function $R_\alpha(t)$ is a Gaussian random function of time with zero mean and variance $2m_\alpha\gamma_\alpha\, k_b\, T_0$ for all degrees of freedom interacting with the heat bath; (iii) the force $R_\alpha(t)$ acting on degree of freedom α is uncorrelated with the force $R_\beta(t)$ which acts on another degree of freedom β; (iv) the instantaneous value of $R_\alpha(t)$ is not affected by its preceding values, i.e., the function $R_\alpha(t)$ is uncorrelated with its time history.
In practice, the random force $R_\alpha(t)$ can be sampled at each time step of a numerical simulation as a random Gaussian variable with zero mean and variance (mean square amplitude) $2m_\alpha\gamma_\alpha\, k_b\, T_0$. The sampling procedure is performed independently for each degree of freedom exposed to the thermal noise. Furthermore, samples for two successive time steps are evaluated independent of each other.

A.5 Canonical transformation

In the Hamiltonian formulation the generalized coordinates and momenta are independent variables. Therefore, it is possible to introduce a transformation of both variables simultaneously. For example, the transformation of the coordinates $(\boldsymbol{q},\boldsymbol{p})$ to $(\boldsymbol{Q},\boldsymbol{P})$ is denoted by

$$\begin{aligned}\boldsymbol{Q} &= \boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})\\ \boldsymbol{P} &= \boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})\end{aligned} \tag{A.102}$$

and the inverse transformation, $(\boldsymbol{Q},\boldsymbol{P})$ into $(\boldsymbol{q},\boldsymbol{p})$, is given by

$$\begin{aligned}\boldsymbol{q} &= \boldsymbol{q}(\boldsymbol{Q},\boldsymbol{P})\\ \boldsymbol{p} &= \boldsymbol{p}(\boldsymbol{Q},\boldsymbol{P})\end{aligned} \tag{A.103}$$

Obviously, the value of any function of the phase space coordinates is unaffected by the coordinate transformation. In the case of the Hamiltonian, this implies that

$$\mathcal{H}(\boldsymbol{p},\boldsymbol{q}) \equiv \mathcal{H}[\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p}),\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})] \equiv \mathcal{H}'(\boldsymbol{P},\boldsymbol{Q}) \tag{A.104}$$

In general, the equations of motion in the new coordinates are not of the canonical form, unless the coordinate transformation is canonical (since time does not appear explicitly in these equations, they are defined restricted canonical transformation). If the coordinate transformation is canonical, the equations of motion for the new phase space coordinates $(\boldsymbol{Q},\boldsymbol{P})$ are

$$\dot{\boldsymbol{Q}} = \left(\frac{\partial\mathcal{H}'(\boldsymbol{P},\boldsymbol{Q})}{\partial\boldsymbol{P}}\right) \tag{A.105}$$

$$\dot{\boldsymbol{P}} = -\left(\frac{\partial\mathcal{H}'(\boldsymbol{P},\boldsymbol{Q})}{\partial\boldsymbol{Q}}\right) \tag{A.106}$$

From equation (A.102) and the Hamilton equations of motion for the coordinates $(\boldsymbol{q},\boldsymbol{p})$ it follows that

$$\dot{\boldsymbol{Q}} = \left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right)\dot{\boldsymbol{p}} = \left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right)\left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{p}}\right) - \left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right)\left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{q}}\right)$$

Equation (A.104) allows to write

$$\left(\frac{\partial\mathcal{H}'(\boldsymbol{P},\boldsymbol{Q})}{\partial\boldsymbol{P}}\right) = \left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{p}}\right)\left(\frac{\partial\boldsymbol{p}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{P}}\right) + \left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{q}}\right)\left(\frac{\partial\boldsymbol{q}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{P}}\right) \tag{A.107}$$

This equation can only be equal to expression (A.105) for $\dot{\boldsymbol{Q}}$ if

$$\left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right) = \left(\frac{\partial\boldsymbol{p}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{P}}\right) \qquad \text{and} \qquad \left(\frac{\partial\boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right) = -\left(\frac{\partial\boldsymbol{q}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{P}}\right) \tag{A.108}$$

Similarly, from equation (A.102) and the Hamilton equations of motion for the coordinates $(\boldsymbol{q},\boldsymbol{p})$ it follows that

$$\dot{\boldsymbol{P}} = \left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right)\dot{\boldsymbol{p}} = \left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right)\left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{p}}\right) - \left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right)\left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{q}}\right)$$

Equation (A.104) allows to write

$$\left(\frac{\partial\mathcal{H}'(\boldsymbol{P},\boldsymbol{Q})}{\partial\boldsymbol{Q}}\right) = \left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{p}}\right)\left(\frac{\partial\boldsymbol{p}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{Q}}\right) + \left(\frac{\partial\mathcal{H}(\boldsymbol{p},\boldsymbol{q})}{\partial\boldsymbol{q}}\right)\left(\frac{\partial\boldsymbol{q}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{Q}}\right) \tag{A.109}$$

This equation can only be equal to expression (A.106) for $\dot{\boldsymbol{P}}$ if

$$\left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{q}}\right) = -\left(\frac{\partial\boldsymbol{p}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{Q}}\right) \qquad \text{and} \qquad \left(\frac{\partial\boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})}{\partial\boldsymbol{p}}\right) = \left(\frac{\partial\boldsymbol{q}(\boldsymbol{Q},\boldsymbol{P})}{\partial\boldsymbol{Q}}\right) \tag{A.110}$$

A.6 Resolution of the Bromwich integral

The following integral has to be resolved

$$\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} dt\ t^{-(N-1)d/2}\ \exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \tag{A.111}$$

The integral (A.111) can be traced back to the Bromwich integral, as shown in the following.
First of all, in order to simplify the derivation, the following notation is introduced

$$n=\frac{d}{2}(N-1) \tag{A.112}$$

$$s=E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho}) \tag{A.113}$$

so that the integral (A.111) can be rewritten as

$$\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} t^{-n}\ e^{ts}\,dt=\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}\frac{e^{ts}}{t^n}\,dt \tag{A.114}$$

The solution to this integral can be found looking at the procedure to found the inverse Laplace transform. If $F(t)$ is known, but the original function $f(s)$ is unknown and the transformation is not even listed in the *Mathematical Handbook of Formulas and Tables*, the main task would then be to perform the inverse operation

$$f(s)=\mathcal{L}^{-1}[F(t)] \tag{A.115}$$

The inverse of the Laplace transformation can be given explicitly by computing the forward Laplace transformation as

$$\frac{1}{\sqrt{2\pi}}F(\gamma+\imath\omega)=\frac{1}{\sqrt{2\pi}}\int_0^\infty f(s)\,e^{-(\gamma+\imath\omega)s}\,ds=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}[f(s)\Theta(s)\,e^{-\gamma\,s}]\,e^{-\imath\omega s}\,ds \tag{A.116}$$

where $\imath$ is the imaginary unit and, by means of the $\Theta(s)$ step function, the integration space has been extended in the range $(-\infty, +\infty)$, so that the corresponding Fourier transform can be easily seen to be given by

$$f(s)\Theta(s)\,e^{-\gamma\,s}=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}F(\gamma+\imath\omega)\,e^{i\omega s}\,d\omega=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\gamma+\imath\omega)\,e^{\imath\omega s}\,d\omega \tag{A.117}$$

It is then straightforward to obtain

$$f(s)\Theta(s)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\gamma+\imath\omega)\,e^{(\gamma+\imath\omega)s}\,d\omega \tag{A.118}$$

Finally, by setting the variable back to $t=\gamma+\imath\omega$ the previous integral can be rewritten as

$$f(s)\Theta(s)=\frac{1}{2\pi\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}F(t)\,e^{ts}\,dt \tag{A.119}$$

where $dt=\imath\,d\omega$. The integral (A.119) is called the Bromwich integral or Bromwich formula. The integration path goes vertically to the real axis, passing through $x=\gamma$. The path should be sufficiently to the right of all the possible poles of the integrand $F(t)e^{ts}$, but one can displace the path to the left by means of analytic continuation. In some physical problems it is practically useful to displace the path to the left ($\Re(t)<0$) so that the integrand of the inverse transformation $F(t)e^{ts}$ goes to zero at the both ends of the integration path.
Therefore, by comparing the integral (A.114) which has to be solved with the equation (A.119), the explicit form of the function $F(t)$ in the case of interest can be written as

$$F(t)=\frac{1}{t^n}\qquad\text{with}\quad n>0 \tag{A.120}$$

Looking at the *Mathematical Handbook of Formulas and Tables*, the function $F(t)$ corresponds to a Laplace transform of the function $f(s)$ given by

$$\mathcal{L}^{-1}[F(t)] = \mathcal{L}^{-1}\left[\frac{1}{t^n}\right] = \frac{s^{n-1}}{\Gamma(n)} = f(s) \qquad \text{where} \quad n > 0 \tag{A.121}$$

Finally, substituting the explicit form for the function $F(t)$ and $f(s)$ in equation (A.119)

$$\frac{s^{n-1}}{\Gamma(n)}\,\Theta(s) = \frac{1}{2\pi\imath}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}\frac{e^{ts}}{t^n}\,dt \tag{A.122}$$

and comparing this expression with the integral (A.114) which has to be solved leads to the solution

$$\int_{\gamma-\imath\infty}^{\gamma+\imath\infty}\frac{e^{ts}}{t^n}\,dt = \frac{2\pi\imath}{\Gamma(n)}\,s^{n-1}\,\Theta(s) \tag{A.123}$$

Moreover, by changing the variables in (A.123) following the transformations (A.112) and (A.113), the solution to the integral (A.111) can be finally written as

$$\begin{aligned}\int_{\gamma-\imath\infty}^{\gamma+\imath\infty} dt\;\; t^{-(N-1)d/2}\;\; \exp\left[t\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\right] \\ = \frac{2\pi\imath}{\Gamma[(N-1)d/2]}\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)^{(N-1)d/2-1}\Theta\left(E-\frac{P^2}{2m}-\phi(\boldsymbol{\rho})\right)\end{aligned} \tag{A.124}$$

A.7 Ensemble generated by Nosé-Hoover equations of motion

The partition function Z for a system of N identical particles is obtained by integrating the equilibrium distribution function $f(x_1, x_2, ...)$ over the whole phase space following the formula

$$Z = \frac{1}{N!\ (2\pi\hbar)^{3N}} \int dx_1 \int dx_2 ... f(x_1, x_2, ...) \equiv \zeta \int dx_1 \int dx_2 ... \rho(x_1, x_2, ...) \tag{A.125}$$

where $\hbar$ is the reduced Planck constant[3] and x_i are generalized coordinates. The constant factors for the equilibrium distribution and the partition functions are hereafter labeled as $\zeta = 1/[N!\ (2\pi\hbar)^{3N}]$. The projection of the equilibrium distribution function from the space (x_1, x_2) onto the space x_1 is carried out by integrating with respect to the variable x_2 as

$$f(x_1) = \int dx_2\, f(x_1, x_2) \tag{A.126}$$

In particular, the aim is to find the distribution function $f(\boldsymbol{q}, \boldsymbol{p})$ that is projected from the extended system onto the physical three dimensional ($d = 3$) system, in order to characterize the real ensemble of particles. In the extended system, the total Hamiltonian of equation (5.247) is conserved. Therefore, this method produces a microcanonical ensemble and the distribution function in virtual variables is expressed as

$$f(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}, \tilde{p}_s, \tilde{s}) = \delta(\tilde{\mathcal{H}} - E) \tag{A.127}$$

Accordingly, the partition function is

$$Z = \zeta \int d\tilde{p}_s \int d\tilde{s} \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}}\ \ \delta\left(\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln \tilde{s} - E \right) \tag{A.128}$$

where the integration variables abbreviations $d\tilde{\boldsymbol{q}} = d\tilde{\boldsymbol{q}}_1 d\tilde{\boldsymbol{q}}_2 \cdots d\tilde{\boldsymbol{q}}_N$ and $d\tilde{\boldsymbol{p}} = d\tilde{\boldsymbol{p}}_1 d\tilde{\boldsymbol{p}}_2 \cdots d\tilde{\boldsymbol{p}}_N$ have been used in the integral to simplify the notation. Then, the virtual momenta $\tilde{\boldsymbol{p}}_i$ and positions $\tilde{\boldsymbol{q}}_i$ can be transformed to the real variables $\boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i/s$ and $\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i$, while the degree of freedom $\tilde{s}$ has the same expression in both virtual and real systems of reference, as reported in (5.244), $\tilde{s} = s$. With these transformations, the volume element changes as

$$d\tilde{\boldsymbol{p}}\, d\tilde{\boldsymbol{q}} = s^{3N}\, d\boldsymbol{p}\, d\boldsymbol{q} \tag{A.129}$$

Hence the partition function becomes

$$Z = \zeta \int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q} \int ds\ \ s^{3N}\ \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln s - E \right) \tag{A.130}$$

or equivalently, using the shortest notation

$$\mathcal{H}_0(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) \quad \rightarrow \quad \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{A.131}$$

the partition function in real variables can be written more simply as

$$Z = \zeta \int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q} \int ds\ \ s^{3N}\ \delta\left(\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln s - E \right) \tag{A.132}$$

[3]The only slightly odd thing is the factor of $1/(2\pi\hbar)^{3N}$ that sits out front. It is a quantity that needs to be there simply on dimensional grounds. Indeed, Z should be dimensionless so $h = 2\pi\hbar$ must have dimension (length·momentum) or, equivalently, Joules per seconds (J·s). The actual value of h won't matter for any physical observable, like heat capacity, because these quantities are always computed by taking $\log Z$ and then differentiating. Despite this, there is actually a correct value for h: it is Planck's constant, $h = 2\pi\hbar \approx 6.6 \cdot 10^{-34}$ J·s. It is very strange to see Planck's constant in a formula that is supposed to be classical. In fact, it is a vestigial object. It is redundant, serving only as a reminder of where the classical world came from. And the classical world came from the quantum.

Since the argument of the δ function in the above equation has only one zero as a function of the variable s, a convenient equivalence relation can be employed to simplify the integral, that is

$$\delta[h(s)] = \frac{\delta(s - s_0)}{h'(s_0)} \tag{A.133}$$

where s_0 is the zero of function $h(s)$ in the argument of the Dirac function, and $h'(s_0)$ is the derivative of the function $h(s)$ evaluated in the point s_0, which has to be read, inside the integral, as the function $h'(s)$, so that the expressions for these two objects are

$$s_0 = \exp\left[-\frac{1}{g k_b T_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E\right)\right] \qquad \text{and} \qquad h'(s) = \frac{g k_b T_0}{s} \tag{A.134}$$

Using the relation (A.133) and the expressions (A.134) in (A.132), the partition function can be rewritten as

$$\begin{aligned} Z &= \frac{\zeta}{g k_b T_0}\int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q} \int ds\ s^{3N+1}\ \delta(s - s_0) = \frac{\zeta}{g k_b T_0}\int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q}\ s_0^{3N+1} \\ &= \frac{\zeta}{g k_b T_0}\int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\frac{3N+1}{g k_b T_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E\right)\right] \\ &= \underbrace{\frac{\zeta}{g k_b T_0}\exp\left[\left(\frac{3N+1}{g k_b T_0}\right)E\right]\int d\tilde{p}_s\ \exp\left[-\left(\frac{3N+1}{g k_b T_0}\right)\frac{\tilde{p}_s^2}{2Q}\right]}_{=\text{ constant}} \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\left(\frac{3N+1}{g k_b T_0}\right)\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right] \end{aligned} \tag{A.135}$$

where the constant terms and integrals have been collected before the rest of the integral expression as above. Defining the constant $g = (3N + 1)$, the partition function generated by the equations of motion (5.249)-(5.252) in virtual time sampling becomes

$$g = 3N+1 \quad : \qquad Z = \underbrace{\frac{\zeta}{g k_b T_0}\exp\left(\frac{E}{k_b T_0}\right)\int d\tilde{p}_s\ \exp\left[-\left(\frac{1}{k_b T_0}\right)\frac{\tilde{p}_s^2}{2Q}\right]}_{=\text{ constant}} \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right)\right]$$

Then, carrying out the integration with respect to $\tilde{p}_s$ variable,[4]

$$\begin{aligned} g = 3N+1 \quad : \qquad Z &= \frac{\zeta}{g k_b T_0}\exp\left(\frac{E}{k_b T_0}\right)(2\pi\, Q\, k_b T_0)^{1/2} \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right)\right] \\ &= \underbrace{\frac{\zeta}{g}\left(\frac{2\pi\, Q}{k_b T_0}\right)^{1/2}\exp\left(\frac{E}{k_b T_0}\right)}_{=\text{ constant } C} \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right)\right] \end{aligned}$$

and grouping all the constant factors, labeling them equal to C, the partition function in virtual time sampling results equal to

$$g = 3N+1 \quad : \qquad Z = C \int d\boldsymbol{p} \int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right)\right] \tag{A.137}$$

Therefore, if the number of degrees of freedom is set equal to $g = 3N + 1$, then the partition function of the extended system is equivalent to that of the physical system in the canonical (NVT) ensemble, except for a constant factor, and the equilibrium distribution function is

$$g = 3N+1 \quad : \qquad f(\boldsymbol{p},\boldsymbol{q}) = \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_b T_0}\right)\right] \tag{A.138}$$

[4]The integral with respect to the variable $\tilde{p}_s$ is a Gaussian integral with a solution given by

$$I_0(a) = \int_{-\infty}^{\infty} e^{-a\,x^2}\, dx = \sqrt{\frac{\pi}{a}} \tag{A.136}$$

With the quasi-ergodic hypothesis which relates the time average along the trajectory to the ensemble average, the averages of any static quantities expressed as functions of $\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{p}}_i/s$ along the trajectory determined by equations (5.249)-(5.252), are exactly those in the canonical ensemble iff the number of degrees of freedom are set equal to $g = 3N + 1$

$$\lim_{t_0\to\infty} \frac{1}{t_0}\int_0^{t_0} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\, d\tilde{t} = \langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\rangle \overset{g=3N+1}{=} \langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\rangle_c \equiv \langle A(\boldsymbol{p}, \boldsymbol{q})\rangle_c \tag{A.139}$$

where $\langle ...\rangle$ and $\langle ...\rangle_c$ denote the ensemble average in the extended system and in the canonical ensemble, respectively. The first equivalence in equation (A.139) is achieved by sampling data points at integer multiples of the virtual time unit $\Delta\tilde{t}$. This is called virtual time sampling. In this sampling, the real time interval of each time step is unequal. However, in practical molecular dynamics simulation is will be more convenient to sample at equal intervals in real time. As demonstrated in Appendix A, Section A.11.1, if the sampling is performed using equal intervals in real time t, with

$$t_1 = \int_0^{t_0} d\tilde{t}/s \tag{A.140}$$

the result is a weighted average in the extended system that is equivalent to a weighted average of the physical system in the canonical ensemble iff the number of degrees of freedom are set equal to $g = 3N$

$$\lim_{t_1\to\infty} \frac{1}{t_1}\int_0^{t_1} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\, dt = \frac{\langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})/s\rangle}{\langle 1/s\rangle} \overset{g=3N}{=} \langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\rangle_c \equiv \langle A(\boldsymbol{p}, \boldsymbol{q})\rangle_c \tag{A.141}$$

Therefore, in virtual time sampling, g should be equal to $3N + 1$, and in real time sampling, g must be equal to $3N$.

A.7.1 Ensemble with linear momentum conservation

The ensemble generated by the molecular dynamics method depends on the boundary conditions.[152] For particle in a box the ensemble generated is the traditional microcanonical (or canonical) ensemble where only the total energy of the system (or the energy associated to the pseudo-Hamiltonian (5.247)) is conserved. If the periodic boundary conditions are employed in the simulations, the ensemble is no longer strictly microcanonical (or canonical) because in this case the total linear momentum is also conserved. Indeed, even though Newton equations of motion also satisfy the conservation of the total angular momentum, periodic boundary conditions destroy the conservation of the total angular momentum.[29] In some cases, the neglect of the conservation of the total momentum will introduce only a small amount of error in the interpretation of molecular dynamics simulation results.[153] However, there are cases where the conservation of the total momentum should play a crucial role in determining the nature of an ensemble generated by the molecular dynamics method. The extended system method of Nosé is precisely such a case.[63] The generation of a canonical ensemble from an ergodic extended system is guaranteed by the Nosé derivation within the Hamiltonian formalism.[46] However, in this derivation, only the conservation of the total extended system energy is used, and the conservation of the total virtual linear momentum and the total virtual angular momentum are ignored.[46] One can safely ignore the conservation of the total virtual angular momentum because it is not conserved during the simulation if a periodic boundary condition is used.[152] However, one cannot ignore the conservation of the total virtual linear momentum because this is conserved during numerical simulations. Therefore the Nosé treatment is no longer strictly valid for actual extended system method simulations. Consequently, any theoretical proof which will determine the conditions under which one can obtain the canonical ensemble of a physical system from the extended system method must include the conservation of the total virtual linear momentum as well as the conservation of the total energy of the extended system.

In the following, a generalization of the Nosé treatment including the conservation of the total virtual linear momentum will be presented, and an analytical prove that a canonical ensemble is generated from the extended system *only if* the total virtual linear momentum is zero will be given. This generalized Nosé treatment shows that the physical system satisfying a canonical ensemble is a $(N - 1)$ particle system with a different mass spectrum from that of the original N particle physical system.

For the proof, the ergodicity of the extended system is assumed, so that the extended system partition

function has a microcanonical ensemble form of energy δ function and linear momentum δ function. The partition function of an ergodic extended system is

$$Z = \zeta \int d\tilde{p}_s \int d\tilde{s} \int\int \prod_{i=1}^{N} d\tilde{\boldsymbol{p}}_i \, d\tilde{\boldsymbol{q}}_i \;\; \delta\Big(\mathcal{H}_0(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln \tilde{s} - E\Big) \; \delta\Big(\sum_{i=1}^{N} \tilde{\boldsymbol{p}}_i - \tilde{\mathbf{P}}_0\Big) \tag{A.142}$$

where $\zeta = 1/[N!\ (2\pi\hbar)^{3N}]$ is a normalization constant for the partition function, $\tilde{\mathbf{P}}_0$ is the total linear angular momenta, while $H_0(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})$ is given by (A.131) and it contains the first two terms (nuclear kinetic and potential) of the extended system Hamiltonian defined in equation (5.247). By introducing the center of mass momenta,

$$\tilde{\boldsymbol{\kappa}}_i = \tilde{\boldsymbol{p}}_i - \tilde{\mathbf{P}}_0/N \tag{A.143}$$

the kinetic energy term becomes a sum of the relative kinetic energy and the kinetic energy of the center of mass follows:

$$\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2} + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2} \qquad \text{where} \quad m = \sum_{i=1}^{N} m_i \tag{A.144}$$

The partition function can be reformulated using the center of mass momenta (A.143), that is

$$Z = \zeta \int d\tilde{p}_s \int d\tilde{s} \int\int \prod_{i=1}^{N} d\tilde{\boldsymbol{\kappa}}_i \, d\tilde{\boldsymbol{q}}_i \;\; \delta(\mathcal{H}_{es} - E) \; \delta\Big(\sum_{i=1}^{N} \tilde{\boldsymbol{\kappa}}_i\Big) \tag{A.145}$$

where H_{es} is the Hamiltonian in the extended system given by

$$\mathcal{H}_{es} = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2} + \frac{\tilde{\mathbf{P}}_0^2}{2M\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln \tilde{s} \tag{A.146}$$

The momentum delta function in (A.145) can be eliminated by performing an integration over the variable $\tilde{\boldsymbol{\kappa}}_N$, in the following way

$$\begin{aligned} Z &= \zeta \int d\tilde{p}_s \int d\tilde{s} \int \prod_{i=1}^{N} d\tilde{\boldsymbol{q}}_i \int \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\kappa}}_i \int d\tilde{\boldsymbol{\kappa}}_N \;\; \delta(\mathcal{H}_{es} - E) \; \delta\Big(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i + \tilde{\boldsymbol{\kappa}}_N\Big) \\ &= \zeta \int d\tilde{p}_s \int d\tilde{s} \int \prod_{i=1}^{N} d\tilde{\boldsymbol{q}}_i \int \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\kappa}}_i \;\; \delta(\mathcal{H}'_{es} - E) \end{aligned} \tag{A.147}$$

where the Hamiltonian $\mathcal{H}_{es}$ has been transformed into the Hamiltonian H'_{es} by the application of the momentum delta function, so that the argument of the remaining energy delta function in the last expression in (A.147) results equal to

$$\mathcal{H}'_{es} - E = \sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2} + \frac{1}{2m_N\tilde{s}^2}\Big(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\Big)^2 + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln \tilde{s} - E \tag{A.148}$$

The first two terms in the previous equation can be rearranged as

$$\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2} + \frac{1}{2m_N\tilde{s}^2}\Big(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\Big)^2 = \frac{1}{2\,\tilde{s}^2}\Big(\frac{1}{m_N} + \sum_{j=1}^{N-1} \frac{1}{m_j}\delta_{ij}\Big)\Big(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\Big)^2 \tag{A.149}$$

Therefore, in order to rewrite more conveniently these two terms, the diagonalization of the inverse mass matrix

$$(M^{-1})_{ij} = \frac{1}{m_N} + \sum_{j=1}^{N-1} \frac{1}{m_j}\delta_{ij} \tag{A.150}$$

has to be performed, introducing normal mode momenta $\tilde{\boldsymbol{\pi}}_i$ such that

$$\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2} + \frac{1}{2m_N\tilde{s}^2}\left(\sum_{i=1}^{N-1}\tilde{\boldsymbol{\kappa}}_i\right)^2 = \sum_{i=1}^{N-1}\frac{\tilde{\boldsymbol{\pi}}_i^2}{2\mu_i\tilde{s}^2} \tag{A.151}$$

After this diagonalization, the partition function becomes

$$Z = \zeta\int d\tilde{p}_s\int d\tilde{s}\int\prod_{i=1}^{N-1}d\tilde{\boldsymbol{\pi}}_i\int\prod_{i=1}^{N}d\tilde{\boldsymbol{q}}_i\ \delta\left(\sum_{i=1}^{N-1}\frac{\tilde{\boldsymbol{\pi}}_i^2}{2\mu_i\tilde{s}^2} + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0\ln\tilde{s} - E\right) \tag{A.152}$$

Finally, one introduces the physical spatial positions $\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i$, the physical momenta $\boldsymbol{p}_i = \tilde{\boldsymbol{\pi}}_i/\tilde{s}$, and the variable $s = \tilde{s}$, so that the variable s in the nuclear kinetic energy term (i.e. in the first term of the delta function argument in (A.152)) can be eliminated, and the following form for partition function is then obtained:

$$Z = \zeta\int d\tilde{p}_s\int\prod_{i=1}^{N-1}d\boldsymbol{p}_i\int\prod_{i=1}^{N}d\boldsymbol{q}_i\int ds\ s^{3N-3}\ \delta\left(\sum_{i=1}^{N-1}\frac{\boldsymbol{p}_i^2}{2\mu_i} + \frac{\tilde{\mathbf{P}}_0^2}{2ms^2} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0\ln s - E\right) \tag{A.153}$$

The integral above has a similar form to the one in the original formulation of Nosé, equation (A.130), except for the presence of the center of mass kinetic energy term in the argument of the energy delta function, and the integration over $3N-3$ momentum variables instead of $3N$. In general, if the total linear momentum is different from zero, $\tilde{\mathbf{P}}_0 \neq \mathbf{0}$, the argument of the energy delta function has two roots with respect to the variable s, which can be called s_1 and s_2 (see Figure A.4), so that the function $h(s)$ in the above integral can be rewritten using the relation

$$\delta[h(s)] = \sum_{i=1}^{2}\frac{\delta(s-s_i)}{|-K_0/s^3 + gk_bT_0/s|_{s=s_i}} \qquad \text{where} \quad K_0 = \frac{\tilde{\mathbf{P}}_0^2}{m} \tag{A.154}$$

where the denominator in the expression above is given by the derivative of the delta function argument in (A.153) with respect to the variable s.

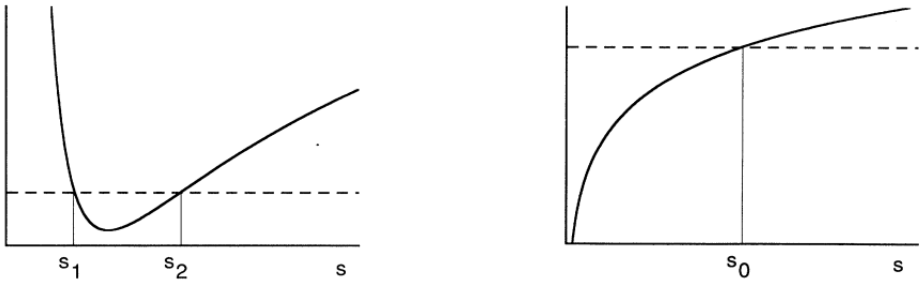

Figure A.4: Two solutions of the equation $K_0/s^2 + gk_bT_0\ln s = E_0$ with $K_0 \neq 0$ (left panel) and the solution of the same equation with $K_0 = 0$ (right panel).

Then, using the relation (A.154) to easily find a convenient form for the partition function, equation (A.153) becomes

$$\begin{aligned} Z &= \zeta\int d\tilde{p}_s\int\prod_{i=1}^{N-1}d\boldsymbol{p}_i\int\prod_{i=1}^{N}d\boldsymbol{q}_i\int ds\ s^{3N-3}\ \sum_{i=1}^{2}\frac{\delta(s-s_i)}{|-K_0/s^3 + gk_bT_0/s|_{s=s_i}} \\ &= \zeta\int d\tilde{p}_s\int\prod_{i=1}^{N-1}d\boldsymbol{p}_i\int\prod_{i=1}^{N}d\boldsymbol{q}_i\sum_{i=1}^{2}\frac{s_i^{3N-3}}{|-K_0/s^3 + gk_bT_0/s|_{s=s_i}} \end{aligned} \tag{A.155}$$

However, it is easy to see in the expression above that the integrand is not the Boltzmann factor of a physical Hamiltonian.

In the case $\tilde{\mathbf{P}}_0 = \mathbf{0}$, the problem simplifies considerably, since the partition function (A.153) can now be written as

$$Z = \zeta \int d\tilde{p}_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \int ds\ s^{3N-3}\ \delta\left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln s - E \right) \tag{A.156}$$

and the integration over the variable s can be trivially performed using the relation (A.133) for the delta function in the integrand: the zero of the function that is the energy delta argument,

$$h(s) = \sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln s - E \tag{A.157}$$

is equal to

$$h(s_0) = 0 \quad \leftrightarrow \quad s_0 = \exp\left[-\frac{1}{g k_b T_0} \pm \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E \right) \right] \tag{A.158}$$

Inserting this expression for s_0 in (A.133) and then using the relation in the partition function integrand (A.156) leads to

$$\begin{aligned}
Z &= \zeta \int d\tilde{p}_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \int ds\ s^{3N-3}\ \frac{s}{g k_b T_0}\ \delta(s - s_0) = \frac{\zeta}{g k_b T_0} \int d\tilde{p}_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ s_0^{3N-2} \\
&= \frac{\zeta}{g k_b T_0} \int d\tilde{p}_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left[-\frac{3N-2}{g k_b T_0} \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E \right) \right] \\
&= \underbrace{\frac{\zeta}{g k_b T_0} \exp\left[\left(\frac{3N-2}{g} \right) \frac{E}{k_b T_0} \right] \int d\tilde{p}_s \exp\left[-\left(\frac{3N-2}{g} \right) \frac{1}{k_b T_0} \frac{\tilde{p}_s^2}{2Q} \right]}_{= \text{constant}} \times \\
&\qquad \times \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left[-\left(\frac{3N-2}{g} \right) \frac{H_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right]
\end{aligned}$$

where the physical Hamiltonian has been labeled as

$$\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}) \tag{A.159}$$

The same result can be obtained starting from the more general expression (A.155) and imposing $K_0 = 0$ in the equation. Since $\tilde{\mathbf{P}}_0 = \mathbf{0}$, there is only one root s_0, so that the summation in (A.155) reduces to a single term $i = 1$ with $s_i = s_0$, leading to the same result. Finally, if $g = 3N - 2$ is chosen, the partition function assumes the form

$$\begin{aligned}
g = 3N-2 \quad : \quad Z &= \frac{\zeta}{g k_b T_0} \exp\left(\frac{E}{k_b T_0} \right) \int d\tilde{p}_s \exp\left[-\left(\frac{1}{k_b T_0} \right) \frac{\tilde{p}_s^2}{2Q} \right] \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left(-\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right) \\
&= \frac{\zeta}{g k_b T_0} \exp\left(\frac{E}{k_b T_0} \right) (2\pi\, Q\, k_b T_0)^{1/2} \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left(-\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right) \\
&= \underbrace{\frac{\zeta}{g} \left(\frac{2\pi\, Q}{k_b T_0} \right)^{1/2} \exp\left(\frac{E}{k_b T_0} \right)}_{= \text{constant } C} \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left(-\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right)
\end{aligned}$$

where the integral with respect to $\tilde{p}_s$ variable has been performed from the first to the second line of the expressions above, and the Hamiltonian $\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})$, given by (A.159), has been introduces. Grouping all the constant factors in front of the relevant integral, and labeling these constant factors C, the partition function finally results equal to

$$g = 3N-2 \quad : \quad Z = C \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i\ \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right) \right] \tag{A.160}$$

The correspondent equilibrium distribution function is

$$g = 3N - 2 \quad : \qquad f(\boldsymbol{p}, \boldsymbol{q}) = \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0}\right)\right] = \exp\left[-\frac{1}{k_b T_0}\left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q})\right)\right] \tag{A.161}$$

The partition function and the equilibrium distribution function obtained in this way corresponds to a canonical ensemble of the $(N-1)$ particle system with mass spectrum $\{\mu_i\}$, starting from the N particle system with the mass spectrum $\{m_i\}$, only if the total linear virtual momentum is zero.
The equilibrium distribution function (A.161) and the partition function (A.160) can be used to compute ensemble averages in the canonical ensemble that corresponds, assuming the ergodicity hypothesis, with time averages over the molecular dynamics simulation trajectory, performing a sampling data points at integer multiples of the virtual time unit $\Delta\tilde{t}$. In order to obtain a partition function for a sampling in real time unit Δt, starting from the expression for the partition function (A.156), the same procedure reported in Appendix A, Section A.11.1, can be followed. The partition function that leads to a canonical ensemble form of the ensemble average (A.291) corresponding to a real time sampling has obviously the same form as the previous derived partition function (A.160), if the number of degrees of freedom are set equal to $g = 3N - 3$. Therefore, a value of $g = 3N - 3$ has to be used in practical simulations, in order to sample correctly the canonical ensemble in the real physical system using the equations of motion (5.249)-(5.252). However, these equations are expressed with virtual variables referred to the extended system. In Section 5.3.4.1, the equations of motion will be computed in real variables, associated to the physical system, and the correspondent conserved quantity form is derived.
Therefore, it has been demonstrated that if the extended system is ergodic and if the total linear virtual momentum of the N particle physical system is zero, then the extended system method will generate a canonical ensemble for a $(N-1)$ particle system with a different mass spectrum.[63] In general, the new masses are different from the original nuclear masses. For the special case of N identical original nuclear masses $m_i = m$, one finds a final mass spectrum $\mu_i = m$ for $i = 1, ..., N-2$ and $\mu_{N-1} = m/N$. This new mass spectrum will introduce, in principle, an error in calculations of the dynamical properties. Of course, the conventional molecular dynamics introduces the same error,[5] and as N becomes large, the contribution of the light mass becomes unimportant in practical considerations. As far as thermodynamic averages are concerned, the difference of mass spectra is always irrelevant because of the equipartition theorem. The only significant change relevant to the thermodynamic averages is the reduction of the number of degrees of freedom from $3N$ to $3N - 3$. Therefore, the consequences of a different mass spectrum are irrelevant to thermodynamic averages, but relevant to the dynamical properties and relaxation times of the system.[63]
As previously demonstrated, if $\tilde{\mathbf{P}}_0 \neq \mathbf{0}$, then the physical system does not satisfy a canonical ensemble. Since the total linear momentum is conserved by the equations of motion (5.249)-(5.252), it can be set to zero at the beginning of the simulation. In this way, the total linear momentum will be conserved near to zero values (due to floating errors in computer simulations) for the whole molecular dynamics trajectory. However, the canonical ensemble sampling by the dynamics trajectory is satisfied even if the total linear momentum has small deviations from the zero value. Indeed, if $|\tilde{\mathbf{P}}_0|$ is small, the following approximation can be made. For small $|\tilde{\mathbf{P}}_0|$, s_1 is very small so that

$$s_1^{3N-3} << s_2^{3N-3}$$

Hence the first integral containing s_1 in the partition function (A.155) can be neglected. The remaining integral containing s_2 can be approximated as follows:

$$\left|-K_0/s_2^3 + g k_b T_0/s_2\right| \approx |g k_b T_0/s_2| \tag{A.162}$$

since $K_0/s_2^2 << g k_b T_0$ for small $|\tilde{\mathbf{P}}_0|$. After these two approximations, the partition function reduces to the partition function of zero total momentum, with a correction of order $O(K_0/g k_b T_0)$. The reliability

[5] It can be performed a demonstration similar to that outlined in this section, applying it to the conventional molecular dynamics equations of motion, starting from a partition function similar to that in equation (A.145) with both energy conservation and linear total momentum conservation delta functions of a physical system. For this case, however, the integration of the momentum delta function leads to the microcanonical ensemble of the $(N-1)$ particle system with the mass spectrum $\{\mu_i\}$ whether $\tilde{\mathbf{P}}_0 = \mathbf{0}$ or not.[63]

of these approximations is also confirmed by theoretical simulations performed by K. Cho *et al.*[63] with different values of the center of mass kinetic energy. They found that, for a small center of mass kinetic energy ($\langle K_0/s^2 \rangle \leq T_0$) the results for the computation of average moments of the instantaneous temperature and thermostat kinetic energy fluctuations are found to be quite similar to those of zero total momentum. This reasoning suggests that quite a good approximate canonical ensemble can be generated with nonzero total linear momentum, even if the total momentum is on the order of the external temperature. However, as the center of mass kinetic energy increases ($\langle K_0/s^2 \rangle >> T_0$), the deviation from the canonical ensemble values becomes large.[63] These analysis guarantee the numerical stability of generating a canonical ensemble by the extended system method in practical applications where computational errors inevitably introduce a small nonzero total linear momentum.

A.8 Ensemble generated by Ferrario equations of motion

Consider a system of N particles moving in a three dimensional space ($d = 3$). The time trajectory generated by the equations of motion (5.451)-(5.456) corresponds, by invoking the ergodic hypothesis, to a microcanonical ensemble distribution for the extended system in the virtual phase space. Therefore, the distribution function in virtual variables is given by

$$f(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}, \tilde{p}_v, \tilde{v}, \tilde{p}_s, \tilde{s}) = \delta(\tilde{\mathcal{H}} - E) \tag{A.163}$$

However, the properties of interest can be quite generally expressed as ensemble averages of an observable $f(\boldsymbol{p}, \boldsymbol{q}, v)$ that is projected from the extended system onto the physical system, in order to characterize the real ensemble of particles. Accordingly to the distribution function (A.163), the partition function is

$$Z = \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}} \;\; \delta(\tilde{\mathcal{H}}(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}, \tilde{p}_s, \tilde{s}, \tilde{p}_v, \tilde{v}) - E) \tag{A.164}$$

where $\zeta = 1/[N!\ (2\pi\hbar)^{3N}]$ is a normalization factor for the partition function, the integration variables abbreviations $d\tilde{\boldsymbol{q}} = d\tilde{\boldsymbol{q}}_1 d\tilde{\boldsymbol{q}}_2 \cdots d\tilde{\boldsymbol{q}}_N$ and $d\tilde{\boldsymbol{p}} = d\tilde{\boldsymbol{p}}_1 d\tilde{\boldsymbol{p}}_2 \cdots d\tilde{\boldsymbol{p}}_N$ have been used in the integral to simplify the notation and the Hamiltonian in the argument of the delta function has the form in (5.450). Then, the virtual coordinates can be transformed to the real variables using the relations[78]

$$\boldsymbol{q}_i = v^{1/3}\, \tilde{\boldsymbol{q}}_i \qquad \boldsymbol{p}_i = \frac{\tilde{\boldsymbol{p}}_i}{s v^{1/3}} \qquad v = \tilde{v} \qquad p_v = \frac{\tilde{p}_v}{s} \tag{A.165}$$

$$s = \tilde{s} \qquad p_s = \tilde{p}_s \qquad t = \int^t \frac{d\tilde{t}}{s} \tag{A.166}$$

With these transformations, the volume element changes as

$$d\tilde{\boldsymbol{p}}\, d\tilde{\boldsymbol{q}}\, d\tilde{p}_v\, d\tilde{v}\, d\tilde{p}_s\, d\tilde{s} = s^{3N+1}\, d\boldsymbol{p}\, d\boldsymbol{q}\, dp_v\, dv\, dp_s\, ds \tag{A.167}$$

Consequently, the Hamiltonian virtual variables form (5.450), converted in real coordinates frame of reference using the relations (A.165) and (A.166), is given by

$$\begin{aligned} \mathcal{H}(\boldsymbol{p}, \boldsymbol{q}, p_s, s, p_v, v) &= \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\{\boldsymbol{q}_i\}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s \\ &= \mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s \end{aligned} \tag{A.168}$$

where the shortest notation

$$\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}, v) = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\{\boldsymbol{q}_i\}, v) \tag{A.169}$$

has been used. Applying the change of variables to the partition function, its expression in real coordinates becomes

$$Z = \zeta \int dp_v \int dv \int dp_s \int ds \int d\boldsymbol{p} \int d\boldsymbol{q}\, s^{3N+1}\; \delta\!\left(\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s - E\right)$$

The argument of the δ function in the above equation, taken as a function of the variable s, has only one zero. Therefore, a convenient equivalence relation can be employed to simplify the integral, that is

$$\delta[h(s)] = \frac{\delta(s - s_0)}{h'(s_0)} \tag{A.170}$$

where s_0 is the zero of function $h(s)$ in the argument of the Dirac function, and $h'(s_0)$ is the derivative of the function $h(s)$ evaluated in the point s_0, which has to be read, inside the integral (i.e. before applying

the delta function $\delta(s-s_0)$ in the integral), as the function $h'(s)$. Looking at the partition function form, the expressions for the zero of the function $h(s)$ and its derivative are

$$s_0 = \exp\left[-\frac{1}{gk_bT_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}-E\right)\right] \qquad \text{and} \qquad h'(s)=\frac{gk_bT_0}{s} \tag{A.171}$$

Using the relation (A.170) and the expressions (A.171) in the previous partition function form, it becomes

$$\begin{aligned} Z &= \frac{\zeta}{gk_bT_0}\int dp_v\int dv\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\int ds\ s^{3N+2}\ \delta(s-s_0) \\ &= \frac{\zeta}{gk_bT_0}\int dp_v\int dv\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\ s_0^{3N+2} \\ &= \frac{\zeta}{gk_bT_0}\int dp_v\int dv\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\frac{3N+2}{gk_bT_0}\ \left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+\frac{p_v^2}{2W}+P_0v+\frac{p_s^2}{2Q}-E\right)\right] \\ &= C_g\int dv\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left\{-\left(\frac{3N+2}{gk_bT_0}\right)[\,\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+P_0v\,]\right\} \end{aligned} \tag{A.172}$$

where the constant terms and integrals that has been collected before the rest of the integral expression and is given by

$$C_g = \frac{\zeta}{gk_bT_0}\exp\left[\left(\frac{3N+2}{gk_bT_0}\right)E\right]\int dp_v\ \exp\left[-\left(\frac{3N+2}{gk_bT_0}\right)\frac{p_v^2}{2W}\right]\int dp_s\ \exp\left[-\left(\frac{3N+2}{gk_bT_0}\right)\frac{p_s^2}{2Q}\right] \tag{A.173}$$

Defining the constant $g=(3N+2)$, the partition function becomes

$$g=3N+2 \quad : \qquad Z = C\int dv\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+P_0v}{k_bT_0}\right)\right] \tag{A.174}$$

with the constant C given by

$$\begin{aligned} g=3N+2 \quad : \qquad C &= \frac{\zeta}{gk_bT_0}\exp\left(\frac{E}{k_bT_0}\right)\int dp_v\ \exp\left[-\left(\frac{1}{k_bT_0}\right)\frac{p_v^2}{2W}\right]\int dp_s\ \exp\left[-\left(\frac{1}{k_bT_0}\right)\frac{p_s^2}{2Q}\right] \\ &= \frac{\zeta}{gk_bT_0}\exp\left(\frac{E}{k_bT_0}\right)(2\pi\,W\,k_bT_0)^{1/2}\ (2\pi\,Q\,k_bT_0)^{1/2} = \frac{2\pi\zeta}{g}\sqrt{WQ}\ \exp\left(\frac{E}{k_bT_0}\right) \end{aligned}$$

where the integration over p_v and p_s variables are carried out very easily. Therefore, the partition function generated by the equations of motion (5.451)-(5.456) in virtual time sampling can be written, in the real physical coordinates frame of reference, as

$$g=3N+2 \quad : \qquad Z = \underbrace{\frac{2\pi\zeta}{g}\sqrt{WQ}\ \exp\left(\frac{E}{k_bT_0}\right)}_{=\text{ constant } C}\int dv\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+P_0v}{k_bT_0}\right)\right] \tag{A.175}$$

Therefore, if the number of degrees of freedom is set equal to $g=3N+2$, then the partition function of the extended system is equivalent to that of the physical system in the constant temperature and constant pressure (NPT) ensemble, except for a constant factor, and the equilibrium distribution function is

$$g=3N+2 \quad : \qquad f(\boldsymbol{p},\boldsymbol{q},v) = \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q},v)+P_0v}{k_bT_0}\right)\right] \tag{A.176}$$

In conclusion, it has been demonstrated that the time averages computed over the dynamical trajectory obtained by integrating the equations of motion (5.451)-(5.456) numerically are equivalent, if the motion is ergodic, to the NPT ensemble averages, iff the number of degrees of freedom are set equal to $g=3N+2$, that is

$$\lim_{t_0\to\infty}\frac{1}{t_0}\int_0^{t_0}A(\boldsymbol{p},\boldsymbol{q},v)\,d\tilde{t} = \langle A(\boldsymbol{p},\boldsymbol{q},v)\rangle \ \overset{g=3N+2}{=}\ \langle A(\boldsymbol{p},\boldsymbol{q},v)\rangle_{\text{NPT}} \tag{A.177}$$

where $\langle ...\rangle$ and $\langle ...\rangle_{\text{NPT}}$ denote the ensemble average in the extended system and in the constant temperature and constant pressure (NPT) ensemble, respectively. The first equivalence in equation (A.177)

is achieved by sampling data points at integer multiples of the virtual time unit $\Delta\tilde{t}$. This is called virtual time sampling. In this sampling, the real time interval of each time step is unequal. However, in practical molecular dynamics simulation is will be more convenient to sample at equal intervals in real time. As demonstrated in Appendix A, Section A.11 (or in Section 5.4, subsection 5.4.1.4), if the sampling is performed using equal intervals in real time t, with

$$t_1 = \int_0^{t_0} d\tilde{t}/s \tag{A.178}$$

the result is a weighted average in the extended system that is equivalent to a weighted average of the physical system in the canonical ensemble iff the number of degrees of freedom are set equal to $g = 3N+1$

$$\lim_{t_1\to\infty} \frac{1}{t_1}\int_0^{t_1} A(\boldsymbol{p},\boldsymbol{q},v)\,dt = \frac{\langle A(\boldsymbol{p},\boldsymbol{q},v)/s\rangle}{\langle 1/s\rangle} \overset{g=3N+1}{=} \langle A(\boldsymbol{p},\boldsymbol{q},v)\rangle_{\text{NPT}} \tag{A.179}$$

Therefore, in virtual time sampling, g should be equal to $3N+2$, and in real time sampling, g must be equal to $3N+1$. The choice of $g = 3N+1$ for real variables sampling stems from the fact that, including the barostat, the number of thermostated degrees of freedom is $3N+1$.

A.8.1 Ensemble with linear momentum conservation

As previously explained and demonstrated in Section A.7.1, the ensemble generated by the molecular dynamics method depends on the boundary conditions.[152] For particle in a box subject to the equations of motion (5.451)-(5.456), the ensemble generated is the traditional constant temperature and constant pressure (NPT) ensemble, where only to the Hamiltonian (5.450)) (or its non-Hamiltonian (A.168) in real physical variables) is conserved. Due to the conservation of the linear total momentum, MD methods produce ensembles that deviate slightly from the statistical mechanical ensembles. Indeed, if the periodic boundary conditions are employed in the simulations, the ensemble is no longer strictly the standard NPT ensemble because in this case the total linear momentum is also conserved.[6] There are cases where the conservation of the total momentum should play a crucial role in determining the nature of an ensemble generated by the molecular dynamics method. The extended system method of Nosé is precisely such a case: its generalization to conservation of linear total momentum has been discussed in Section A.7.1 following the demonstration of Ref. [63]. Since the Ferrario approach to a constant temperature and pressure MD method explained above (see Section 5.4.1) is based on the extended system technique as the Nosé-Hoover thermostat approach, it is obvious to extend the treatment for the conservation of total linear momentum also to this method, following the procedure in Section (A.7.1).
In the following, a generalization of the Ferrario treatment including the conservation of the total virtual linear momentum will be presented, and an analytical prove that a constant temperature and pressure (NPT) ensemble is generated from the extended system *only if* the total virtual linear momentum is zero will be given. This generalized Ferrario treatment shows that the physical system satisfying a canonical ensemble is a $(N-1)$ particle system with a different mass spectrum from that of the original N particle physical system.
For the proof, the ergodicity of the extended system is assumed, so that the extended system partition function is associated to a microcanonical ensemble in the energy and in the linear momentum through delta functions forms. The partition function of an ergodic extended system subject to equations of motion with form given by (5.451)-(5.456) is

$$Z = \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int\int \prod_{i=1}^{N} d\tilde{\boldsymbol{p}}_i\, d\tilde{\boldsymbol{q}}_i \;\; \delta(\mathcal{H}_{es} - E)\; \delta\left(\sum_{i=1}^{N} \tilde{\boldsymbol{p}}_i - \tilde{\mathbf{P}}_0\right) \tag{A.180}$$

where $\zeta = 1/[N!\,(2\pi\hbar)^{3N}]$ is a normalization constant for the partition function, $\tilde{\mathbf{P}}_0$ is the total linear angular momenta, while $\mathcal{H}_{es}$ is the Hamiltonian of the extended system expressed in virtual variables, given by (5.450) and here reported

$$\mathcal{H}_{es} = \frac{1}{2}\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\,\tilde{s}^2\,\tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3}\tilde{\boldsymbol{q}}_i\},\tilde{v}) + \frac{\tilde{p}_v^2}{2\tilde{s}^2 W} + P_0\tilde{v} + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln\tilde{s} \tag{A.181}$$

[6] Remember: Even though Newtonian equations of motion also satisfy the conservation of the total angular momentum, periodic boundary conditions destroy the conservation of the total angular momentum.[29]

By introducing the center of mass momenta,

$$\tilde{\boldsymbol{\kappa}}_i = \tilde{\boldsymbol{p}}_i - \tilde{\mathbf{P}}_0/N \tag{A.182}$$

the kinetic energy term in (A.181) becomes a sum of the relative kinetic energy and the kinetic energy of the center of mass follows:

$$\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2\,\tilde{v}^{2/3}} = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2\,\tilde{v}^{2/3}} + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2\,\tilde{v}^{2/3}} \qquad \text{where} \quad m = \sum_{i=1}^{N} m_i \tag{A.183}$$

The partition function can be reformulated using the center of mass momenta (A.182), that is

$$Z = \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int\int \prod_{i=1}^{N} d\tilde{\boldsymbol{\kappa}}_i\, d\tilde{\boldsymbol{q}}_i \;\; \delta(\mathcal{H}_{es} - E)\; \delta\left(\sum_{i=1}^{N} \tilde{\boldsymbol{\kappa}}_i\right) \tag{A.184}$$

The momentum delta function in (A.184) can be eliminated by performing an integration over the variable $\tilde{\boldsymbol{\kappa}}_N$, in the following way

$$\begin{aligned} Z &= \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int \prod_{i=1}^{N} d\tilde{\boldsymbol{q}}_i \int \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\kappa}}_i \int d\tilde{\boldsymbol{\kappa}}_N \;\; \delta(\mathcal{H}_{es} - E)\; \delta\left(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i + \tilde{\boldsymbol{\kappa}}_N\right) \\ &= \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int \prod_{i=1}^{N} d\tilde{\boldsymbol{q}}_i \int \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\kappa}}_i \;\; \delta(\mathcal{H}'_{es} - E) \end{aligned} \tag{A.185}$$

where the Hamiltonian $\mathcal{H}_{es}$ has been transformed into the Hamiltonian $\mathcal{H}'_{es}$ by the application of the momentum delta function in the integration over the variable $\tilde{\boldsymbol{\kappa}}_N$, so that the argument of the remaining energy delta function in the last expression in (A.185) contains the Hamiltonian $\mathcal{H}'_{es}$ and results equal to

$$\begin{aligned} \mathcal{H}'_{es} - E = &\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2\,\tilde{v}^{2/3}} + \frac{1}{2m_N\tilde{s}^2\,\tilde{v}^{2/3}} \left(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\right)^2 + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2\,\tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3}\tilde{\boldsymbol{q}}_i\}, \tilde{v}) \\ &+ \frac{\tilde{p}_v^2}{2\tilde{s}^2 W} + P_0\tilde{v} + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln\tilde{s} - E \end{aligned} \tag{A.186}$$

The first two terms in the previous equation can be rearranged as

$$\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2\,\tilde{v}^{2/3}} + \frac{1}{2m_N\tilde{s}^2\,\tilde{v}^{2/3}} \left(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\right)^2 = \frac{1}{2\,\tilde{s}^2\,\tilde{v}^{2/3}} \left(\frac{1}{m_N} + \sum_{j=1}^{N-1} \frac{1}{m_j}\delta_{ij}\right) \left(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\right)^2 \tag{A.187}$$

In order to rewrite more conveniently this two terms, a diagonalization of the inverse mass matrix

$$(M^{-1})_{ij} = \frac{1}{m_N} + \sum_{j=1}^{N-1} \frac{1}{m_j}\delta_{ij} \tag{A.188}$$

can be performed, introducing normal mode momenta $\tilde{\boldsymbol{\pi}}_i$ such that

$$\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\kappa}}_i^2}{2m_i\tilde{s}^2\,\tilde{v}^{2/3}} + \frac{1}{2m_N\tilde{s}^2\,\tilde{v}^{2/3}} \left(\sum_{i=1}^{N-1} \tilde{\boldsymbol{\kappa}}_i\right)^2 = \sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\pi}}_i^2}{2\mu_i\tilde{s}^2\,\tilde{v}^{2/3}} \tag{A.189}$$

After the diagonalization of the inverse mass matrix, the partition function becomes

$$\begin{aligned} Z = \zeta \int d\tilde{p}_v \int d\tilde{v} \int d\tilde{p}_s \int d\tilde{s} \int \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\pi}}_i \prod_{i=1}^{N} \int d\tilde{\boldsymbol{q}}_i \;\; \delta\Bigg(&\sum_{i=1}^{N-1} \frac{\tilde{\boldsymbol{\pi}}_i^2}{2\mu_i\tilde{s}^2\,\tilde{v}^{2/3}} + \frac{\tilde{\mathbf{P}}_0^2}{2m\tilde{s}^2\,\tilde{v}^{2/3}} + \phi(\{\tilde{v}^{1/3}\tilde{\boldsymbol{q}}_i\}, \tilde{v}) \\ &+ \frac{\tilde{p}_v^2}{2\tilde{s}^2 W} + P_0\tilde{v} + \frac{\tilde{p}_s^2}{2Q} + gk_bT_0 \ln\tilde{s} - E\Bigg) \end{aligned} \tag{A.190}$$

Finally, one introduces the physical spatial positions and momenta as given by the transformations (A.165), so that

$$\boldsymbol{q}_i = v^{1/3}\,\tilde{\boldsymbol{q}}_i \qquad\qquad \boldsymbol{p}_i = \frac{\tilde{\boldsymbol{\pi}}_i}{s v^{1/3}} \tag{A.191}$$

together with the equivalence between virtual and real frame of reference for the variables associated to the thermostat and the barostat, as given by (A.165) and (A.166),

$$s = \tilde{s} \qquad\qquad p_s = \tilde{p}_s \qquad\qquad v = \tilde{v} \tag{A.192}$$

Moreover, the transformation in (A.165) from virtual to real frame of reference for the variable p_v associated to the barostat momentum is essential to eliminate the dependence from the variable s in the delta function argument,

$$p_v = \tilde{p}_v / s \tag{A.193}$$

The introduction of these transformations from virtual to real variables leads to the elimination of the variable s in the nuclear kinetic energy term (i.e. in the first term of the delta function argument in (A.190)), so that the partition function (A.190) takes the form

$$\begin{aligned} Z = \zeta \int dp_v \int dv \int dp_s \int ds \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \;\; s^{3N-2}\, v^{-1} \;\; \delta\Bigg(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \frac{\tilde{\mathbf{P}}_0^2}{2ms^2\, v^{2/3}} + \phi(\{\boldsymbol{q}_i\}, v) \\ + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s - E \Bigg) \end{aligned} \tag{A.194}$$

where the change of variables is responsible for the comparison of Jacobian terms in the integrals above, that can be justified with the following calculation

$$\begin{aligned} d\tilde{p}_v\, d\tilde{v}\, d\tilde{p}_s\, d\tilde{s} \prod_{i=1}^{N-1} d\tilde{\boldsymbol{\pi}}_i \prod_{i=1}^{N} d\tilde{\boldsymbol{q}}_i &= s\, dp_v\, dv\, dp_s\, ds \prod_{i=1}^{N-1} (s v^{1/3})^{3N-3}\, d\boldsymbol{p}_i \prod_{i=1}^{N} (v^{-1/3})^{3N} d\boldsymbol{q}_i \\ &= s^{3N-2}\, v^{-1}\, dp_v\, dv\, dp_s\, ds \prod_{i=1}^{N-1} d\boldsymbol{p}_i \end{aligned} \tag{A.195}$$

The integrand in (A.194) has a similar form to the one in the original Ferrario formulation, except for the presence of the center of mass kinetic energy term in the argument of the energy delta function, and the integration over $3N-3$ momentum variables instead of $3N$. In general, if the total linear momentum is different from zero, $\tilde{\mathbf{P}}_0 \neq \mathbf{0}$, the argument of the energy delta function has two roots with respect to the variable s, which can be called s_1 and s_2, and the function $h(s)$ in the above integral can be rewritten using the relation

$$\delta[h(s)] = \sum_{i=1}^{2} \frac{\delta(s - s_i)}{|-K_0/s^3 + g k_b T_0/s|_{s=s_i}} \qquad \text{where} \quad K_0 = \frac{\tilde{\mathbf{P}}_0^2}{m\, v^{2/3}} \tag{A.196}$$

where the denominator in the expression above is given by the derivative of the delta function argument in (A.194) with respect to the variable s. Then, using the relation (A.196), the partition function form in equation (A.194) can be written as

$$\begin{aligned} Z &= \zeta \int dp_v \int dv \int dp_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; v^{-1} \int ds \;\; s^{3N-2} \;\; \sum_{i=1}^{2} \frac{\delta(s - s_i)}{|-K_0/s^3 + g k_b T_0/s|_{s=s_i}} \\ &= \zeta \int dp_v \int dv \int dp_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; v^{-1} \;\; \sum_{i=1}^{2} \frac{s_i^{3N-2}}{|-K_0/s^3 + g k_b T_0/s|_{s=s_i}} \end{aligned} \tag{A.197}$$

However, it is easy to see in the expression above that the integrand has not a Boltzmann form related to a physical Hamiltonian, from which the constant temperature and pressure (NPT) ensemble distribution function can be derived.

On the other hand, in the case $\tilde{\mathbf{P}}_0 = \mathbf{0}$, the problem simplifies considerably, since the partition function (A.194) can now be written as

$$Z = \zeta \int dp_v \int dv \int dp_s \int ds \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \;\; s^{3N-2}\, v^{-1} \;\; \delta\Bigg(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\{\boldsymbol{q}_i\}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s - E \Bigg) \tag{A.198}$$

and the integration over the variable s can be trivially performed using the relation (A.170) for the delta function in the integrand: the zero of the function that is the energy delta argument,

$$h(s) = \sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} + g k_b T_0 \ln s - E \tag{A.199}$$

is equal to

$$h(s_0) = 0 \quad \leftrightarrow \quad s_0 = \exp\left[-\frac{1}{g k_b T_0} \left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} - E \right) \right] \tag{A.200}$$

Inserting this expression for s_0 in (A.170) and then using the relation in the partition function integrand (A.198) leads to

$$\begin{aligned}
Z &= \zeta \int dp_v \int dv \int dp_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \int ds \; s^{3N-2}\, v^{-1} \;\; \frac{s}{g k_b T_0} \; \delta(s - s_0) \\
&= \frac{\zeta}{g k_b T_0} \int dp_v \int dv \int dp_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; s_0^{3N-1}\, v^{-1} \\
&= \frac{\zeta}{g k_b T_0} \int dp_v \int dv \int dp_s \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; \frac{1}{v} \exp\left[-\frac{3N-1}{g k_b T_0} \left(\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q}, v) + \frac{p_v^2}{2W} + P_0 v + \frac{p_s^2}{2Q} - E \right) \right] \\
&= \underbrace{\frac{\zeta}{g k_b T_0} \exp\left[\left(\frac{3N-1}{g} \right) \frac{E}{k_b T_0} \right] \int dp_v \exp\left[-\left(\frac{3N-1}{g} \right) \frac{1}{k_b T_0} \frac{p_v^2}{2W} \right] \int dp_s \exp\left[-\left(\frac{3N-1}{g} \right) \frac{1}{k_b T_0} \frac{p_s^2}{2Q} \right]}_{= \text{constant}} \times \\
&\qquad \times \int dv \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; \exp\left[-\left(\frac{3N-1}{g} \right) \frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q}, v) + P_0 v}{k_b T_0} - \ln v \right]
\end{aligned}$$

where the real variable Hamiltonian has been labeled as

$$\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q}, v) = \sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q}, v) \tag{A.201}$$

The same result can be obtained starting from the more general expression (A.197) and imposing $K_0 = 0$ in the equation. Since $\tilde{\mathbf{P}}_0 = \mathbf{0}$, there is only one root s_0 for the delta function argument in the partition function (A.194), so that the summation in (A.197) reduces to a single term $i = 1$ with $s_i = s_0$, leading to the same result as with the derivation above. Finally, if $g = 3N - 1$ is chosen, the partition function assumes the form

$$\begin{aligned}
g = 3N - 1 \quad : \quad Z = \frac{\zeta}{g k_b T_0} \exp\left(\frac{E}{k_b T_0} \right) \int dp_v \exp\left(-\frac{1}{k_b T_0} \frac{p_v^2}{2W} \right) \int dp_s \exp\left(-\frac{1}{k_b T_0} \frac{p_s^2}{2Q} \right) \times \\
\times \int dv \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \; \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p}, \boldsymbol{q}, v) + P_0 v}{k_b T_0} \right) - \ln v \right]
\end{aligned} \tag{A.202}$$

where the integral with respect to p_v and p_s variables are simple Gaussian integrals that can be easily performed, so that the partition function is given by

$$\begin{aligned} g = 3N-1 \quad : \quad Z = & \frac{\zeta}{g k_b T_0} \exp\left(\frac{E}{k_b T_0}\right) (2\pi\, W\, k_b T_0)^{1/2} (2\pi\, Q\, k_b T_0)^{1/2} \times \\ & \times \int dv \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int \prod_{i=1}^{N} d\boldsymbol{q}_i \ \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p},\boldsymbol{q},v) + P_0 v}{k_b T_0}\right) - \ln v\right] \end{aligned} \tag{A.203}$$

resulting in the final form

$$g = 3N-1 \quad : \quad Z = \underbrace{\frac{2\pi\zeta}{g} \exp\left(\frac{E}{k_b T_0}\right) \sqrt{WQ}}_{= \text{ constant } C} \int dv \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int d\boldsymbol{q} \ \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p},\boldsymbol{q},v) + P_0 v}{k_b T_0}\right) - \ln v\right]$$

where the variable of integration given by the product of N position coordinates has been simplified using the notation $d\boldsymbol{q} = d\boldsymbol{q}_1\, d\boldsymbol{q}_2 \cdots d\boldsymbol{q}_N$. Therefore, grouping all the constant factors in front of the relevant integral, and labeling these constant factors C, the partition function finally results equal to

$$g = 3N-1 \quad : \quad Z = C \int dv \int \prod_{i=1}^{N-1} d\boldsymbol{p}_i \int d\boldsymbol{q} \ \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p},\boldsymbol{q},v) + P_0 v}{k_b T_0} + \ln v\right)\right] \tag{A.204}$$

The correspondent equilibrium distribution function is

$$g = 3N-1 \quad : \qquad f(\boldsymbol{p},\boldsymbol{q},v) = \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p},\boldsymbol{q},v) + P_0 v}{k_b T_0} + \ln v\right)\right] \tag{A.205}$$

where the Hamiltonian function is given by (A.201), so that the explicit form of the distribution function is given by the expression

$$g = 3N-1 \quad : \qquad f(\boldsymbol{p},\boldsymbol{q},v) = \exp\left[-\frac{1}{k_b T_0}\left(\sum_{i=1}^{N-1} \frac{\boldsymbol{p}_i^2}{2\mu_i} + \phi(\boldsymbol{q},v) + P_0 v\right) - \ln v\right] \tag{A.206}$$

The partition function and the equilibrium distribution function obtained in this way does not correspond to a isothermal-isobaric ensemble of the $(N-1)$ particle system with mass spectrum $\{\mu_i\}$, as instead obtained for the canonical ensemble generated by the Nosé-Hoover equations of motion with linear momentum conservation, derived in Appendix A.7.1. Indeed, the distribution function (A.205) differs from the correct isothermal-isobaric equilibrium distribution function, whose form is given by equation (A.176), for the term $\ln v$ in the argument of the exponential, that can be written as a factor $1/v$ in front of the correct isothermal-isobaric distribution function, that is

$$g = 3N-1 \quad : \qquad f(\boldsymbol{p},\boldsymbol{q},v) = \frac{1}{v} \exp\left[-\left(\frac{\mathcal{H}_0''(\boldsymbol{p},\boldsymbol{q},v) + P_0 v}{k_b T_0}\right)\right] \tag{A.207}$$

This additional term $1/v$ derives from the Jacobian computed in (A.195), and used for the change of variable from the virtual to the real set of coordinates. Anyway, the equilibrium distribution function (A.206) and the partition function (A.204) can be used to compute ensemble averages that corresponds, assuming the ergodicity hypothesis, with time averages over the molecular dynamics simulation trajectory, performing a sampling data points at integer multiples of the virtual time unit $\Delta\tilde{t}$. In order to obtain a partition function for a sampling in real time unit Δt, starting from the expression for the partition function (A.198), the same procedure reported in Appendix A, Section A.11, can be followed. The partition function that leads to a distribution function of the form in equation (A.206), but that corresponds to a real time sampling, has obviously the same form as the previous derived partition function (A.204), with the number of degrees of freedom set equal to $g = 3N-2$. Therefore, a value of $g = 3N-2$ has to be used in practical simulations.

A.8.2 Solution of integral (5.569) with respect to variable p_v

Starting from the partition function (5.569), that is reported here below,

$$Z(N,C_1,C_2) = \frac{\zeta}{C_2}\sqrt{\frac{Q}{2}}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dp_v\int dv\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}\times \\ \times\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0v-gk_bT_0\ln\left(\frac{C_2}{\Pi\,v^{1/d}}\right)\right]^{-1/2} \tag{A.208}$$

the following substitution can be performed

$$p_v=\sqrt{2W}\,a\sin\theta \qquad\qquad \text{with} \qquad -\frac{\pi}{2}\le\theta\le\frac{\pi}{2} \tag{A.209}$$

where

$$a^2 = C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-P_0v-gk_bT_0\ln\left(\frac{C_2}{\Pi\,v^{1/d}}\right) \tag{A.210}$$

Differentiating both members of equation (A.209) leads to

$$dp_v=\sqrt{2W}\,a\cos\theta\,d\theta \tag{A.211}$$

Using the definition (A.209) and (A.210), together with the equation (A.211) in the partition function form (A.208), it becomes

$$\begin{aligned} Z(N,C_1,C_2) &= \frac{\zeta}{C_2}\sqrt{\frac{Q}{2}}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dv\int_{-\pi/2}^{\pi/2}d\theta\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}\frac{\sqrt{2W}\,a\cos\theta}{\sqrt{a^2-a^2\sin^2\theta}} \\ &= \frac{\zeta}{C_2}\sqrt{QW}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dv\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2}\int_{-\pi/2}^{\pi/2}d\theta\;\frac{\cos\theta}{\sqrt{1-\sin^2\theta}} \end{aligned}$$

The last integral in the previous equation can be easily solved with the substitution

$$t=\sin\theta \qquad dt=\cos\theta\,d\theta \qquad \cos\theta=\sqrt{1-t^2} \qquad d\theta=\frac{dt}{\sqrt{1-t^2}} \tag{A.212}$$

so that the integral can be rewritten as

$$\int_{-\pi/2}^{\pi/2}d\theta\;\frac{\cos\theta}{\sqrt{1-\sin^2\theta}}=\int_{-1}^{1}\frac{dt}{\sqrt{1-t^2}}\frac{\sqrt{1-t^2}}{\sqrt{1-t^2}}=\int_{-1}^{1}\frac{1}{\sqrt{1-t^2}}\,dt=\arcsin(t)|_{t=-1}^{t=1}=\frac{\pi}{2}-\left(-\frac{\pi}{2}\right)=\pi$$

Therefore, the integral with respect to the variable p_v is given by

$$\int dp_v\;\left[C_1-\mathcal{H}_0(\boldsymbol{\pi},\Pi,\boldsymbol{\rho})-\frac{p_v^2}{2W}-P_0v-gk_bT_0\ln\left(\frac{C_2}{\Pi\,v^{1/d}}\right)\right]^{-1/2}=\pi\sqrt{2W} \tag{A.213}$$

and the partition function (A.208) becomes

$$Z(N,P_0,T,C_1,C_2)=\frac{\pi\zeta}{C_2}\sqrt{QW}\int d\boldsymbol{\pi}\int d\boldsymbol{\rho}\int d\Pi\int dv\;\frac{1}{v^{(N-1)+1/d}}\left(\frac{C_2}{\Pi}\right)^{(N-1)d+2} \tag{A.214}$$

A.8.3 Conserved quantities

Consider the Ferrario equations of motion (5.506) - (5.511), that are reported here below

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{vd}\frac{dv}{dt} = \frac{\boldsymbol{p}_i}{m_i} + \frac{\boldsymbol{q}_i}{vd}\frac{p_v}{W} \qquad i = 1, ..., N \tag{A.215}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \frac{\boldsymbol{p}_i}{s}\frac{ds}{dt} - \frac{\boldsymbol{p}_i}{vd}\frac{vd}{dt} = -\frac{\partial\phi}{\partial\boldsymbol{q}_i} - \boldsymbol{p}_i\frac{p_s}{Q} - \frac{\boldsymbol{p}_i}{vd}\frac{p_v}{W} \qquad i = 1, ..., N \tag{A.216}$$

$$\frac{dv}{dt} = \frac{p_v}{W} \tag{A.217}$$

$$\frac{dp_v}{dt} = \frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 - \frac{p_v p_s}{Q} \equiv F_v - \frac{p_v p_s}{Q} \tag{A.218}$$

$$\frac{d\eta}{dt} = \frac{p_s}{Q} \tag{A.219}$$

$$\frac{dp_s}{dt} = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \equiv F_s \tag{A.220}$$

where the two force variables used in equations (A.218) and (A.220) are given by

$$F_v \equiv \frac{1}{vd}\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{m_i} - \frac{\partial\phi}{\partial\boldsymbol{q}_i}\cdot\boldsymbol{q}_i\right) - \frac{\partial\phi}{\partial v} - P_0 \qquad F_s \equiv \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i} + \frac{p_v^2}{W} - gk_bT_0 \tag{A.221}$$

where all the positions and momenta in (A.215) - (A.220) and (A.221) are d dimensional vectors.
The aim of this section is to analyze some quantities, that are extensions of the conservation of total linear and angular momentum which are conserved for the Hamiltonian flow generated by the simple Hamiltonian equations of motion, and to perform calculations in order to understand if these quantities can be constant of motion for the non Hamiltonian flow generated by the Ferrario equations of motion (A.215) - (A.220). These quantities are

$$\mathbf{P}\, e^{\eta} \tag{A.222}$$

$$\mathbf{P}\, v^{1/d}\, e^{\eta} \tag{A.223}$$

$$\mathbf{L}\, e^{\eta} \tag{A.224}$$

$$\mathbf{L}\, v^{1/d}\, e^{\eta} \tag{A.225}$$

As it can be observed, all these quantities are extensions of the expression for the total linear and angular momentum (which are conserved for the Hamiltonian flow generated by the simple Hamiltonian equations of motion), with the inclusion of thermostat and barostat degrees of freedom via the variables η and v, respectively. The calculation is simple: the total time derivative of each quantity is taken, the equations of motion (A.215) - (A.220) are used to find a final expression for this derivative, and finally a condition which involves the total force acting on the system or similar is search so that the computed time derivative is zero. Obviously, this condition has to be related to some symmetry property of the system.
The total time derivatives of quantities (A.223) and (A.224) have been already computed in equations (5.518) and (5.524). These two derivations permit to conclude that (A.223) and (A.224) are conserved quantities for the non Hamiltonian flow generated by the equations of motion (A.215) - (A.220), iff the conditions (5.515) and (5.521) are satisfied, respectively. In the following, the calculations of the total derivatives of the quantities (A.222) and (A.225) permit to conclude that they are not constant of motion for the Ferrario time evolution equations (A.215) - (A.220), i.e. it is not possible to find a condition related to a symmetry property of the physical system for which the total time derivatives of the two quantities (A.222) and (A.225) are equal to zero.
Total time derivative of quantity (A.222):

$$\frac{d}{dt}(\mathbf{P}\, e^{\eta}) = \dot{\mathbf{P}}\, e^{\eta} + \mathbf{P}\, e^{\eta}\, \dot{\eta}$$

$$
\begin{aligned}
&= e^{\eta}\left(\sum_{i=1}^{N}\frac{d\boldsymbol{p}_i}{dt}+\frac{p_s}{Q}\,\mathbf{P}\right)\\
&= e^{\eta}\left[\sum_{i=1}^{N}\left(-\frac{\partial\phi}{\partial\boldsymbol{q}_i}-\boldsymbol{p}_i\,\frac{p_s}{Q}-\frac{\boldsymbol{p}_i}{vd}\,\frac{p_v}{W}\right)+\frac{p_s}{Q}\,\mathbf{P}\right]\\
&= e^{\eta}\left(\sum_{i=1}^{N}\mathbf{F}_i-\frac{p_s}{Q}\sum_{i=1}^{N}\boldsymbol{p}_i-\frac{1}{vd}\,\frac{p_v}{W}\sum_{i=1}^{N}\boldsymbol{p}_i+\frac{p_s}{Q}\,\mathbf{P}\right)\\
&= e^{\eta}\left(\sum_{i=1}^{N}\mathbf{F}_i-\frac{p_s}{Q}\,\mathbf{P}+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P}+\frac{p_s}{Q}\,\mathbf{P}\right)=e^{\eta}\left(\sum_{i=1}^{N}\mathbf{F}_i+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{P}\right)
\end{aligned}
\tag{A.226}
$$

Total time derivative of quantity (A.223): see derivation (5.518)

Total time derivative of quantity (A.224): see derivation (5.524)

Total time derivative of quantity (A.225):

$$
\begin{aligned}
&\frac{d}{dt}(\mathbf{L}\,v^{1/d}\,e^{\eta})=\dot{\mathbf{L}}\,v^{1/d}\,e^{\eta}+\frac{1}{d}\,\mathbf{L}\,v^{1/d-1}\,e^{\eta}\,\dot{v}+\mathbf{L}\,v^{1/d}\,e^{\eta}\,\dot{\eta}\\
&= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\left(\frac{d\boldsymbol{q}_i}{dt}\wedge\boldsymbol{p}_i+\boldsymbol{q}_i\wedge\frac{d\boldsymbol{p}_i}{dt}\right)+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{L}+\frac{p_s}{Q}\,\mathbf{L}\right]\\
&= e^{\eta}\,v^{1/d}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i}{m_i}\wedge\boldsymbol{p}_i+\frac{p_v}{vdW}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i-\boldsymbol{q}_i\wedge\frac{\partial\phi}{\partial\boldsymbol{q}_i}-\frac{p_s}{Q}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i-\frac{p_v}{vdW}\,\boldsymbol{q}_i\wedge\boldsymbol{p}_i\right)+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{L}+\frac{p_s}{Q}\,\mathbf{L}\right]\\
&= e^{\eta}\,v^{1/d}\left(-\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\frac{\partial\phi}{\partial\boldsymbol{q}_i}-\frac{p_s}{Q}\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\boldsymbol{p}_i+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{L}+\frac{p_s}{Q}\,\mathbf{L}\right)\\
&= e^{\eta}\,v^{1/d}\left(\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i-\frac{p_s}{Q}\,\mathbf{L}+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{L}+\frac{p_s}{Q}\,\mathbf{L}\right)=e^{\eta}\,v^{1/d}\left(\sum_{i=1}^{N}\boldsymbol{q}_i\wedge\mathbf{F}_i+\frac{1}{vd}\,\frac{p_v}{W}\,\mathbf{L}\right)
\end{aligned}
\tag{A.227}
$$

A.9 Gaussian thermostat through extended system

In Section 5.3.1 the simple scaling technique has been introduced. The basic principle of this method is the introduction of the Langevin equation of motion (5.174) on which the non-holonomic constraint (5.165) on the instantaneous kinetic energy has been imposed. This method can produce the canonical distribution in the coordinate space if $g = 3N - 1$ is set, for a system of N particles moving in a three dimensional system ($d = 3$). In the following, the previous assessment will be demonstrated in detail starting from the extended system method as introduced by S. Nosé.[46]
The distribution function in momentum space is usually simple and the contribution of this term can be easily calculated in the canonical ensemble. Therefore, any method that produces the canonical distribution, even if only in coordinate space, can be useful in some situations. The standard way for this approach is to constrain the total instantaneous kinetic energy term

$$\frac{1}{2}\sum_{i=1}^{N} m_i \boldsymbol{v}_i^2(t) = \frac{1}{2}\sum_{i=1}^{N} m_i \left(\frac{d\boldsymbol{q}_i}{dt}\right)^2 = \frac{g}{2} k_b T_0 \tag{A.228}$$

where k_b is the Boltzmann constant, T_0 the temperature of the heat bath and the parameter g is essentially equal to the number of degrees of freedom of the physical system (however, its exact value will be chosen to satisfy the canonical distribution of configurations exactly at equilibrium). Fluctuations of the total kinetic energy are suppressed by imposing constraint (A.228). An additional degree of freedom s is introduced, that acts as an external system on the physical system of N particles, together with the *virtual* variables $\tilde{\boldsymbol{q}}_i$, $\tilde{\boldsymbol{p}}_i$, $\tilde{s}$ and $\tilde{t}$ which are related to the *real* variables $\boldsymbol{q}_i$, $\boldsymbol{p}_i$ and t by

$$\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i \qquad \boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i / s \qquad s = \tilde{s} \qquad \tilde{t} = \int^t \frac{dt}{s} \tag{A.229}$$

The Hamiltonian of the extended system of the particles and the variable $\tilde{s} = s$ in terms of the virtual variables is postulated as

$$\tilde{\mathcal{H}} = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) + \frac{\tilde{p}_s^2}{2Q} + g k_b T_0 \ln\tilde{s} \tag{A.230}$$

where $\tilde{p}_s$ is the conjugate momentum of $\tilde{s} = s$ and Q is a parameter of dimension energy$\cdot$(time)^2 that behaves as a mass for the motion of $\tilde{s} = s$. The Hamiltonian (A.230) is constrained by the conditions

$$\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}} = -\frac{1}{s}\left[\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i\tilde{s}^2} - g k_b T_0\right] = 0 \tag{A.231}$$

$$\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{p}_s} = \frac{\tilde{p}_s}{Q} = 0 \tag{A.232}$$

Equation (A.232) is trivial, and the term $\tilde{p}_s^2/2Q$ will be ignored hereafter. The equations of motion for $\tilde{\boldsymbol{q}}_i$ and $\tilde{\boldsymbol{p}}_i$ have the form

$$\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{p}}_i} + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}}\frac{\partial\tilde{s}}{\partial\tilde{\boldsymbol{p}}_i} = \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{p}}_i} = \frac{\tilde{\boldsymbol{p}}_i}{m_i\tilde{s}^2} \tag{A.233}$$

$$\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} = -\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{q}}_i} - \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}}\frac{\partial\tilde{s}}{\partial\tilde{\boldsymbol{q}}_i} = -\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{q}}_i} = -\frac{\partial\phi(\tilde{\boldsymbol{q}})}{\partial\tilde{\boldsymbol{q}}_i} \tag{A.234}$$

but the value of $\tilde{s} = s$ must be determined from the constraint (A.231), that is

$$\tilde{s} = s = \left[\frac{1}{g k_b T_0}\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i}\right]^{1/2} \equiv \tilde{s}_0 = s_0 \tag{A.235}$$

The Hamiltonian (A.230) is conserved by the previous defined equations of motion, indeed

$$\frac{d\tilde{\mathcal{H}}}{d\tilde{t}} = \sum_{i=1}^{N}\left(\frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{p}}_i}\frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{\boldsymbol{q}}_i}\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}}\right) + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{p}_s}\frac{d\tilde{p}_s}{d\tilde{t}} + \frac{\partial\tilde{\mathcal{H}}}{\partial\tilde{s}}\frac{d\tilde{s}}{d\tilde{t}} = 0 \tag{A.236}$$

The partition function in this case is equal to

$$Z = \zeta \int d\tilde{s} \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}}\ \delta(\mathcal{H} - E)\delta(\tilde{s} - \tilde{s}_0) \tag{A.237}$$

$$= \zeta \int d\tilde{s} \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}}\ \delta[\mathcal{H}_0(\tilde{\boldsymbol{p}}/\tilde{s}, \tilde{\boldsymbol{q}}) + gk_bT_0 \ln\tilde{s} - E]\, \delta(\tilde{s} - \tilde{s}_0) \tag{A.238}$$

where $\zeta = 1/[N!\ (2\pi\hbar)^{3N}]$ is a normalization constant for the partition function, the second delta function represents the constraint (A.231) for the value of the variable $\tilde{s} = s$, while $\mathcal{H}_0(\tilde{\boldsymbol{p}}/\tilde{s}, \tilde{\boldsymbol{q}})$ corresponds to the first two terms in the complete Hamiltonian (A.230) and s_0 is the value of s defined by the constraint in (A.235), namely,

$$\mathcal{H}_0(\tilde{\boldsymbol{p}}/\tilde{s}, \tilde{\boldsymbol{q}}) = \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{2m_i\tilde{s}^2} + \phi(\tilde{\boldsymbol{q}}) \qquad \text{and} \qquad s_0 = \left[\frac{1}{gk_bT_0}\sum_{i=1}^{N}\frac{\tilde{\boldsymbol{p}}_i^2}{m_i}\right]^{1/2} \tag{A.239}$$

Taking into account that $\tilde{s} = s$, the partition function (A.237) can rewritten slightly simplifying the notations as

$$Z = \zeta \int ds \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}}\ \delta[\mathcal{H}_0(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) + gk_bT_0 \ln s - E]\, \delta(s - s_0) \tag{A.240}$$

The last delta function in (A.240) can be manipulated using the relation

$$\delta(s - s_0) = \left|h'(s)\right| \delta[h(s)] \tag{A.241}$$

where s_0 is the root of the function $h(s)$ and $h'(s)$ is the derivative of the function $h(s)$ with respect to s variable. It is easy to find a function $h(s)$ whose only root is s_0 with expression given by equation (A.235), that is

$$\delta(s - s_0) = sgk_bT_0\ \delta\left(\frac{1}{2}\, s^2 gk_bT_0 - \sum_{i=1}^{N}\frac{\tilde{\boldsymbol{p}}_i^2}{2m_i}\right) \tag{A.242}$$

The partition function (A.237) then becomes

$$Z = \zeta \int ds \int d\tilde{\boldsymbol{p}} \int d\tilde{\boldsymbol{q}}\ \delta[\mathcal{H}_0(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) + gk_bT_0 \ln s - E]\, sgk_bT_0\ \delta\left(\frac{1}{2}\, s^2 gk_bT_0 - \sum_{i=1}^{N}\frac{\tilde{\boldsymbol{p}}_i^2}{2m_i}\right) \tag{A.243}$$

With the transformations $\tilde{\boldsymbol{q}}_i = \boldsymbol{q}_i$ and $\tilde{\boldsymbol{p}}_i = s\boldsymbol{p}_i$ from the virtual to the real coordinates, the previous expression becomes

$$Z = \zeta \int ds \int s^{3N}\, d\boldsymbol{p} \int d\boldsymbol{q}\ \delta[\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + gk_bT_0 \ln s - E]\, sgk_bT_0\ \frac{1}{s^2}\ \delta\left(\frac{1}{2}\, gk_bT_0 - \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}\right) \tag{A.244}$$

where for the last delta function has been used the relation $\delta(ax) = \delta(x)/|a|$ to divide its argument in (A.243) by the factor s^2 and perform the transformation between the virtual and real coordinates. Simplifying the equation above, re-introducing the complete expression for $\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q})$ given by (A.239) and applying the last delta function on the argument of the first delta function in the integral leads to

$$\begin{aligned}
Z &= \zeta \int ds \int s^{3N-1}\, d\boldsymbol{p} \int d\boldsymbol{q}\ \delta[\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q}) + gk_bT_0 \ln s - E]\, gk_bT_0\ \delta\left(\frac{1}{2}\, gk_bT_0 - \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}\right) \\
&= \zeta \int ds \int s^{3N-1}\, d\boldsymbol{p} \int d\boldsymbol{q}\ \delta\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) + gk_bT_0 \ln s - E\right) gk_bT_0\ \delta\left(\frac{1}{2}\, gk_bT_0 - \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}\right) \\
&= \zeta \int d\boldsymbol{p}\ \delta\left(\frac{1}{2}\, gk_bT_0 - \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i}\right) \int d\boldsymbol{q} \int s^{3N-1}\, gk_bT_0\ \delta\left(\frac{1}{2}\, gk_bT_0 + \phi(\boldsymbol{q}) + gk_bT_0 \ln s - E\right)
\end{aligned}$$

The argument $h(s)$ of the last delta function has a root s_0 given by

$$h(s_0) = 0 \quad \rightarrow \quad s_0 = \exp\left[-\frac{1}{g}\left(\frac{g k_b T_0}{2} + \phi(\boldsymbol{q}) - E\right)\frac{1}{k_b T_0}\right] \tag{A.245}$$

and the derivative of the function with respect to s variable is equal to

$$h'(s) = g k_b T_0 / s \tag{A.246}$$

Therefore, using the inverse relation of (A.241), namely,

$$\delta[h(s)] = \frac{\delta(s - s_0)}{|h'(s)|} \tag{A.247}$$

on the last delta function in the previous expression for the partition function, it becomes

$$\begin{aligned}
Z &= \zeta \int d\boldsymbol{p}\ \delta\left(\frac{1}{2}\, g k_b T_0 - \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right) \int d\boldsymbol{q} \int s^{3N-1}\, g k_b T_0\ \frac{s}{g k_b T_0}\ \delta(s - s_0) \\
&= \zeta \int d\boldsymbol{p}\ \delta\left(\frac{1}{2}\, g k_b T_0 - \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right) \int d\boldsymbol{q}\ s_0^{3N} \\
&= \zeta \int d\boldsymbol{p}\ \delta\left(\frac{1}{2}\, g k_b T_0 - \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i}\right) \int d\boldsymbol{q}\ \exp\left[-\frac{3N}{g}\left(\frac{g k_b T_0}{2} + \phi(\boldsymbol{q}) - E\right)\frac{1}{k_b T_0}\right] \\
&= \zeta \exp\left[-\frac{3N}{g}\left(\frac{g}{2} - \frac{E}{k_b T_0}\right)\right] \int d\boldsymbol{p}\ \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{1}{2}\, g k_b T_0\right) \int d\boldsymbol{q}\ \exp\left[-\frac{3N}{g}\left(\frac{\phi(\boldsymbol{q})}{k_b T_0}\right)\right]
\end{aligned}$$

Therefore, the partition function is given by the equation

$$Z = \zeta \exp\left[-\frac{3N}{g}\left(\frac{g}{2} - \frac{E}{k_b T_0}\right)\right] \int d\boldsymbol{p}\ \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{1}{2}\, g k_b T_0\right) \int d\boldsymbol{q}\ \exp\left[-\frac{3N}{g}\left(\frac{\phi(\boldsymbol{q})}{k_b T_0}\right)\right] \tag{A.248}$$

where the exponent outside the integrals is a constant factor. If $g = 3N$, the partition function reported above becomes

$$Z = \zeta \exp\left[-\left(\frac{3N}{2} - \frac{E}{k_b T_0}\right)\right] \int d\boldsymbol{p}\ \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{1}{2}\, g k_b T_0\right) \int d\boldsymbol{q}\ \exp\left[-\left(\frac{\phi(\boldsymbol{q})}{k_b T_0}\right)\right] \tag{A.249}$$

which corresponds to the equilibrium distribution function

$$f(\boldsymbol{q}, \boldsymbol{p}) = \delta\left(\sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} - \frac{1}{2}\, g k_b T_0\right) \exp\left[-\left(\frac{\phi(\boldsymbol{q})}{k_b T_0}\right)\right] \tag{A.250}$$

Therefore, the equations of motion (A.233) and (A.234) with the constraint (A.235) produce the canonical distribution in coordinate space.

A.9.1 Equations of motion in real variables

The transformation laws between real and virtual variables, given by (A.229), can be used to write the equations of motion (A.233) and (A.234) for the real physical variables $\boldsymbol{q}_i$, $\boldsymbol{p}_i$ and t as follows

$$\frac{d\boldsymbol{q}_i}{dt} = s\frac{d\tilde{\boldsymbol{q}}_i}{d\tilde{t}} = \frac{\tilde{\boldsymbol{p}}_i}{m_i s} = \frac{\boldsymbol{p}_i}{m_i} \tag{A.251}$$

$$\frac{d\boldsymbol{p}_i}{dt} = s\frac{d\boldsymbol{p}_i}{d\tilde{t}} = s\frac{d}{d\tilde{t}}\left(\frac{\tilde{\boldsymbol{p}}_i}{s}\right) = -\frac{\partial\phi(\boldsymbol{q})}{\partial\boldsymbol{q}_i} - \frac{ds}{d\tilde{t}}\,\boldsymbol{p}_i \tag{A.252}$$

The derivative $ds/d\tilde{t}$ is obtained via the differentiation of equation (A.231), that is

$$\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i s^2} - g k_b T_0 = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i} - s^2 g k_b T_0 = 0 \tag{A.253}$$

$$\frac{d}{d\tilde{t}} \left[\sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i^2}{m_i} - s^2 g k_b T_0 \right] = 0 \quad \rightarrow \quad \sum_{i=1}^{N} \frac{\tilde{\boldsymbol{p}}_i}{m_i} \frac{d\tilde{\boldsymbol{p}}_i}{d\tilde{t}} = g k_b T_0 s \, \frac{ds}{d\tilde{t}} \tag{A.254}$$

and using the transformations $\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i$ and $\boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i / s$ and the equation of motion (A.234), the time derivative of the variable s is given by

$$\frac{ds}{d\tilde{t}} = \frac{1}{g k_b T_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \left(-\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \right) \tag{A.255}$$

Finally, substituting this expression in (A.252), the equations of motion in real physical variables become

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \tag{A.256}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} + \left[\frac{1}{g k_b T_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i}{m_i} \left(\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \right) \right] \cdot \boldsymbol{p}_i = \mathbf{F}_i(\boldsymbol{q}) - \left[\frac{1}{g k_b T_0} \sum_{i=1}^{N} \boldsymbol{v}_i \cdot \mathbf{F}_i(\boldsymbol{q}) \right] \cdot \boldsymbol{p}_i \tag{A.257}$$

where in the last equivalence the derivative of the potential with respect to the i-th nuclear positions has been identified with minus the force acting on the i-th nucleus, i.e.

$$\mathbf{F}_i(\boldsymbol{q}) = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} \tag{A.258}$$

Rewriting the equation of motion (A.257) for the velocities instead of the momenta $\boldsymbol{p}_i = m_i \boldsymbol{v}_i$, and recovering the fact that the positions $\boldsymbol{q}_i \equiv \boldsymbol{q}_i(t)$ and the velocities $\boldsymbol{v}_i \equiv \boldsymbol{v}_i(t)$ are time-dependent functions, the equation of motion (A.257) can be written as

$$m_i \frac{d\boldsymbol{v}_i(t)}{dt} = \mathbf{F}_i(\boldsymbol{q}(t)) - \frac{m_i}{g k_b T_0} \left[\sum_{i=1}^{N} \boldsymbol{v}_i(t) \cdot \mathbf{F}_i(\boldsymbol{q}(t)) \right] \cdot \boldsymbol{v}_i(t) \tag{A.259}$$

which perfectly coincides with the equation of motion (5.177) introduced for the simple scaling velocity thermostat in Section 5.3.2.

A.10 Integration of the equations of motion

The Liouville approach introduced in Section 4.2 permits to obtain in a simple way the integrator associated to a given equations of motion, which can be used in a computational program to propagate in time the positions and momenta of the nuclei in a condensed matter system during a molecular dynamics simulation. As usual, the first step is to write the Hamilton equations of motion associated to our ensemble, in this case equations (5.171) and (5.172), which are here rewritten for completeness

$$\frac{d\boldsymbol{q}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i} \tag{A.260}$$

$$\frac{d\boldsymbol{p}_i}{dt} = -\frac{\partial \phi(\boldsymbol{q})}{\partial \boldsymbol{q}_i} - \gamma \boldsymbol{p}_i = \mathbf{F}_i(\boldsymbol{q}(t)) - \gamma \boldsymbol{p}_i \qquad i = 1, ..., N \tag{A.261}$$

Then, the total Liouville operator is written, following equation (4.62), as

$$\begin{aligned} i\mathcal{L} = \dot{\boldsymbol{\Gamma}} \cdot \frac{\partial}{\partial \boldsymbol{\Gamma}} &= (\dot{\boldsymbol{q}}, \dot{\boldsymbol{p}}) \cdot \left(\frac{\partial}{\partial \boldsymbol{q}}, \frac{\partial}{\partial \boldsymbol{p}} \right) = \sum_{\alpha=1}^{Nd} \left[\dot{q}_\alpha \frac{\partial}{\partial q_\alpha} + \dot{p}_\alpha \frac{\partial}{\partial p_\alpha} \right] \\ &= \sum_{i=1}^{N} \left[\dot{\boldsymbol{q}}_i \cdot \frac{\partial}{\partial \boldsymbol{q}_i} + \dot{\boldsymbol{p}}_i \cdot \frac{\partial}{\partial \boldsymbol{p}_i} \right] \equiv \sum_{i=1}^{N} \left[\dot{\boldsymbol{q}}_i \cdot \nabla_{\boldsymbol{q}_i} + \dot{\boldsymbol{p}}_i \cdot \nabla_{\boldsymbol{p}_i} \right] \end{aligned}$$

where d is the dimensionality of the system. Substituting the expressions for the Hamilton equations of motion (A.260) and (A.261) in the equation above, the Liouville operator for the case of Gaussian thermostat equations of motion is given by

$$i\mathcal{L} = \sum_{i=1}^{N} \left[\frac{\boldsymbol{p}_i(t)}{m_i} \cdot \nabla_{\boldsymbol{q}_i} + [\mathbf{F}_i(\boldsymbol{q}(t)) - \gamma \boldsymbol{p}_i(t)] \cdot \nabla_{\boldsymbol{p}_i} \right] \tag{A.262}$$

which can be recalled as the sum of two contributions

$$i\mathcal{L} = i\mathcal{L}_1 + i\mathcal{L}_2 \tag{A.263}$$

where

$$i\mathcal{L}_1 = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(t)}{m_i} \cdot \nabla_{\boldsymbol{q}_i} \qquad \text{and} \qquad i\mathcal{L}_2 = \sum_{i=1}^{N} [\mathbf{F}_i(\boldsymbol{q}(t)) - \gamma \boldsymbol{p}_i(t)] \cdot \nabla_{\boldsymbol{p}_i} \tag{A.264}$$

The approximate evolution of the system ruled by the Gaussian thermostat equations of motion over a time step Δt is obtained by acting with a Suzuki-Trotter factorized operator with the form

$$e^{i\mathcal{L}\Delta t} \approx e^{i\mathcal{L}_2 \Delta t/2} e^{i\mathcal{L}_1 \Delta t} e^{i\mathcal{L}_2 \Delta t/2} \tag{A.265}$$

on an initial condition $\boldsymbol{q}(0), \boldsymbol{p}(0)$. The action of each of the operators in this factorization can be evaluated analytically. First of all, the action of the operator $\exp(i\mathcal{L}_2 \Delta t/2)$ refers to the evolution of the momenta, hence it propagates the positions $\boldsymbol{p}_i(t)$. The equations of motion (A.261), which are here reported as

$$\dot{\boldsymbol{p}}_i(t) = \mathbf{F}_i(\boldsymbol{q}(t)) - \gamma \boldsymbol{p}_i \qquad\qquad i = 1, ..., N \tag{A.266}$$

or equivalently, using a more simple notation,

$$\dot{p}_\alpha(t) = F_\alpha(\boldsymbol{q}(t)) - \gamma p_\alpha(t) \qquad\qquad \alpha = 1, ..., Nd \tag{A.267}$$

(where d is the number of degrees of freedom) can be used to solve analytically the evolution of the momenta $\boldsymbol{p}_i(t)$ for $\Delta t/2$ by solving the coupled first order differential equations (with $\boldsymbol{q}_i(t)$ and $\mathbf{F}_i(\boldsymbol{q}(t))$ held fixed at their value at time t). Considering the expression (5.176) for the coefficient γ, the momentum propagation is given by

$$\dot{p}_\alpha(t) = F_\alpha(\boldsymbol{q}(t)) - \left[\frac{1}{g k_b T_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(t)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t)) \right] p_\alpha(t) = F_\alpha(\boldsymbol{q}(t)) - \dot{h}(t) p_\alpha(t) \qquad \alpha = 1, ..., Nd \tag{A.268}$$

where the nuclear positions $\boldsymbol{q} = (q_1, ..., q_\alpha)$, $\alpha = 1, ..., Nd$ are held fixed.
Defining a function

$$y(t) = \frac{y(0) + a \int_0^t e^{h(t')}\, dt'}{e^{h(t)}} \tag{A.269}$$

its time derivative is

$$\dot{y}(t) = \frac{a e^{h(t)}\, e^{h(t)} - [y(0) + a \int_0^t e^{h(t')}\, dt'] e^{h(t)} \dot{h}(t)}{(e^{h(t)})^2} = a - y(t)\dot{h}(t) \tag{A.270}$$

where a is a constant value with respect to time. Hence, the function form (A.269) generates a derivative with the same form given by (A.268), so that it can be assumed to has the right form for the solution of the differential equations (A.268). From the comparison between (A.270) and the equations of motion (A.268) the constant a and the function $h(t)$ can be identified as

$$a = F_\alpha(\boldsymbol{q}(t)) \qquad\qquad \dot{h}(t) = \frac{1}{g k_b T_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(t)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t)) \tag{A.271}$$

Hence the solution to (A.268) can be written in the form (A.269) as

$$p_\alpha(t) = \frac{p_\alpha(0) + F_\alpha(\boldsymbol{q}(t)) \int_0^t e^{h(t')}\, dt'}{e^{h(t)}} \tag{A.272}$$

Furthermore, defining

$$\dot{s}(t) = e^{h(t)} \qquad \rightarrow \qquad s(t) = \int_0^t e^{h(t')}\, dt' \tag{A.273}$$

the equation (A.272) can be rewritten more simply as

$$p_\alpha(t) = \frac{p_\alpha(0) + F_\alpha(\boldsymbol{q}(t))s(t)}{\dot{s}(t)} \tag{A.274}$$

Further deriving the function $\dot{s}(t)$ one gets

$$\ddot{s}(t) = \frac{d}{dt}(e^{h(t)}) = e^{h(t)}\, \dot{h}(t) = \dot{s}(t)\, \dot{h}(t) \tag{A.275}$$

and substituting the expression for the derivative of $h(t)$ given by (A.271) the previous equation becomes

$$\ddot{s}(t) = \frac{d}{dt}(e^{h(t)}) = \left[\frac{1}{gk_bT_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(t)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t))\right] \dot{s}(t) = \left[\frac{1}{gk_bT_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(t)\dot{s}(t)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t))\right] \tag{A.276}$$

Since equation (A.274) can be written as

$$\boldsymbol{p}_i(t)\dot{s}(t) = \boldsymbol{p}_i(0) + \mathbf{F}_i(\boldsymbol{q}(t))s(t) \tag{A.277}$$

the second derivative $\ddot{s}(t)$ in (A.276) results equal to

$$\ddot{s}(t) = \frac{d}{dt}(e^{h(t)}) = \frac{1}{gk_bT_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(0)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t)) + \frac{s(t)}{gk_bT_0} \sum_{i=1}^{N} \frac{1}{m_i} [\mathbf{F}_i(\boldsymbol{q}(t)) \cdot \mathbf{F}_i(\boldsymbol{q}(t))] \tag{A.278}$$

The second order differential equation of the form

$$\ddot{s}(t) = a + bs(t) \tag{A.279}$$

has a solution

$$s(t) = \frac{a}{b}\left(\cosh(t\sqrt{b}) - 1\right) + \frac{1}{\sqrt{b}} \sinh(t\sqrt{b}) \tag{A.280}$$

where

$$a = \frac{1}{gk_bT_0} \sum_{i=1}^{N} \frac{\boldsymbol{p}_i(0)}{m_i} \cdot \mathbf{F}_i(\boldsymbol{q}(t)) \qquad \text{and} \qquad b = \frac{1}{gk_bT_0} \sum_{i=1}^{N} \frac{1}{m_i} [\mathbf{F}_i(\boldsymbol{q}(t)) \cdot \mathbf{F}_i(\boldsymbol{q}(t))] \tag{A.281}$$

Similarly, one can calculate exactly the expression for $\dot{s}(t)$ by deriving equation (A.280), so that the propagation in time for the nuclear momenta, given by (A.274), is completely defined. Therefore, the operator $\exp(i\mathcal{L}_2\Delta t/2)$ is applied by simply evaluating equation (A.280) and its derivative at time $t = \Delta t/2$, when it is applied on momenta coordinates, while the nuclear positions are let unaltered by the same operator, i.e.

$$e^{i\mathcal{L}_2\Delta t/2}\, \boldsymbol{p}_i(t) = \frac{\boldsymbol{p}_i(t) + \mathbf{F}_i(\boldsymbol{q}(t))s(\Delta t/2)}{\dot{s}(\Delta t/2)} \tag{A.282}$$

$$e^{i\mathcal{L}_2\Delta t/2}\, \boldsymbol{q}_i(t) = \boldsymbol{q}_i(t) \tag{A.283}$$

Then, following the operators order in Trotter expansion (A.265), the action of the operator $\exp(i\mathcal{L}_1\Delta t)$ on a state $\boldsymbol{\Gamma} = (\boldsymbol{q}, \boldsymbol{p})$ yields (see Section 4.2.1)

$$e^{i\mathcal{L}_1\Delta t}\, \boldsymbol{p}_i(t) = \boldsymbol{p}_i(t) \tag{A.284}$$

$$e^{i\mathcal{L}_1\Delta t}\, \boldsymbol{q}_i(t) = \boldsymbol{q}_i(t) + \Delta t\, \frac{\boldsymbol{p}_i(t)}{m_i} \tag{A.285}$$

so that the application of this second operator lets the momenta coordinates unvaried but produces a translation for the nuclear positions during the propagation.

The combined action of the three operators in the Trotter factorization (A.265) leads to the following reversible, kinetic energy conserving algorithm for integrating the isokinetic equations:

Step 0. Evaluate new values for : $s(\Delta t/2),\ \dot{s}(\Delta t/2)$

Step 1. Propagator $e^{i\mathcal{L}_2\Delta t/2}$: $\boldsymbol{v}_i(t) \leftarrow \dfrac{1}{\dot{s}(\Delta t/2)}\left[\boldsymbol{v}_i(t) + s(\Delta t/2)\,\dfrac{\mathbf{F}_i(\boldsymbol{q}(t))}{m_i}\right]$

Step 2. Propagator $e^{i\mathcal{L}_1\Delta t}$: $\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t) + \Delta t\,\dfrac{\boldsymbol{p}_i(t)}{m_i}$ $\quad i = 1, ..., N$

Step 3. Updating forces : $\mathbf{F}_i[\{\boldsymbol{q}_i(t)\}] \leftarrow \mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$

Step 4. Evaluate new values for : $s(\Delta t/2),\ \dot{s}(\Delta t/2)$

Step 5. Propagator $e^{i\mathcal{L}_2\Delta t/2}$: $\boldsymbol{v}_i(t+\Delta t) \leftarrow \dfrac{1}{\dot{s}(\Delta t/2)}\left[\boldsymbol{v}_i(t) + s(\Delta t/2)\,\dfrac{\mathbf{F}_i(\boldsymbol{q}(t+\Delta t))}{m_i}\right]$

Note that $s(\Delta t/2)$ and $\dot{s}(\Delta t/2)$ are evaluated by substituting the present momentum and the forces into equation (A.281) with $t = \Delta t/2$. The symbol $\leftarrow$ indicates that on the computer, the values on the left-hand side are overwritten in memory by the values on the right-hand side.

A.11 From virtual to real sampling

A.11.1 Nosé-Hoover thermostat

In the extended methods, an additional degree of freedom s is introduced which acts as an external system on the physical system of N particles in three dimensions ($d = 3$), with real coordinates $\boldsymbol{q}_i$, masses m_i ($i = 1, ..., N$) and potential energy $\phi(\boldsymbol{q})$. Virtual variables are also introduced (positions $\tilde{\boldsymbol{q}}_i$, momenta $\tilde{\boldsymbol{p}}_i$ and time $\tilde{t}$), which are related to the real variables ($\boldsymbol{q}_i$, $\boldsymbol{p}_i$ and t) by

$$\boldsymbol{q}_i = \tilde{\boldsymbol{q}}_i \qquad \boldsymbol{p}_i = \tilde{\boldsymbol{p}}_i / s \qquad s = \tilde{s} \qquad \tilde{t} = \int^t \frac{dt}{s} \tag{A.286}$$

The virtual variables define the extended system, while the real variables describe the real physical system. In the following, the ensemble average of a given observable in the extended system will be compared and related to the ensemble average of the same observable in the real physical system.
In Section 5.3.4 has been demonstrated that, with the quasi-ergodic hypothesis, which relates the time average along the trajectory to the ensemble average, the averages of any static quantities expressed as functions of $\tilde{\boldsymbol{p}}_i/s$, $\tilde{\boldsymbol{q}}_i$ along the trajectory determined by the equations of motion, are exactly those in the canonical ensemble:

$$\lim_{t_0 \to \infty} \frac{1}{t_0} \int_0^{t_0} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\, d\tilde{t} = \langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \rangle \overset{g=3N+1}{=} \langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \rangle_c \tag{A.287}$$

where $\langle ... \rangle$ and $\langle ... \rangle_c$ denote the ensemble average in the extended system and in the canonical ensemble, respectively. The first equivalence in equation (A.287) is achieved by sampling data points at integer multiples of the virtual time unit $\Delta\tilde{t}$. This sampling is called virtual time sampling. In this sampling, the virtual time interval of each time step is equal, but the real time interval of each time step is unequal. In particular, given the partition function in the canonical ensemble (A.137) and the associated equilibrium distribution function (A.138), the ensemble average (A.287) is written as

$$\langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \rangle \equiv \langle A(\boldsymbol{p}, \boldsymbol{q}) \rangle \overset{g=3N+1}{=} \langle A(\boldsymbol{p}, \boldsymbol{q}) \rangle_c = \frac{1}{Z} \left\{ C \int d\boldsymbol{p} \int d\boldsymbol{q}\ A(\boldsymbol{p}, \boldsymbol{q})\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p}, \boldsymbol{q})}{k_b T_0} \right) \right] \right\} \tag{A.288}$$

However, it will be more convenient in practical molecular dynamics simulations to sample at equal intervals in real time. In the following, the transition from virtual to real time sampling will be performed, and the formula (A.141) of Section 5.3.4 will be demonstrated.
If the sample has to be performed using equal intervals in real time t with

$$t_1 = \int_0^{t_0} \frac{d\tilde{t}}{s} \tag{A.289}$$

the result is a weighted average

$$\begin{aligned} \lim_{t_1 \to \infty} \frac{1}{t_1} \int_0^{t_1} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})\, dt &= \lim_{t_1 \to \infty} \frac{t_0}{t_1} \frac{1}{t_0} \int_0^{t_0} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \frac{d\tilde{t}}{s} = \left[\lim_{t_0 \to \infty} \frac{1}{t_0} \int_0^{t_0} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \frac{d\tilde{t}}{s} \right] \left(\lim_{t_0 \to \infty} \frac{t_1}{t_0} \right)^{-1} \\ &= \left[\lim_{t_0 \to \infty} \frac{1}{t_0} \int_0^{t_0} A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}}) \frac{d\tilde{t}}{s} \right] \left(\lim_{t_0 \to \infty} \frac{1}{t_0} \int_0^{t_0} \frac{d\tilde{t}}{s} \right)^{-1} = \frac{\langle A(\tilde{\boldsymbol{p}}/s, \tilde{\boldsymbol{q}})/s \rangle}{\langle 1/s \rangle} \end{aligned}$$

In order to compute the last average above, equation (A.135) for the partition function can be used, which is here reported for completeness

$$Z = \frac{\zeta}{g k_b T_0} \int d\tilde{p}_s \int d\boldsymbol{p} \int d\boldsymbol{q} \int ds\ \ s^{3N+1}\ \delta(s - s_0) \tag{A.290}$$

With this partition function, the ensemble average sampled in real variables becomes

$$\begin{aligned}\frac{\langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})/s\rangle}{\langle 1/s\rangle} \equiv \frac{\langle A(\boldsymbol{p},\boldsymbol{q})/s\rangle}{\langle 1/s\rangle} &= \frac{\frac{1}{Z}\frac{\zeta}{gk_bT_0}\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\int ds\ \left[A(\boldsymbol{p},\boldsymbol{q})/s\right]s^{3N+1}\ \delta(s-s_0)}{\frac{1}{Z}\frac{\zeta}{gk_bT_0}\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\int ds\ (1/s)\,s^{3N+1}\ \delta(s-s_0)} \\ &= \frac{\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\int ds\ A(\boldsymbol{p},\boldsymbol{q})\,s^{3N}\ \delta(s-s_0)}{\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\int ds\ s^{3N}\ \delta(s-s_0)} \\ &= \frac{\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\ A(\boldsymbol{p},\boldsymbol{q})\,s_0^{3N}}{\int dp_s\int d\boldsymbol{p}\int d\boldsymbol{q}\ s_0^{3N}}\end{aligned} \tag{A.291}$$

Substituting the expression for s_0, that is the zero of the function $h(s)$ as computed in equation (A.134), and reported in the following,

$$s_0 = \exp\left[-\frac{1}{gk_bT_0}\left(\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) + \frac{\tilde{p}_s^2}{2Q} - E\right)\right] \tag{A.292}$$

in the last equation in (A.291), the result is

$$\begin{aligned}&\frac{\langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})/s\rangle}{\langle 1/s\rangle} \equiv \frac{\langle A(\boldsymbol{p},\boldsymbol{q})/s\rangle}{\langle 1/s\rangle} \\ &= \frac{1}{Z}\underbrace{\frac{\zeta}{gk_bT_0}\exp\left[\left(\frac{3N}{gk_bT_0}\right)E\right]\int d\tilde{p}_s\ \exp\left[-\left(\frac{3N}{gk_bT_0}\right)\frac{\tilde{p}_s^2}{2Q}\right]}_{=\text{ constant}}\int d\boldsymbol{p}\int d\boldsymbol{q}\ A(\boldsymbol{p},\boldsymbol{q})\ \exp\left[-\left(\frac{3N}{gk_bT_0}\right)\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right]\end{aligned}$$

where the constant terms and integrals has been collected before the rest of the integral expression as above and the partition Z is given by

$$Z = \underbrace{\frac{\zeta}{gk_bT_0}\exp\left[\left(\frac{3N}{gk_bT_0}\right)E\right]\int d\tilde{p}_s\ \exp\left[-\left(\frac{3N}{gk_bT_0}\right)\frac{\tilde{p}_s^2}{2Q}\right]}_{=\text{ constant}}\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\left(\frac{3N}{gk_bT_0}\right)\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})\right] \tag{A.293}$$

Defining the constant $g = 3N$, the partition function generated by the equations of motion (5.249)-(5.252) in real time sampling becomes

$$g = 3N \quad : \qquad Z = \underbrace{\frac{\zeta}{gk_bT_0}\exp\left(\frac{E}{k_bT_0}\right)\int d\tilde{p}_s\ \exp\left[-\left(\frac{1}{k_bT_0}\right)\frac{\tilde{p}_s^2}{2Q}\right]}_{=\text{ constant } C}\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_bT_0}\right)\right]$$

and grouping all the constant factors, labeling them equal to C, the partition function in real time sampling results equal to

$$g = 3N \quad : \qquad Z = C\int d\boldsymbol{p}\int d\boldsymbol{q}\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_bT_0}\right)\right] \tag{A.294}$$

Using these results, if the number of degrees of freedom is fixed to a value $g = 3N$, the ensemble average in real time sampling (A.291) takes the same form as the ensemble average in virtual time sampling with $g = 3N + 1$, given by equation (A.11.1), i.e. a form equal to the ensemble average in the canonical ensemble, that is

$$\frac{\langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})/s\rangle}{\langle 1/s\rangle} \equiv \frac{\langle A(\boldsymbol{p},\boldsymbol{q})/s\rangle}{\langle 1/s\rangle} \overset{g=3N}{=} \langle A(\boldsymbol{p},\boldsymbol{q})\rangle_c = \frac{1}{Z}\left\{C\int d\boldsymbol{p}\int d\boldsymbol{q}\ A(\boldsymbol{p},\boldsymbol{q})\ \exp\left[-\left(\frac{\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q})}{k_bT_0}\right)\right]\right\}$$

where the Hamiltonian in the exponential argument is given by equation (A.131) as the sum of the nuclear kinetic and potential energy

$$\mathcal{H}_0(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + \phi(\boldsymbol{q}) \tag{A.295}$$

Therefore, the formulae of the ensemble averages for the virtual and real time sampling (t.s.) are

virtual t.s.
$$\lim_{t_0\to\infty}\frac{1}{t_0}\int_0^{t_0} A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})\,d\tilde{t} = \langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})\rangle \overset{g=3N+1}{=} \langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})\rangle_c \equiv \langle A(\boldsymbol{p},\boldsymbol{q})\rangle_c \tag{A.296}$$

real t.s.
$$\lim_{t_1\to\infty}\frac{1}{t_1}\int_0^{t_1} A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})\,dt = \frac{\langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})/s\rangle}{\langle 1/s\rangle} \overset{g=3N}{=} \langle A(\tilde{\boldsymbol{p}}/s,\tilde{\boldsymbol{q}})\rangle_c \equiv \langle A(\boldsymbol{p},\boldsymbol{q})\rangle_c \tag{A.297}$$

A.12 Alternative integrator for Nosé-Hoover equations of motion

Form of the time propagator operator using the Suzuki-Trotter decomposition scheme:

$$e^{i\mathcal{L}\tau} \approx e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_4\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_3\,\tau/2}\, e^{i\mathcal{L}_2\,\tau/2}\, e^{i\mathcal{L}_1\,\tau}\,. \\ \cdot\; e^{i\mathcal{L}_2\,\tau/2}\, e^{i\mathcal{L}_3\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_4\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4} + O(\tau^{11}) \tag{A.298}$$

so that a phase space point evolves following the equation

$$\mathbf{\Gamma}(t_0+\tau) \approx e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_4\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_3\,\tau/2}\, e^{i\mathcal{L}_2\,\tau/2}\, e^{i\mathcal{L}_1\,\tau}\,. \\ \cdot\; e^{i\mathcal{L}_2\,\tau/2}\, e^{i\mathcal{L}_3\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4}\, e^{i\mathcal{L}_4\,\tau/2}\, e^{i\mathcal{L}_5\,\tau/4}\; \mathbf{\Gamma}(t_0) + O(\tau^{11}) \tag{A.299}$$

For the specific forms of the Liouville operators $\{\mathcal{L}_i\}$ $(i = 1, ..., 5)$ see equations (5.283) and (5.284) in Section 5.3.4.2.

Starting point (initial conditions)	:	$\boldsymbol{q}_i(t),\ \boldsymbol{p}_i(t),\ \mathbf{F}_i[\{\boldsymbol{q}_i(t)\}],\ \eta(t),\ p_\eta(t)$
Step 1. Propagator $e^{i\mathcal{L}_5\Delta t/4}$	:	$p_\eta(t+\Delta t/4) \leftarrow p_\eta(t) + \frac{\Delta t}{4}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t)}{m_i} - gk_bT_0\right)$
Step 2. Propagator $e^{i\mathcal{L}_4\Delta t/2}$	:	$\eta(t+\Delta t/2) \leftarrow \eta(t) + \frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/4)$
Step 3. Propagator $e^{i\mathcal{L}_5\Delta t/4}$	:	$p_\eta(t+\Delta t/2) \leftarrow p_\eta(t+\Delta t/4) + \frac{\Delta t}{4}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t)}{m_i} - gk_bT_0\right)$
Step 4. Propagator $e^{i\mathcal{L}_3\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t)\exp\left[-\frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)\right]$
Step 5. Propagator $e^{i\mathcal{L}_2\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t/2) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \frac{\Delta t}{2}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t)\}]$
Step 6. Propagator $e^{i\mathcal{L}_1\Delta t}$	:	$\boldsymbol{q}_i(t+\Delta t) \leftarrow \boldsymbol{q}_i(t) + \frac{\Delta t}{m_i}\,\boldsymbol{p}_i(t+\Delta t/2)$
Step 7. Updating forces	:	compute $\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 8. Propagator $e^{i\mathcal{L}_2\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t) \leftarrow \boldsymbol{p}_i(t+\Delta t/2) + \frac{\Delta t}{2}\,\mathbf{F}_i[\{\boldsymbol{q}_i(t+\Delta t)\}]$
Step 9. Propagator $e^{i\mathcal{L}_3\Delta t/2}$	:	$\boldsymbol{p}_i(t+\Delta t) \leftarrow \boldsymbol{p}_i(t+\Delta t)\exp\left[-\frac{\Delta t}{2Q}\,p_\eta(t+\Delta t/2)\right]$
Step 10. Propagator $e^{i\mathcal{L}_5\Delta t/4}$	:	$p_\eta(t+3\Delta t/4) \leftarrow p_\eta(t+\Delta t/2) + \frac{\Delta t}{4}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t+\Delta t)}{m_i} - gk_bT_0\right)$
Step 11. Propagator $e^{i\mathcal{L}_4\Delta t/2}$	:	$\eta(t+\Delta t) \leftarrow \eta(t+\Delta t/2) + \frac{\Delta t}{2Q}\,p_\eta(t+3\Delta t/4)$
Step 12. Propagator $e^{i\mathcal{L}_5\Delta t/4}$	:	$p_\eta(t+\Delta t) \leftarrow p_\eta(t+3\Delta t/4) + \frac{\Delta t}{4}\left(\sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2(t+\Delta t)}{m_i} - gk_bT_0\right)$

Table A.1: Nosé-Hoover thermostat integrator algorithm (vverlet_nose_I0), applied $\forall\, i = 1, ..., N$.

Appendix B

Molecular Dynamics module details

B.1 Code workflow (moldyn.f90)

SUBROUTINE: readMD

comments mark	code
physical constants	a0 = par(32)·1.0e-10_float
	upres = ha/(a0^3)
	pconv = 1.0_float/(upres*1.0e-9_float)
	xconv = par(32)
	vconv = xconv·tconv
defaults	naf = inf(24)
	nfree = 3·naf - 3
	nsteps = 1000
	timestep = 0.1_float*tconv ! fs → a.u.
	temperature = 300 ! K
	nve = .true.
	thstat = 0
	nvt = .false.
	nose_integ = 1
	npt = .false.
	npt_integ = 0
	vrtprt = .false.
	cellprt = .false.
	norot = .false.
	guessp = .true.
	mdrest = .false.
	mdfixind = .false.
	openfiles = .true.
	comprt = .false.
	timingprt = .false.
	initvelprt = .false.
	velandstop = .false.
	xyzunits = .false.
	debug = .false.
	computePcf = .false.
	analysisMD = .false.
default for fragment	inf(129) = inf(24) ! number of active atoms: all
	inf129 = inf(129)
	fragment = .false.
internal check	timedef = .false.
read the input file	read the input file, cases for the different keywords
	if a keyword is not find in the case statements : an error is printed in the output file and the code stops
	for the list of keywords: see Manual in Section E.1

SUBROUTINE: md

comment mark	code
write header	call headmd : write in output the header of the Moldyn module
open i/o files	if(openfiles) : open files with MD trajectory info if a restart is required, the files are opened as : fort.221, fort.188, fort.189, fort.190 and fort.191 : opened with status old, because they have to be read fort.222, fort.192, fort.193, fort.186 and fort.187 : opened in append mode
which ensemble	print in output the ensemble of the simulation, the conserved quantity and the integrator used for the equations of motion, on the base of the input keyword and logical variables initialized in subroutine readMD
initialize	call initialize : initialize positions, velocities and energies
if mdfixind = true	if fixind strategy used, inf(15) = 2
if guessp = true	if guess the density matrix, inf(57) = 2
integrate the eom	loop on the number of steps with call to the subroutine move (call move) for the integration of the equations of motion, calculation of kinetic energy, temperature and update of trajectories data files
finalize	call finalMD : after the end of MD simulation a post processing analysis of the trajectory can be performed (pair correlation function, frequency, self-diffusion coefficient)
print final timings	print the final time (call timvrs) in output file and close the main output file unit (ioutmd)

SUBROUTINE: headmd

comments mark	code
write conversion units	the header of the MD module is written in output file, with conversion factors to atomic units explicitly given and written in a table on the output file

SUBROUTINE: initialize

comments mark	code
checks on geometry	check on the input system geometry that should be P_1 (inf(2) = 1) : if inf(2) /= 1 then call symmremo_moldyn for removal of the symmetry in direct and reciprocal space
allocation begin	allocate: vector mass(naf), matrix vel(3,naf), matrix forces(3,naf)
if fragment (0)	if fragment, allocate matrices xaallat(3,naf), velallat(3,naf) and forcesallat(3,naf)
masses initialization	initialization of (mass(i), i = 1,naf) vector with the masses of each atom calculation of total mass of the system (totmass)

if norot	if NOROT keyword is used (system rotations not allowed):
	if(inf(10).eq.0) nfree = nfree - 3 (for 0D systems: $g \leftarrow g - 3$)
	if(inf(10).eq.1) nfree = nfree - 1 (for 1D systems: $g \leftarrow g - 1$)

p1 printings	print in the output file : naf (number of atoms), nfree (number of degrees of freedom) and totmass
scratch values	initialize to zero : timepre = 0._float, nstepre = 0
initialize virtual degrees	initialization to zero of virtual variables (degrees of freedom) for NVT and NPT ensembles
initialize with restarting	if(mdrest) initialize with restarting : reading of the trajectory files for restarting positions, velocities,

begin –	time and eventually virtual degrees of freedom
	call readEn, call readPos, call readVel
	if(vrtprt) call readVirt
	if(cellprt) call readCell

if fragment (1)	if(fragment) call reset_velfa :
end –	if fragment, reset to zero the velocities of the nuclei not included in the fragment
step index and time	istep = nstepre and time = timepre

p2 printings	print in the output file :
	timestep, number of steps, initial temperature,
	if(thstat.eq.2) i.e. if Berendsen thermostat, rise time of thermostat
	if(nvt) or if(npt) thermostat info from input file

if(npt) cicp	if(npt) call parlat(paret,tmpinv,mdcell) [see Chapter F, Section F.4] : initialization of mdcell(1:7) vector with initial cell parameters and volume of the unit cell

p3 printings	print in the output file :
	initial virtual degrees of freedom, initial nuclear positions
	if(fragment) print info on the atoms in the fragment (moving atoms)
	if(.not.mdrest) write in the output info about random Gaussian initialization of the nuclear velocities

initialize from scratch	if(.not.mdrest) Box-Muller algorithm for nuclear velocities initialization (see paragraph B.1.1)
begin –	vel(j,i) = uran·sqrt(k_bT/mass(i)) where uran is the random number Gaussian-distributed
if fragment (3)	if(fragment) call reset_velfa : if fragment, reset to zero the velocities of the nuclei not included in the fragment
if initvelprt 0	if(initvelprt) write atomic masses and random nuclear velocities on file fort.193 (call printmat)
velocities translated	write in output file that the nuclear velocities are translated with respect to the center of mass motion call vel_tranremo : the nuclear velocities are translated with respect to the center of mass motion
if initvelprt 1	if(initvelprt) write the velocity of the center of mass and write the translated nuclear velocities on file fort.193 (call printmat)
remove system rotation	if(norot) :
begin –	write in output file that system rotation components are removed from the nuclear velocities call vel_rotremo : the system rotation components are removed from the nuclear velocities (this option is valid only for 0D and 1D systems)
if initvelprt 2	if(initvelprt) write translated nuclear velocities on file fort.193 (call printmat)
end –	
scwrttt	compute kinetic energy and temperature (compute_kinene(andtemp), see paragraph B.1.2), then scale the initial nuclear velocities with respect to the target temperature (see paragraph B.1.2)
if initvelprt 3	if(initvelprt) write the temperature computed from translated nuclear velocities and the scaling factor on file fort.193
end –	
p4 printings	print in the output file initial nuclear velocities (both for restarted simulation and from scratch)
if initvelprt 4	if(initvelprt) write the initial nuclear velocities on file fort.193 (call printmat)
compute the kinetic	computation of the kinetic energy and temperature (call compute_kinene(andtemp), see paragraph B.1.2)
p5 printings	print in the output file : initial temperature

if testing	if(velandstop) : write in the output file a warning with testvel directive, call md_closefiles : close files, call md_deallocate : deallocate vectors and matrices, call stoppp : write end time elapsed, write termination date and stop (exit the code)
time for first SCF begin	call cpu_time(tbeg_firstscf) : time at the beginning of the first SCF cycle
compute energy and forces	call moldyn_scf_driver : compute initial energy and forces (first main call to CRYSTAL)
time for first SCF end	call cpu_time(tend_firstscf) : time at the end of the first SCF cycle
cctfs	cpu_time_init = tend_firstscf - tbeg_firstscf : compute CPU time for the first SCF cycle
com motion	if(comprt) call compute_com : calculation of position com(1:3), velocity vcom(1:3), momentum sump(1:3) of system center of mass
ccq compute conserved	call compute_conserved : compute conserved quantity (ensemble dependent)
initialize trajectory files	if((.not.mdrest).and.openfiles), i.e. if not restart and if openfiles :
begin –	call update_traj_real : initialize trajectory files with energies, positions, velocities and timings if(vrtprt) call update_traj_vrt : if virtual variables are used, initialize file with virtual variables
end –	if(iameq0) call extprt('fort.34') : create (or overwrite) fort.34 file with geometry data settings
p6 printings	redirect SCF output on file SCFOUT.LOG, write on output that the redirection is done, write on output the name of the file created during the MD simulation, write on output that the MD simulation is started and print the first line with data (energies, temperature, timings) of the output file MD table

SUBROUTINE: symmremo_moldyn

comments mark	code
delete the use of symmetry	removal of symmetry in direct and reciprocal spaces

SUBROUTINE: readEn

comments mark	code
analyze file en	read the file fort.221 (ENERGIES.DAT, see Section B.3) and save relevant data (nstepre and timepre): rewind the file read the first row (header of the file) read the MD step 0 (initial energies, temperature and time) read the file until EOF save the number of previous steps (nstepre) and the number of previous time (timepre) write in output file that the total time of previous MD run is restarted from file fort.221 compute the number of lines (nline) expected in the file broadcast nstepre and timepre

SUBROUTINE: readPos

comments mark	code
analyze file pos	read the file fort.188 (POSITIONS.DAT, see Section B.3) and save relevant data (last MD step positions): rewind the file read the first naf+1 rows (naf = number of atoms) read naf rows with positions at MD step 0 (initial positions) read nstepre·naf rows with positions at each MD step save the nuclear positions of the last MD step in xa(1:3,1:naf) matrix write in output file that the positions are restarted from last MD run from file fort.188 broadcast xa(3,naf) matrix

SUBROUTINE: readVel

comments mark	code
analyze file vel	read the file fort.189 (VELOCITIES.DAT, see Section B.3) and save relevant data (last MD step velocities): rewind the file read the first naf+1 rows (naf = number of atoms) read naf rows with velocities at MD step 0 (initial velocities) read nstepre·naf rows with velocities at each MD step save the nuclear velocities of the last MD step in vel(1:3,1:naf) matrix write in output file that the velocities are restarted from last MD run from file fort.189 broadcast vel(3,naf) matrix

SUBROUTINE: readVirt

comments mark	code
analyze file virt	
begin –	if(nvt) then read file fort.190 (VIRTUAL.DAT, see Section B.3) and save relevant data (last MD step virtual DOF): rewind the file read the first 10 rows (header with data) read the first row with initial virtual DOF at MD step 0 (initial virtual degrees of freedom) read nstepre rows with virtual degrees of freedom at each MD step save the virtual degrees of freedom of the last MD step (p_nose, zeta_nose) write in output file that the virtual DOF are restarted from last MD run from file fort.190 write in output file the value of the virtual DOF of the last MD run saved from file fort.190
end –	broadcast p_nose and zeta_nose
begin –	if(npt) then read file fort.190 (VIRTUAL.DAT, see Section B.3) and save relevant data (last MD step virtual DOF): rewind the file read the first 18 rows (header with data) read the first row with initial virtual DOF at MD step 0 (initial virtual degrees of freedom) read nstepre rows with virtual degrees of freedom at each MD step save the virtual degrees of freedom of the last MD step (p_nose, zeta_nose, p_barost, mdcell(7) = volume) write in output file that the virtual DOF are restarted from last MD run from file fort.190 write in output file the value of the virtual DOF of the last MD run saved from file fort.190
end –	broadcast p_nose, zeta_nose, p_barost, mdcell(7)

SUBROUTINE: vel_tranremo

comments mark	code
if fragment	if(fragment) call removefrozat(.false.,.true.,.false.,.true.) : remove from vel(3,naf) matrix the velocities of fixed atoms remove from mass(naf) vector the masses of fixed atoms
subtract com velocity	subtraction of the center of mass velocity from each nuclear velocity (the components of center of mass velocity are subtracted to the correspondent components of the nuclear velocities) $P_j = \sum_{i=1}^{N_F} m_i\, v_{ji} \qquad v_{ji} \leftarrow v_{ji} - \frac{P_j}{M} \qquad \text{where} \qquad j = x,y,z \qquad M = \sum_{i=1}^{N} m_i$ (B.1) with N_F = number of atoms free to move
if fragment	if(fragment) call insertfrozat(.false.,.false.,.true.,.true.,.false.,.true.) : insert in vel(3,inf(129)) matrix the velocities of fixed atoms insert in mass(inf(129)) vector the masses of fixed atoms

SUBROUTINE: vel_rotremo

comments mark	code

inf(10) = 0
begin –
	if(inf(10).eq.0) (if the system is an isolated molecule 0D)
	write in the output file that the option NOROT has been selected
	allocate matrices : xa_com(3,naf) (x^c_{ji}), vel_com(3,naf) (v^c_{ji})
	if(fragment) allocate matrices : xa_comall(3,naf), vel_comall(3,naf)
	compute position of the center of mass:
	$p^c_j = \frac{1}{M}\sum_{i=1}^{N} m_i\, x_{ji} \qquad M = \sum_{i=1}^{N} m_i \qquad j = x,y,z$ (B.2)
	compute nuclear positions with respect to the center of mass:
	$x^c_{ji} = x_{ji} - p^c_j \qquad j = x,y,z \qquad i = 1,...,N$ (B.3)
	if(fragment) call removefrozat(.false.,.true.,.false.,.true.)
	compute velocity of the center of mass:
	$k^c_j = \frac{1}{M}\sum_{i=1}^{N_F} m_i\, v_{ji} \qquad M = \sum_{i=1}^{N} m_i \qquad j = x,y,z$ (B.4)
	with N_F = number of atoms free to move
	if(fragment) call removefrozat_mat(vel_com,vel_comall,natfree)
	compute nuclear velocities with respect to the center of mass:
	$v^c_{ji} = v_{ji} - k^c_j \qquad j = x,y,z \qquad i = 1,...,\text{inf}(129)$ (B.5)
	if(fragment) call insertmodfrozat_mat(vel_com,vel_comall,natfree)
	if(fragment) call insertfrozat(.false.,.false.,.true.,.false.,.false.,.true.)
	compute angular momentum angmom(1:3):
	$\mathbf{L} = \sum_{i=1}^{N} m_i\, (\boldsymbol{x}^c_i \wedge \boldsymbol{v}^c_i)$ (B.6)

compute momentum of inertia inertm(1:3,1:3):

$$\text{sqnorm} = m_i(\boldsymbol{x}_i^c \cdot \boldsymbol{x}_i^c) = m_i[(x_{1i}^c)^2 + (x_{2i}^c)^2 + (x_{3i}^c)^2] \tag{B.7}$$

$$I_{11} = \sum_{i=1}^{N}\{m_i\,[(x_{1i}^c)^2 + (x_{2i}^c)^2 + (x_{3i}^c)^2] - m_i\,(x_{1i}^c)^2\} = \sum_{i=1}^{N} m_i\,[(x_{2i}^c)^2 + (x_{3i}^c)^2] \tag{B.8}$$

$$I_{12} = -\sum_{i=1}^{N}[m_i\,(x_{1i}^c)(x_{2i}^c)] \tag{B.9}$$

$$I_{13} = -\sum_{i=1}^{N}[m_i\,(x_{1i}^c)(x_{3i}^c)] \tag{B.10}$$

$$I_{21} = -\sum_{i=1}^{N}[m_i\,(x_{2i}^c)(x_{1i}^c)] \tag{B.11}$$

$$I_{22} = \sum_{i=1}^{N}\{m_i\,[(x_{1i}^c)^2 + (x_{2i}^c)^2 + (x_{3i}^c)^2] - m_i\,(x_{2i}^c)^2\} = \sum_{i=1}^{N} m_i\,[(x_{1i}^c)^2 + (x_{3i}^c)^2] \tag{B.12}$$

$$I_{23} = -\sum_{i=1}^{N}[m_i\,(x_{2i}^c)(x_{3i}^c)] \tag{B.13}$$

$$I_{31} = -\sum_{i=1}^{N}[m_i\,(x_{3i}^c)(x_{1i}^c)] \tag{B.14}$$

$$I_{32} = -\sum_{i=1}^{N}[m_i\,(x_{3i}^c)(x_{2i}^c)] \tag{B.15}$$

$$I_{33} = \sum_{i=1}^{N}\{m_i\,[(x_{1i}^c)^2 + (x_{2i}^c)^2 + (x_{3i}^c)^2] - m_i\,(x_{3i}^c)^2\} = \sum_{i=1}^{N} m_i\,[(x_{1i}^c)^2 + (x_{2i}^c)^2] \tag{B.16}$$

call minv(inertm,3,inertmod) : compute the inverse of angular momentum matrix $\mathbf{I}^{-1}$
compute angular velocities angv(1:3):

$$\omega_1 = I_{11}^{-1}L_1 + I_{12}^{-1}L_2 + I_{13}^{-1}L_3 \tag{B.17}$$

$$\omega_2 = I_{21}^{-1}L_1 + I_{22}^{-1}L_2 + I_{23}^{-1}L_3 \tag{B.18}$$

$$\omega_3 = I_{31}^{-1}L_1 + I_{32}^{-1}L_2 + I_{33}^{-1}L_3 \tag{B.19}$$

write on output file that nuclear velocities are shifted with respect to system angular velocity components
if(fragment) :
call removefrozat(.false.,.true.,.false.,.false.) and call removefrozat_mat(xa_com,xa_comall,natfree)
remove system angular velocity components from nuclear velocities:

$$\text{tmp}(1:3) = \boldsymbol{t} = \boldsymbol{\omega} \wedge \boldsymbol{x}_i^c \qquad i = 1, ..., \text{inf}(129) \tag{B.20}$$

if(initvelprt) write the nuclear tangential rotational velocity tmp(1:3) per each atom on file fort.193

$$v_{ji} \leftarrow v_{ji} - t_j = v_{ji} - (\boldsymbol{\omega} \wedge \boldsymbol{x}_i^c) \cdot \mathbf{e}_j \qquad j = x, y, z \qquad i = 1, ..., \text{inf}(129) \tag{B.21}$$

if(fragment) :
call insertfrozat(.false.,.false.,.true.,.true.,.false.,.false.) and call insertfrozat_mat(xa_com,xa_comall)
recompute the angular momentum with the new velocities v_{ji}

$$\mathbf{L} = \sum_{i=1}^{N} m_i\,(\boldsymbol{x}_i^c \wedge \boldsymbol{v}_i) \tag{B.22}$$

if(fragment) deallocate matrices : vel_comall(3,naf), xa_comall(3,naf)
end – deallocate matrices : vel_com(3,naf) (v_{ji}^c), xa_com(3,naf) (x_{ji}^c)

inf(10) = 1
begin –
if(inf(10).eq.1) (if the system is a polymer 1D)
write in the output file that the option NOROT has been selected
allocate matrices : xa_com(3,naf) (x^c_{ji}), vel_com(3,naf) (v^c_{ji})
if(fragment) allocate matrices : xa_comall(3,naf), vel_comall(3,naf)
compute position of the center of mass:

$$p^c_x = 0 \qquad \text{and} \qquad p^c_j = \frac{1}{M}\sum_{i=1}^{N} m_i\, x_{ji} \qquad M = \sum_{i=1}^{N} m_i \qquad j = y, z \tag{B.23}$$

compute nuclear positions with respect to the center of mass:

$$x^c_{xi} \equiv x^c_{1i} = 0 \qquad \text{and} \qquad x^c_{ji} = x_{ji} - p^c_j \qquad j = y, z \qquad i = 1, ..., N \tag{B.24}$$

if(fragment) call removefrozat(.false.,.true.,.false.,.true.)
compute velocity of the center of mass:

$$k^c_x = 0 \qquad \text{and} \qquad k^c_j = \frac{1}{M}\sum_{i=1}^{N_F} m_i\, v_{ji} \qquad M = \sum_{i=1}^{N} m_i \qquad j = y, z \tag{B.25}$$

with N_F = number of atoms free to move
if(fragment) call removefrozat_mat(vel_com,vel_comall,natfree)
compute nuclear velocities with respect to the center of mass:

$$v^c_{xi} \equiv v^c_{1i} = 0 \qquad \text{and} \qquad v^c_{ji} = v_{ji} - k^c_j \qquad j = y, z \qquad i = 1, ..., \inf(129) \tag{B.26}$$

if(fragment) call insertmodfrozat_mat(vel_com,vel_comall,natfree)
if(fragment) call insertfrozat(.false.,.false.,.true.,.false.,.false.,.true.)
compute angular momentum angmom(1:3):

$$L_1 \equiv L_x = \sum_{i=1}^{N} m_i\,(x^c_{2i} v^c_{3i} - x^c_{3i} v^c_{2i}) \qquad L_2 \equiv L_y = 0 \qquad L_3 \equiv L_z = 0 \tag{B.27}$$

compute momentum of inertia inertmod:

$$\text{sqnorm} = m_i[(x^c_{2i})^2 + (x^c_{3i})^2] \tag{B.28}$$

$$I_{11} \equiv I_{xx} = \sum_{i=1}^{N} m_i\,[(x^c_{2i})^2 + (x^c_{3i})^2] \tag{B.29}$$

compute angular velocities angv(1:3):

$$\omega_1 \equiv \omega_x = L_1/I_{11} \equiv L_x/I_{xx} \qquad \omega_2 \equiv \omega_y = 0 \qquad \omega_3 \equiv \omega_z = 0 \tag{B.30}$$

write on output file that nuclear velocities are shifted with respect to system angular velocity x component
if(fragment) :
call insertfrozat(.false.,.false.,.true.,.true.,.false.,.false.) and call insertfrozat_mat(xa_com,xa_comall)
remove system angular velocity x component from nuclear velocities:

$$\text{tmp}(1:3) = \boldsymbol{t} = \boldsymbol{\omega} \wedge \boldsymbol{x}^c_i \qquad i = 1, ..., \inf(129) \tag{B.31}$$

if(initvelprt) write the nuclear tangential rotational velocity tmp(1:3) per each atom on file fort.193

$$v_{ji} \leftarrow v_{ji} - t_j = v_{ji} - (\boldsymbol{\omega} \wedge \boldsymbol{x}^c_i) \cdot \mathbf{e}_j \qquad j = y, z \qquad i = 1, ..., \inf(129) \tag{B.32}$$

if(fragment) :
call insertfrozat(.false.,.true.,.true.,.false.,.false.) and call insertfrozat_mat(xa_com,xa_comall,natfree)

recompute the angular momentum with the new velocities v_{ji}

$$L_1 \equiv L_x = \sum_{i=1}^{N} m_i \left(x_{2i}^c v_{3i} - x_{3i}^c v_{2i}\right) \qquad L_2 \equiv L_y = 0 \qquad L_3 \equiv L_z = 0 \tag{B.33}$$

end – deallocation of x_{ji}^c and v_{ji}^c matrices

SUBROUTINE: compute_com

comments mark	code
cpv_com_linmom	$\boldsymbol{p}^c \equiv$ com(1:3) = 0._float (global variable) $\boldsymbol{v}^c \equiv$ vcom(1:3) = 0._float (global variable) $\boldsymbol{P} \equiv$ sump(1:3) = 0._float (global variable) compute center of mass position: $\boldsymbol{p}^c = \frac{1}{M}\sum_{i=1}^{N} m_i \boldsymbol{x}_i \qquad M = \sum_{i=1}^{N} m_i$ (B.34) compute center of mass velocity: $\boldsymbol{v}^c = \frac{1}{M}\sum_{i=1}^{N} m_i \boldsymbol{v}_i \qquad M = \sum_{i=1}^{N} m_i$ (B.35) compute system total linear momentum: $\boldsymbol{P} = \sum_{i=1}^{N} m_i \boldsymbol{v}_i$ (B.36) compute kinetic energy of the center of mass $E_k^c = \frac{1}{2} M (\boldsymbol{v}^c)^2 = \frac{1}{2} M [(v_1^c)^2 + (v_2^c)^2 + (v_3^c)^2]$ (B.37)

SUBROUTINE: compute_conserved

comments mark	code
computed conserved	compute conserved quantity (ensemble dependent) :
begin –	if(nve) conserved = kinetic + energy ($C = E_k + E_p$)

end –	if((thstat.eq.1).or.(thstat.eq.2)) conserved = kinetic ($C = E_k$)
	else if(nvt) conserved = kinetic + energy + kins + pots ($C = E_k + E_p + p_n^2/(2Q) + gk_bT_0\eta$)
	else if(npt.and.(npt_integ.eq.0)) then
	conserved = kinetic + energy + kins + pots + kinv + potv
	($C = E_k + E_p + p_n^2/(2Q) + gk_bT_0\eta + p_v^2/(2W) + P_0 v$)
	endif

SUBROUTINE: update_traj_real

comments mark	code
pyes logical variable	pyes = (istep.eq.0)
UE Update energies	file: fort.221
	if(pyes) write file header (1 row)
	update trajectory at each step :
	step - - time - - E_k - - E_p - - $E_t = E_k + E_p$ - - conserved - - T
UP Update positions	file: fort.188
	if(pyes) write file header ($N+1$ rows) :
	1r: number of atoms
	Nr: atom index, atomic number, mass [atomic unit]
	write nuclear positions at each step [Angstrom]: ((xa(j,i)*par(32), j=1,3), i=1,naf) :
	$x(i)$ - - $y(i)$ - - $z(i)$ $\quad i = 1, ..., N$ (N lines per step)
UV Update velocities	file: fort.189
	if(pyes) write file header ($N+1$ rows) :
	1r: number of atoms
	Nr: atom index, atomic number, mass [atomic unit]
	write nuclear velocities at each step [Angstrom/fs]: ((vel(j,i)*par(32)*tconv, j=1,3), i=1,naf) :
	$v_x(i)$ - - $v_y(i)$ - - $v_z(i)$ $\quad i = 1, ..., N$ (N lines per step)
UT Update timings	if(timingprt)
	file: fort.222
	if(pyes) write file header (1 row)
	update trajectory at each step :
	step - - time - - T_{CPU} - - N_{cycles}
UXYZ Update	if(xyzunits)
	file: fort.186
	update positions block at each step :
	1r: number of atoms
	1r: "STEP =" - - step - - "TIME [fs] =" - - time - - "E [HA] =" - - conserved
	Nr: i-th atomic symbol - - $x(i)$ - - $y(i)$ - - $z(i)$ $\quad$ [Angstrom] ($i = 1, ..., N$)
	file: fort.187
	update velocities block at each step :
	1r: number of atoms
	1r: "STEP =" - - step - - "TIME [fs] =" - - time - - "E [HA] =" - - conserved
	Nr: i-th atomic symbol - - $v_x(i)$ - - $v_y(i)$ - - $v_z(i)$ $\quad$ [Angstrom/fs] ($i = 1, ..., N$)
UMDC Update cell	if(cellprt)
	file: fort.191
	if(pyes) write file header (8 rows)
	write cell parameters info at each step :
	1r: "STEP =" - - step - - "TIME [fs] =" - - time - - "E [HA] =" - - conserved
	3r: update lattice parameters : (paret(i,j)*par(32), j=1,3)
	1r: update lattice parameters a, b, c [Angstrom] : (mdcell(i)*par(32), i=1,3)
	1r: update lattice parameters α, β, γ [degrees] : (mdcell(i), i=4,6)
	1r: update primitive cell volume [Angstrom^3] : mdcell(7)*(par(32)**3)
UCOM Update	if(comprt)
	file: fort.192
	if(pyes) write file header (3 rows) :
	1r: string header
	1r: total mass of the system [Hartree a.u. (me)]
	1r: string header
	update center of mass info at each step :
	step - - p_x^c - - p_y^c - - p_z^c - - P_x - - P_y - - P_z - - E_k^c

SUBROUTINE: update_traj_vrt

comments mark	code
pyes virt logical variable	pyes = (istep.eq.0)
begin –	if(nvt)
	if(pyes) write file header (10 rows)
	update trajectory at each step :
end –	step - - time - - p_nose (p_s) - - kins - - zeta_nose (η) - - pots
begin –	if(npt)
	if(pyes) write file header (18 rows)
	update trajectory at each step :
end –	step - - p_nose (p_n) - - kins - - zeta_nose (η) - - pots - - p_barost - - kinv - - mdcell(7) - - potv

SUBROUTINE: printmat(mat,row,col,conv,outfile)

if conversion factor	if(present(conv)) write(outfile,100) i, nat(i), symbat(nat(i)), (mat(j,i)·conv,j=1,row) with i=1,...,ncol
else if	else write(outfile,100) i, nat(i), symbat(nat(i)), (mat(j,i),j=1,row) with i=1,...,ncol

SUBROUTINE: extprt(filename)

see subroutine extprt(filename) in libxf_com.f :
rewind and update (overwrite) file fort.34 with system geometry setting and details

SUBROUTINE: move

comments mark	code
compute md time	time = timepre + (istep-nstepre)·timestep/tconv : compute MD time at each step $t = t_0 + (n - n_0)\Delta t$
time for a MD step begin	if(iameq0) call cpu_time(tbeg_move) : time at the beginning of a MD step

eom int	integration of equations of motion :
	if(nve) call vverlet_NVE
	if(thstat.eq.1) call vverlet_NVE + call velrescale
	if(thstat.eq.2) call vverlet_NVE + call berendsen
	if(nvt) and if(nose_integ.eq.0) call vverlet_nose_I0
	if(nvt) and if(nose_integ.eq.1) call vverlet_nose_I1
	if(npt) and if(npt_integ.eq.0) call vverlet_npt_I0

if fragment	if(fragment) call reset_velfa : if fragment, reset to zero the velocities of the nuclei not included in the fragment
calculate kt	calculate kinetic energy and temperature (call compute_kinene(andtemp), see paragraph B.1.2)
compute cq	call compute_conserved : compute conserved quantity (ensemble dependent)
time for a MD step end	if(iameq0) call cpu_time(tend_move) : time at the end of a MD step
cctms	cpuSteptime = tend_move - tbeg_move : compute CPU time for a MD step
com motion update	if(comprt) call compute_com : calculation of position com(1:3), velocity vcom(1:3), momentum sump(1:3) of system center of mass
update out file	update output file MD table with energies, temperature and timings
update traj files	if(openfiles) :
begin –	call update_traj_real : update trajectory files with energies, positions, velocities and timings if(vrtprt) call update_traj_vrt : if virtual variables are used, update file with virtual variables
end –	if(iameq0) call extprt('fort.34') : overwrite fort.34 file with geometry data settings

SUBROUTINE: vverlet_NVE

comments mark	code
fragment atom remove 0	if(fragment) call removefrozat(.true.,.true.,.true.,.true.)
new nuclear positions	calculation of nuclear positions at time $t + \Delta t$:
	$v_{ji} \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i}$ $\quad i = 1, ..., \inf(129)$ $\quad j = x, y, z$ (B.38)
	$x_{ji} \leftarrow x_{ji} + \Delta t\, v_{ji}$ $\quad i = 1, ..., \inf(129)$ $\quad j = x, y, z$ (B.39)
fragment atom insert 0	if(fragment) call insertfrozat(.true.,.true.,.true.,.true.,.true.,.true.)
cfe calculation	call moldyn_scf_driver : calculation of new forces F_{ji} (forces(3:naf)) and potential energy E_p (energy)
fragment atom remove 1	if(fragment) call removefrozat(.false.,.true.,.true.,.true.)
new nuclear velocities	calculation of nuclear velocities at time $t + \Delta t$:
	$v_{ji} \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i}$ $\quad i = 1, ..., \inf(129)$ $\quad j = x, y, z$ (B.40)
fragment atom insert 1	if(fragment) call insertfrozat(.false.,.false.,.true.,.true.,.true.,.true.)

SUBROUTINE: simplevel_scale

comments mark	code
avoid flying ice cube	call vel_tranremo : remove center of mass linear velocity from nuclear velocities (remove translations) if((inf(10).eq.0).or.(inf(10).eq.1)) call vel_rotremo : remove center of mass angular velocity from nuclear velocities (remove rotations)
compute temperature	compute kinetic energy and temperature (call compute_kinene(andtemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.41)
	$T = \frac{2}{gk_b}\, E_k$ (B.42)
simple scaling factor	compute scaling factor
	$\lambda = \sqrt{\frac{T_0}{T}}$ with T_0 = target temperature (B.43)
scale nuclear velocities	scale nuclear velocities with respect to the computed scaling factor
	$v_{ji} \leftarrow \lambda v_{ji} \qquad j = x, y, z \qquad i = 1, ..., N$ (B.44)

SUBROUTINE: berendsen_scale

comments mark	code
avoid flying ice cube	call vel_tranremo : remove center of mass linear velocity from nuclear velocities (remove translations) if((inf(10).eq.0).or.(inf(10).eq.1)) call vel_rotremo : remove center of mass angular velocity from nuclear velocities (remove rotations)
compute temperature	compute kinetic energy and temperature (call compute_kinene(andtemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.45)
	$T = \frac{2}{gk_b}\, E_k$ (B.46)
berendsen scaling factor	compute Berendsen scaling factor
	$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_0}{T} - 1\right)}$ with T_0 = target temperature and τ = rise time (B.47)
scale nuclear velocities	scale nuclear velocities with respect to the computed scaling factor
	$v_{ji} \leftarrow \lambda v_{ji} \qquad j = x, y, z \qquad i = 1, ..., N$ (B.48)

SUBROUTINE: vverlet_nose_I0

comments mark	code
compute kinetic energy	compute kinetic energy (call compute_kinene(notemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.49)
$F_n(dt/2)$	$F_n(dt/2) = 2E_k - gk_bT_0$
eta$(dt/2)$ and $p_n(dt/2)$	update η and p_n
	$p_n \leftarrow p_n + \frac{\Delta t}{4} F_n(dt/2)$ (B.50)
	$\eta(dt/2) \leftarrow \eta + \frac{\Delta t}{2}\frac{p_n}{Q}$ (B.51)
	$p_n(dt/2) \leftarrow p_n + \frac{\Delta t}{4} F_n(dt/2)$ (B.52)
$s_n(dt/2)$	$s_n(dt/2) = \exp[-p_n\, \Delta t/(2Q)]$
fragment atom remove	if(fragment) call removefrozat(.true.,.true.,.true.,.true.)
$v(dt/2)$ and $q(dt)$	update velocities and positions
	$v_{ji} \leftarrow v_{ji} \cdot s_n(dt/2) \qquad i = 1, ..., \text{inf}(129)$ and $j = x, y, z$ (B.53)
	$v_{ji}(dt/2) \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i} \qquad i = 1, ..., \text{inf}(129)$ and $j = x, y, z$ (B.54)
	$x_{ji}(dt) \leftarrow x_{ji} + v_{ji}\, \Delta t \qquad i = 1, ..., \text{inf}(129)$ and $j = x, y, z$ (B.55)
fragment atom insert	if(fragment) call insertfrozat(.true.,.true.,.true.,.true.,.true.,.true.)
scf	call moldyn_scf_driver
fragment atom remove	if(fragment) call removefrozat(.false.,.true.,.true.,.true.)
$v(dt)$	update velocities
	$v_{ji} \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i} \qquad i = 1, ..., \text{inf}(129)$ and $j = x, y, z$ (B.56)
	$v_{ji}(dt) \leftarrow v_{ji} \cdot s_n(dt/2) \qquad i = 1, ..., \text{inf}(129)$ and $j = x, y, z$ (B.57)
fragment atom insert	if(fragment) call insertfrozat(.false.,.false.,.true.,.true.,.true.,.true.)
compute kinetic energy	compute kinetic energy (call compute_kinene(notemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.58)
$F_n(dt)$	$F_n(dt) = 2E_k - gk_bT_0$
eta(dt) and $p_n(dt)$	update η and p_n
	$p_n \leftarrow p_n + \frac{\Delta t}{4} F_n(dt)$ (B.59)
	$\eta(dt) \leftarrow \eta + \frac{\Delta t}{2}\frac{p_n}{Q}$ (B.60)
	$p_n(dt) \leftarrow p_n + \frac{\Delta t}{4} F_n(dt)$ (B.61)

SUBROUTINE: vverlet_nose_I1

comments mark	code
compute kinetic energy	compute kinetic energy (call compute_kinene(notemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.62)
$F_n(dt/2)$	$F_n(dt/2) = 2E_k - gk_bT_0$
$p_n(dt/2)$ and eta$(dt/2)$	update p_n and η
	$p_n(dt/2) \leftarrow p_n + \frac{\Delta t}{2} F_n(dt/2)$ (B.63)
	$\eta(dt/2) \leftarrow \eta + \frac{\Delta t}{2}\frac{p_n}{Q}$ (B.64)
$s_n(dt/2)$	$s_n(dt/2) = \exp[-p_n\,\Delta t/(2Q)]$
fragment atom remove	if(fragment) call removefrozat(.true.,.true.,.true.,.true.)
$v(dt/2)$ and $q(dt)$	update velocities and positions
	$v_{ji} \leftarrow v_{ji} \cdot s_n(dt/2) \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.65)
	$v_{ji}(dt/2) \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i} \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.66)
	$x_{ji}(dt) \leftarrow x_{ji} + v_{ji}\,\Delta t \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.67)
fragment atom insert	if(fragment) call insertfrozat(.true.,.true.,.true.,.true.,.true.,.true.)
scf	call moldyn_scf_driver
fragment atom remove	if(fragment) call removefrozat(.false.,.true.,.true.,.true.)
$v(dt)$	update velocities
	$v_{ji} \leftarrow v_{ji} + \frac{\Delta t}{2}\frac{F_{ji}}{m_i} \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.68)
	$v_{ji}(dt) \leftarrow v_{ji} \cdot s_n(dt/2) \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.69)
fragment atom insert	if(fragment) call insertfrozat(.false.,.false.,.true.,.true.,.true.,.true.)
compute kinetic energy	compute kinetic energy (call compute_kinene(notemp))
	$E_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.70)
$F_n(dt)$	$F_n(dt) = 2E_k - gk_bT_0$
eta(dt) and $p_n(dt)$	update η and p_n
	$\eta(dt) \leftarrow \eta + \frac{\Delta t}{2}\frac{p_n}{Q}$ (B.71)
	$p_n(dt) \leftarrow p_n + \frac{\Delta t}{2} F_n(dt)$ (B.72)

SUBROUTINE: vverlet_npt_I0

comments mark	code
save the previous volume	vol0 = mdcell(7) = $v(0)$
compute xF	
	$L = \sum_{i=1}^{N} \sum_{j=1}^{3} x_{ji} F_{ji} \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.73)
compute kinetic energy	compute kinetic energy (call compute_kinene(notemp))
	$E_k = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{3} m_i v_{ji}^2 \qquad j = x, y, z(\equiv 1, 2, 3)$ (B.74)
F_eta(0)	$F_n(0) = 2E_k + p_v^2/W - g k_b T_0$
F_v(0)	$F_v(0) = [1/(3v(0))](2E_k + L) - P_0$
p_eta(dt/2)	update p_n
	$p_n(dt/2) \leftarrow p_n + \frac{\Delta t}{2} F_n(0)$ (B.75)
s_n(dt/2)	$s_n(dt/2) = \frac{\Delta t}{2} \frac{p_n}{Q}$
eta(dt/2)	update η
	$\eta(dt/2) \leftarrow \eta + s_n(dt/2)$ (B.76)
p_v(dt/2)	update p_v
	$p_v(dt/2) \leftarrow p_v \, e^{-s_n(dt/2)} + \frac{\Delta t}{2} F_v(0)$ (B.77)
s_v(dt/2)	$s_v(dt/2) = \frac{\Delta t}{2} \frac{p_v(dt/2)}{3v(0)W}$
fragment atom remove 0	if(fragment) call removefrozat(.true.,.true.,.true.,.true.)
p_i(dt/2) v_i(dt/2)	update nuclear velocities and positions
	$v_{ji}(dt/2) \leftarrow v_{ji} \, e^{-s_n(dt/2) - s_v(dt/2)} \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.78)
	$v_{ji}(dt/2) \leftarrow v_{ji}(dt/2) + \frac{\Delta t}{2} \frac{F_{ji}}{m_i} \qquad i = 1, ..., \inf(129)$ and $j = x, y, z$ (B.79)
volume(dt/2)	update volume
	$v(dt/2) \leftarrow v(0) + \frac{\Delta t}{2} \frac{p_v}{W}$ (B.80)
update dlv	update direct lattice vectors paret (3,3) matrix
	paret $\leftarrow$ paret $\cdot \sqrt[3]{\frac{v(dt/2)}{v(0)}}$ (B.81)
compute new cell	call parlat(paret,tmpinv,mdcell) : recompute cell lattice parameters
s_v(dt/2)	$s_v(dt/2) = \frac{\Delta t}{2} \frac{p_v(dt/2)}{3v(dt/2)W}$

r_i(dt) update nuclear positions

$$x_{ji}(dt/2) \leftarrow x_{ji}\, e^{s_v(dt/2)} + v_{ji}\, \Delta t \qquad i = 1, ..., \inf(129) \text{ and } j = x, y, z \tag{B.82}$$

$$x_{ji}(dt) \leftarrow x_{ji}(dt/2)\, e^{s_v(dt/2)} \qquad i = 1, ..., \inf(129) \text{ and } j = x, y, z \tag{B.83}$$

fragment atom insert 0 if(fragment) call insertfrozat(.true.,.true.,.true.,.true.,.true.,.true.)
scf call moldyn_scf_driver
save volume vol1 = mdcell(7) = $v(dt/2)$
volume(dt) update volume

$$v(dt) \leftarrow v(dt/2) + \frac{\Delta t}{2}\, \frac{p_v}{W} \tag{B.84}$$

update dlv update direct lattice vectors paret (3,3) matrix

$$\text{paret} \leftarrow \text{paret} \cdot \sqrt[3]{\frac{v(dt)}{v(dt/2)}} \tag{B.85}$$

compute new cell call parlat(paret,tmpinv,mdcell) : recompute cell lattice parameters
s_v(dt) $s_v(dt) = \frac{\Delta t}{2}\, \frac{p_v(dt/2)}{3v(dt)W}$
fragment atom remove 1 if(fragment) call removefrozat(.false.,.true.,.true.,.true.)
p_i(dt) v_i(dt) update nuclear velocities and positions

$$v_{ji}(dt/2) \leftarrow v_{ji}(dt/2) + \frac{\Delta t}{2}\, \frac{F_{ji}}{m_i} \qquad i = 1, ..., \inf(129) \text{ and } j = x, y, z \tag{B.86}$$

$$v_{ji}(dt) \leftarrow v_{ji}(dt/2)\, e^{-s_n(dt/2) - s_v(dt)} \qquad i = 1, ..., \inf(129) \text{ and } j = x, y, z \tag{B.87}$$

fragment atom insert 1 if(fragment) call insertfrozat(.false.,.false.,.true.,.true.,.true.,.true.)
compute xF

$$L = \sum_{i=1}^{N} \sum_{j=1}^{3} x_{ji}\, F_{ji} \qquad j = x, y, z (\equiv 1, 2, 3) \tag{B.88}$$

compute kinetic energy compute kinetic energy (call compute_kinene(notemp))

$$E_k = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{3} m_i\, v_{ji}^2 \qquad j = x, y, z (\equiv 1, 2, 3) \tag{B.89}$$

F_v(dt) $F_v(dt) = [1/(3v(dt))](2E_k + L) - P_0$
p_v(dt) update p_v

$$p_v(dt) \leftarrow [p_v(dt/2) + \frac{\Delta t}{2}\, F_v(dt)]\, e^{-s_n(dt/2)} \tag{B.90}$$

eta(dt) update η

$$\eta(dt) \leftarrow \eta(dt/2) + s_n(dt/2) \tag{B.91}$$

F_eta(dt) $F_n(dt) = 2E_k + p_v^2/W - gk_bT_0$
p_eta(dt) update p_n

$$p_n(dt) \leftarrow p_n(dt/2) + \frac{\Delta t}{2}\, F_n(dt) \tag{B.92}$$

assign to par(5)	par(5) = mdcell(7) : assign to par(5) the actual volume so that the actual volume can be used in Scf calculation

Subroutine: removefrozat_mat(mat,matallat,natfree)

comments mark	code
be sure matrix	be sure matrix matallat have the correct dimensions : it must be a (3,naf) matrix if not, deallocate matallat and allocate it with the correct (3,naf) dimensions
save all	matallat(:,:) = mat(:,:) : copy the content of mat in matallat (mat → matallat)
deallocate the (3,inf(24))	call crydealloc(mat,znamz,'MAT') : deallocate the matrix mat
allocate the (3,inf(129))	call cryalloc(mat,3,inf(129),znamz,'MAT') : allocate the matrix mat with dimensions (3,inf(129))
save in the mat	if(any(natfree == i)) mat(j,k) = matallat(j,i) with j=1,2,3 and k=1,...,inf(129) : save in the reshaped matrix mat *only* the elements with index contained in natfree vector (natfree vector has dimension = inf(129)) ! output : ! (3,inf(24)) matrix matallat with all data, ! (3,inf(129)) matrix mat with data only of those atoms whose index is contained in natfree vector

Subroutine: removefrozat_vec(vec,vecallat,natfree)

comments mark	code
be sure vector	be sure vector vecallat has the correct dimension : it must be a (naf) vector if not, deallocate vecallat and allocate it with the correct (naf) dimension
save all	vecallat(:) = vec(:) : copy the content of vec in vecallat (vec → vecallat)
deallocate the (inf(24))	call crydealloc(vec,znamz,'VEC') : deallocate the vector vec
allocate the (inf(129))	call cryalloc(vec,inf(129),znamz,'VEC') : allocate the vector vec with dimension (inf(129))
save in the vec	if(any(natfree == i)) vec(k) = vecallat(i) with i=1,...,naf and k=1,...,inf(129) : save in the reshaped vector vec *only* the elements with index contained in natfree vector (natfree vector has dimension = inf(129)) ! output : ! (inf(24)) vector vecallat with all data, ! (inf(129)) vector vec with data only of those atoms whose index is contained in natfree vector

Subroutine: removefrozat(l1,l2,l3,l4)

comments mark	code

	if(l1 == .true.) call removefrozat_mat(xa,xaallat,natfree)
	if(l2 == .true.) call removefrozat_mat(vel,velallat,natfree)
	if(l3 == .true.) call removefrozat_mat(forces,forcesallat,natfree)
	if(l4 == .true.) call removefrozat_vec(mass,massallat,natfree)

SUBROUTINE: insertmodfrozat_mat(mat,matallat,natfree)

comments mark	code
refresh values	if(any(natfree == i)) matallat(j,i) = mat(j,k) with j=1,2,3 and k=1,...,inf(129) : refresh values in matallat with values in mat related to moving atoms (moving atoms = atoms with index in natfree)
deallocate the (3,inf(129))	call crydealloc(mat,znamz,'MAT') : deallocate the matrix mat
allocate the (3,inf(24))	call cryalloc(mat,3,inf(24),znamz,'MAT')
save all	mat(:,:) = matallat(:,:) : copy the content of matallat in the reshaped matrix mat (matallat → mat) ! output : ! (3,inf(24)) matrix matallat with all data, ! (3,inf(24)) matrix mat with all data (moving and frozen atoms)

SUBROUTINE: insertfrozat_mat(mat,matallat)

comments mark	code
deallocate the (3,inf(129))	call crydealloc(mat,znamz,'MAT') : deallocate the matrix mat
allocate the (3,inf(24))	call cryalloc(mat,3,inf(24),znamz,'MAT')
save all	mat(:,:) = matallat(:,:) : copy the content of matallat in the reshaped matrix mat (matallat → mat) ! output : ! (3,inf(24)) matrix matallat with all data, ! (3,inf(24)) matrix mat with all data (moving and frozen atoms)

SUBROUTINE: insertfrozat_vec(vec,vecallat)

comments mark	code
deallocate the (inf(129))	call crydealloc(vec,znamz,'VEC') : deallocate the vector vec
allocate the (inf(24))	call cryalloc(vec,inf(24),znamz,'VEC')
save all	vec(:) = vecallat(:) : copy the content of vecallat in the reshaped vector vec (vecallat → vec) ! output : ! (inf(24)) vector vecallat with all data, ! (inf(24)) vector vec with all data (moving and frozen atoms)

SUBROUTINE: insertfrozat(l1,m1,l2,m2,l3,l4)

comments mark	code

	if(l1 == .true.) :
	if(m1 == .true.) call insertmodfrozat_mat(xa,xaallat,natfree)
	else (m1 == .false.) call insertfrozat_mat(xa,xaallat)
	if(l2 == .true.) :
	if(m2 == .true.) call insertmodfrozat_mat(vel,velallat,natfree)
	else (m2 == .false.) call insertfrozat_mat(vel,velallat)
	if(l3 == .true.) call insertfrozat_mat(forces,forcesallat)
	if(l4 == .true.) call insertfrozat_vec(mass,massallat)

SUBROUTINE: finalMD

comments mark	code
final output strings	write in the output file that the MD calculation is ended (the job is done) if(mdrest) write the previous and the total number of MD steps
close i/o files	call md_closefiles : close all the file units opened, they will be eventually reopened in the post processing subroutines
Post-processing	if(computePcf == .true.) call pcf_md : compute pair correlation functions if(analysisMD == .true.) call analysis_md : perform analysis of MD trajectory
deallocate v/m	call md_deallocate : deallocation of vectors and matrices allocated (the deallocation proceeds in inverse order with respect to the allocation order, following the stack architecture)

SUBROUTINE: md_closefiles

comments mark	code
close i/o files	close all the output files opened in subroutine md and written (updated) in the moldyn module

SUBROUTINE: md_deallocate(znamz)

comments mark	code
deallocate v/m	deallocate all the vectors and matrices allocated and used in the moldyn module the input argument znamz is a character containing the name of the subroutine in which this function is called

SUBROUTINE: moldyn_scf_driver

comments mark	code
proc dependency	call make_processor_dependent_suffix(suffix)
London subroutine	call gupdte : update geometry (shell positions, **G** vectors list and such)
if not mdfixind	if(.not.mdfixind) call int_screen(1) : reclassify the integrals
integral calculation	call int_calc : calculation of the integrals
basis set infos	if(guessp) lprint(72) = 0 : deactivates the print of the basis set infos after the first SCF cycle
scf cycle	call scf : perform the SCF cycle
if too many cycles	if(inf(35).eq.1) : if it is the first SCF cycle, then stop with error 'FIRST SCF ENDED WITH TOO MANY CYCLES' if it is not the first SCF cycle and GUESSP option is used, then restart SCF with atomic guess → → if even with atomic guess the SCF ends with too many cycles, then stop with error 'SCF RESTARTED WITH ATOMIC P - ENDED WITH TOO MANY CYCLES' → otherwise (if SCF converges well) continue to the next MD step if it is not the first SCF cycle and GUESSP option is not used, it means that the convergence failed even with an atomic guess, then stop with error 'SCF ENDED WITH TOO MANY CYCLES'
save final P(g) matrix	if(guessp) : call outo3b(iunit(20)) : print on file fort.20 the final SCF density matrix (ground state density matrix) rewind(iunit(20)) : rewind the fort.20 unit file
set energy and forces	energy = par(6) : save final SCF (ground state) energy forces(j,i) = atnug((i-1)*3+j) [i=1,naf ; j=1,3] : define forces (global matrix of the Moldyn module) from CRYSTAL global array atnug in which final SCF (ground state) forces acting on the nuclei are allocated
free memory	call tidy_memory : deallocate arrays and/or matrices left allocated (if(inf(170).eq.2) and if(allocated(itreni))) call free_dft_batches and call crydealloc(itreni,znamz,'itreni')

$$\{x_{ij}\} = \mathsf{pos} = \begin{pmatrix} x_1 & x_2 & ... & x_n \\ y_1 & y_2 & ... & y_n \\ z_1 & z_2 & ... & z_n \end{pmatrix} \qquad \text{units: [Bohr]} \tag{B.93}$$

$$\{v_{ij}\} = \mathsf{vel} = \begin{pmatrix} v_x^1 & v_x^2 & ... & v_x^n \\ v_y^1 & v_y^2 & ... & v_y^n \\ v_z^1 & v_z^2 & ... & v_z^n \end{pmatrix} \qquad \text{units: [Bohr/(a.u. time)]} \tag{B.94}$$

B.1.1 Box-Muller implementation

```
! BOX-MULLER algorithm (gaussian distribution function)
do i=1,naf
   do j=1,3
      call random_number(u1)
      call random_number(u2)
      uran = cos(2*par(1)*u1)*sqrt(-2*log(u2))
      vel(j,i) = uran*sqrt(kboltz*temperature/mass(i))
   enddo
enddo
```

u_1, u_2 random numbers

$$v_{ij} = \cos(2\pi u_1)\sqrt{-2 \cdot \log(u_2)} \tag{B.95}$$

$$v_{ij} \leftarrow v_{ij}\sqrt{\frac{k_b T}{m_i}} \qquad i = 1, ..., N \qquad j = x, y, z \tag{B.96}$$

units:

k_b [Ha/K] (E_h/T) T [K] m_i [u] (m_e) $\rightarrow$ v_{ij} $[\sqrt{\text{Ha/u}}]$ $(\sqrt{E_h/m_e})$

momentum $p_{ij} = m_i v_{ij}$ $(\hbar/a_0)$ mass m_i (m_e) velocity v_{ij} $(a_0 E_h/\hbar)$

$\rightarrow$ $\hbar/a_0 = m_e\, a_0 E_h/\hbar$ $\rightarrow$ $m_e = \hbar^2/(a_0^2\, E_h)$

$\rightarrow$ v_{ij} $(\sqrt{E_h/m_e} = \sqrt{a_0^2\, E_h^2/\hbar^2} = a_0\, E_h/\hbar)$

B.1.2 Kinetic energy and temperature calculations

Kinetic energy : $$E_k(t) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{3} m_i v_{ij}^2(t) \qquad j = x, y, z(\equiv 1, 2, 3) \tag{B.97}$$

Temperature : $$T(t) = \frac{2}{g k_b} E_k(t) \tag{B.98}$$

Initial temperature scaling : $$v_{ij} \leftarrow v_{ij}\sqrt{\frac{T}{T(t)}} \qquad i = 1, ..., N \qquad j = x, y, z \tag{B.99}$$

where T is the initial target temperature.

B.2 Constants and conversion units

	Variable	Value [real(float)]	Description	NIST link
Declared and initialized as global parameters in module moldyn				
0.	emass	$5.48579909065 \cdot 10^{-4}$	1 [me] = emass [u=Dalton]	https://physics.nist.gov/emass
1.	amu	1.0/emass	1 [u] = amu [me]	derived from 0.
2.	ha	$4.3597447222071 \cdot 10^{-18}$	1 [Ha] = ha [Joules]	https://physics.nist.gov/ha
3.	kb_joules	$1.380649 \cdot 10^{-23}$	Boltzmann constant [Joules/K]	https://physics.nist.gov/kb_joules
4.	kboltz	kb_joules/ha	Boltzmann constant [Hartree/K]	derived from 2. and 3.
5.	avonum	$6.02214076 \cdot 10^{23}$	Avogadro constant [1/mol]	https://physics.nist.gov/avonum
6.	kb_kJmol	kb_joules·avonum·10^{-3}	Boltzmann constant [kJ/(mol K)]	derived from 3. and 5.
7.	utime	$2.4188843265857 \cdot 10^{-17}$	1 [a.u. time] = utime [sec]	https://physics.nist.gov/utime
8.	tconv	$1.0/(\text{utime} \cdot 10^{15})$	1 [fs] = tconv [a.u. time]	derived from 7.
Declared as global variables in module moldyn and initialized in subroutines readMD and readFIRE				
9.	a0	par(32)·10^{-10}	1 [Bohr radius] = a0 [m]	https://physics.nist.gov/a0
10.	upres	ha/($a0^3$)	1 [a.u. pres] = upres [Pa]	derived from 2. and 9.
11.	pconv	$1.0/(\text{upres} \cdot 10^{-9})$	1 [GPa] = pconv [a.u. pres]	derived from 10.
12.	xconv	par(32) = 0.5291772083	1 [Bohr radius] = xconv [Angstrom]	internal code constant
13.	vconv	xconv·tconv	1 [a.u. vel] = vconv [Angstrom/fs]	derived from 8. and 12.

Table B.36: Variables used as conversion factors, the correspondent assigned values, module and/or subroutine in which they are initialized and description.

[au] v_{ji} · vconv → v_{ji} [Angstrom/fs]
[Angstrom/fs] v_{ji} / vconv → v_{ji} [au]

[au length] x_{ji} · xconv → t [Angstrom]
[Angstrom] x_{ji} / xconv → t [au length]

[au time] t / tconv → t [fs]
[fs] t · tconv → t [au time]

CRYSTAL code

Input : temperature T_0 [K] and oscillation time τ [fs]
Constants : Boltzmann constant k_b [Ha/K] - t_c conversion from fs to au time

$$Q = 2gk_bT_0\left(\frac{t_c\,\tau}{2\pi}\right)^2 \qquad \text{(Ha/K)(K)(au time)}^2 = \text{Ha(au time)}^2$$

VASP code script

Input : temperature T_0 [K] and oscillation time τ [fs]
Default : $\tau = 40$ fs
Constants : Boltzmann constant k_b [eV/K] with 1 eV = e J where e is the electron charge

$$Q = 2g\,k_b\,e\,T_0\left(10^{-15}\,\frac{\tau}{2\pi}\right)^2 \qquad \text{(J/K)(K)(sec}^2\text{)} = \text{J(sec}^2\text{)}$$

B.3 Output files

The files created by the molecular dynamics module are reported in the following table.

File name	old unit	NVE	Rescaling	Berendsen	NVT	NPT	Keyword	Condition to activate
ENERGIES.DAT	fort.221	T	T	T	T	T	none	none (active by default)
POSITIONS.DAT	fort.188	T	T	T	T	T	none	none (active by default)
VELOCITIES.DAT	fort.189	T	T	T	T	T	none	none (active by default)
TIMES.DAT	fort.222	F	F	F	F	F	TIMEPRT	timingprt = true
VIRTUAL.DAT	fort.190	F	F	F	T	T	none	vrtprt = true
MDCELL.DAT	fort.191	F	F	F	F	T	none	cellprt = true
COM.DAT	fort.192	F	F	F	F	F	COMPRT	comprt = true
ATCHARGES.DAT	–	F	F	F	F	F	CHARGESPRT	chargesprt = true
INITVEL.DAT	fort.193	F	F	F	F	F	INVELPRT	initvelprt = true
POSITIONS.XYZ	fort.186	F	F	F	F	F	XYZUNITS	xyzunits = true
VELOCITIES.XYZ	fort.187	F	F	F	F	F	XYZUNITS	xyzunits = true

Table B.37: T : printing of file enabled by default. F : printing of file disabled by default, to activate the print use the keyword in the penultimate column, which enables the logical condition in the last column. The filed printed by default (T) are necessary to restart the calculation, so that they can change on the base of the ensemble.

The files required to restart a molecular dynamics calculations are reported in the following table.

Ensemble	Files necessary for restart	Information contained
NVE	ENERGIES.DAT	nstepre - timepre
	POSITIONS.DAT	xa (positions matrix)
	VELOCITIES.DAT	vel (velocities matrix)
NVT	ENERGIES.DAT	nstepre - timepre
	POSITIONS.DAT	xa (positions matrix)
	VELOCITIES.DAT	vel (velocities matrix)
	VIRTUAL.DAT	zeta_nose - p_nose
NPT	ENERGIES.DAT	nstepre - timepre
	POSITIONS.DAT	xa (positions matrix)
	VELOCITIES.DAT	vel (velocities matrix)
	VIRTUAL.DAT	zeta_nose - p_nose - p_barost - dEdV
	MDCELL.DAT	$a, b, c, \alpha, \beta, \gamma$

B.4 Merging the moldyn_post.f90 library in CRYSTAL code

In order to insert the subroutine pcf_md in the CRYSTAL code, the module moldyn_post is created in the moldyn_post.f90 library. The created module is very useful for two different purposes:

(i) it makes the code more generalizable, i.e. if other molecular dynamics post processing calculations will be implemented in the future, they can be collected in the same module moldyn_post of the library moldyn_post.f90

(ii) it makes the module moldyn in the library moldyn.f90 more readable, because it contains only the molecular dynamics main calculations, letting all the post processing subroutines to be declared and written apart (i.e. in the moldyn_post.f90 library, moldyn_post module)

The makefile and the following libraries are then modified in the way explained below, in order to include the moldyn_post.f90 library:

1. makefile:
   (a) add the new object file in the variable MODF90 (→ Free format Fortran 90 modules) after the $(OBJDIR)/moldyn.o string:
   MODF90 = $(OBJDIR)/moldyn_post.o
   (b) add the new object file in the variable SCFCOM (→ Files common to crystal / Pcrystal / MPPcrystal, but not properties) after the $(OBJDIR)/moldyn.o string:
   SCFCOM = $(OBJDIR)/moldyn_post.o
   (c) add the new object file in the variable PROPS:
   PROPS = $(OBJDIR)/moldyn_post.o
   (d) add the new object file as a new link to the moldyn object in the cerber section:
   $(OBJDIR)/moldyn.o: moldyn.f90 $(COMMONDEP) $(OBJDIR)/moldyn_post.o
   (e) add the following line in the cerber section:
   $(OBJDIR)/moldyn_post.o: moldyn_post.f90
2. properties.f90:
   (a) add the following line in the subroutine f90main3:
   Use moldyn_post
   (b) add the following lines in the subroutine f90main3 (inside the do while(ok)):
   case ('PCFMD')
   call pcf_md
3. crystal.f90:
   (a) add the following line in the subroutine f90main:
   Use moldyn_post

In this way, the subroutine for the calculation of the pair correlation function can be used both in the CRYSTAL input files .d12 and .d3, i.e. both as a post processing calculation in the main input file .d12 (with the initial keyword PCFMD and the closing keyword END inside the MD section), and as a post processing calculation in the properties file .d3 (with the same keywords used in the .d12 file). This has the advantage that, if the user wants to perform a molecular dynamics calculation and, after that, compute the pair correlation function, he/she can insert the PCFMD section in the MD section in the file .d12, otherwise, if the user has already done a molecular dynamics simulation and have already been generated the POSITIONS.DAT file, he/she can use use the PCFMD section in the .d3 file, reading and analyzing the trajectory file without performing a new molecular dynamics simulation. Inside the PCFMD section, the keywords which can be inserted are the same for both the .d12 and the .d3 files (see manual in Appendix E.1).

B.4.1 The module read_moldyn_post_module: reading of the input file

A module read_moldyn_post_module is implemented in moldyn_post.f90 library, which contains a series of subroutines for the reading of the input file sections dedicated to the post processing calculations that can be done after a molecular dynamic simulation. In particular, the subroutines in read_moldyn_post_module read the keywords in the input file .d12 or .d3 for the calculation of the radial pair correlation function, the analysis of the trajectory, the calculation of the velocity autocorrelation function and of the power spectrum. The activation of the reading subroutines and the subsequent calculation of the associated quantities are triggered by the initial keywords PCFMD, ANALYSIS, AVCORR or FREQCALC in the input file .d12 or .d3, and terminated with the keyword END. The other keywords useful for the setting of the parameters for the calculations have to be enclosed in each section, and reported more specifically in the manual in Appendix E.1.

B.4.2 Molecular dynamics and post processing calculations using input file .d12

In order to make the subroutines in read_moldyn_post_module module visible by the CRYSTAL code when reading the input files .d12, a new case is added in the subroutine readMD (module read_moldyn_module, library moldyn.f90), so that if the keywords for the post processing are read in the input file inside the MOLDYN section, a specific subroutine in read_moldyn_post_module module is called and executed. At the same time, if for example the keyword PCFMD is present in the input file, in the readMD subroutine the flag computePcf for the computation of the pair correlation function is turned on (i.e. it assumes the value .true.). Then, during the execution of the md subroutine in the library moldyn.f90, the subroutine finalMD is in turns called: at the end of this subroutine, if the flag computePcf is true, then the subroutine pcf_md is called and the calculation of the radial pair correlation functions is performed. The same workflow is followed (using other logical variables to active the calling to the specific subroutines) when the keywords ANALYSIS, AVCORR or FREQCALC are used in the input file for the analysis of the trajectory, the calculation of the velocity autocorrelation function and of the power spectrum, respectively. The scheme of the links between libraries and subroutines for this part is reported in Figure B.1.

As seen in Figure B.1, the first library called by the CRYSTAL code is input.f90, which reads the keywords in the input file .d12. If the keyword MOLDYN is found, then the 189-th element of the vector inf is set equal to one and the subroutine readMD is called and executed. This subroutine reads all the keywords in the input file under the section MOLDYN until the END keyword is find. If among the keywords read by readMD there is, for example, the string PCFMD, then the logical variable computePcf is set true and the subroutine readMD_post_pcf of read_moldyn_post_module module is called, which finally reads all the keyword of the PCF section, until the END word is found. Subsequently, the CRYSTAL code call the libraries crystal.f90 and crystal06.f90. If the 189-th element of the vector inf is different by zero, then the subroutine md is called and executed. In the last part of that subroutine, the subroutine finalMD is executed, and the calculation of the radial pair correlation functions is done here, with a call to the pcf_md subroutine of the library moldyn_post.f90. The same workflow is also followed by the others post processing properties.

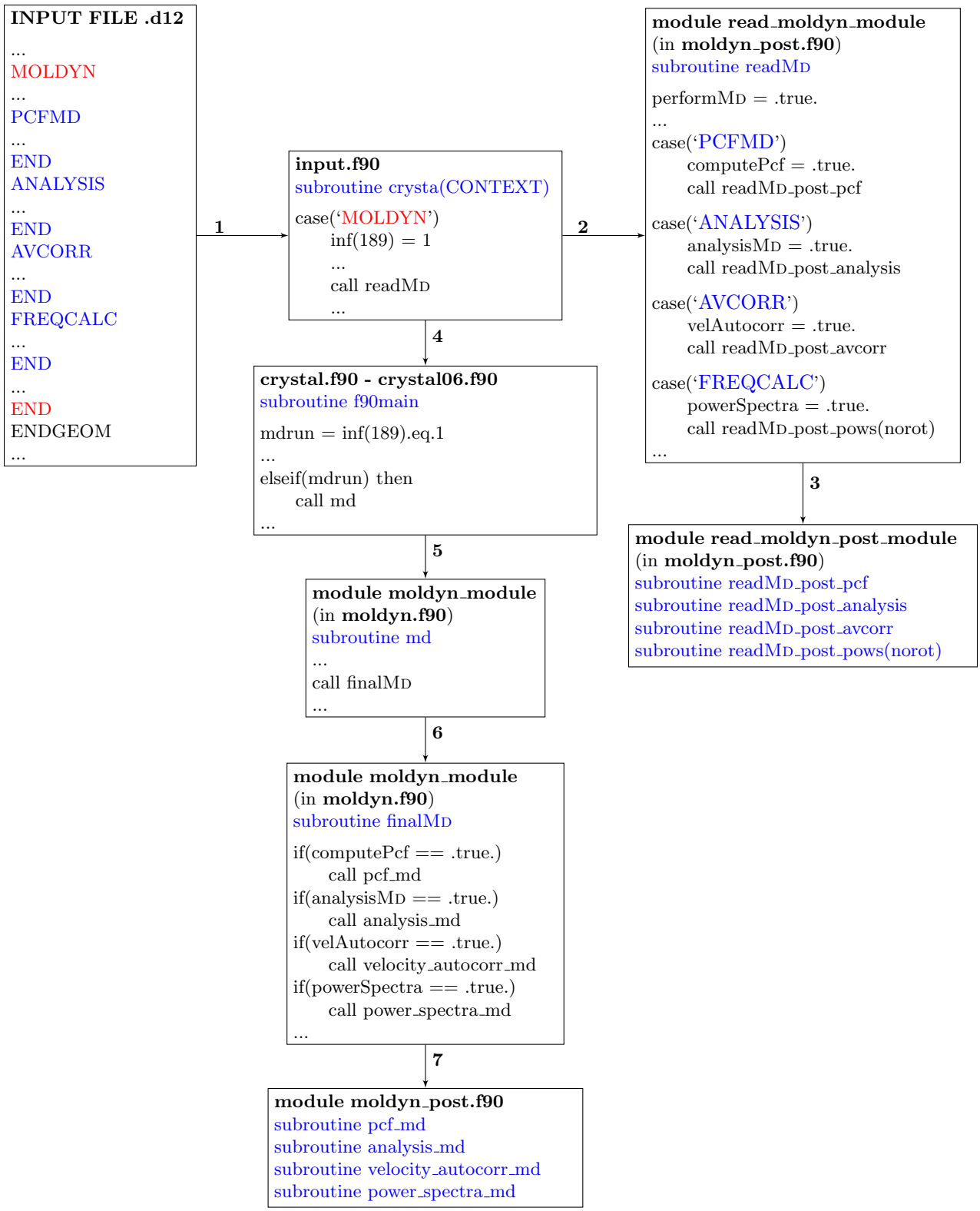

Figure B.1: Links and dependencies of libraries and subroutines involved in the calculation of the radial pair correlation function, the analysis of the trajectory and the calculation of the velocity autocorrelation function and of the power spectrum (post processing analysis calculation derived from the molecular dynamics trajectories), starting from the input file .d12. The MOLDYN section block must be the last section of the geometry block.

B.4.3 Post processing calculations starting from the input file .d3

In order to compute the post processing properties starting from the properties file .d3, a call first to specific subroutines in read_moldyn_post_module module for the reading and then to the subroutines in moldyn_post module for the calculation is done in the library properties.f90, which reads the input file .d3. The scheme for the calculation of the radial pair correlation function, the analysis of the trajectory, the calculation of the velocity autocorrelation function and the power spectrum, from the input file .d3, is reported in Figure B.2.

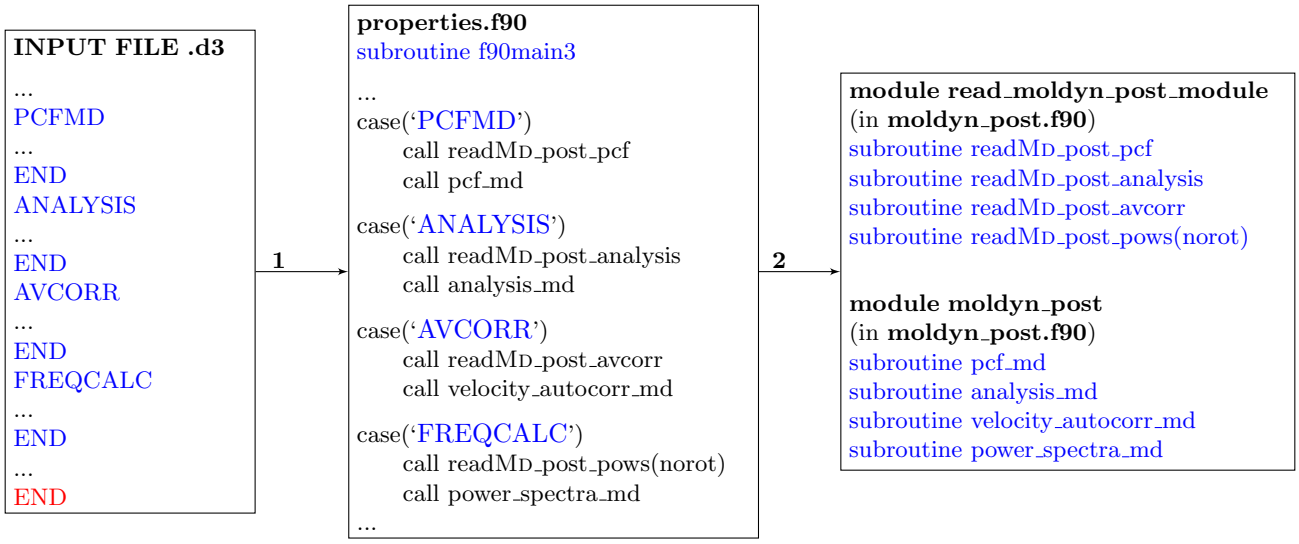

Figure B.2: Links and dependencies of libraries and subroutines involved in the calculation of the radial pair correlation function, the analysis of the trajectory and the calculation of the velocity autocorrelation function and of the power spectrum (post processing analysis calculation derived from the molecular dynamics trajectories), starting from the input file .d3.

The specific subroutines of read_moldyn_post_module read the correspondent sections in the input file .d3 until the END keyword is found. The second END keyword (marked in red in Figure B.2) is referred to the end of the properties section. After that the reading of the input file is done, the parameters for the calculation of the selected post processing properties are set, and the calling to the subroutines of moldyn_post module performed in the library properties.f90 finally execute the correspondent calculation.

B.5 FIRE algorithm in the framework of the CRYSTAL code

In this section the implementation of the FIRE algorithm in the CRYSTAL code is described. The subroutine fire accomplishes the task of structural minimization with FIRE algorithm. It is a public subroutine of the fire_module module (in moldyn.f90), and it is called by the subroutine f90main in crystal.f90 and crystal06.f90 if the logical variable firerun = inf(189).eq.2 is .true. (this logical variable is defined in input.f90 and it is initialized to .true. if the keyword FIRE is used in the input file). In this framework, the use of the keyword FIRE in the input file, with a subsequent minimization using the fire subroutine, is not connected to the use of the keyword MOLDYN in the input file that performs a molecular dynamics simulation. The subroutine fire is therefore disconnected with respect to the subroutine moldyn, indeed, the belong to two different modules in moldyn.f90. At the same time, the FIRE algorithm is a structural minimization tool which uses molecular dynamics concepts to find the optimized geometry of a structure. Therefore, the FIRE algorithm needs to use some molecular dynamics integrator, such as the Velocity Verlet algorithm, which has been already implemented in the subroutine vverlet_NVE in the moldyn_module module. Therefore, the subroutine fire for the FIRE algorithm has been implemented inside the fire module in the moldyn.f90 library. In this framework, some global variables and constants defined in the declaration and initialization part of the moldyn module are shared between the fire and the md subroutines. Since the fire subroutine and the md subroutine are mutually exclusive, the possibility of conflict between global variables initialized in one of these two subroutines is automatically excluded: if the code call the md subroutine, the fire subroutine is not called, and vice versa, so that the global variables declared in the first part of the moldyn module are initialized in the md subroutine *or* in the fire subroutine.

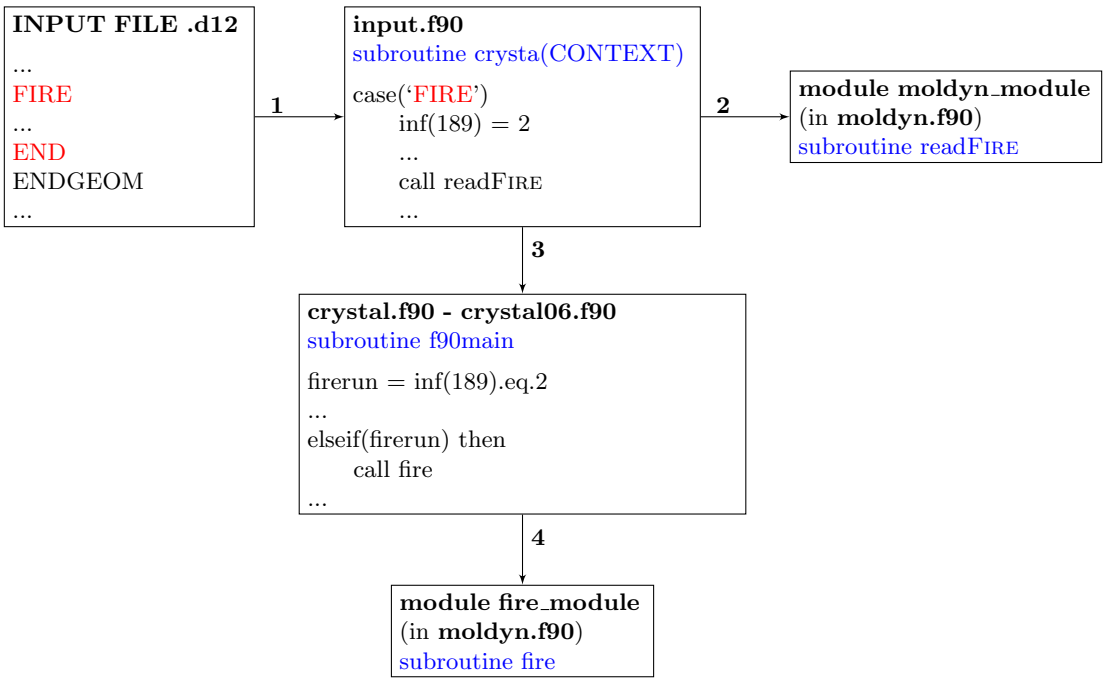

Figure B.3: Links and dependencies of the modules and subroutines involved in the FIRE structural minimization algorithm, starting from the input file .d12, are sketched in the scheme. The order of subroutines calls is indicated through numbers near the connecting arrows of the diagram. The FIRE section block must be the last section of the geometry block.

B.5.1 Module fire_module

The implementation of the FIRE algorithm, described in Chapter 8, is carried out in the fire_module module, whose source code is written in moldyn.f90. In the following, a description of this source code and of the FIRE algorithms it contains is given. The dependencies and connections of the Moldyn module with the other parts of the CRYSTAL code outside the module are specified in Section B.5. In

the following, the main steps in the execution statements of each subroutine used for the FIRE algorithm are described.

B.5.1.1 Subroutine fire

The subroutine fire is the main subroutine of the fire_module module which performs a structural relaxation using molecular dynamics concepts: in this subroutine, the main body of the FIRE algorithm is written and the other important subroutines for this algorithm are called (such as the subroutine velocity_NVE for the molecular dynamics integration, the subroutine moldyn_scf_driver for the self consistent cycle, the calculation of forces and energies, and the subroutine testcon_fire for the check of the convergence of the FIRE algorithm). The subroutine fire is a public subroutine of the fire_module module, and it is called by the subroutine f90main in crystal.f90 and crystal06.f90 if the logical variable firerun = inf(189).eq.2 is .true. (this logical variable is defined in input.f90 and it is set to .true. if the keyword FIRE is used in the input file). In the following, the subroutine fire, which implements the FIRE algorithm, is described.

Check on the input geometry An initial check on the input geometry is done at the beginning of the subroutine fire. The FIRE algorithm is based on molecular dynamics concepts, assigning to each atom an equation of motion which would lead the whole atomic structure towards a minimum in the potential energy surface. As a consequence, symmetry properties of the crystalline or molecular structure are not exploited in the algorithm itself. In this framework, the input atomic structure has supposed not to have any symmetry (the symmetry is removed, the space group is set to P_1). If a space group different from P_1 (no symmetry) is inserted in the input file, the code stops printing in the output an explanatory error:

ERROR **** FIRE (MD-LIKE) **** ONLY P1 SYMMETRY SUPPORTED

Open of input/output files The following files (units) are opened:

- fort.188 in which the positions of the nuclei for each step will be written
- fort.189 in which the velocities of the nuclei for each step will be written

If required in the input file through the keyword XYZUNITS, open the files:

- fort.186 in which the positions of the nuclei for each step will be written
- fort.187 in which the velocities of the nuclei for each step will be written

These two files are in a format that can be written by external programs, such as VMD (Visual Molecular Dynamics), with which the positions of the nuclei step by step can be visualized.

Initialization of positions, velocities and forces In this part, the important quantities needed for the calculation are declared, such as the number of atoms, the mass of each atom and the total mass of the system, the initial positions and velocities of the nuclei. In the CRYSTAL code, the initial positions of the nuclei are read in the input file and allocated in the matrix pos, that is a $3 \times n$ pos matrix (where n is the number of atoms) in which the positions of the nuclei are stored as follows:

$$\mathsf{pos} = \begin{pmatrix} x_1 & x_2 & ... & x_\mathrm{n} \\ y_1 & y_2 & ... & y_\mathrm{n} \\ z_1 & z_2 & ... & z_\mathrm{n} \end{pmatrix} \qquad \text{units: [Bohr]} \tag{B.100}$$

The velocities of the nuclei are allocated in the matrix vel, which is a $3 \times n$ matrix in which the elements are organized as follows:

$$\mathsf{vel} = \begin{pmatrix} v_x^1 & v_x^2 & ... & v_x^\mathrm{n} \\ v_y^1 & v_y^2 & ... & v_y^\mathrm{n} \\ v_z^1 & v_z^2 & ... & v_z^\mathrm{n} \end{pmatrix} \qquad \text{units: [Bohr/(a.u. time)]} \tag{B.101}$$

At the beginning of the algorithm, the velocities of the nuclei are set all equal to zero (vel = 0.0), so that the velocities matrix are here initialized to a zeros matrix.
Then, the calculation of the maximum and minimum timestep, the computation of the thresholds on energy and on the euclidean norm of nuclei forces are performed, on the basis on input values or default values set in readFIRE subroutine. All these initial information, together with the masses of the atoms, the initial positions of the nuclei and the initial parameters for the FIRE algorithm (defined in the input file of by default in readFIRE subroutine), which constitute the initial setup for the algorithm, are printed in the output file.

Calculation of the initial forces on the nuclei Once the initial positions and velocities have been defined, the code proceeds with the calculation of forces on the nuclei. The subroutine moldyn_scf_driver, defined in moldyn.f90, is called to execute the following tasks: (i) perform a self consistent calculation which find the ground state electronic wavefunction for that particular nuclear configuration, (ii) compute the forces on the nuclei with the Hellman-Feynman theorem. At the end of this process, an initial set of positions, velocities and forces are defined, together with the initial parameters for FIRE algorithm and with the thresholds on quantities which are used as stopping criteria for the algorithm.

Fire algorithm: main body The main body of the FIRE algorithm is then implemented. If in the input file the keyword FIRE is used, then the basic FIRE algorithm is performed. This task is accomplished through the function firealg1, which is called at this point by the fire subroutine. The subroutine firealg1 is described in Section B.5.1.2.

Writing final information Once the FIRE algorithm finishes, the program exits the firealg1 subroutine and returns to the fire subroutine. At this point, all the final important information related to the optimized atomic structure are printed in the output file: the final band gap or the Fermi energy, the list of the nearest neighbors in the first five shell for each atom and the final optimized geometry (with positions of all the nuclei belonging to the asymmetric unit given in Angstrom units). Finally, all the vectors and matrices used for the calculation are deallocated (using the crydealloc subroutine, which deallocates vectors and matrices initially allocated with the cryalloc subroutine).

B.5.1.2 Subroutine firealg1

The subroutine firealg1 implements the main body of the basic FIRE algorithm, introduced in Ref. [125] and described in Ref. [127]. The pseudo code of the FIRE algorithm in the CRYSTAL code, together with the list of tasks performed by the firealg1 subroutine, is sketched in the flowchart reported in the following page. The firealg1 calls two important subroutines: (i) the vverlet_NVE subroutine, which performs the integration of the equation of motion through the Velocity Verlet algorithm and also performs a single point energy calculation (by calling the moldyn_scf_driver subroutine) in order to compute a new set of forces used in the Velocity Verlet integration; and (ii) the testcon_fire subroutine, which prints in the standard output the state of the convergence on the four quantities of interest (the maximum value and the root-mean-square of the nuclei displacements, the euclidean norm of the 3×naf forces vector and the difference between)

B.5.1.3 Subroutine readFire

If in the input file the keyword FIRE is found in input.f90, the subroutine readFIRE is called by that same module. The subroutine readFIRE sets the default values for some input variables and reads all the keywords and numbers in the input file between the FIRE keyword and the terminating END string, through a series of case statements. If the user writes a keyword in the input file (in the section reserved to the minimization setup with the FIRE algorithm) which is not found among all those listed in the subroutine readFIRE, the code stops, printing in the standard output an explanatory error.
The list of allowed keywords and default values are reported in Section E.2.

FULLOPTG algorithm

libopt.f
SUBROUTINE CONOPT_

The input file in the block OPTGEOM ... END is read.
In particular, the variable inf(115) is here initialized depending on the time the type of geometry optimization performed

FIRE algorithm

moldyn.f90
SUBROUTINE readFIRE

The input file in the block FIRE ... END is read.
The variable GENSAED_MD is set to true if the user wants to perform a structural optimization of both the atomic coordinates and the cell parameters, or a structural optimization of the cell parameters only, otherwise it is false

geometry.f
SUBROUTINE SYMMETRIZE_DIR

```
3632       INF115=INF(115)

3641 ! CRBL ADDS CONDITION ON GENSAEDFIRE IN THE LINE HERE BELOW ! FALG11
3642       IF(INF115.GE.3.AND.NSAED.EQ.0.OR.LELAPIEZ.OR.GENSAED_MD)THEN
3643       CALL GENERATE_SAED(.TRUE.)
3644       ENDIF
```

geometry.f
SUBROUTINE GENERATE_SAED(LOGIK)

The symmetry allowed elastic distortions are generated

libopt.f
SUBROUTINE OPTGEN

CALL SCF

The SCF cycle is performed at each optimization step

moldyn.f90
SUBROUTINE MOLDYN_SCF_DRIVER

CALL SCF

The SCF cycle is performed at each optimization step

libx5_scf.f
SUBROUTINE SCF

```
2397       IF(INF(35).NE.1.OR.INF(66).NE.0)THEN
2398       IF(INF(201).NE.0)THEN
2399       CALL TOTGRA_C
2400       ELSE
2401       CALL TOTGRA
2402       ENDIF
2403       ENDIF
```

libforce6.f
SUBROUTINE TOTGRA_C

CALL SYMMETRIZE_GRAD_C

libforce6.f
SUBROUTINE SYMMETRIZE_GRAD_C

Compute cartesian forces in Hartree/Bohr (analytical)

Compute symmetry allowed forces (analytical), atomic part and cell part (gradient of energy with respect to the cell parameters)

Compute stress tensor in Hartree/Bohr^3

```
1289       write(iout,101)
1290       if(nsaed.eq.0)then
1291       write(iout,102)
1292       else
1293       m=nsadir+1
1294       forze(m:nsadir+nsaed)=0._float
1295 c KD 3 Sept 09
1296       WRITE(IOUT,120)
1297       IF(IDIM.EQ.1)WRITE(IOUT,121) FORZE_C(1)
1298       IF(IDIM.EQ.2)THEN
1299       WRITE(IOUT,122)
1300       WRITE(IOUT,123) FORZE_C(1),FORZE_C(2),FORZE_C(3),FORZE_C(4)
1301       ENDIF
1302       IF(IDIM.EQ.3)THEN
1303       WRITE(IOUT,124)
1304       WRITE(IOUT,125)FORZE C(1:9)
```

...

Figure B.4: Workflow of the main subroutines invoked for the calculation of the gradient of the energy with respect to the nuclei positions and the cell parameters. The FULLOPTG (left) and the FIRE (right) algorithms call the same subroutines, they differ only at the beginning and when the SCF calculation is performed.

subroutine: readFire

Initial parameters

maxcycle M_c	(keyword: MAXCYCLE)
ndelay N_d	(keyword: NDELAY)
alphastart α_{in}	(keyword: ALPHASTART)
alphashrink f_α	(keyword: ALPHASHRINK)
dtstart dt_{in}	(keyword: DTSTART)
dtgrow f_{inc}	(keyword: DTGROW)
dtshrink f_{dec}	(keyword: DTSHRINK)
tmax t_{max}	(keyword: TMAX)

Thresholds

onlyfrth	(keyword: ONLYFRTH)
expfthr t_f	(keyword: TOLFORCE)
onlyenth	(keyword: ONLYENTH)
expethr t_e	(keyword: TOLDEE)
onlydsth	(keyword: ONLYDSTH)
thdrms t_{drms}	(keyword: TOLDEX)
thdmax t_{dmax}	= thdrms · 1.5_float
ncstpcon n_{stc}	(keyword: NCSTPCON)

subroutine: fire

Check on the symmetry of the input system: only P_1 space group supported

Initialization of initial positions $\mathbf{x}(t_0)$ of the nuclei

Initialization of initial velocities $\mathbf{v}(t_0)$ of the nuclei: vel(3,naf) = 0.0_float

dtmax $dt_{max} = dt_{in} \cdot t_{max}$

ethr $= 10^{-t_e}$
fthr $= 10^{-t_f}$

Calculation of the initial energy and initial forces $\mathbf{F}(\mathbf{x}(t_0))$ on the nuclei (using Hellman-Feynman theorem on the electronic ground state, moldyn_scf_driver subroutine)
calltoforce = 1

call firealg1

subroutine: firealg1

npgzero $N_{P>0} = 0$
fictime t = 0.0_float
alphanow $\alpha = \alpha_{in}$
dtnow $dt = dt_{in}$
nconv = 0
deltae ΔE = energy (initial energy)

Print information on actual nuclei coordinates, velocities, forces and energy with call write_cofoven_fire, the nuclei displacements, actual timestep dt and α parameter

! FIRE LOOP BEGIN
do i = 1, M_c
  $P(t) = \mathbf{F}(\mathbf{x}(t)) \cdot \mathbf{v}(t)$! compute power
  energyprev = energy ! save current energy
  if $P(t) > 0$ **then** ! downhill motion
    $N_{P>0} = N_{P>0} + 1$
    $\mathbf{v}(t) = (1-\alpha)\mathbf{v}(t) + \alpha\mathbf{F}(\mathbf{x}(t))\|\mathbf{v}(t)\|/\|\mathbf{F}(\mathbf{x}(t))\|$
    if $N_{P>0} > N_d$ **then** ! equilibration
      dtnow dt = min($dt\, f_{inc}$, dt_{max}) ! increase the timestep
      alphanow $\alpha = \alpha\, f_\alpha$! decrease alpha
    endif
    print motion information
  else if $P(t) \leq 0$ **then** ! uphill motion
    $N_{P>0} = 0$
    vel(3,naf) = 0.0_float ! freeze the velocities
    dtnow $dt = dt\, f_{dec}$! decrease the timestep
    alphanow $\alpha = \alpha_{in}$! reset alpha to alphastart
    print motion information
  endif
  $t = t$ + dtnow ! increase fictitious time
  xa_prev = xa ! save current nuclei positions
  call vverlet_nve ! MD integration and energy calculation
  calltoforce = calltoforce + 1

  Print information on actual nuclei coordinates, velocities, forces and energy with call write_cofoven_fire, the nuclei displacements, actual timestep dt and α parameter

  ΔE = energyprev - energy ! compute energy difference between steps
  eunormf $= \|\mathbf{F}(\mathbf{x}(t))\|/\sqrt{3N}$! compute forces vector euclidean norm

  ! compute RMS of nuclei displacements:
  ds = xa - xa_prev
  displa(:) = sqrt(ds(1,:)^2 + ds(2,:)^2 + ds(3,:)^2)
  dmax = maxval(displa(1:naf))
  drms = sqrt(sum(displa^2)/size(displa))

  call testcon_fire ! check the convergence

  if(firefinish) exit ! if FIRE ended (converged or maxcycle)
  ! exit the FIRE loop
enddo
! FIRE LOOP END

subroutine: fire

Print the info on the optimized final geometry: band gap or Fermi energy, list of nearest neighbors and positions of the nuclei in the final optimized geometry

subroutine: vverlet_nve

```
! calculation of new positions x(t+dt)
! v(t+dt/2) = v(t) + (dt/2)(F(x(t))/m)
! x(t+dt) = x(t) + dt · v(t+dt/2)
do i = 1,naf
    do k = 1,3
        vel(k,i) = vel(k,i) + (dt/2) [F(k,i)/mass(i)]
        xa(k,i) = xa(k,i) + dt vel(k,i)
    enddo
enddo

! calculation of new energy and forces F(x(t+dt))
call moldyn_scf_driver

! calculation of new velocities v(t+dt)
! v(t+dt) = v(t+dt/2) + (dt/2)(F(x(t+dt))/m)
do i = 1,naf
    do k = 1,3
        vel(k,i) = vel(k,i) + (dt/2) [F(k,i)/mass(i)]
    enddo
enddo
```

subroutine: testcon_fire

```
fireconv = .false.
firefinish = .false.

nconv_prev = nconv

if(onlyfrth == .true.) then
    if(eunormf ≤ fthr) nconv = nconv + 1
else if(onlyenth == .true.) then
    if(|ΔE| ≤ ethr) nconv = nconv + 1
else if(onlydsth == .true.) then
    if(dmax ≤ thdmax .and. drms ≤ thdrms) then
        nconv = nconv + 1
    endif
else if(eunormf ≤ fthr .and. |ΔE| ≤ ethr .and. &
        dmax ≤ thdmax .and. drms ≤ thdrms) then
    nconv = nconv + 1
endif

if(nconv == ncstpcon) then
    fireconv = .true.
    firefinish = .true.
else if(nconv == nconv_prev) then
    nconv = 0
endif

! compute actual number of FIRE steps
nstepfire = calltoforce - 1

if(nstepfire == maxcycle M_c) firefinish = .true.
```

If FIRE is finished (converged or reached maxcycle), then print info in the standard output file

subroutine: readFire

Initial parameters

maxcycle M_c	(keyword: MAXCYCLE)
ndelay N_d	(keyword: NDELAY)
nplezeromax $N_{P\leq 0}^{max}$	(keyword: NPLEZMAX)
alphastart α_{in}	(keyword: ALPHASTART)
alphashrink f_α	(keyword: ALPHASHRINK)
dtstart dt_{in}	(keyword: DTSTART)
dtgrow f_{inc}	(keyword: DTGROW)
dtshrink f_{dec}	(keyword: DTSHRINK)
tmax t_{max}	(keyword: TMAX)
tmin t_{min}	(keyword: TMIN)

Thresholds

onlyfrth	(keyword: ONLYFRTH)
expfthr t_f	(keyword: TOLFORCE)
onlyenth	(keyword: ONLYENTH)
expethr t_e	(keyword: TOLDEE)
onlydsth	(keyword: ONLYDSTH)
thdrms t_{drms}	(keyword: TOLDEX)
thdmax t_{dmax}	= thdrms · 1.5_float
ncstpcon n_{stc}	(keyword: NCSTPCON)

subroutine: vverlet_firealg2(alphanow)

! mixing and calculation of new positions $\boldsymbol{x}(t+dt)$
! $\boldsymbol{v}(t+dt/2) = \boldsymbol{v}(t) + (dt/2)(\boldsymbol{F}(\boldsymbol{x}(t))/m)$
! $\mathbf{v}(t) = (1-\alpha)\mathbf{v}(t) + \alpha\mathbf{F}(\mathbf{x}(t))\|\mathbf{v}(t)\|/\|\mathbf{F}(\mathbf{x}(t))\|$
! $\boldsymbol{x}(t+dt) = \boldsymbol{x}(t) + dt \cdot \boldsymbol{v}(t+dt/2)$
do i = 1,naf
  do k = 1,3
    vel(k,i) = vel(k,i) + $(dt/2)$ [F(k,i)/mass(i)]
    vel(k,i) = (1 - α) vel(k,i) + α F(k,i) $\|$vel$\|/\|$F$\|$
    xa(k,i) = xa(k,i) + dt vel(k,i)
  enddo
enddo

! calculation of new energy and forces $\boldsymbol{F}(\boldsymbol{x}(t+dt))$
call moldyn_scf_driver

! calculation of new velocities $\boldsymbol{v}(t+dt)$
! $\boldsymbol{v}(t+dt) = \boldsymbol{v}(t+dt/2) + (dt/2)(\boldsymbol{F}(\boldsymbol{x}(t+dt))/m)$
do i = 1,naf
  do k = 1,3
    vel(k,i) = vel(k,i) + $(dt/2)$ [F(k,i)/mass(i)]
  enddo
enddo

subroutine: testcon_fire

fireconv = .false.
firefinish = .false.

nconv_prev = nconv

if(onlyfrth == .true.) **then**
  if(eunormf $\leq$ fthr) nconv = nconv + 1
else if(onlyenth == .true.) **then**
  if($|\Delta E| \leq$ ethr) nconv = nconv + 1
else if(onlydsth == .true.) **then**
  if(dmax $\leq$ thdmax .and. drms $\leq$ thdrms) **then**
    nconv = nconv + 1
  endif
else if(eunormf $\leq$ fthr .and. $|\Delta E| \leq$ ethr .and. &
    dmax $\leq$ thdmax .and. drms $\leq$ thdrms) **then**
  nconv = nconv + 1
endif

if(nconv == ncstpcon) **then**
  fireconv = .true.
  firefinish = .true.
else if(nconv == nconv_prev) **then**
  nconv = 0
endif

! compute actual number of FIRE steps
nstepfire = calltoforce - 1

if(nstepfire == maxcycle M_c) firefinish = .true.

If FIRE is finished (converged or reached maxcycle), then print info in the standard output file

subroutine: fire

Check on the symmetry of the input system: only P_1 space group supported

Initialization of initial positions $\mathbf{x}(t_0)$ of the nuclei

Initialization of initial velocities $\mathbf{v}(t_0)$ of the nuclei: vel(3,naf) = 0.0_float

dtmax $dt_{max} = dt_{in} \cdot t_{max}$
dtmin $dt_{min} = dt_{in} \cdot t_{min}$

ethr $= 10^{-t_e}$
fthr $= 10^{-t_f}$

Calculation of the initial energy and initial forces $\mathbf{F}(\mathbf{x}(t_0))$ on the nuclei (using Hellman-Feynman theorem on the electronic ground state, moldyn_scf_driver subroutine)
calltoforce = 1

call firealg2

subroutine: firealg2

npgzero $N_{P>0} = 0$
nplezero $N_{P\leq 0} = 0$
fictime t = 0.0_float
alphanow $\alpha = \alpha_{in}$
dtnow $dt = dt_{in}$
nconv = 0
deltae ΔE = energy (initial energy)

Print information on actual nuclei coordinates, velocities, forces and energy with call write_cofoven_fire, the nuclei displacements, actual timestep dt and α parameter

! FIRE LOOP BEGIN
do i = 1, M_c
  $P(t) = \mathbf{F}(\mathbf{x}(t)) \cdot \mathbf{v}(t)$! compute power
  energyprev = energy ! save current energy
  if $P(t) > 0$ **then** ! downhill motion
    $N_{P>0} = N_{P>0} + 1$
    if $N_{P>0} > N_d$ **then** ! equilibration
      dtnow dt = min($dt\, f_{inc}$, dt_{max}) ! increase the timestep
      alphanow $\alpha = \alpha\, f_\alpha$! decrease alpha
    endif
    print motion information
  else if $P(t) \leq 0$ **then** ! uphill motion
    $N_{P>0} = 0$
    vel(3,naf) = 0.0_float ! freeze the velocities
    dtnow $dt = dt\, f_{dec}$! decrease the timestep
    alphanow $\alpha = \alpha_{in}$! reset alpha to alphastart
    print motion information
  endif
  $t = t$ + dtnow ! increase fictitious time
  xa_prev = xa ! save current nuclei positions
  call vverlet_firealg2(alphanow) ! MD integration and energy calculation
  calltoforce = calltoforce + 1

  Print information on actual nuclei coordinates, velocities, forces and energy with call write_cofoven_fire, the nuclei displacements, actual timestep dt and α parameter

  ΔE = energyprev - energy ! compute energy difference between steps
  eunormf = $\|\mathbf{F}(\mathbf{x}(t))\|/\sqrt{3N}$! compute forces vector euclidean norm

  ! compute RMS of nuclei displacements:
  ds = xa - xa_prev
  displa(:) = sqrt(ds(1,:)^2 + ds(2,:)^2 + ds(3,:)^2)
  dmax = maxval(displa(1:naf))
  drms = sqrt(sum(displa^2)/size(displa))

  call testcon_fire ! check the convergence

  if(firefinish) exit ! if FIRE ended (converged or maxcycle)
  ! exit the FIRE loop
enddo
! FIRE LOOP END

subroutine: fire

Print the info on the optimized final geometry: band gap or Fermi energy, list of nearest neighbors and positions of the nuclei in the final optimized geometry

B.5.2 Changes in libopt.f library and memory_opt.f90 module

In CRYSTAL code, the implementation of the Conjugate gradient (CG) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian updating algorithms for minimization is performed in the libopt.f library. In order to compare the CG and BFGS hessian updating schemes with the FIRE algorithm for structural minimization, it is necessary to introduce the same quantities adopted in FIRE algorithm as stopping criteria, i.e. the maximum nuclei displacements and the root-mean-square displacement, together with the euclidean norm of the $3N$ forces vector, as defined in Chapter 8. In this way, when the CG or BFGS algorithm complete their calculation of an optimized atomic structure, these quantities are printed at convergence and the efficiency of the Hessian updating schemes minimization can be compared with the FIRE one through these quantities. In the Hessian updating schemes, the convenient system of coordinates for both the gradients and the displacements is the system of normal coordinates, while the more convenient one in the case of FIRE algorithm is the system of cartesian coordinates. Therefore, even if the maximum value and the root-mean-square of the displacements are already computed quantity in CG and BFGS schemes, they are computed in normal coordinates which are not useful for a comparison with the same quantities in the FIRE algorithm. Therefore, the calculation of the maximum value of the nuclei displacements (in cartesian coordinates), of the root-mean-square of the nuclei displacements (in cartesian coordinated) and the euclidean norm of the $3N$ forces vector is implemented in the libopt.f library. The stopping criteria for CG and BFGS remains the default one, based on the absolute value of the largest component of both nuclei displacements and gradients and on the root-mean-square of both the nuclei displacements and the gradients. Therefore, the printed information about the maximum value and root-mean-square of the displacements and the euclidean norm of atomic forces vector are *not* used, in the CG and BFGS schemes, as stopping criteria. The intervention made for comparison purposes in the libopt.f library is therefore a question of merely printing some information.
Some new keywords are introduced to switch on these printing options related to the calculation and the output writings of the quantities described above, which allow comparison between the CG or BFGS algorithms and the FIRE minimization scheme. In the subroutine CONOPT_ of the libopt.f library, the input keywords related to the geometry optimization (OPTGEOM) are read from the input file, so that the case statements for the new keywords are added in this subroutine. Since the global variables common to all the subroutines in libopt.f library are declared in the Memory_opt module written in memory_opt.f90, also this module is modified, with the insertion of the four variables, each corresponding to an input keyword which is a variable used both by the subroutine OPTGEN and by the subroutine CONOPT_ in the libopt.f library. Overall, the modifications done in the libopt.f library and in memory_opt.f90 module are listed below (in the following page), together with the new keywords introduced. The new keywords have to be inserted in the OPTGEOM block section of the input file.

	rec	variable	meaning
•	*A*	**FIREUP**	activate the writing in the standard output file (per each optimization step) of the four convergence criteria adopted by the FIRE algorithm, which is (i) the value of the euclidean norm of the $3N$ nuclei forces vector, (ii) the difference in energy between two consecutive optimization steps, (iii) the largest value of the nuclei displacements (in cartesian coordinates), (iv) the root-mean-square of the nuclei displacements (in cartesian coordinates). The value of the thresholds on the different convergence criteria are the default ones associated with the three keywords described below [default: false]
•	*A*	**THEUNORMF**	modify the threshold on the euclidean norm of $3N$ forces vector computed in cartesian coordinates
	∗	THEUNORMF	$\|\mathbf{F}\| < 10^{-THEUNORMF}$ eV/Å [default: 10]
•	*A*	**THDEF**	modify the threshold on the energy change between optimization steps
	∗	THDEFIRE	$\|\Delta E\| < 10^{-THDEFIRE}$ Ha [default: 8]
•	*A*	**THDRMSF**	modify the threshold on the RMS of the nuclei displacements computed in cartesian coordinates (units: Bohr)
	∗	THDRMS_FIRE	maximum RMS of the nuclei displacements computed in cartesian coordinates [default: 0.0012 Bohr]

Modification to the libopt.f library and memory_opt.f90 module to allow comparison of stopping criteria in CG or BFGS updating scheme in geometry optimization with respect to the Fire structural minimization algorithm $\longrightarrow$ commented as : ! CRBL

Modifications done in the memory_opt.f90 module (**module Memory_opt**):

1. (! FALG1 block) declaration of the following global variables (global variables for the module):
   (a) (logical) fireup : for the printing of (i) the euclidean norm nuclei forces vector, (ii) the difference in the total energy between two consecutive optimization steps and (iii) the maximum value and the root-mean square of the nuclei displacements per each CG or BFGS step, with the state of convergence (satisfied or failed) on these four quantities, with respect to the *default* associated thresholds
   (b) (real) theunormf : threshold related to the euclidean norm of the nuclei $3\times$naf forces vector
   (c) (real) thdefire : threshold related to the difference in the total energy between two consecutive optimization steps
   (d) (real) thdmax_fire : threshold related to the maximum value of the nuclei displacement
   (e) (real) thdrms_fire : threshold related to the root-mean square of the nuclei displacements

Modifications done in the libopt.f library (**subroutine CONOPT_** which reads input OPTGEOM section keywords):

1. (! FALG2 block) set of the default thresholds associated to the variables fireup, thdmax_fire, thdrms_fire, theunormf and thdefire
2. (! FALG3 block) add of five case statements for the following five new keywords:
   (a) keyword FIREUP: allow the printing of (i) the maximum value and the root-mean square of the nuclei displacements, (ii) the difference in the total energy between two steps and (iii) the euclidean norm nuclei forces vector per each CG or BFGS step, with the state of convergence (satisfied or failed) on these four quantities, with respect to the *default* associated thresholds
   (b) keyword THEUNORMF: modify the threshold related to the euclidean norm of the nuclei $3\times$naf forces vector (variable theunormf in the code)
   (c) keyword THDEF: modify the threshold related to the difference in the total energy of two consecutive optimization steps (variable thdefire in the code)
   (d) keyword THDRMSF: modify the threshold related to the root-mean square of the nuclei displacements (variable thdrms_fire in the code). The value associated to thdrms_fire influences the value of the threshold related to the maximum value of the nuclei displacement (variable thdmax_fire in the code), in the following way: thdmax_fire $= 1.5 \cdot$ thdrms_fire

Modifications done in the libopt.f library (**subroutine OPTGEN** which performs standard geometry optimization):

1. (! FALG4 block) declaration of
   (a) matrices xa_prevfire and xa_diff (dimension $3 \times$ naf)
   (b) vector displa_fire (dimension naf)
   (c) real variables dmax_fire, drms_fire, eunormf, en_prevfire and de_fire
2. (! FALG5 block) only if fireup == .true. : allocation (using subroutine CRYALLOC) of
   (a) matrices xa_prevfire and xa_diff (dimension $3 \times$ naf)
   (b) vector displa_fire (dimension naf)

3. (! FALG6 block) only if fireup == .true. : first optimization cycle: calculation of the euclidean norm of the initial nuclei forces vector eunormf and check for the convergence (this is no a stopping criteria, only printings information about the success of the convergence check or about its failure)

4. (! FALG7 block) only if fireup == .true. : storage of the positions of the nuclei of the current $(i-1)$-th step in matrix xa_prevfire, and storage of the value of the total energy of the current $(i-1)$-th step in the variable en_prevfire, before moving atoms in the i-th step through the minimization algorithm

5. (! FALG8 block) only if fireup == .true. : compute the relevant quantities and check convergence:

   (a) per each minimization step, calculation of the euclidean norm of nuclei forces vector eunormf (in cartesian coordinates as reference system, and atomic units), calculation of the difference in total energy de_fire between the actual i-th step and the previous $(i-1)$-th one, calculation of the maximum value of nuclei displacements dmax_fire and the root-mean-square of nuclei displacements drms_fire (in cartesian coordinates as reference system, and atomic units)

   (b) per each minimization step, comparison between the quantities computed in point (a) and the correspondent thresholds set by default or read from the input file in subroutine CONOPT_ (libopt.f library), and check for the convergence (this is no a stopping criteria, only printings information about the success of the convergence check or about its failure)

   The part of the code which performs the previous (a) and (b) tasks, together with the first convergence check on euclidean force norm described at point 3. and the format statements used in this section (and described at point 7.) is reported in the following picture

```
1486 ! CRBL BEGIN 02/2021 compute DMAX_FIRE, DRMS_FIRE, EUNORMF, DE_FIRE ! FALG8
1487 ! Calculations added for comparison with FIRE algorithm (moldyn module)
1488 ! 1. calculation of the euclidean norm of the 3naf forces vector (forces
1489 !    in cartesian coordinates, units Hartree/Bohr)
1490 ! 2. calculation of the total energy difference between two optimization
1491 !    steps
1492 ! 3. calculation of root-mean-square of atomic displacements and maximum
1493 !    atomic displacement (in cartesian coordinates, units Bohr)
1494 ! 4. check for convergence of those quantities (no stopping criteria!)
1495       FALG8: IF(FIREUP) THEN
1496         EUNORMF = NORM2(ATNUG)/SQRT(3*REAL(NAF))
1497         DE_FIRE = EN_PREVFIRE - PAR(6)
1498         XA_DIFF = XA - XA_PREVFIRE
1499         DISPLA_FIRE(:)  = SQRT(XA_DIFF(1,:)**2 + XA_DIFF(2,:)**2
1500      *   + XA_DIFF(3,:)**2)
1501         DMAX_FIRE = MAXVAL(DISPLA_FIRE(1:NAF))
1502         DRMS_FIRE = SQRT(SUM(DISPLA_FIRE**2)/SIZE(DISPLA_FIRE))
1503         PRINTINFO_1: IF(IAMEQ0) THEN
1504           WRITE(IOUT,2221)
1505           IF(EUNORMF > THEUNORMF) THEN
1506             WRITE(IOUT,2222) EUNORMF, THEUNORMF
1507           ELSE
1508             WRITE(IOUT,2223) EUNORMF, THEUNORMF
1509           ENDIF
1510           IF(DE_FIRE > THDEFIRE) THEN
1511             WRITE(IOUT,2224) DE_FIRE, THDEFIRE
1512           ELSE
1513             WRITE(IOUT,2225) DE_FIRE, THDEFIRE
1514           ENDIF
1515           IF(DMAX_FIRE > THDMAX_FIRE) THEN
1516             WRITE(IOUT,2226) DMAX_FIRE, THDMAX_FIRE
1517           ELSE
1518             WRITE(IOUT,2227) DMAX_FIRE, THDMAX_FIRE
1519           ENDIF
1520           IF(DRMS_FIRE > THDRMS_FIRE) THEN
1521             WRITE(IOUT,2228) DRMS_FIRE, THDRMS_FIRE
1522           ELSE
1523             WRITE(IOUT,2229) DRMS_FIRE, THDRMS_FIRE
1524           ENDIF
1525         ENDIF PRINTINFO_1
1526       ENDIF FALG8
1527 ! CRBL END 1202/20
```

```
801 ! CRBL BEGIN 03/2021 compute EUNORMF ! FALG6
802 ! Calculations added for comparison with FIRE algorithm (moldyn module)
803 ! 1. calculation of the euclidean norm of the 3naf forces vector (forces
804 !    in cartesian coordinates, units Hartree/Bohr) and check for convergence
805       FALG6: IF(FIREUP) THEN
806         EUNORMF = NORM2(ATNUG)/SQRT(3*REAL(NAF))
807         PRINTINFO_0: IF(IAMEQ0) THEN
808           WRITE(IOUT,2221)
809           IF(EUNORMF > THEUNORMF) THEN
810             WRITE(IOUT,2222) EUNORMF, THEUNORMF
811           ELSE
812             WRITE(IOUT,2223) EUNORMF, THEUNORMF
813           ENDIF
814         ENDIF PRINTINFO_0
815       ENDIF FALG6
816 ! CRBL END 1202/30
```

```
1771 ! CRBL BEGIN format statements ! FALG10
1772 2221  FORMAT(/1X'PARAMETERS RELATED TO FIRE MINIMIZATION',
1773      * ' (AU UNITS):')
1774 2222  FORMAT(' EUNORM FORC.',T14,ES14.6,T30,'THRESHOLD ',T52,ES9.2,
1775      * ' CONVERGED NO')
1776 2223  FORMAT(' EUNORM FORC.',T14,ES14.6,T30,'THRESHOLD ',T52,ES9.2,
1777      * ' CONVERGED YES')
1778 2224  FORMAT(' ENERGY DIFF.',T14,ES14.6,T30,'THRESHOLD ',T52,ES9.2,
1779      * ' CONVERGED NO')
1780 2225  FORMAT(' ENERGY DIFF.',T14,ES14.6,T30,'THRESHOLD ',T52,ES9.2,
1781      * ' CONVERGED YES')
1782 2226  FORMAT(' MAX DISPLAC.',T19,F9.6,T30,'THRESHOLD ',T52,F9.6,
1783      * ' CONVERGED NO')
1784 2227  FORMAT(' MAX DISPLAC.',T19,F9.6,T30,'THRESHOLD ',T52,F9.6,
1785      * ' CONVERGED YES')
1786 2228  FORMAT(' RMS DISPLAC.',T19,F9.6,T30,'THRESHOLD ',T52,F9.6,
1787      * ' CONVERGED NO')
1788 2229  FORMAT(' RMS DISPLAC.',T19,F9.6,T30,'THRESHOLD ',T52,F9.6,
1789      * ' CONVERGED YES')
1790 ! CRBL END
```

6. (! FALG9 block) only if fireup == .true. : deallocation (using subroutine CRYDEALLOC) of

   (a) matrices xa_prevfire and xa_diff (dimension $3 \times$ naf)

   (b) vector displa_fire (dimension naf)

7. (! FALG10 block) definition of some printings format for writing in the output the values of the quantities computed (maximum value and root-mean-square of the nuclei displacements, euclidean norm of the nuclei forces vector and the difference in total energy between two consecutive steps), the associated thresholds and the state of the convergence process

B.5.3 Changes in geometry.f and libforce6.f libraries

Modifications A and B to perform FULLOPTG and CELLONLY with Fire (i.e. to compute the energy gradient with respect to cell parameters):

A. Modification to the geometry.f library in order to call the subroutine generate_saed in Fire minimization $\longrightarrow$ commented as : ! CRBL

The subroutine generate_saed computes the symmetry allowed elastic distortion, useful for the optimization of the cell parameters and volume in FIRE algorithm (FULLOPTG and CELLONLY FIRE structural optimization options can thus be activated in this way)

Modifications done in the geometry.f library (**subroutine SYMMETRIZE_DIR**):

1. (! FALG11 block) In order to compute the symmetry allowed elastic distortion (so that the variable nsaed != 0) the subroutine generate_saed(logik) has to be used. This is done at the beginning of the computation, in the subroutine symmetrize_dir in geometry.f library, if some conditions are respected. Among these conditions, a further one is adjoined, that is, if the logical variable gensaed_md is true, then the subroutine generate_saed(logik) is called, otherwise not (if the other conditions are not fulfilled). This logical variable is declared in the moldyn_interface module (in moldyn.f90) and set to true in subroutine readFIRE in moldyn.f90, if the keyword FULLOPTG or CELLONLY is found in the input file in the FIRE ... END block (i.e., if the user wants to perform, respectively, a geometry optimization of both the atomic coordinates and the cell parameters or a structural optimization of the cell parameters only, using the FIRE minimization scheme).

   The part of the code modified in geometry.f is reported in the figures below

```
3611 ! CRBL BEGIN ! FALG11
3612       USE MOLDYN_INTERFACE
3613 ! CRBL END
```

```
3641 ! CRBL ADDS CONDITION ON GENSAED_MD IN THE LINE HERE BELOW ! FALG11
3642       IF(INF115.GE.3.AND.NSAED.EQ.0.OR.LELAPIEZ.OR.GENSAED_MD)THEN
3643       CALL GENERATE_SAED(.TRUE.)
3644       ENDIF
```

B. Modification to the libforce6.f library in order to disable the deallocation of the matrix forze_c when FULLOPTG or CELLONLY minimization is performed with Fire algorithm $\longrightarrow$ commented as : ! CRBL

Modifications done in the libforce6.f library (**subroutine TOTGRA_C**):

1. (! FALG12 block) In order to perform a variable cell structural optimization, the information on the energy gradients with respect to the components of the lattice vectors has to be collected. The energy gradients with respect to the cell vectors components are computed in subroutines totgra_c and symmetrize_grad_c in libforce6.f library, and they are stored in forze_c vector, whose size depends on the dimension of the system (forze_c vector has 9, 4 and 1 components for 3D, 2D and 1D systems, respectively). In order to pick the information about the values of forze_c vector elements in the moldyn module, and in particular in the moldyn_scf_driver subroutine, (i) the module cellgrad_memory has to be used to include the variables initialized in that module, and (ii) the vector forze_c has to be yet allocated, filled with values and read inside the moldyn_scf_driver subroutine. However, at the end of the totgra_c subroutine, the forze_c vector is deallocated if inf(194).eq.0 (inf(194) is related to elastic calculation options), so that, if that condition is respected (as in the case of a FIRE minimization calculation), the information about the energy cell gradients in forze_c vector is destroyed. In order to made available the forze_c vector after the subroutine totgra_c, a condition is added for the deallocation of that vector. The first change in the subroutine totgra_c is the insertion of the use of moldyn_interface module, to include a logical variable dealloc_forcell_md that defines a new condition for the deallocation of forze_c vector at the end of the subroutine. The changes made in totgra_c and on the condition for the deallocation of forze_c vector are reported in the following two pieces of code.

```
394 ! CRBL BEGIN ! FALG12
395       USE MOLDYN_INTERFACE
396 ! CRBL END
```

```
828 !wfp
829 !      if(inf(194).eq.0)CALL CRYdealloc(forze_c,ZNAMZ,'FORZE_C')
830 ! CRBL BEGIN ! FALG12
831       IF(INF(194).EQ.0.AND.DEALLOC_FORCELL_MD) THEN
832       CALL CRYDEALLOC(FORZE_C,ZNAMZ,'FORZE_C')
833       ENDIF
834 ! CRBL END
835 !wfp
```

The variable dealloc_forcell_md is a logical variable declared in the moldyn_interface module in moldyn.f90 and it is initialized there as .true.. It becomes .false. when the keyword FULLOPTG and CELLONLY are used in the FIRE ... END block of the input file, i.e. when a variable cell structural optimization is performed and the vector forze_c has to be available in the moldyn_scf_driver subroutine in moldyn.f90.

WARNING: since at the beginning of totgra_c subroutine the vector forze_c is allocated, and since the subroutine totgra_c is called at each self consistent cycle by the subroutine SCF, then the allocation of the vector forze_c is done at each self consistent cycle. Thus, in the case of a FULLOPTG or CELLONLY FIRE minimization calculation (inf(201) = 1), the vector forze_c has to be deallocated in moldyn.f90 before another self consistent cycle is performed through the calling of the subroutine SCF, which in turns calls the subroutine totgra_c where the vector forze_c is allocated. This deallocation is done in the subroutine moldyn_scf_driver (defined in moldyn.f90), after that the forze_c vector has been saved in another vector (forcell), which is a global variable in the moldyn module and it is used to update the components of the lattice vectors during the optimization (this task is performed by the subroutine update_cell). Defining a new vector forcell in which the information contained in forze_c vector is stored is more expensive in the code in terms of memory use, but, at the same time, it makes the code clearer, and avoid the risk to stain the values contained in the vector forze_c. Note that the cost in memory is very low in all cases, because the vector forze_c has at most 9 components (and at least only one). Finally, in the case of a ATOMONLY FIRE minimization calculation, instead, inf(201) = 0 in moldyn.f90, so that the self consistent cycle is performed by the subroutine SCF by calling the subroutine totgra, so the problem does not subsist in this case.

B.6 Black List of changes in CRYSTAL SVN_DEV1218

Modified version: CRYSTAL SVN_DEV1218

For MOLDYN algorithm purpose (moldyn_module module):
Personal changes are released in the code under the comment: ! CRBL

1. input.f90 [subroutine crysta(CONTEXT)]
2. crystal.f90 [subroutine f90main]
3. crystal06.f90 [subroutine f90main]
4. moldyn.f90 is the main of molecular dynamics implementation
5. moldyn_post.f90 is the main of post-processing molecular dynamics implementation

For FIRE algorithm purpose (fire_module module):
Personal changes are released in the code under the comment: ! CRBL
Other comments (if distinctive of the changes) are reported below

1. input.f90 [subroutine crysta(CONTEXT)]
2. crystal.f90 [subroutine f90main]
3. crystal06.f90 [subroutine f90main]
4. memory_opt.f90 [memory_opt module] comment ! FALG1
5. libopt.f [subroutine conopt_] comment ! FALG2 and ! FALG3
   [subroutine optgen] comment ! FALG4, ! FALG5, ! FALG6, ! FALG7, ! FALG8, ! FALG9 and ! FALG10
6. geometry.f [subroutine symmetrize_dir] comment ! FALG11
7. libforce6.f [subroutine totgra_c] comment ! FALG12
8. moldyn.f90 is the main of structural optimization implementation with FIRE algorithm

The **makefile** has been modified

Bug fix in subvdwd3.f related to HF3C functional

Appendix C

Electronic Transport Properties module details

C.1 Code workflow (boltzatorb.f90) for ab initio electronic transport properties calculation

$d = \inf(10)$ dimensionality of the system
tensdim $= d \cdot (d+1)/2 \equiv d_{sym}$
matdim $= d$

In the following, the dimensionality of the system is taken equal to $d = 3$

Allocated arrays and matrices :

- tdf(d_{sym}, ε, χ) $\equiv \Xi$ (d_{sym}, ε, χ) $\quad \chi = 1, ..., \chi_{max}$ [$\chi_{max} = 1$ closed and $\chi_{max} = 2$ open shell]
- int_array(d_{sym}, ε)
- elcond(d_{sym}, μ, T)
- elcond_mat(d, d)
- sigmas(d_{sym}, μ, T)
- sigmas_mat(d, d)
- seebeck(d^2, μ, T)
- seebeck_mat(d, d)
- kappa(d_{sym}, μ, T)
- power(d^2, μ, T)

The Transport Distribution Function tdf(d_{sym}, ε, χ) $\equiv \Xi$ (d_{sym}, ε, χ) is stored as follows :

1. if $d = 1$: $\boldsymbol{\Xi}$(1, ε, χ) $= \begin{pmatrix} \Xi_{xx} \end{pmatrix} = \begin{pmatrix} \Xi_{11} \end{pmatrix}$
2. if $d = 2$: $\boldsymbol{\Xi}$(3, ε, χ) $= \begin{pmatrix} \Xi_{xx} & \Xi_{xy} & \Xi_{yy} \end{pmatrix} = \begin{pmatrix} \Xi_{11} & \Xi_{12} & \Xi_{22} \end{pmatrix}$
3. if $d = 3$: $\boldsymbol{\Xi}$(6, ε, χ) $= \begin{pmatrix} \Xi_{xx} & \Xi_{xy} & \Xi_{xz} & \Xi_{yy} & \Xi_{yz} & \Xi_{zz} \end{pmatrix} = \begin{pmatrix} \Xi_{11} & \Xi_{12} & \Xi_{13} & \Xi_{22} & \Xi_{23} & \Xi_{33} \end{pmatrix}$

Tasks performed :

1. Calculation of the array

$$\text{int\_array\_1}(:,\varepsilon) = \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi(:,\varepsilon,\alpha) \tag{C.1}$$

2. Calculation of elcond array (electrical conductivity $\boldsymbol{\sigma}$)

$$\text{elcond}(:,\mu,T) = \sum_{\varepsilon=1}^{n_\varepsilon} \underbrace{\left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi(:,\varepsilon,\alpha)}_{=\,\text{int\_array\_1}(:,\varepsilon)}\,\Delta\varepsilon \tag{C.2}$$

$$\text{elcond}(\ d_{sym},\ \mu,\ T\ ) = \begin{pmatrix}\sigma_1 & \sigma_2 & \dots & \sigma_{d_{sym}}\end{pmatrix} \tag{C.3}$$

$$\sigma_{ij}(\mu,T) = \int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi_{ij}(\varepsilon) \tag{C.4}$$

3. Calculation of the array

$$\text{int\_array\_2}(:,\varepsilon) = \underbrace{\left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi(:,\varepsilon,\alpha)}_{=\,\text{int\_array\_1}}(\varepsilon-\mu) \tag{C.5}$$

4. Calculation of sigmas array (electrical conductivity $\boldsymbol{\sigma}$ times Seebeck coefficient $\boldsymbol{S}$)

$$\text{sigmas}(:,\mu,T) = \frac{1}{T}\sum_{\varepsilon=1}^{n_\varepsilon} \underbrace{\left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi(:,\varepsilon,\alpha)\,(\varepsilon-\mu)}_{=\,\text{int\_array\_2}(:,\varepsilon)}\,\Delta\varepsilon \tag{C.6}$$

$$\text{sigmas}(\ d_{sym},\ \mu,\ T\ ) = \begin{bmatrix}(\sigma\,S)_1 & (\sigma\,S)_2 & \dots & (\sigma\,S)_{d_{sym}}\end{bmatrix} \tag{C.7}$$

$$\Sigma_{ij}(\mu,T) \equiv (\sigma\,S)_{ij}(\mu,T) = \frac{1}{T}\int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi_{ij}(\varepsilon)\,(\varepsilon-\mu) \tag{C.8}$$

5. Calculation of elcond_mat matrix by symmetrization of elcond array :

Example : $d = 3$ ($d_{sym} = 6$)

$$\text{elcond}(\ 6,\ \mu,\ T\ ) = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6\end{pmatrix}$$

$$\text{elcond\_mat}(\ 3,\ 3\ ) = \begin{pmatrix}\sigma_{11} & \sigma_{12} & \sigma_{13}\\ \sigma_{21} & \sigma_{22} & \sigma_{23}\\ \sigma_{31} & \sigma_{32} & \sigma_{33}\end{pmatrix} = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3\\ \sigma_2 & \sigma_4 & \sigma_5\\ \sigma_3 & \sigma_5 & \sigma_6\end{pmatrix}$$

6. Calculation of sigmas_mat matrix by symmetrization of sigmas array :

Example : $d = 3$ ($d_{sym} = 6$)

$$\text{sigmas}(\ 6,\ \mu,\ T\ ) = \begin{bmatrix}(\sigma\,S)_1 & (\sigma\,S)_2 & (\sigma\,S)_3 & (\sigma\,S)_4 & (\sigma\,S)_5 & (\sigma\,S)_6\end{bmatrix}$$

$$\text{sigmas\_mat}(\ 3,\ 3\ ) = \begin{pmatrix}\Sigma_{11} & \Sigma_{12} & \Sigma_{13}\\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23}\\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33}\end{pmatrix} = \begin{pmatrix}(\sigma\,S)_1 & (\sigma\,S)_2 & (\sigma\,S)_3\\ (\sigma\,S)_2 & (\sigma\,S)_4 & (\sigma\,S)_5\\ (\sigma\,S)_3 & (\sigma\,S)_5 & (\sigma\,S)_6\end{pmatrix}$$

7. Calculation of the Seebeck coefficient $\boldsymbol{S}$:

$$\boldsymbol{\Sigma}(\mu,T) \equiv (\boldsymbol{\sigma}\,\boldsymbol{S})(\mu,T) = \frac{1}{T}\int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial \varepsilon}\right) \boldsymbol{\Xi}(\varepsilon)\,(\varepsilon-\mu) \tag{C.9}$$

$$\boldsymbol{\sigma}^{-1}(\mu,T)\,\boldsymbol{\Sigma}(\mu,T) = \boldsymbol{\sigma}^{-1}(\mu,T)\left[\frac{1}{T}\int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial \varepsilon}\right) \boldsymbol{\Xi}(\varepsilon)\,(\varepsilon-\mu)\right] \tag{C.10}$$

$$\boldsymbol{S}(\mu,T) = \boldsymbol{\sigma}^{-1}(\mu,T)\left[\frac{1}{T}\int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial \varepsilon}\right) \boldsymbol{\Xi}(\varepsilon)\,(\varepsilon-\mu)\right] \equiv \boldsymbol{\sigma}^{-1}(\mu,T)\,\boldsymbol{\Sigma}(\mu,T) \tag{C.11}$$

Three cases, on the base of the system dimensionality, that are

(a) if $d = 1$:

$$S_{11} = -\frac{\Sigma_{11}}{\sigma_{11}} \qquad \rightarrow \qquad \text{seebeck\_mat(1, 1)} = S_{11}$$

(b) if $d = 2$:

$$\det(\boldsymbol{\sigma}) = \sigma_{11}\,\sigma_{22} - \sigma_{12}\,\sigma_{12}$$

$$\boldsymbol{\sigma}^{-1}(\mu,T) = \frac{1}{\det(\boldsymbol{\sigma})}\begin{pmatrix} \sigma_{22} & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11} \end{pmatrix}$$

$$\boldsymbol{S}(\mu,T) = -\boldsymbol{\sigma}^{-1}(\mu,T)\,\boldsymbol{\Sigma}(\mu,T) \qquad [\text{ matmul Fortran intrinsic function }]$$

$$\rightarrow \qquad \text{seebeck\_mat(2, 2)} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

(c) if $d = 3$:

calculation of $\boldsymbol{\sigma}^{-1}(\mu,T)$ using CRYSTAL function inv_mat3 $\rightarrow \boldsymbol{\sigma}^{-1}(\mu,T)$

$$\boldsymbol{S}(\mu,T) = -\boldsymbol{\sigma}^{-1}(\mu,T)\,\boldsymbol{\Sigma}(\mu,T) \qquad [\text{ matmul Fortran intrinsic function }]$$

$$\rightarrow \qquad \text{seebeck\_mat(3, 3)} = \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{pmatrix}$$

Note : the sign of the Seebeck coefficient matrix has been inverted because the electron charge is negative ($-e < 0$)

8. Reshape of Seebeck matrix in array : seebeck(:, μ, T) = pack(seebeck_mat, .true.)

$$\text{if } d = 1 \qquad \rightarrow \qquad \text{seebeck(1, } \mu, T \text{)} = \begin{pmatrix} S_{11} \end{pmatrix} \tag{C.12}$$

$$\text{if } d = 2 \qquad \rightarrow \qquad \text{seebeck(4, } \mu, T \text{)} = \begin{pmatrix} S_{11} & S_{21} & S_{12} & S_{22} \end{pmatrix} \tag{C.13}$$

$$\text{if } d = 3 \qquad \rightarrow \qquad \text{seebeck(9, } \mu, T \text{)} = \begin{pmatrix} S_{11} & S_{21} & S_{31} & S_{12} & S_{22} & S_{32} & S_{13} & S_{23} & S_{33} \end{pmatrix} \tag{C.14}$$

9. Calculation of the array

$$\text{int\_array\_3}(:,\varepsilon) = \underbrace{\left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial \varepsilon}\right) \Xi(:,\varepsilon,\alpha)\,(\varepsilon-\mu)}_{=\text{ int\_array\_2}}\,(\varepsilon-\mu) \tag{C.15}$$

10. Calculation of kappa array (electrical contribution to the thermal conductivity $\boldsymbol{\kappa}$) :

$$\text{kappa}(:,\mu,T) = \frac{1}{T}\sum_{\varepsilon=1}^{n_\varepsilon} \underbrace{\left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi(:,\varepsilon,\alpha)\,(\varepsilon-\mu)^2}_{=\text{ int\_array\_3}(:,\varepsilon)}\,\Delta\varepsilon \tag{C.16}$$

$$\text{kappa}(\ d_{sym},\ \mu,\ T\ ) = \begin{bmatrix}\kappa_1 & \kappa_2 & \dots & \kappa_{d_{sym}}\end{bmatrix} \tag{C.17}$$

$$\kappa_{ij}(\mu,T) = \frac{1}{T}\int d\varepsilon \left(-\frac{\partial f_{eq}(\varepsilon,\mu,T)}{\partial\varepsilon}\right)\Xi_{ij}(\varepsilon)\,(\varepsilon-\mu)^2 \tag{C.18}$$

11. Calculation of the power array (power factor $\boldsymbol{P}$) :
power(ij, :, :) = elcond_mat(j, i) * seebeck(ij, :, :)**2

(a) if $d = 1$ ($d_{sym} = 1$) :

elcond(1, μ, T) $= \begin{pmatrix}\sigma_1\end{pmatrix}$

elcond_mat(1, 1) $= \begin{pmatrix}\sigma_{11}\end{pmatrix} = \begin{pmatrix}\sigma_1\end{pmatrix}$

seebeck(1, μ, T) $= \begin{pmatrix}S_{11}\end{pmatrix} = S_1$

power(1, μ, T) $= \sigma_{11}\,(S_1)^2 = \sigma_{11}\,(S_{11})^2 = P_{xx}$

(b) if $d = 2$ ($d_{sym} = 3$) :

elcond(3, μ, T) $= \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3\end{pmatrix}$

elcond_mat(2, 2) $= \begin{pmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{pmatrix} = \begin{pmatrix}\sigma_1 & \sigma_2\\ \sigma_2 & \sigma_3\end{pmatrix}$

seebeck(4, μ, T) $= \begin{pmatrix}S_{11} & S_{21} & S_{12} & S_{22}\end{pmatrix} = \begin{pmatrix}S_1 & S_2 & S_3 & S_4\end{pmatrix}$

$$\begin{aligned}\text{power}(\ 4,\ \mu,\ T\ ) &= \begin{bmatrix}\sigma_{11}(S_1)^2 & \sigma_{21}(S_2)^2 & \sigma_{12}(S_3)^2 & \sigma_{22}(S_4)^2\end{bmatrix}\\ &= \begin{bmatrix}\sigma_{11}(S_{11})^2 & \sigma_{21}(S_{21})^2 & \sigma_{12}(S_{12})^2 & \sigma_{22}(S_{22})^2\end{bmatrix}\\ &= \begin{bmatrix}P_{xx} & P_{yx} & P_{xy} & P_{yy}\end{bmatrix}\end{aligned}$$

(c) if $d = 3$ ($d_{sym} = 6$) :

elcond(6, μ, T) $= \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6\end{pmatrix}$

elcond_mat(3, 3) $= \begin{pmatrix}\sigma_{11} & \sigma_{12} & \sigma_{13}\\ \sigma_{21} & \sigma_{22} & \sigma_{23}\\ \sigma_{31} & \sigma_{32} & \sigma_{33}\end{pmatrix} = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3\\ \sigma_2 & \sigma_4 & \sigma_5\\ \sigma_3 & \sigma_5 & \sigma_6\end{pmatrix}$

$$\begin{aligned}\text{seebeck}(\ 9,\ \mu,\ T\ ) &= \begin{pmatrix}S_{11} & S_{21} & S_{31} & S_{12} & S_{22} & S_{32} & S_{13} & S_{23} & S_{33}\end{pmatrix}\\ &= \begin{pmatrix}S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9\end{pmatrix}\end{aligned}$$

$$\begin{aligned}&\text{power}(\ 9,\ \mu,\ T\ ) =\\ &= \begin{bmatrix}\sigma_{11}\,S_1^2 & \sigma_{21}\,S_2^2 & \sigma_{31}\,S_3^2 & \sigma_{12}\,S_4^2 & \sigma_{22}\,S_5^2 & \sigma_{32}\,S_6^2 & \sigma_{13}\,S_7^2 & \sigma_{23}\,S_8^2 & \sigma_{33}\,S_9^2\end{bmatrix}\\ &= \begin{bmatrix}\sigma_{11}\,S_{11}^2 & \sigma_{21}\,S_{21}^2 & \sigma_{31}\,S_{31}^2 & \sigma_{12}\,S_{12}^2 & \sigma_{22}\,S_{22}^2 & \sigma_{32}\,S_{32}^2 & \sigma_{13}\,S_{13}^2 & \sigma_{23}\,S_{23}^2 & \sigma_{33}\,S_{33}^2\end{bmatrix}\\ &= \begin{bmatrix}P_{xx} & P_{yx} & P_{zx} & P_{xy} & P_{yy} & P_{zy} & P_{xz} & P_{yz} & P_{zz}\end{bmatrix}\end{aligned}$$

C.2 Bug report 1 (boltzatorb.f90)

Summary : In boltzatorb.f90 the array power was being multiplied by the array elcond. In 2 and 3 dimensions these have incompatible dimensions, thus the operation is illegal. The bug is detected by the gnu compiler when debugging flags are turned on, but not by the Intel compiler

$d = \inf(10)$ dimensionality of the system
tensdim $= d \cdot (d+1)/2$
matdim $= d$

In the following, the dimensionality of the system is taken equal to $d = 3$

Allocated arrays and matrices :

- elcond(6, μ, T)
- elcond_mat(3, 3)
- seebeck(9, μ, T)
- seebeck_mat(3, 3)
- power(9, μ, T)

Tasks performed :

1. Calculation of elcond three-dimensional array :

$$\text{elcond(6, } \mu, T \text{)} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 \end{pmatrix} \tag{C.19}$$

2. Calculation of elcond_mat matrix by symmetrization of elcond array :

$$\text{elcond\_mat(3, 3)} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_2 & \sigma_4 & \sigma_5 \\ \sigma_3 & \sigma_5 & \sigma_6 \end{pmatrix} \tag{C.20}$$

3. Calculation of Seebeck matrix :

$$\text{seebeck\_mat(3, 3)} = \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{pmatrix} \tag{C.21}$$

4. Reshape of Seebeck matrix in array : seebeck(:, μ, T) = pack(seebeck_mat, .true.)

$$\text{seebeck(9, } \mu, T \text{)} = \begin{pmatrix} S_{11} & S_{21} & S_{31} & S_{12} & S_{22} & S_{32} & S_{13} & S_{23} & S_{33} \end{pmatrix} \tag{C.22}$$

5. Calculation of the power factor : power = elcond*seebeck**2
the power factor is allocated as a three dimensional array power(9, μ, T) is here reshaped in a three dimensional array with dimension along the first direction equal to 6 instead of 9, that is it becomes power(6, μ, T)

$$\begin{aligned} \text{power(6, } \mu, T \text{)} &= \begin{pmatrix} \sigma_1 S_{11}^2 & \sigma_2 S_{21}^2 & \sigma_3 S_{31}^2 & \sigma_4 S_{12}^2 & \sigma_5 S_{22}^2 & \sigma_6 S_{32}^2 \end{pmatrix} \\ &= \begin{pmatrix} \sigma_{11} S_{11}^2 & \sigma_{21} S_{21}^2 & \sigma_{31} S_{31}^2 & \sigma_{22} S_{12}^2 & \sigma_{32} S_{22}^2 & \sigma_{33} S_{32}^2 \end{pmatrix} \end{aligned} \tag{C.23}$$

```
ij = 0
do i = 1, size( elcond_mat, dim = 2 )
   do j = 1, size( elcond_mat, dim = 1 )
      ij = ij + 1
      power2( ij, :, : ) = elcond_mat( j, i ) * seebeck( ij, :, : ) ** 2
   enddo
enddo
```

6. New formula for the power factor :

$$\begin{aligned}&\text{power}(\ 9,\ \mu,\ T\ )\\ &\quad= \left(\sigma_{11}\,S_{11}^2 \quad \sigma_{21}\,S_{21}^2 \quad \sigma_{31}\,S_{31}^2 \quad \sigma_{12}\,S_{12}^2 \quad \sigma_{22}\,S_{22}^2 \quad \sigma_{32}\,S_{32}^2 \quad \sigma_{13}\,S_{13}^2 \quad \sigma_{23}\,S_{23}^2 \quad \sigma_{33}\,S_{33}^2\right)\\ &\quad= \left(\sigma_{1}\,S_{11}^2 \quad \sigma_{2}\,S_{21}^2 \quad \sigma_{3}\,S_{31}^2 \quad \sigma_{2}\,S_{12}^2 \quad \sigma_{4}\,S_{22}^2 \quad \sigma_{5}\,S_{32}^2 \quad \sigma_{3}\,S_{13}^2 \quad \sigma_{5}\,S_{23}^2 \quad \sigma_{6}\,S_{33}^2\right)\end{aligned}$$

C.2.1 Tests

Version : SVN_DEV1286 Pcrystal
ARCH file used for compiling : Linux-mpigfortran-debug.inc

- **Test case 0**.
  Silicon crystalline system
  – Input files : silicon_bug.d12 and silicon_bug.d3
  – Output files : silicon_bug.out and silicon_bug.outp
  The output file of the property calculation contains the error message

Version : SVN_DEV1288 Pcrystal
ARCH file used for compiling : Linux-ifort_openmpi_i64_clut_debug.inc

- **Test case 1**.
  Bi_2Te_3 hexagonal crystalline system
  – Input files : bi2te3.d12 and bi2te3.d3
  – Output files : bi2te3.out and bi2te3.outp
  The output file .outp does not show the error message (the compilator hides it)
  See Figure C.1 for debug printings from output file (that follows the equations above)

- **Test case 2**.
  TiNiSn cubic crystalline system
  – Input files : tinisn_pbe.d12 and tinisn_pbe.d3
  – Output files : tinisn_pbe.out and tinisn_pbe.outp
  The output file .outp does not show the error message (the compilator hides it)
  See Figure C.7 for debug printings from output file (that follows the equations above)

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE      216.26 TCPU      215.50
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY,XZ,YZ,ZZ DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE      401.64 TCPU      400.83

*********************************************************************************
 TEMPERATURE :    251   CHEMICAL POTENTIAL :      3
*********************************************************************************

 SHAPE ELCOND             6         251           3
 elcond(   1 ) :   1.46083411E+07
 elcond(   2 ) :  -1.26883724E+01
 elcond(   3 ) :  -8.37828986E-03
 elcond(   4 ) :   1.46083264E+07
 elcond(   5 ) :   1.45115969E-02
 elcond(   6 ) :   1.54662534E+07

 SHAPE ELCOND_MAT             3           3
  1.46083411E+07 -1.26883724E+01 -8.37828986E-03
 -1.26883724E+01  1.46083264E+07  1.45115969E-02
 -8.37828986E-03  1.45115969E-02  1.54662534E+07

 SHAPE SEEBECK_MAT             3           3
  3.06841583E-06 -4.20263040E-10  1.60397202E-14
 -4.20263040E-10  3.06793055E-06 -2.77821951E-14
  2.70552851E-14 -4.68606794E-14 -1.89078758E-05

 SHAPE SEEBECK             9         251           3
 seebeck(   1 ) :   3.06841583E-06
 seebeck(   2 ) :  -4.20263040E-10
 seebeck(   3 ) :   2.70552851E-14
 seebeck(   4 ) :  -4.20263040E-10
 seebeck(   5 ) :   3.06793055E-06
 seebeck(   6 ) :  -4.68606794E-14
 seebeck(   7 ) :   1.60397202E-14
 seebeck(   8 ) :  -2.77821951E-14
 seebeck(   9 ) :  -1.89078758E-05

 SHAPE POWER BEFORE MULTIPLICATION            9         251           3

 HERE THE CALCULATION POWER = ELCOND*SEEBECK**2 IS PERFORMED ...

 SHAPE POWER AFTER CALCULATION                6         251           3
 power(   1 ) :   1.37540098E-04
 power(   2 ) :  -2.24103330E-18
 power(   3 ) :  -6.13281142E-30
 power(   4 ) :   2.58013756E-12
 power(   5 ) :   1.36586021E-13
 power(   6 ) :   3.39627059E-20


 CORRECT FORMULA FOR POWER ....

 SHAPE POWER              9         251           3
 power(   1 ) :   1.37540098E-04
 power(   2 ) :  -2.24103330E-18
 power(   3 ) :  -6.13281142E-30
 power(   4 ) :  -2.24103330E-18
 power(   5 ) :   1.37496459E-04
 power(   6 ) :   3.18663534E-29
 power(   7 ) :  -2.15550461E-30
 power(   8 ) :   1.12007814E-29
 power(   9 ) :   5.52930575E-03

*********************************************************************************
```

Figure C.1: Debug printings for transport properties calculation of Bi_2Te_3 hexagonal crystalline system.

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE        3.15 TCPU        3.04
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY,XZ,YZ,ZZ DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE       11.35 TCPU       11.24

*********************************************************************************
 TEMPERATURE :    901   CHEMICAL POTENTIAL :      7
*********************************************************************************

 SHAPE ELCOND             6         901           7
 elcond(   1 ) :   6.38287317E+06
 elcond(   2 ) :   6.62217137E-09
 elcond(   3 ) :   5.92912316E-09
 elcond(   4 ) :   6.38287317E+06
 elcond(   5 ) :   9.42029139E-09
 elcond(   6 ) :   6.48535517E+06

 SHAPE ELCOND_MAT             3           3
  6.38287317E+06  6.62217137E-09  5.92912316E-09
  6.62217137E-09  6.38287317E+06  9.42029139E-09
  5.92912316E-09  9.42029139E-09  6.48535517E+06

 SHAPE SEEBECK_MAT             3           3
  3.17238219E-05 -3.73551138E-20 -2.28927716E-20
 -3.73551138E-20  3.17238219E-05  4.25945896E-20
 -2.30756210E-20  4.10562336E-20  3.11281283E-05

 SHAPE SEEBECK             9         901           7
 seebeck(   1 ) :   3.17238219E-05
 seebeck(   2 ) :  -3.73551138E-20
 seebeck(   3 ) :  -2.30756210E-20
 seebeck(   4 ) :  -3.73551138E-20
 seebeck(   5 ) :   3.17238219E-05
 seebeck(   6 ) :   4.10562336E-20
 seebeck(   7 ) :  -2.28927716E-20
 seebeck(   8 ) :   4.25945896E-20
 seebeck(   9 ) :   3.11281283E-05

 SHAPE POWER BEFORE MULTIPLICATION            9         901           7

 HERE THE CALCULATION POWER = ELCOND*SEEBECK**2 IS PERFORMED ...

 SHAPE POWER AFTER CALCULATION                6         901           7
 power(   1 ) :   6.42372913E-03
 power(   2 ) :   9.24060793E-48
 power(   3 ) :   3.15716491E-48
 power(   4 ) :   8.90669014E-33
 power(   5 ) :   9.48058948E-18
 power(   6 ) :   1.09318075E-32


 CORRECT FORMULA FOR POWER ....

 SHAPE POWER              9         901           7
 power(   1 ) :   6.42372913E-03
 power(   2 ) :   9.24060793E-48
 power(   3 ) :   3.15716491E-48
 power(   4 ) :   9.24060793E-48
 power(   5 ) :   6.42372913E-03
 power(   6 ) :   1.58789780E-47
 power(   7 ) :   3.10732889E-48
 power(   8 ) :   1.70912259E-47
 power(   9 ) :   6.28405217E-03

*********************************************************************************
```

Figure C.2: Debug printings for transport properties calculation of TiNiSn cubic crystalline system.

C.3 Bug report 1 (boltzatorb.f90)

Summary : In boltzatorb.f90 the arrays elcond and sigmas are used to define the symmetric matrices elcond_mat and sigmas_mat. However, the way in which the indexes are computed for the mapping from the arrays to the matrices is not general. In particular, it is wrong for the case of two-dimensional systems. The bug is detected by the GNU compiler and by the Intel compiler when debugging flags are turned on.

d = matdim = inf(10) dimensionality of the system
t = tensdim = $d \cdot (d+1)/2$

Allocated arrays and matrices :

- elcond(t, μ, T)
- elcond_mat(d, d)
- sigmas(t, μ, T)
- sigmas_mat(d, d)

Tasks performed : (case $d = 3 \rightarrow t = 6$)

1. Calculation of elcond three-dimensional array :

$$\text{elcond}(\ 6,\ \mu,\ T\ ) = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6\end{pmatrix} \tag{C.24}$$

2. Calculation of elcond_mat matrix by symmetrization of elcond array :

$$\text{elcond\_mat}(\ 3,\ 3\ ) = \begin{pmatrix}\sigma_{11} & \sigma_{12} & \sigma_{13}\\ \sigma_{21} & \sigma_{22} & \sigma_{23}\\ \sigma_{31} & \sigma_{32} & \sigma_{33}\end{pmatrix} = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3\\ \sigma_2 & \sigma_4 & \sigma_5\\ \sigma_3 & \sigma_5 & \sigma_6\end{pmatrix} \tag{C.25}$$

Algorithm used:

```
. do i = 1, inf(10)
.     do j = 1, i
.         rpunt = i + (j - 1) · j/2^(j-2)
.         elcond_mat( i, j ) = elcond( rpunt, ichempot, itemp )
.         elcond_mat( j, i ) = elcond_mat( i, j )
.     enddo
. enddo
```

3. The same steps 1. and 2. are performed for the array sigmas and the matrix sigmas_mat, and the mapping between the array and the matrix is done using the same algorithm described before.

Problem :
In the case of system dimensionality equal to $d = 2 \rightarrow t = 3$, the elcond array is given by

$$\text{elcond}(\ 3,\ \mu,\ T\ ) = \begin{pmatrix}\sigma_1 & \sigma_2 & \sigma_3\end{pmatrix} \tag{C.26}$$

However, the previous algorithm performs the following map from the elcond array to elcond_mat matrix:

$$\text{elcond\_mat}(\ 2,\ 2\ ) = \begin{pmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{pmatrix} = \begin{pmatrix}\sigma_1 & \sigma_2\\ \sigma_2 & \sigma_4\end{pmatrix} \tag{C.27}$$

But the σ_4 element has not been defined nor calculated, so that the σ_{22} element in the matrix is not correctly computed.

Suggested formula :

```
! elcond to elcond_mat mapping
ij = 1
k  = 1
do i = 1, inf(10)
   do j = k, inf(10)
      elcond_mat( i, j ) = elcond( ij, ichempot, itemp )
      elcond_mat( j, i ) = elcond_mat( i, j )
      ij = ij + 1
   enddo
   k = k + 1
enddo

! sigmas to sigmas_mat mapping
ij = 1
k  = 1
do i = 1, inf(10)
   do j = k, inf(10)
      sigmas_mat( i, j ) = sigmas( ij, ichempot, itemp )
      sigmas_mat( j, i ) = sigmas_mat( i, j )
      ij = ij + 1
   enddo
   k = k + 1
enddo
```

Figure C.3: Formula for the mapping from arrays to symmetric matrices.

Other nasty bug :
The logical variable info_det in boltzatorb.f90 has been declared but not always initialized. It is initialized to true only if a series of if statements are satisfied. If none of these conditions are satisfied, the variable remains initialized and when it is used later on in the code, the program does not know what value has to be assigned to it
The error associated to this bug is detected with the Intel compiler only if debug flags are activated (see Linux-ifort_openmpi_i64_clut_debug.inc)
Suggestion : initialized info_det = .false. at the beginning of boltzao_main subroutine

C.3.1 Tests

Version : SVN_DEV1288 Pcrystal
ARCH file used for compiling : Linux-ifort_openmpi_i64_clut_debug.inc

- **Test case 0**.
  Berillium 4 layers slab system

  Calculations run with public version:
  – Input files : P_test06.d12 and P_test06.d3
  – Output files : P_test06.out and P_test06.outp
  The output file of the property calculation contains the error message (see Figure C.4)

  Calculations run with new version:
  – Input files : P_test06.d12 and P_test06_bug_fix.d3
  – Output files : P_test06.out and P_test06_bug_fix.outp
  See Figure C.5 for debug printings in the output file (that follows the algorithm in Figure C.3)

- **Test case 1**.
  TiNiSn cubic crystalline system

  Calculations run with old version:
  – Input files : tinisn_pbe.d12 and tinisn_pbe_bug_fix.d3
  – Output files : tinisn_pbe.out and tinisn_pbe.outp
  See Figure C.6 for debug printings in the output file (that follows the algorithm in Figure C.3)

  Calculations run with new version:
  – Input files : tinisn_pbe.d12 and tinisn_pbe_bug_fix.d3
  – Output files : tinisn_pbe.out and tinisn_pbe_bug_fix.outp

  See Figure C.7 for debug printings in the output file
  (that follows the algorithm in Figure C.3).

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE        0.72 TCPU        0.69
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE        0.93 TCPU        0.90
forrtl: severe (408): fort: (2): Subscript #1 of the array ELCOND has value 4 which is greater than the upper bound of 3

Image              PC                Routine            Line        Source
Pproperties-30044  000000000D78F00F  Unknown               Unknown  Unknown
Pproperties-30044  00000000012400F8  boltzao_mp_boltza         392  boltzatorb.f90
Pproperties-30044  000000000040F47B  f90main3_                 299  properties.f90
Pproperties-30044  00000000004197CE  MAIN__                      3  properties.f90
Pproperties-30044  000000000040DA22  Unknown               Unknown  Unknown
libc-2.28.so       00007FE2839EB09B  __libc_start_main     Unknown  Unknown
Pproperties-30044  000000000040D92A  Unknown               Unknown  Unknown
forrtl: severe (408): fort: (2): Subscript #1 of the array ELCOND has value 4 which is greater than the upper bound of 3
```

Figure C.4: Error in the output file for transport properties calculation of Berillium 4 layers slab system (public version).

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE        0.71 TCPU        0.68
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE        0.92 TCPU        0.89

** NEW VERSION : BEGIN **************

 elcond to elcond_mat ****************
 i =            1    j =            1    ij =            1
 i =            1    j =            2    ij =            2
 i =            2    j =            2    ij =            3
 elcond to elcond_mat ****************

 sigmas to sigmas_mat ****************
 i =            1    j =            1    ij =            1
 i =            1    j =            2    ij =            2
 i =            2    j =            2    ij =            3
 sigmas to sigmas_mat ****************

*******************************************************************************
 TEMPERATURE :    801   CHEMICAL POTENTIAL :      1
*******************************************************************************

 SHAPE ELCOND             3         801           1
 elcond(   1 ) :   0.00000000E+00
 elcond(   2 ) :   0.00000000E+00
 elcond(   3 ) :   0.00000000E+00

 SHAPE ELCOND_MAT             2           2
  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00

 SHAPE SIGMAS                 3         801           1
 sigmas(   1 ) :   0.00000000E+00
 sigmas(   2 ) :   0.00000000E+00
 sigmas(   3 ) :   0.00000000E+00

 SHAPE SIGMAS_MAT             2           2
  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00

** NEW VERSION : END ****************
```

Figure C.5: Debug printings for transport properties calculation of Berillium 4 layers slab system (new version).

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE        3.15 TCPU        3.04
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY,XZ,YZ,ZZ DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE       11.35 TCPU       11.24

*****************************************************************************
 TEMPERATURE :    901   CHEMICAL POTENTIAL :      7
*****************************************************************************

 SHAPE ELCOND             6         901           7
 elcond(   1 ) :   6.38287317E+06
 elcond(   2 ) :   6.62217137E-09
 elcond(   3 ) :   5.92912316E-09
 elcond(   4 ) :   6.38287317E+06
 elcond(   5 ) :   9.42029139E-09
 elcond(   6 ) :   6.48535517E+06

 SHAPE ELCOND_MAT             3           3
  6.38287317E+06  6.62217137E-09  5.92912316E-09
  6.62217137E-09  6.38287317E+06  9.42029139E-09
  5.92912316E-09  9.42029139E-09  6.48535517E+06

 SHAPE SEEBECK_MAT             3           3
  3.17238219E-05 -3.73551138E-20 -2.28927716E-20
 -3.73551138E-20  3.17238219E-05  4.25945896E-20
 -2.30756210E-20  4.10562336E-20  3.11281283E-05

 SHAPE SEEBECK             9         901           7
 seebeck(   1 ) :   3.17238219E-05
 seebeck(   2 ) :  -3.73551138E-20
 seebeck(   3 ) :  -2.30756210E-20
 seebeck(   4 ) :  -3.73551138E-20
 seebeck(   5 ) :   3.17238219E-05
 seebeck(   6 ) :   4.10562336E-20
 seebeck(   7 ) :  -2.28927716E-20
 seebeck(   8 ) :   4.25945896E-20
 seebeck(   9 ) :   3.11281283E-05

 SHAPE POWER BEFORE MULTIPLICATION            9         901           7

 HERE THE CALCULATION POWER = ELCOND*SEEBECK**2 IS PERFORMED ...

 SHAPE POWER AFTER CALCULATION                6         901           7
 power(   1 ) :   6.42372913E-03
 power(   2 ) :   9.24060793E-48
 power(   3 ) :   3.15716491E-48
 power(   4 ) :   8.90669014E-33
 power(   5 ) :   9.48058948E-18
 power(   6 ) :   1.09318075E-32


 CORRECT FORMULA FOR POWER ....

 SHAPE POWER              9         901           7
 power(   1 ) :   6.42372913E-03
 power(   2 ) :   9.24060793E-48
 power(   3 ) :   3.15716491E-48
 power(   4 ) :   9.24060793E-48
 power(   5 ) :   6.42372913E-03
 power(   6 ) :   1.58789780E-47
 power(   7 ) :   3.10732889E-48
 power(   8 ) :   1.70912259E-47
 power(   9 ) :   6.28405217E-03

*****************************************************************************
```

Figure C.6: Debug printings for transport properties calculation of TiNiSn cubic crystalline system (old version).

```
 TDF CALCULATION IN PROGRESS...
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTT SMAT        TELAPSE        9.33 TCPU        8.68
 TDF CALCULATION PERFORMED
 TDF CALCULATED ALONG THE XX,XY,YY,XZ,YZ,ZZ DIRECTIONS
 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTT TDF_CALC    TELAPSE       76.57 TCPU       75.91

** NEW VERSION : BEGIN **************

 elcond to elcond_mat ****************
 i =            1    j =            1    ij =            1
 i =            1    j =            2    ij =            2
 i =            1    j =            3    ij =            3
 i =            2    j =            2    ij =            4
 i =            2    j =            3    ij =            5
 i =            3    j =            3    ij =            6
 elcond to elcond_mat ****************

 sigmas to sigmas_mat ****************
 i =            1    j =            1    ij =            1
 i =            1    j =            2    ij =            2
 i =            1    j =            3    ij =            3
 i =            2    j =            2    ij =            4
 i =            2    j =            3    ij =            5
 i =            3    j =            3    ij =            6
 sigmas to sigmas_mat ****************

********************************************************************************
 TEMPERATURE :    901   CHEMICAL POTENTIAL :      7
********************************************************************************

 SHAPE ELCOND             6         901           7
 elcond(   1 ) :   1.72601733E+00
 elcond(   2 ) :   1.09543857E-15
 elcond(   3 ) :   1.72629800E-15
 elcond(   4 ) :   1.72601733E+00
 elcond(   5 ) :   6.31295301E-16
 elcond(   6 ) :   1.75373006E+00

 SHAPE ELCOND_MAT             3           3
  1.72601733E+00  1.09543857E-15  1.72629800E-15
  1.09543857E-15  1.72601733E+00  6.31295301E-16
  1.72629800E-15  6.31295301E-16  1.75373006E+00

 SHAPE SIGMAS                 6         901           7
 sigmas(   1 ) :  -5.47563016E-05
 sigmas(   2 ) :  -1.02263160E-19
 sigmas(   3 ) :  -1.17625237E-19
 sigmas(   4 ) :  -5.47563016E-05
 sigmas(   5 ) :  -4.94336450E-20
 sigmas(   6 ) :  -5.45907657E-05

 SHAPE SIGMAS_MAT             3           3
 -5.47563016E-05 -1.02263160E-19 -1.17625237E-19
 -1.02263160E-19 -5.47563016E-05 -4.94336450E-20
 -1.17625237E-19 -4.94336450E-20 -5.45907657E-05

** NEW VERSION : END ****************
```

Figure C.7: Debug printings for transport properties calculation of TiNiSn cubic crystalline system (new version).

Appendix D

Computational parameters and input setup for molecular dynamics simulations

The computational parameters adopted in structural optimizations and molecular dynamics (MD) simulations for the two systems (crystalline ice with $Pna2_1$ space group (P-ice) and liquid-like cubic system with 32 water molecules [$(H_2O)_{32}$]) are reported in Table D.1. Details about the cell parameters, the optimized geometry configuration, CRYSTAL Input Block 3 (related to the Hamiltonian and the associated computational parameters) and the MD Section of Input Block 1 as used in the simulations are given in Sections D.1 and D.2, for each atomic system analyzed. The nuclear positions obtained with a full structural optimization, used as starting geometry for the molecular dynamics simulations of the the P-ice crystal, are reported in the Supporting Information of Ref. [92], together with the initial nuclear positions used as starting geometry for the molecular dynamics simulations for the $(H_2O)_{32}$ liquid water periodic system.

System	Basis set	Functional	TOLDEE	TOLINTEG	$n_{\mathbf{k}}$	Grid
P-ice (3D)	Gatti[104]	PBE	8	10 10 20 20 20	112	XXLGRID
	Gatti[104]	PBE-D3	8	10 10 20 20 20	112	XXLGRID
	Gatti[104]	PBE0	8	10 10 20 20 20	112	XXLGRID
	Gatti[104]	B3LYP	8	10 10 20 20 20	112	XXLGRID
	Gatti[104]	B3LYP-D3	8	10 10 20 20 20	112	XXLGRID
$(H_2O)_{32}$ (3D)	Gatti[104]	PBE	8	10 10 20 20 20	36	XXLGRID
	Gatti[104]	PBE-D3	8	10 10 20 20 20	36	XXLGRID
	Gatti[104]	PBE0	8	10 10 20 20 20	36	XXLGRID
	Gatti[104]	B3LYP	8	10 10 20 20 20	36	XXLGRID
	Gatti[104]	B3LYP-D3	8	10 10 20 20 20	36	XXLGRID

Table D.1: Basis set, exchange-correlation functional, threshold on total energy (TOLDEE), thresholds for Coulomb and exchange integrals overlap criteria (TOLINTEG), number of $\boldsymbol{k}$-points in the IBZ ($n_{\boldsymbol{k}}$) and DFT grid used in structural optimizations and molecular dynamics simulations for different systems. The first two values of TOLINTEG parameters are referred to Coulomb integrals, while the last three set the thresholds for Hartree-Fock exchange integrals. The XXLGRID pruned grid for DFT integration have 99 radial points, with a maximum number of 1454 angular points in the region relevant for chemical bonding. For further details see CRYSTAL23 Manual.[114]

D.1 Crystalline ice with P_{na2_1} space group (P-ice) - 3D system

Space group: P_1
Number of atoms in the asymmetric unit cell: 24
Basis set for oxygen and hydrogen: Gatti basis set[104]

CRYSTAL Block 3
DFT
B3LYP
XXLGRID
END
TOLINTEG
10 10 20 20 20
SHRINK
6 6
FMIXING
30
TOLDEE
8
SCFDIR
ENDSCF

CRYSTAL Block 1 MD Section (NVE ensemble)
MOLDYN
TIMESTEP
0.15
NSTEPS
6667
TEMP
300
COMPRT
INVELPRT
TIMEPRT
XYZUNITS
ENDMD

CRYSTAL Block 1 MD Section (NVT ensemble)
MOLDYN
TIMESTEP
0.15
NSTEPS
6667
TEMP
300
NVT
300 20.0
COMPRT
INVELPRT
TIMEPRT
XYZUNITS
ENDMD

Table D.2: CRYSTAL Input Block 3 for the setup of self-consistent calculation used for structural optimization and molecular dynamics simulations (left panel) with B3LYP functional (for the other functionals the same identical input is used except for the name of the functional), MD Section in Input Block 1 used for molecular dynamics calculations with timestep of 0.15 fs, for a total simulation time of 1.0 ps in the NVE (central panel) and NVT (right panel) ensembles, both with an initial temperature of $T = 300$ K.

	initial	PBE	PBE-D3	B3LYP	B3LYP-D3	PBE0	PBE0-D3
ΔE [eV]	0.000	-0.3564	-0.4831	-0.0927	-0.2059	-0.1541	-0.9079
a [Å]	7.362568	7.027025	6.947098	7.168653	7.048932	7.081031	7.002097
b [Å]	7.744386	7.468170	7.387431	7.592544	7.474845	7.501811	7.421250
c [Å]	4.533529	4.303437	4.258816	4.396057	4.335274	4.349973	4.307872
v [Å^3]	258.495338	225.840116	218.567598	239.269961	228.424197	231.072971	223.855616
α [°]	90.000000	89.996140	90.000096	89.994506	89.994755	90.001785	89.999219
β [°]	90.000000	89.998355	90.003458	90.001829	90.005460	90.006733	90.007456
γ [°]	90.000000	89.996947	89.994200	90.000491	89.998431	89.998143	89.994863

Functional	E_{in} [Ha]	E_{fin} [Ha]	E_g^{in} [eV]	E_g^{fin} [eV]
PBE	-610.9384020285	-610.9515008155	6.0364	5.9117
PBE-D3	-610.9646513252	-610.9824040682	6.0364	5.8996
B3LYP	-611.3073988797	-611.3108045671	8.1569	8.0853
B3LYP-D3	-611.3445446430	-611.3521108944	8.1569	8.0469
PBE0	-610.9515213552	-610.9571845752	8.8405	8.7541
PBE0-D3	-610.9515213550	-610.9848880396	8.8405	8.7268

Table D.3: Cell parameters (upper table), total energy per unit cell and direct energy band gap (lower table) of the P-ice crystal, before and after the geometry optimization with different functionals. Precision on the absolute energy values is fixed by the self-consistent threshold value of 10^{-8} Ha (i.e. TOLDEE equal to 8). In the upper table, the energy gained by means of structural optimization with respect to the initial nuclear configuration (computed as $\Delta E = E_{\text{fin}} - E_{\text{in}}$) is also reported for each functional considered.

D.2 Liquid-like water $(H_2O)_{32}$ - 3D system

Space group: P_1
Number of atoms in the asymmetric unit cell: 96
Basis set for oxygen and hydrogen: Gatti basis set[104]

CRYSTAL Block 3
DFT
B3LYP
XXLGRID
END
TOLINTEG
10 10 20 20 20
SHRINK
4 4
FMIXING
30
TOLDEE
8
SCFDIR
ENDSCF

CRYSTAL Block 1 MD Section (NVE ensemble)
MOLDYN
TIMESTEP
0.25
NSTEPS
4000
TEMP
300
COMPRT
INVELPRT
TIMEPRT
XYZUNITS
ENDMD

CRYSTAL Block 1 MD Section (NVT ensemble)
MOLDYN
TIMESTEP
0.25
NSTEPS
4000
TEMP
300
NVT
300 38.0
COMPRT
INVELPRT
TIMEPRT
XYZUNITS
ENDMD

Table D.4: CRYSTAL Input Block 3 for the setup of self-consistent calculation used for molecular dynamics simulations (left panel) with B3LYP functional (for the other functionals the same identical input is used except for the name of the functional), MD Section in Input Block 1 used for molecular dynamics calculations with timestep of 0.25 fs, for a total simulation time of 1.0 ps in the NVE (central panel) and NVT (right panel) ensembles, both with an initial temperature of $T = 300$ K. The input file for the simulation in NVT ensemble with $T = 600$ K is exactly equal to that reported in the right panel, except for the initial temperature value, modified using the keyword TEMP.

The lattice parameter for the cubic cell in which the 32 water molecules have been modeled is computed on the basis of the water density ρ under ambient conditions, which is equal to 1 g/cm^3. The value of the lattice parameter a is then computed with the formula

$$\rho = \frac{m}{a^3} \quad \rightarrow \quad a = \sqrt[3]{\frac{m}{\rho}} = \sqrt[3]{\frac{(m_h + m_o)\, n_w}{\rho\, N_A}} \tag{D.1}$$

where m is the mass of n_w water molecules, m_h is the hydrogen mass, m_o is the oxygen mass and N_A is the Avogadro constant, namely,

$$\begin{aligned} m_h &= 1.0080 \text{ g/mol} \\ m_o &= 15.999 \text{ g/mol} \\ N_A &= 0.602214076 \cdot 10^{24} \\ \rho &= 1.0 \text{ g/cm}^3 = 10^{-24} \text{ g/Å}^3 \end{aligned}$$

Inserting these data in the formula (D.1), the lattice parameter $a = b = c$ of the cubic system can be computed as

$$ab = c = \sqrt[3]{\frac{(2 \cdot 1.0080 + 15.999) \cdot 32 \text{ g}}{0.602214076 \text{ g/Å}^3}} = 9.855 \text{ Å} \tag{D.2}$$

Appendix E

Manuals

E.1 Molecular dynamics : manual and keywords

rec		variable	meaning
•	*A*	**NSTEPS**	number of molecular dynamics steps [default: 1000]
	∗	NSTEPS	
•	*A*	**TIMESTEP**	time step for integration of nuclear equations of motion [default: 0.1 fs]
	∗	TIMESTEP	
•	*A*	**TEMP**	initial temperature [default: 300 K]
	∗	TEMPERATURE	
•	*A*	**INVELKT**	logical keyword [default: false] using this keyword, the initial nuclear velocities are multiplied by $\sqrt{k_b T}$ just after being initialized through the Box-Muller algorithm
•	*A*	**NEWSEEDVEL**	logical keyword [default: false] using this keyword, a new seed is generated for the initialization of the initial nuclear velocities (initialized through the Box-Muller algorithm). This keyword is useful to generate different sets of initial nuclear velocities if the compiler used to compile the code generates an executable that returns always the same set of random numbers
•	*A*	**VELOCITIES**	initial nuclear velocities given from input file
		insert a number of input records equal to the number of atoms n	
	∗	LB(i), v_x(i), v_y(i), v_z(i), i=1,n	label and initial velocity components (along each direction) of the each nucleus
•	*A*	**ZEROINITF**	logical keyword [default: false] using this keyword, the initial nuclear forces are set all equal to zero
•	*A*	**DEGFREE**	set the number of degrees of freedom g to be considered in the calculation of the temperature and of all the other molecular dynamics quantities [default: see Table 5.12]
	∗	INPUT_NFREE	number of degrees of freedom
•	*A*	**THERMOST**	type of thermostat that can be set
	∗	TEMPCONST	constant temperature to be maintained by the thermostat [K]
		THSTAT	type of thermostat : 1 = simple scaling 2 = Berendsen thermostat
		if thstat = 2	
	∗	TS	rise time of Berendsen thermostat (it must be larger than the time step) [fs]
•	*A*	**YESFLY**	if true, at each step of the simple scaling and the Berendsen thermostat methods, the center of mass rotational and translational velocities components are not subtracted from the nuclear velocities (flying ice cube effect) [default: false]
•	*A*	**NVT**	Nosé-Hoover thermostat (NVT ensemble)
	∗	TEMPCONST	constant temperature to be maintained by the thermostat [K]
		TS	time associated to the thermostat mass [fs]
•	*A*	**NVTINT**	keyword allowed only after the keyword NOSE [default: 1]
	∗	NOSE_INTEG	procedure for integration of nuclear equations of motion in NVT ensemble : 0 = integrator I0_NVT (see Table A.1) 1 = integrator I1_NVT (see Table 5.5)

•	*A*	**NPT**	thermostat and barostat (NPT ensemble)
	∗	TEMPCONST	constant temperature to be maintained by the thermostat [K]
		TS	time associated to the thermostat mass [fs]
		PRESCONST	constant pressure to be maintained by the barostat [GPa]
		TV	time associated to the barostat mass [fs]
•	*A*	**NPTINT**	keyword allowed only after the keyword NPT [default: 0]
	∗	NPT_INTEG	procedure for integration of nuclear equations of motion in NPT ensemble : 0 = integrator I0_NPT (see Table 5.8)
•	*A*	**NOROT**	logical keyword allowed only for 0D and 1D systems [default: false] using this keyword, the 0D and 1D systems are not allowed to rotate during the molecular dynamics simulation (center of mass angular velocity subtracted from nuclear velocities)
•	*A*	**GUESSP**	logical keyword [default: false] by default, the density matrix guess at each molecular dynamics step is taken to be the atomic one. If this keyword is used, the density matrix at each step is taken to be the final density matrix of the previous step (for the first molecular dynamics step, the atomic guess is used)
•	*A*	**INITGUESSP**	logical keyword [default: false] by default, the density matrix guess at the first molecular dynamics step is taken to be the atomic density matrix guess. If this keyword is used, the density matrix used at the first molecular dynamics step is read from file fort.20 (calculation restarted with external density matrix from file fort.20)
•	*A*	**RESTART**	logical keyword [default: false] using this keyword, the molecular dynamics calculation is restarted from a previous run (some restarting units are needed, depending on the calculation)
•	*A*	**FIXIND**	logical keyword [default: false] by default, the Coulomb and exchange integrals involved in the calculation (based on the system geometry and on input thresholds), are computed at each step of a molecular dynamics simulation, since the geometry changes at each step. Using this keyword, the integrals are computed only once, at the beginning of the calculation, and they are not updated during the molecular dynamics simulation
•	*A*	**COMPRT**	set to true the logical variable for the printing of file fort.192 with center of mass position, velocity and kinetic energy information at each step [default: false]
•	*A*	**MASSPRT**	set to true the logical variable for the printing of the atomic masses in the output file [default: false]
•	*A*	**CHARGESPRT**	set to true the logical variable for the printing of the atomic charges in file ATCHARGES.DAT [default: false]
•	*A*	**TIMEPRT**	set to true the logical variable for the printing of file fort.222 with the number of SCF cycles and the time for a complete SCF cycle at each step [default: false]
•	*A*	**INVELPRT**	set to true the logical variable for the printing of file fort.193 with information on values of initial nuclear velocities (before and after the shifting with respect to center of mass velocity, before and after the rescaling with respect to initial temperature) [default: false]
•	*A*	**TESTVEL**	test of initial nuclear velocities distribution: the code computes the initial nuclear velocities distribution, print information about this calculation in the output file fort.193 (as for the keyword INITVELPRT) and then it stops. This keyword can be used to test and study the initial nuclear velocities distribution of the Molecular Dynamics module, which represents an important part of the nuclear equations of motion initial conditions [default: false]
•	*A*	**XYZUNITS**	activate the creation of files fort.186 e fort.187, containing respectively the nuclear positions and velocities at each step (useful to visualize the trajectories during optimization) [default: false]
•	*A*	**DEBUG**	activate debug modality : more info in output printings [default: false]
•	*A*	**FRAGMENT**	partial molecular dynamics simulation [default: global molecular dynamics]
	∗	NF	number of atoms free to move
			if NF > 0 insert NF input records
	∗	LB(i), i=1,NF	labels of the atoms to move
			if NF < 0 insert two input indexes
	∗	IFRAGSTART, IFRAGSTOP	atoms starting from label ifragstart to label ifragstop are free to move
•	*A*	**ISOTOPES**	atomic masses modified

	∗	NL	number of atoms whose atomic mass must be modified
			insert NL input records
	∗	LB(i), AMASS(i), i=1,NL	label and new atomic mass (amu) of the atom
•	*A*	**PCFMD**	activate the post processing analysis of molecular dynamics trajectory in order to compute the Pair Correlation Function [default: false]
•	*A*	**ANALYSIS**	activate the post processing analysis of molecular dynamics trajectory in order to compute mean values and standard deviations of energies and temperature [default: false]
•	*A*	**AVCORR**	activate the post processing analysis of molecular dynamics trajectory in order to compute the Velocity Autocorrelation Function [default: false]
•	*A*	**FREQCALC**	activate the post processing analysis of molecular dynamics trajectory in order to compute the Vibrational Density of States (Power Spectrum) [default: false]

E.2 Fast Inertial Relaxation Engine : manual and keywords

	rec	variable	meaning
•	*A*	**BASIC**	activate the basic FIRE algorithm. If this keyword is not inserted, the improved FIRE2.0 algorithm is used by default.
•	*A*	**ONLYFRTH**	activate the stopping criteria only on the basis of the convergence with respect to the euclidean norm of $3N$ forces vector of the actual FIRE step [default: false]
•	*A*	**TOLFORCE**	modify the threshold on the euclidean norm of $3N$ forces vector computed in cartesian coordinates
	∗	EXPFTHR	$\|\mathbf{F}\| < 10^{-EXPFTHR}$ Ha/Bohr [default: 10]
•	*A*	**ONLYENTH**	activate the stopping criteria only on the basis of the convergence with respect to the energy difference between two FIRE steps [default: false]
•	*A*	**TOLDEE**	threshold on the energy change between two optimization FIRE steps
	∗	EXPETHR	$\|\Delta E\| < 10^{-EXPETHR}$ Ha [default: 8]
•	*A*	**ONLYDSTH**	activate the stopping criteria only on the basis of the convergence with respect to the maximum and the root-mean-square of the nuclei displacements [default: false]
•	*A*	**TOLDEX**	convergence criterion on the RMS of the nuclei displacements computed in cartesian coordinates (units: Bohr)
	∗	THDRMS	maximum RMS of the nuclei displacements computed in cartesian coordinates [default: 0.0012 Bohr]
•	*A*	**NCSTPCON**	modify the number of consecutive steps the convergence criteria have to be satisfied before stopping and considering the minimization as converged
	∗	NCSTPCON	number of consecutive steps the convergence criteria have to be satisfied before stopping the minimization [default: 2]
•	*A*	**MAXCYCLE**	modify the maximum number of FIRE iterations
	∗	MAXCYCLE	maximum number of iterations [default: 500]
•	*A*	**NPLEZMAX**	only for FIRE 2.0 method: modify the number of consecutive steps the power factor is negative or equal to zero ($P(t) \leq 0$) before stopping the FIRE research of the PES minimum (if the power factor continue to remain negative or equal to zero, probably the system is stuck in a narrow valley in the PES)
	∗	NPLEZEROMAX	number of consecutive steps the power factor is negative or equal to zero ($P(t) \leq 0$) before stopping the minimization [default: 2000]
•	*A*	**TMAX**	modify the maximum value of the timestep so that $dt_{max} = \text{TMAX} \cdot dt_{in}$
	∗	TMAX	factor used to compute the maximum value of the timestep [default: 10.0]
•	*A*	**TMIN**	only for FIRE 2.0 method: modify the minimum value of the timestep so that $dt_{min} = \text{TMIN} \cdot dt_{in}$
	∗	TMIN	factor used to compute the minimum value of the timestep [default: 0.02]
•	*A*	**NDELAY**	modify the number of steps to wait after the power factor became negative ($P(t) < 0$) before increasing the timestep
	∗	NDELAY	number of steps to wait after the power factor became negative ($P(t) < 0$) before increasing the timestep [default: 5]
•	*A*	**DTSTART**	modify the initial value of the timestep dt_{in} (units: fs)
	∗	DTSTART	initial value of the timestep [default: ...]
•	*A*	**DTGROW**	modify the factor by which the timestep is eventually increased
	∗	DTGROW	factor by which the timestep is eventually increased [default: 1.1]
•	*A*	**DTSHRINK**	modify the factor by which the timestep is eventually decreased
	∗	DTSHRINK	factor by which the timestep is eventually decreased [default: 0.5]
•	*A*	**ALPHASTART**	modify the initial coefficient α_{in} for velocities and forces vectors mixing
	∗	ALPHASTART	initial coefficient α_{in} for velocities and forces vectors mixing [default: 0.25]
•	*A*	**ALPHASHRINK**	modify the factor by which the coefficient α is eventually decreased
	∗	ALPHASHRINK	factor by which the coefficient α is eventually decreased [default: 0.99]
•	*A*	**UNUCMASSES**	logical keyword [default = false] using this keyword, all the atomic masses are set equal to 1 amu
•	*A*	**NUCMASSES**	set all the atomic masses equal to a unique value specified below
	∗	SAME_MASS	value of the atomic mass (amu)

- *A* **ATOMICP** at each FIRE step, the density matrix $\mathbf{P}(\mathbf{G})$ is restarted by default from the density matrix obtained at the end of the self-consistent (SCF) calculation of the previous step. This keyword deactivates the default restarting option: if used, at each step the density matrix taken as input for SCF calculation is constructed from an atomic guess
- *A* **RESTART** logical keyword [default: false]
  using this keyword, the structural optimization calculation is restarted from a previous run (some restarting units are needed)
- *A* **INITGUESSP** logical keyword [default: false]
  by default, the density matrix guess at the first FIRE step is taken to be the atomic density matrix guess. If this keyword is used, the density matrix used at the first FIRE step is read from file fort.20 (calculation restarted with external density matrix from file fort.20)
- *A* **FIXIND** logical keyword [default: false]
  by default, the Coulomb and exchange integrals involved in the calculation (based on the system geometry and on input thresholds), are computed at each step of a FIRE structural optimization, since the geometry changes at each step. Using this keyword, the integrals are computed only once, at the beginning of the calculation, and then they are not updated during the FIRE structural optimization

---

- *A* **FRAGMENT** Partial geometry optimization [default: global optimization]
  - ∗ NL number of atoms free to move (to be optimized)

  if NL > 0 insert NL input records
  - ∗ LB(L), L=1,NL labels of the atoms to move

  if NL < 0 insert two input indexes
  - ∗ IFRAGSTART IFRAGSTOP atoms starting from label IFRAGSTART to label IFRAGSTOP are free to move

---

- *A* **ISOTOPES** atomic masses modified
  - ∗ NL number of atoms whose atomic mass must be modified

  insert NL input records
  - ∗ LB(i), AMASS(i), i=1,NL label and new atomic mass (amu) of the atom

---

- *A* **XYZUNITS** activate the creation of two .XYZ files: one with the nuclei positions and one with the velocities of the nuclei at each FIRE optimization step (useful to visualize the trajectories during optimization)
  [default: false]
- *A* **FORCEPRT** activate the printing in the standard output file of the cartesian forces on the nuclei at each FIRE optimization step [default: false]
- *A* **VELPRT** activate the printing in the standard output file of the cartesian velocities of the nuclei at each FIRE optimization step [default: false]
- *A* **DISPLPRT** activate the printing in the standard output file of the cartesian displacements of the nuclei at each FIRE optimization step
  [default: false]

Appendix F

Documentation of some CRYSTAL core subroutines

Contents

F.1 Subroutine PRIMST (libx7_scf.f)

The subroutine PRIMST is called, for example, in input.f90 library, when keyword SYMMREMO is used, with the aim to delete symmetry in solids.

line	**case SYMMREMO in input.f90**	CRYSTAL version: SVN_DEV1218
1652	case ('SYMMREMO')	
	!****	
	!**** DELETE THE USE OF SYMMETRY	
	!****	
	!CIPRM STORE THE NUMBER OF SYMMETRY OPERATORS IN INF(157)	
	INF(157)=INF(2)	Save the previous number of symmetry operators in INF(157)
	INF(2) = 1	Reset the number of symmetry operators = INF(2) to 1
	CALL PRIMST	See Table F.3
	WRITE (IOUT,707)	Print: USE OF SYMMETRY WILL BE SUBSEQUENTLY SUSPENDED
1661	LB5 = .TRUE.	Logical variable LB5: if LB5.EQ.TRUE, CALL XYVINV
		if LB5.EQ.TRUE and (LPRINT(4).NE.0), CALL XYVPRT
1799	707 FORMAT(/' USE OF SYMMETRY WILL BE SUBSEQUENTLY SUSPENDED')	

Matrix $\mathbf{W} \equiv$ W1R is the conversion matrix from primitive to conventional (crystallographic) cell. It is printed in the output file (in row order) through the subroutine COOPRT, as reported in the following

line	**SUBROUTINE PRIMST: defined in libx7_scf.f**	CRYSTAL version: SVN_DEV1218
306	SUBROUTINE PRIMST	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE GVECT_MODULE	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	C...	
	C... TO RECLASSIFY CELL AS PRIMITIVE	
	C...	
	INF(62)=1	INF(62) = numeric code for centering type : 1(P) 2(A) 3(B) 4(C) 5(F) 6(I) 7(R)
	INF(63)=1	INF(63) = numeric code for crystal system
317	DO I=1,3	
	W1R(1,I)=0._FLOAT	$\mathbf{W} \equiv$ W1R(3,3) conversion matrix from primitive to conventional cell
	W1R(2,I)=0._FLOAT	
	W1R(3,I)=0._FLOAT	Lines (317:322) : W1R is set equal to the unit matrix $\mathbb{1}$, i.e. the conventional
	W1R(I,I)=1._FLOAT	cell is forced to be the primitive cell
322	ENDDO	
	RETURN	
324	END	

Table F.3: Code and description of subroutine PRIMST (libx7_scf.f)

table.

line	**Lines 1184 and 1287-1288 of SUBROUTINE COOPRT: defined in both4.f**	CRYSTAL version: SVN_DEV1218
1184	WRITE(IOUT,204)((W1R(I,J),J=1,3),I=1,3)	Writing matrix W1R in the output file
1287	204 FORMAT(/' TRANSFORMATION MATRIX PRIMITIVE-CRYSTALLOGRAPHIC CELL'/	
1288	*9F8.4/)	

If $\mathbf{a}_1^c$, $\mathbf{a}_2^c$, $\mathbf{a}_3^c$ are the direct lattice vectors of the conventional cell, and $\mathbf{a}_1^p$, $\mathbf{a}_2^p$, $\mathbf{a}_3^p$ are the direct lattice vectors of the primitive cell, then the conversion matrix W1R = $\mathbf{W}$ has to be applied on the primitive direct lattice vectors to obtain the conventional direct lattice vectors, as follows

$$\begin{pmatrix} \mathbf{a}_1^c & \mathbf{a}_2^c & \mathbf{a}_3^c \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^p & \mathbf{a}_2^p & \mathbf{a}_3^p \end{pmatrix} \mathbf{W} \tag{F.1}$$

where $\mathbf{a}_i^c = (a_{ix}^c, a_{iy}^c, a_{iz}^c)$ with $i = 1, 2, 3$ and the same for $\mathbf{a}_i^p$ vectors. The matrix $\mathbf{W}$ defined as

$$\mathbf{W} = \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{pmatrix} \qquad \text{(primitive} \rightarrow \text{crystallographic cell) conversion matrix} \tag{F.2}$$

Analogously, applying the transpose operation matrix on both members of equation (F.1)

$$\begin{pmatrix} \mathbf{a}_1^c \\ \mathbf{a}_2^c \\ \mathbf{a}_3^c \end{pmatrix} = {}^{\mathbf{t}}\mathbf{W} \begin{pmatrix} \mathbf{a}_1^p \\ \mathbf{a}_2^p \\ \mathbf{a}_3^p \end{pmatrix} \qquad \rightarrow \qquad \mathbf{a}_i^c = \sum_{j=1}^{3} W_{ij}\, \mathbf{a}_j^p \quad \text{with } i = 1, 2, 3 \tag{F.3}$$

which in components is read as

$$\begin{pmatrix} a_{1x}^c & a_{1y}^c & a_{1z}^c \\ a_{2x}^c & a_{2y}^c & a_{2z}^c \\ a_{3x}^c & a_{3y}^c & a_{3z}^c \end{pmatrix} = \begin{pmatrix} W_{11} & W_{21} & W_{31} \\ W_{12} & W_{22} & W_{32} \\ W_{13} & W_{23} & W_{33} \end{pmatrix} \begin{pmatrix} a_{1x}^p & a_{1y}^p & a_{1z}^p \\ a_{2x}^p & a_{2y}^p & a_{2z}^p \\ a_{3x}^p & a_{3y}^p & a_{3z}^p \end{pmatrix} \tag{F.4}$$

In the same way as in (G.3), the fractional coordinates $\mathbf{x}^c = (x_1^c, x_2^c, x_3^c)$ of a nucleus in the conventional cell can be obtained from the fractional coordinates $\mathbf{x}^p = (x_1^p, x_2^p, x_3^p)$ of the same nucleus in the primitive cell through the equation

$$\begin{pmatrix} x_1^c \mathbf{a}_1^c \\ x_2^c \mathbf{a}_2^c \\ x_3^c \mathbf{a}_3^c \end{pmatrix} = {}^{\mathbf{t}}\mathbf{W} \begin{pmatrix} x_1^p \mathbf{a}_1^p \\ x_2^p \mathbf{a}_2^p \\ x_3^p \mathbf{a}_3^p \end{pmatrix} \qquad \rightarrow \qquad x_i^c \mathbf{a}_i^c = \sum_{j=1}^{3} W_{ij}\, x_j^p\, \mathbf{a}_j^p \quad \text{with } i = 1, 2, 3 \tag{F.5}$$

F.2 Subroutine MINV3 (libx5_com.f)

Input : (3,3) matrix **P** ; (3,3) matrix PINV ; real number DET.
The subroutine modifies the input values of PINV and DET, so that PINV matrix returns the inverse of the input matrix **P**, and DET returns the determinant of the input matrix **P**.

line	**SUBROUTINE MINV3: defined in libx5_com.f**	CRYSTAL version: SVN_DEV1218
256	SUBROUTINE MINV3(P,PINV,DET)	
	C... CALCULATES DETERMINANT AND INVERSE OF 3 * 3 MATRIX	
	USE NUMBERS	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	PARAMETER (TOLL=1E-16_FLOAT)	
	DIMENSION P(3,3),PINV(3,3)	P ≡ **P** : (3,3) matrix
262	F1=P(2,2)*P(3,3)-P(2,3)*P(3,2)	
	F2=P(2,3)*P(3,1)-P(2,1)*P(3,3)	Lines 262-265: calculation of the determinant of a
	F3=P(2,1)*P(3,2)-P(2,2)*P(3,1)	(3,3) matrix, see eq. (F.7)
265	DET=P(1,1)*F1+P(1,2)*F2+P(1,3)*F3	DET ≡ det(**P**) computed as in eq. (F.7)
	IF(ABS(DET).LT.TOLL)THEN	If $\|\det(\mathbf{P})\| < 10^{-15}$, print an error in the output file
	CALL ERRVRS(1,'MINV3 ',	and stop the program (because the inverse matrix
	*'THE DETERMINANT OF THE DIRECT LATTICE VECTORS IS VERY SMALL')	cannot be safely calculated)
	ELSE	Else, compute the inverse of the (3,3) input matrix **P**
270	DETI=1._FLOAT/DET	
	PINV(1,1)=F1*DETI	
	PINV(2,1)=F2*DETI	
	PINV(3,1)=F3*DETI	
	F1=P(1,1)*DETI	
	F2=P(1,2)*DETI	Lines 270-282: calculation of the inverse of a (3,3)
	F3=P(1,3)*DETI	matrix, see eqs. (F.8) and (F.9)
	PINV(1,2)=F3*P(3,2)-F2*P(3,3)	
	PINV(2,2)=F1*P(3,3)-F3*P(3,1)	
	PINV(3,2)=F2*P(3,1)-F1*P(3,2)	
	PINV(1,3)=F2*P(2,3)-F3*P(2,2)	
	PINV(2,3)=F3*P(2,1)-F1*P(2,3)	
282	PINV(3,3)=F1*P(2,2)-F2*P(2,1)	PINV ≡ $\mathbf{P}^{-1}$: (3,3) matrix
	ENDIF	
	RETURN	
285	END	

Table F.6: Code and description of subroutine MINV3 (libx5_com.f)

The determinant of a matrix

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix} \tag{F.6}$$

is computed as follows:

$$\begin{aligned} \det(\mathbf{P}) &= p_{11}\underbrace{[p_{22}p_{33} - p_{23}p_{32}]}_{=f_1} - p_{12}\underbrace{[p_{21}p_{33} - p_{23}p_{31}]}_{=-f_2} + p_{13}\underbrace{[p_{21}p_{32} - p_{22}p_{31}]}_{=f_3} \\ &= p_{11}f_1 + p_{12}f_2 + p_{13}f_3 \end{aligned} \tag{F.7}$$

To compute the inverse of a (3,3) matrix the following procedure has to be followed:

1. Compute the transpose of the matrix $\mathbf{P} \rightarrow {}^t\mathbf{P} = \begin{pmatrix} p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{32} \\ p_{13} & p_{23} & p_{33} \end{pmatrix}$

2. Compute the matrix of cofactors of $\mathbf{P} \rightarrow \mathbf{A} = \begin{pmatrix} p_{22}p_{33} - p_{32}p_{23} & p_{32}p_{13} - p_{12}p_{33} & p_{12}p_{23} - p_{22}p_{13} \\ p_{31}p_{23} - p_{21}p_{33} & p_{11}p_{33} - p_{31}p_{13} & p_{21}p_{13} - p_{11}p_{23} \\ p_{21}p_{32} - p_{31}p_{22} & p_{31}p_{12} - p_{11}p_{32} & p_{11}p_{22} - p_{21}p_{12} \end{pmatrix}$

   which can be rewritten, using (F.7): $\mathbf{A} = \begin{pmatrix} f_1 & p_{32}p_{13} - p_{12}p_{33} & p_{12}p_{23} - p_{22}p_{13} \\ f_2 & p_{11}p_{33} - p_{31}p_{13} & p_{21}p_{13} - p_{11}p_{23} \\ f_3 & p_{31}p_{12} - p_{11}p_{32} & p_{11}p_{22} - p_{21}p_{12} \end{pmatrix}$

3. Divide $\mathbf{A}$ by $\det(\mathbf{P}) \rightarrow \mathbf{P}^{-1} = \dfrac{1}{\det(\mathbf{P})} \begin{pmatrix} f_1 & p_{32}p_{13} - p_{12}p_{33} & p_{12}p_{23} - p_{22}p_{13} \\ f_2 & p_{11}p_{33} - p_{31}p_{13} & p_{21}p_{13} - p_{11}p_{23} \\ f_3 & p_{31}p_{12} - p_{11}p_{32} & p_{11}p_{22} - p_{21}p_{12} \end{pmatrix}$ (F.8)

So that the inverse matrix $\mathbf{P}^{-1}$ has the following components:

$$\begin{aligned} P_{11}^{-1} &= f_1/\det(\mathbf{P}) \\ P_{21}^{-1} &= f_2/\det(\mathbf{P}) \\ P_{31}^{-1} &= f_3/\det(\mathbf{P}) \\ P_{12}^{-1} &= [p_{32}p_{13} - p_{12}p_{33}]/\det(\mathbf{P}) \\ P_{22}^{-1} &= [p_{11}p_{33} - p_{31}p_{13}]/\det(\mathbf{P}) \\ P_{32}^{-1} &= [p_{31}p_{12} - p_{11}p_{32}]/\det(\mathbf{P}) \\ P_{13}^{-1} &= [p_{12}p_{23} - p_{22}p_{13}]/\det(\mathbf{P}) \\ P_{23}^{-1} &= [p_{21}p_{13} - p_{11}p_{23}]/\det(\mathbf{P}) \\ P_{33}^{-1} &= [p_{11}p_{22} - p_{21}p_{12}]/\det(\mathbf{P}) \end{aligned} \quad \text{(F.9)}$$

which corresponds to the calculations in lines 270-282 of the subroutine MINV3 previously reported (see Table F.6).

NOTE: the **determinant** of a (3,3) matrix is equal to the 3-dimensional **volume** defined by the three vectors whose components are the row-by-row components of the matrix itself

F.3 Subroutine INV123 (libx4_com.f)

Input : (3,3) matrix PARET ; (3,3) matrix P2INV ; real number VOL1 ; LDIM = INF(10) = system dimensionality.
The subroutine modifies the input values of P2INV and VOL1, depending on the dimension LDIM of the system, so that P2INV matrix returns the inverse of the input matrix PARET, and VOL1 the volume of the system.

Note: PARET is a (3,3) matrix (in which only (LDIM,LDIM) elements are initialized) containing row-by-row the direct lattice vectors of the primitive cell (in cartesian components) in Bohr units.

PARET MATRIX:
(3D CASE) PARET(1X) PARET(1Y) PARET(1Z)
PARET(2X) PARET(2Y) PARET(2Z)
PARET(3X) PARET(3Y) PARET(3Z)
(2D CASE) PARET(1X) PARET(1Y)
PARET(2X) PARET(2Y)
(1D CASE) PARET(1X)

line	**SUBROUTINE INV123: defined in libx4_com.f**	CRYSTAL version: SVN_DEV1218
208	SUBROUTINE INV123(PARET,P2INV,VOL1,LDIM)	VOL1 $\equiv v$: volume, LDIM $\equiv D$ dimension of the system
	USE NUMBERS	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	PARAMETER (TOLL=1E-16_FLOAT)	
	DIMENSION PARET(3,3),P2INV(3,3)	PARET $\equiv \mathbf{a}$: (3,3) matrix, P2INV $\equiv \mathbf{a}^{-1}$: (3,3) matrix
213	IF(LDIM.EQ.0)THEN	If $D = 0$, $v = 1$
	VOL1=1._FLOAT	
	ELSE	
216	GOTO (3,4,5),LDIM	
217	3 VOL1=PARET(1,1)	Else if $D = 1$, $v = a_{11}$
	IF(ABS(VOL1).LT.TOLL)GOTO 999	if $\|v\| < 10^{-15}$ print an error in the output file and exit
	P2INV(1,1)=1._FLOAT/VOL1	else, P2INV(1,1) $\equiv a_{11}^{-1} = 1/v = 1/a_{11}$
	RETURN	and return
221	4 VOL1=PARET(1,1)*PARET(2,2)-PARET(1,2)*PARET(2,1)	Else if $D = 2$, $v = a_{11}a_{22} - a_{12}a_{21}$
	IF(ABS(VOL1).LT.TOLL)GOTO 999	if $\|v\| < 10^{-15}$ print an error in the output file and exit
223	DETI=1._FLOAT/VOL1	else, compute the inverse of the (3,3) input matrix **a** as
	P2INV(1,1)=PARET(2,2)*DETI	it was a (2,2) matrix, see eq. (F.12)
	P2INV(2,1)=-PARET(2,1)*DETI	
	P2INV(1,2)=-PARET(1,2)*DETI	lines 223-227: calculation of the inverse of a (2,2)
227	P2INV(2,2)=PARET(1,1)*DETI	matrix, see eq. (F.12)
	RETURN	and return
229	5 CALL MINV3(PARET,P2INV,VOL1)	Else if $D = 3$, CALL MINV3 (see section F.2)
	RETURN	and return
	999 CALL ERRVRS(1,'INV123',	
	*'THE DETERMINANT OF THE DIRECT LATTICE VECTORS IS VERY SMALL')	
	ENDIF	Endif
	RETURN	
235	END	

Table F.8: Code and description of subroutine INV123 (libx4_com.f)

If LDIM = 1, PARET $\equiv \mathbf{a}$ is a (3,3) matrix with the last two rows and the last two columns of zeros, so that it can be considered a number PARET(1,1) $= a_{11}$. In this case the volume is $v = a_{11}$ (stored in VOL1), the inverse is $a_{11}^{-1} = 1/a_{11}$ (stored in P2INV(1,1) element).

If LDIM = 2, PARET $\equiv \mathbf{a}$ is a (3,3) matrix with the last row and the last column of zero, i.e. it can be considered as a (2,2) matrix:

$$\mathbf{a} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{F.10}$$

so that the volume (stored in VOL1) is equal to the determinant of matrix **a**:

$$v = \det(\mathbf{a}) = a_{11}a_{22} - a_{12}a_{21} \tag{F.11}$$

and the inverse $\mathbf{a}^{-1}$ is computed as

$$\mathbf{a}^{-1} = \frac{1}{v}\begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \tag{F.12}$$

and stored in P2INV matrix.

If LDIM = 3, the subroutine MINV3 is called (see Section F.2), so that the determinant and the inverse of the (3,3) input matrix is calculated, i.e. the inverse of the (3,3) matrix PARET $\equiv \mathbf{a}$ (stored in P2INV) and its determinant (stored in VOL1) are computed.

F.4 Subroutine PARLAT(C,D,A) (libx6_com.f)

Input : (3,3) matrix **C** ; (3,3) matrix **D** ; 7-elements vector **A**.
The subroutine modifies the value of the input matrix **C** and of the input vector **A**, in particular (i) it computes the lengths of the 3 vectors whose components are stored in each row of the matrix **C**, and it saves the lengths of these vectors in the three elements $\mathbf{A}(k)$ with $k = 1, 2, 3$, (ii) it computes the three fundamental angles between the three vectors and saves them in the three elements $\mathbf{A}(k)$ with $k = 4, 5, 6$, and finally (iii) it computes the 3-dimensional volume defined by the three row-by-row vectors in matrix **C**, saving the volume value in the element $\mathbf{A}(7)$, and it computes the inverse of the (3,3) matrix **C**, storing this inverse in the (3,3) matrix **D**.

line	**SUBROUTINE PARLAT: defined in libx6_com.f**	CRYSTAL version: SVN_DEV1218
240	SUBROUTINE PARLAT(C,D,A)	
	USE NUMBERS	
	USE PARINF_MODULE	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	DIMENSION C(3,3),D(3,3),A(7)	Declaration of: (3,3) matrix **C** ; (3,3) matrix **D** ; 7-elements vector **A**
	DO 2 I=1,3	
246	2 A(I)=SQRT(C(I,1)*C(I,1)+C(I,2)*C(I,2)+C(I,3)*C(I,3))	Computation of the lattice parameters $\mathbf{A}(1)$, $\mathbf{A}(2)$, $\mathbf{A}(3)$, see eq. (F.15)
247	A(4)=ACOS((C(2,1)*C(3,1)+C(2,2)*C(3,2)+C(2,3)*C(3,3))	Computation of angle $\alpha = \mathbf{A}(4)$ between 2-th and 3-th vectors of **C** matrix, see eq. (F.20)
	*/(A(2)*A(3)))*PAR(33)	PAR(33) = 1./PAR(3) ; PAR(3) = conversion factor degrees-radians = $\pi/180$.
249	A(5)=ACOS((C(1,1)*C(3,1)+C(1,2)*C(3,2)+C(1,3)*C(3,3))	Computation of angle $\beta = \mathbf{A}(5)$ between 1-th and 3-th vectors of **C** matrix, see eq. (F.21)
	*/(A(1)*A(3)))*PAR(33)	
251	A(6)=ACOS((C(1,1)*C(2,1)+C(1,2)*C(2,2)+C(1,3)*C(2,3))	Computation of angle $\gamma = \mathbf{A}(6)$ between 1-th and 2-th vectors of **C** matrix, see eq. (F.22)
	*/(A(1)*A(2)))*PAR(33)	
253	CALL INV123(C,D,A(7),INF(10))	Calculation of volume = $\mathbf{A}(7)$ defined by the row-by-row vectors in matrix **C**
	RETURN	(with subroutine INV123, see section F.3)
255	END	

Table F.10: Code and description of subroutine PARLAT (libx6_com.f)

Given the (3,3) input matrix **C** defined as

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{pmatrix} \tag{F.13}$$

which, from a physical point of view, represents the direct lattice vectors of the primitive cell in cartesian components (PARLAT matrix), the following quantities are computed and saved in the input **A** vector:

$$\mathbf{A}(1) = \sqrt{c_{11}^2 + c_{12}^2 + c_{13}^2} \qquad \mathbf{A}(2) = \sqrt{c_{21}^2 + c_{22}^2 + c_{23}^2} \qquad \mathbf{A}(1) = \sqrt{c_{31}^2 + c_{32}^2 + c_{33}^2} \tag{F.14}$$

which can be resumed in the following equation

$$\mathbf{A}(i) = \sqrt{c_{i1}^2 + c_{i2}^2 + c_{i3}^2} \qquad \text{with } i = 1, 2, 3 \tag{F.15}$$

and are the lengths of the three vectors $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ whose components are stored in each line of the matrix **C**. From a physical point of view, these computed lengths are the lengths of the lattice parameters in the direct space, and the matrix **C** contains in each row a direct lattice vector. Then, in lines 247-252, the three fundamental angles between the three vectors are computed, starting from the following equation which relates the 3-dimensional vectors **a** and **b** with the angle $\hat{\mathbf{ab}}$ between them

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\|\cos(\hat{\mathbf{ab}}) \tag{F.16}$$

where $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the lengths of the two vectors. In the same way,

$$\text{angle } \alpha \text{ between } \mathbf{c}_2 \text{ and } \mathbf{c}_3 : \cos\alpha = \frac{\mathbf{c}_2 \cdot \mathbf{c}_3}{\|\mathbf{c}_2\|\|\mathbf{c}_3\|} = \frac{c_{21}c_{31} + c_{22}c_{32} + c_{23}c_{33}}{\|\mathbf{c}_2\|\|\mathbf{c}_3\|} \tag{F.17}$$

$$\text{angle } \beta \text{ between } \mathbf{c}_1 \text{ and } \mathbf{c}_3 : \cos\beta = \frac{\mathbf{c}_1 \cdot \mathbf{c}_3}{\|\mathbf{c}_1\|\|\mathbf{c}_3\|} = \frac{c_{11}c_{31} + c_{12}c_{32} + c_{13}c_{33}}{\|\mathbf{c}_1\|\|\mathbf{c}_3\|} \tag{F.18}$$

$$\text{angle } \gamma \text{ between } \mathbf{c}_1 \text{ and } \mathbf{c}_2 : \cos\gamma = \frac{\mathbf{c}_1 \cdot \mathbf{c}_2}{\|\mathbf{c}_1\|\|\mathbf{c}_2\|} = \frac{c_{11}c_{21} + c_{12}c_{22} + c_{13}c_{23}}{\|\mathbf{c}_1\|\|\mathbf{c}_2\|} \tag{F.19}$$

Since the lengths of the vectors involved have been previously computed at line 246, they can be set equal to $\|\mathbf{c}_1\| = \mathbf{A}(1)$, $\|\mathbf{c}_2\| = \mathbf{A}(2)$, $\|\mathbf{c}_3\| = \mathbf{A}(3)$, so that the angles α, β and γ can be computed from the previous equations as:

$$\text{angle } \alpha \text{ between } \mathbf{c}_2 \text{ and } \mathbf{c}_3 : \qquad \alpha = \mathbf{A}(4) = \arccos\left(\frac{c_{21}c_{31} + c_{22}c_{32} + c_{23}c_{33}}{\mathbf{A}(2)\mathbf{A}(3)}\right) \quad \text{[degrees]} \tag{F.20}$$

$$\text{angle } \beta \text{ between } \mathbf{c}_1 \text{ and } \mathbf{c}_3 : \qquad \beta = \mathbf{A}(5) = \arccos\left(\frac{c_{11}c_{31} + c_{12}c_{32} + c_{13}c_{33}}{\mathbf{A}(1)\mathbf{A}(3)}\right) \quad \text{[degrees]} \tag{F.21}$$

$$\text{angle } \gamma \text{ between } \mathbf{c}_1 \text{ and } \mathbf{c}_2 : \qquad \gamma = \mathbf{A}(6) = \arccos\left(\frac{c_{11}c_{21} + c_{12}c_{22} + c_{13}c_{23}}{\mathbf{A}(1)\mathbf{A}(2)}\right) \quad \text{[degrees]} \tag{F.22}$$

These angles are saved in the 4-th, 5-th and 6-th elements of the vector $\mathbf{A}$ as indicated in the previous equations (F.20)-(F.22).
Finally, at line 253 the subroutine INV123 is called (see Section F.3) with the following input parameters:

INV123(C,D,A(7),INF(10))

so that it modifies and returns the following quantities (depending on the system dimensionality INF(10) given in input):

1. (3,3) matrix $\mathbf{D} = \mathbf{C}^{-1}$: inverse of the matrix $\mathbf{C}$
2. real value $\mathbf{A}(7)$: value of the 3-dimensional volume defined by the row-by-row vectors in matrix $\mathbf{C}$ (i.e. by the three vectors $\mathbf{c}_i$, $i = 1, 2, 3$) in a 3-dimensional space

F.5 Subroutine MCM (libx5_com.f)

Input : integers IS1, IS2, IS3 (shrinking factors) ; integers IS, ISJ1, ISJ2, ISJ3.
The subroutine modifies the values of the integers IS, ISJ1, ISJ2, ISJ3 on the base of the values of the integers IS1, IS2, IS3 representing the shrinking factors.
The variable IS returns the maximum common multiple (mcm) among the shrinking factors IS1, IS2, IS3, while ISJ1 = mcm(IS1,IS2,IS3)/IS1, ISJ2 = mcm(IS1,IS2,IS3)/IS2 and ISJ3 = mcm(IS1,IS2,IS3)/IS3.

line	**SUBROUTINE MCM: defined in libx5_com.f**	CRYSTAL version: SVN_DEV1218
238	SUBROUTINE MCM(IS1,IS2,IS3,IS,ISJ1,ISJ2,ISJ3)	
	USE NUMBERS	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	IS=IS1	
	N=IS2	
243	1 DO 2 L=1,N	BEGIN Loop to find the mcm among IS1, IS2 and IS3
	M=IS*L	
	IF(MOD(M,N).EQ.0)GOTO 3	
	2 CONTINUE	See text and (F.23) for the description of the algorithm
	3 IS=M	IS = M = mcm(IS1,IS2,IS3)
	IF(N.EQ.IS3)GOTO 5	
	N=IS3	
250	GOTO 1	END Loop to find the mcm among IS1, IS2 and IS3
	5 ISJ1=M/IS1	ISJ1 = mcm(IS1,IS2,IS3)/IS1
	ISJ2=M/IS2	ISJ2 = mcm(IS1,IS2,IS3)/IS1
	ISJ3=M/N	ISJ3 = mcm(IS1,IS2,IS3)/IS3
	RETURN	
255	END	

Table F.12: Code and description of subroutine MCM (libx5_com.f)

The main part of this subroutine is the algorithm performed in lines 243-250 to find the maximum common multiple (mcm) among the shrinking factors IS1, IS2, IS3. This is a simple algorithm and can be schematized with a structured code paradigm as follows:

```
DO L = 1,IS2
            M = IS1*L
            IF(MOD(M,IS2) == 0) EXIT LOOP
ENDDO
IS = M
IF(IS2 != IS3)
            DO L = 1,IS3
                        M = IS*L
                        IF(MOD(M,IS3) == 0) EXIT LOOP
            ENDDO
            IS = M
ENDIF
```
(F.23)

After that the maximum common multiple among the shrinking factors has been computed and stored in the variable IS, other three variables are computed as follows:

$$\mathrm{ISJ}i = \frac{\mathrm{mcm(IS1,IS2,IS3)}}{\mathrm{IS}i} \qquad i = 1, 2, 3 \tag{F.24}$$

The four integers IS, ISJ1, ISJ2, ISJ3 are passed as argument in the function and are modified by the function itself, so they have an in-out intent and they are modified after the subroutine mcm is called in the code.

F.6 Subroutine VRSLAT (both1.f)

Input : (3,n_k) matrix JJ containing the shrank coordinates of the points in the reciprocal space (where n_k is the number of $\boldsymbol{k}$ points considered in the reciprocal space) and shrinking factors IS1, IS2, IS3 have to be previously initialized and well-defined.
The subroutine modifies (i) the vector LATVRS(NKF), where NKF is the number of $\boldsymbol{k}$ points, (ii) the integer LATSUM, (iii) the following INF parameters: INF(88), INF(89), INF(90), INF(91), and (iv) the vectors of real variables COSSMA(INF(88)) and SINSMA(INF(88)).

line	**SUBROUTINE VRSLAT: defined in both1.f**	CRYSTAL version: SVN_DEV1218
5074	SUBROUTINE VRSLAT	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE RETIC_MODULE	
	USE MEMORY_USE	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	CHARACTER(LEN=6),PARAMETER :: ZNAMZ='VRSLAT'	
	LATSUM=0	
5083	DO K=1,NKF	BEGIN loop on NKF = total number of $\boldsymbol{k}$ points
	LATVRS(K)=0	LATVRS(K) = 0 if the K-th $\boldsymbol{k}$ point is complex
	IF(MOD(JJ(1,K)*2,IS1).NE.0)CYCLE	JJ(1,K) = k_x for the K-th $\boldsymbol{k}$ point, IS1 = n_x, see condition (F.28)
	IF(MOD(JJ(2,K)*2,IS2).NE.0)CYCLE	JJ(2,K) = k_y for the K-th $\boldsymbol{k}$ point, IS2 = n_y, see condition (F.29)
	IF(MOD(JJ(3,K)*2,IS3).EQ.0)THEN	JJ(3,K) = k_z for the K-th $\boldsymbol{k}$ point, IS3 = n_z, see condition (F.30)
	LATVRS(K)=1	LATVRS(K) = 1 if the K-th $\boldsymbol{k}$ point is real
	LATSUM=LATSUM+1	Increment LATSUM if the K-th $\boldsymbol{k}$ point is real
	ENDIF	
5091	ENDDO	END loop on NKF = total number of $\boldsymbol{k}$ points
	IF(LATSUM.EQ.NKF) THEN	If LATSUM = NKF, then all the $\boldsymbol{k}$ points are real
	INF(81)=1	INF(81) $\neq$ 0, all $\boldsymbol{k}$ points are real
	ELSE	Else
	INF(81)=0	INF(81) = 0, there is at least one $\boldsymbol{k}$ complex point
	ENDIF	Endif
	CALL MCM(IS1,IS2,IS3,ISM,ISJ1,ISJ2,ISJ3)	Computes the MCM among IS1, IS2 and IS3, see Section F.5
	IF(ALLOCATED(COSSMA)) THEN	
	CALL CRYDEALLOC(COSSMA,ZNAMZ,'COSSMA')	
	CALL CRYDEALLOC(SINSMA,ZNAMZ,'SINSMA')	
	ENDIF	
	CALL CRYALLOC(COSSMA,ISM,ZNAMZ,'COSSMA')	
	CALL CRYALLOC(SINSMA,ISM,ZNAMZ,'SINSMA')	Modified by MCM subroutine, see Section F.5:
	INF(88)=ISM	INF(88) = mcm(IS1,IS2,IS3)
	INF(89)=ISJ1	INF(89) = mcm(IS1,IS2,IS3)/IS1
	INF(90)=ISJ2	INF(90) = mcm(IS1,IS2,IS3)/IS2
	INF(91)=ISJ3	INF(91) = mcm(IS1,IS2,IS3)/IS3
	VRS=PAR(34)/ISM	PAR(34) = 2·PAR(1) = 2π $\rightarrow$ VRS = 2π/mcm(IS1,IS2,IS3)
	COSSMA(1)=1._FLOAT	COSSMA(1) = 1 (K = 0 first $\boldsymbol{k}$ point : $\boldsymbol{k} = (0,0,0)$)
	SINSMA(1)=0._FLOAT	SINSMA(1) = 0 (K = 0 first $\boldsymbol{k}$ point : $\boldsymbol{k} = (0,0,0)$)
	DO K=1,ISM-1	BEGIN Loop on K=1,mcm(IS1,IS2,IS3)-1
	WRS=K*VRS	
	COSSMA(K+1)=COS(WRS)	COSSMA(K+1) = COS(2πK/mcm(IS1,IS2,IS3))
	SINSMA(K+1)=SIN(WRS)	SINSMA(K+1) = SIN(2πK/mcm(IS1,IS2,IS3))
	ENDDO	END Loop on K=1,mcm(IS1,IS2,IS3)-1
	RETURN	
	END	

Table F.14: Code and description of subroutine VRSLAT (both1.f)

Consider a point $\boldsymbol{g}$ in the direct space and a point $\boldsymbol{k}$ in the reciprocal space defined as

$$\boldsymbol{g} = \sum_{i=1}^{3} g_i \mathbf{a}_i \qquad \text{and} \qquad \boldsymbol{k} = \sum_{i=1}^{3} k_i \mathbf{b}_i \qquad i = 1,2,3\ (= x,y,z) \tag{F.25}$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ are the primitive lattice vectors in the direct and in the reciprocal space, respectively, while $g_i \in \mathbb{N}$ and k_i are their components, respectively. A point $\boldsymbol{k}$ in the reciprocal space is defined real or complex depending on the phase associated to it, that is written as

$$\exp\left[i\,\boldsymbol{g}\cdot\left(\frac{\boldsymbol{k}}{\boldsymbol{n_k}}\right)\right] = \exp\left(i\sum_{i=1}^{3}\sum_{j=1}^{3}\frac{g_i\,k_j}{n_j}\,\mathbf{a}_i\cdot\mathbf{b}_j\right) = \exp\left(i\sum_{i=1}^{3}\sum_{j=1}^{3}\frac{g_i\,k_j}{n_j}\,2\pi\delta_{ij}\right)$$

$$= \exp\left(2\pi\, i \sum_{i=1}^{3} \frac{g_i\, k_i}{n_i}\right) = \exp\left[2\pi\, i \left(\frac{g_x\, k_x}{n_x} + \frac{g_y\, k_y}{n_y} + \frac{g_z\, k_z}{n_z}\right)\right]$$
$$= \cos\left[\pi \left(g_x \frac{2k_x}{n_x} + g_y \frac{2k_y}{n_y} + g_z \frac{2k_z}{n_z}\right)\right] + i \sin\left[\pi \left(g_x \frac{2k_x}{n_x} + g_y \frac{2k_y}{n_y} + g_z \frac{2k_z}{n_z}\right)\right] \quad \text{(F.26)}$$

where $\boldsymbol{n_k}$ is the vector that defines the number of points in which the first Brillouine zone is sampled along the three space directions, i.e. $\boldsymbol{n_k} = (n_x, n_y, n_z)$.
Since the terms g_i are integers ($g_i \in \mathbb{N}$, $i = x, y, z$), the argument of the complex part in (F.26) is a multiple of $i\pi$ if the terms $2k_i/n_i$ ($i = x, y, z$) are integer numbers. In this case, we can write

$$\frac{2k_i}{n_i} = m_i \in \mathbb{N} \qquad (i = x, y, z) \quad \rightarrow \quad \sin\left(\pi \sum_{i=1}^{3} g_i \frac{2k_i}{m_i}\right) = 0 \quad \text{(F.27)}$$

so that the complex part in (F.26) is zero, and the phase associated to the $\boldsymbol{k}$ point is real, so that the correspondent $\boldsymbol{k}$ point is real. Otherwise, if the three conditions in (F.27) are not satisfied, the phase associated to the $\boldsymbol{k}$ point is complex, so that the correspondent $\boldsymbol{k}$ point is complex.
The three conditions (F.27) about the phase factor are checked in lines 5083-5091, with a loop over the total number of points in the reciprocal space. Per each $\boldsymbol{k}$ point, the following algorithm is applied:

DO K = 1,NKF

$$\text{(i) if mod}\left(\frac{2k_x}{n_x}\right) \neq 0 \quad \rightarrow \quad \text{LATVRS(K)} = 0 \quad \rightarrow \quad \text{K-th point } \boldsymbol{k} \in \mathbb{C} \quad \text{(F.28)}$$

$$\text{(ii) if mod}\left(\frac{2k_y}{n_y}\right) \neq 0 \quad \rightarrow \quad \text{LATVRS(K)} = 0 \quad \rightarrow \quad \text{K-th point } \boldsymbol{k} \in \mathbb{C} \quad \text{(F.29)}$$

$$\text{(iii) if mod}\left(\frac{2k_z}{n_z}\right) = 0 \quad \rightarrow \quad \begin{cases} \text{LATVRS(K)} = 1 \\ \text{LATSUM++} \end{cases} \quad \rightarrow \quad \text{K-th point } \boldsymbol{k} \in \mathbb{R} \quad \text{(F.30)}$$

ENDDO

In this way, if one of the ratios $2k_i/n_i$ ($i = x, y, z$) is not an integer, the entire sum in the arguments in (F.26) is not an integer, then the complex part is not equal to zero and the K-th $\boldsymbol{k}$ point is complex (LATVRS(K) = 0). Otherwise, if *all* the three ratios $2k_i/n_i$ ($i = x, y, z$) are integers, the sum in the arguments in (F.26) is an integer, then the complex part is equal to zero, and the K-th $\boldsymbol{k}$ point is real (LATVRS(K) = 1). The variable LATSUM counts the number of real $\boldsymbol{k}$ points.
Finally, the subroutine computes the cosine and the sine value associated at the $\boldsymbol{k}$ points using the

F.7 Subroutine EXPU (both4.f)

Input : integers J1, J2 and J3 (components of the $\boldsymbol{k}$ point vector in the reciprocal space : J1 = JJ(1, K) = k_x(K), J2 = JJ(2, K) = k_y(K) and J3 = JJ(3, K) = k_z(K), with K = $1, ..., n_k$ where n_k is the total number of $\boldsymbol{k}$ points in the reciprocal space and JJ is the $(3, n_k)$ matrix containing the shrank coordinates of the correspondent $\boldsymbol{k}$ points in the reciprocal space).
The subroutine modifies the (2, INF(28)) matrix EX, where INF(28) is the maximum number of $\boldsymbol{g}$ vectors for bielectronic integrals (which is set equal to INF(79) (= number of direct lattice vectors available) in locali.f library, before calling the subroutine EXPU).
Notes: (i) the parameters INF(88), INF(89), INF(90) and INF(91) used here have been previously modified by the VRSLAT subroutine, see Section F.6, (ii) the vectors COSSMA and SINSMA here used have been previously initialized in the VRSLAT subroutine, see Section F.6.

		CRYSTAL version: SVN_DEV1218
line	**SUBROUTINE EXPU: defined in both4.f**	Complex case of subroutine EXPT (see Section F.8)
5778	SUBROUTINE EXPU(J1,J2,J3)	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE GVECT_MODULE	
	USE RETIC_MODULE	
	USE EXPO_MODULE	INF(28) = max $\boldsymbol{g}$ vectors for biel. integrals (in locali is set = INF(79))
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	INF(88)-INF(91) modified by VRSLAT subroutine, see Section F.6
	J1VRS=J1*INF(89)	J1VRS = J1·INF(89) = J1·mcm(IS1,IS2,IS3)/IS1
	J2VRS=J2*INF(90)	J2VRS = J2·INF(90) = J2·mcm(IS1,IS2,IS3)/IS2
	J3VRS=J3*INF(91)	J3VRS = J3·INF(91) = J3·mcm(IS1,IS2,IS3)/IS3
	ISM=INF(88)	ISM = INF(88) = mcm(IS1,IS2,IS3)
	IQ=ISM*8192	IQ = mcm(IS1,IS2,IS3)·8192
	EX(1,1)=1._FLOAT	COSSMA(1) = 1 → EX(1,1) = 1
	EX(2,1)=0._FLOAT	SINSMA(1) = 0 → EX(2,1) = 0
	DO MG=2,INF(28),2	BEGIN Loop on MG = 2,INF(28),2 (direct lattice vectors available)
5794	JL=MOD(J1VRS*LG(1,MG)+J2VRS*LG(2,MG)+ *J3VRS*LG(3,MG)+IQ,ISM)	Computation of index JL, see eq. (F.35)
	VRS=COSSMA(JL+1)	
	WRS=SINSMA(JL+1)	
	EX(1,MG)=VRS	Initialization of EX(1,MG) and EX(2,MG), see eq.s (F.36)-(F.37)
	EX(2,MG)=WRS	
	EX(1,MG+1)=VRS	Initialization of EX(1,MG+1) and EX(2,MG+1), see eq.s (F.36)-(F.37)
	EX(2,MG+1)=-WRS	
	ENDDO	END Loop on MG = 2,INF(28),2 (direct lattice vectors available)
	RETURN	
5803	END	

Table F.16: Code and description of subroutine EXPU (both4.f)

The subroutine initializes the matrix EX in the following way:

$$\text{EX(1,1)} = \text{COSSMA(1)} = 1 \tag{F.31}$$

$$\text{EX(2,1)} = \text{SINSMA(1)} = 0 \tag{F.32}$$

Considering the argument of the cosine and sine functions related to the exponents

$$\exp\left[i\,\boldsymbol{g}\cdot\left(\frac{\boldsymbol{k}}{\boldsymbol{n_k}}\right)\right] = \cos\left[\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right)\right] + i\sin\left[\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right)\right] \tag{F.33}$$

they can be rewritten as follows

$$\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right) = \frac{2\pi}{m}\left(g_x\,k_x\,\frac{m}{n_x} + g_y\,k_y\,\frac{m}{n_y} + g_z\,k_z\,\frac{m}{n_z}\right) \tag{F.34}$$

where m = mcm(IS1,IS2,IS3) = mcm(n_x, n_y, n_z). The index JL is computed at line 5794 with the equation

$$\text{JL}(g) = \text{mod}\left(k_x\,g_x\,\frac{m}{n_x} + k_y\,g_y\,\frac{m}{n_y} + k_z\,g_z\,\frac{m}{n_z} + 8192\cdot m, m\right) \qquad \forall\ g \tag{F.35}$$

where g is the number of the direct lattice vectors available. The matrix EX is initialized as

$$\text{EX(1,g)} = \text{COSSMA[JL(g)+1]} \qquad \text{EX(1,g+1)} = \text{COSSMA[JL(g)+1]} \tag{F.36}$$

$$\text{EX(2,g)} = \text{SINSMA[JL(g)+1]} \qquad \text{EX(2,g+1)} = -\text{SINSMA[JL(g)+1]} \tag{F.37}$$

where COSSMA and SINSMA are vectors computed in subroutine VRSLAT (see Section F.6).

F.8 Subroutine EXPT (both4.f)

Input : integers J1, J2 and J3 (components of the $\boldsymbol{k}$ point vector in the reciprocal space : J1 = JJ(1, K) = k_x(K), J2 = JJ(2, K) = k_y(K) and J3 = JJ(3, K) = k_z(K), with K = $1, ..., n_k$ where n_k are the number of $\boldsymbol{k}$ points in the reciprocal space and JJ is the $(3, n_k)$ matrix containing the shrank coordinates of the correspondent $\boldsymbol{k}$ points in the reciprocal space).
The subroutine modifies the (2, INF(28)) matrix EX, where INF(28) is the maximum number of $\boldsymbol{g}$ vectors for bielectronic integrals (which is set equal to INF(79) (= number of direct lattice vectors available) in locali.f library, before calling the subroutine EXPT).
Notes: (i) the parameters INF(88), INF(89), INF(90) and INF(91) used here have been previously modified by the VRSLAT subroutine, see Section F.6, (ii) the vectors COSSMA and SINSMA here used have been previously initialized in the VRSLAT subroutine, see Section F.6.

		CRYSTAL version: SVN_DEV1218
line	**SUBROUTINE EXPT: defined in both4.f**	Real case of subroutine EXPU (see Section F.7)
5756	SUBROUTINE EXPT(J1,J2,J3)	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE GVECT_MODULE	
	USE RETIC_MODULE	
	USE EXPO_MODULE	INF(28) = max $\boldsymbol{g}$ vectors for biel. integrals (in locali is set = INF(79))
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	INF(88)-INF(91) modified by VRSLAT subroutine, see Section F.6
	J1VRS=J1*INF(89)	J1VRS = J1·INF(89) = J1·mcm(IS1,IS2,IS3)/IS1
	J2VRS=J2*INF(90)	J2VRS = J2·INF(90) = J2·mcm(IS1,IS2,IS3)/IS2
	J3VRS=J3*INF(91)	J3VRS = J3·INF(91) = J3·mcm(IS1,IS2,IS3)/IS3
	ISM=INF(88)	ISM = INF(88) = mcm(IS1,IS2,IS3)
	IQ=ISM*8192	IQ = mcm(IS1,IS2,IS3)·8192
	EX(1,1)=1._FLOAT	COSSMA(1) = 1 → EX(1,1) = 1
	DO MG=2,INF(28),2	BEGIN Loop on MG = 2,INF(28),2 (direct lattice vectors available)
5771	JL=MOD(J1VRS*LG(1,MG)+J2VRS*LG(2,MG)+ *J3VRS*LG(3,MG)+IQ,ISM)	Computation of index JL, see eq. (F.42)
	VRS=COSSMA(JL+1)	
	EX(1,MG)=VRS	Initialization of EX(1,MG) see eq. (F.43)
	EX(1,MG+1)=VRS	Initialization of EX(1,MG+1) see eq. (F.43)
	ENDDO	END Loop on MG = 2,INF(28),2 (direct lattice vectors available)
	RETURN	
5777	END	

Table F.18: Code and description of subroutine EXPT (both4.f)

The subroutine initializes the matrix EX in the following way:

$$\text{EX(1,1)} = \text{COSSMA(1)} = 1 \tag{F.38}$$

The subroutine EXPT is called when considering real $\boldsymbol{k}$ points (contrary to the subroutine EXPU, which is called for complex $\boldsymbol{k}$ points, see Section F.7), so that the complex part in the following exponent expression

$$\exp\left[i\,\boldsymbol{g}\cdot\left(\frac{\boldsymbol{k}}{\boldsymbol{n_k}}\right)\right] = \cos\left[\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right)\right] + i\sin\left[\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right)\right] \tag{F.39}$$

is equal to zero (since the sine function is equal to zero for real $\boldsymbol{k}$ points). Equation (F.39) then becomes, in the case of real $\boldsymbol{k}$ points:

$$\exp\left[i\,\boldsymbol{g}\cdot\left(\frac{\boldsymbol{k}}{\boldsymbol{n_k}}\right)\right] = \cos\left[\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right)\right] \tag{F.40}$$

The argument of the cosine function in the previous equation can be rewritten as follows

$$\pi\left(g_x\,\frac{2k_x}{n_x} + g_y\,\frac{2k_y}{n_y} + g_z\,\frac{2k_z}{n_z}\right) = \frac{2\pi}{m}\left(g_x\,k_x\,\frac{m}{n_x} + g_y\,k_y\,\frac{m}{n_y} + g_z\,k_z\,\frac{m}{n_z}\right) \tag{F.41}$$

where m = mcm(IS1,IS2,IS3) = mcm(n_x, n_y, n_z). The index JL is computed at line 5771 following the equation

$$\mathrm{JL}(g) = \mathrm{mod}\left(k_x\, g_x\, \frac{m}{n_x} + k_y\, g_y\, \frac{m}{n_y} + k_z\, g_z\, \frac{m}{n_z} + 8192 \cdot m, m \right) \qquad \forall\ g \tag{F.42}$$

where g is the number of the direct lattice vectors available. The matrix EX is initialized as

$$\mathrm{EX(1,g)} = \mathrm{COSSMA[JL(g)+1]} \qquad \mathrm{EX(1,g+1)} = \mathrm{COSSMA[JL(g)+1]} \tag{F.43}$$

where COSSMA and SINSMA are vectors computed in subroutine VRSLAT (see Section F.6).

F.9 Subroutine SYMHEQ (libxa.f)

Input : NSIZE·NSIZE vector **A** (representing a (NSIZE,NSIZE) real-elements square matrix) with float type elements, and integer NSIZE (which is equal to the number or rows = number of columns of the real-elements square matrix associated to the input vector **A**).
The subroutine modifies the vector **A**, by making the correspondent matrix symmetric.

		CRYSTAL version: SVN_DEV1218
line	**SUBROUTINE SYMHEQ: defined in libxa.f**	Real case of subroutine SYMHER (see Section F.10)
2	SUBROUTINE SYMHEQ(A,NSIZE)	NSIZE $\equiv n$ (number of rows and columns of the matrix $\mathbf{A}_m$)
	USE NUMBERS	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	DIMENSION A(*)	
	MM=0	MM $\equiv k = 0$
	DO M=2,NSIZE	BEGIN Loop on m = 2,n (L1)
	MM=MM+NSIZE	$k = k + n$
	NN=M	NN $\equiv s = m$
	DO N=1,M-1	BEGIN Loop on N = 1,$m-1$ (L2)
	A(MM+N)=A(NN)	$\mathbf{A}(k+N) = \mathbf{A}(s)$
	NN=NN+NSIZE	$s = s + n$
	ENDDO	END Loop on N = 1,$m-1$
	ENDDO	END Loop on m = 2,n
	RETURN	
16	END	

Table F.20: Code and description of subroutine SYMHEQ (libxa.f)

Example

Suppose that matrix $\mathbf{A}_m$ is a real (3,3) matrix $\rightarrow n = 3$
The matrix can be written in the following way:

$$\mathbf{A}_m = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{F.44}$$

and it can be rearranged as a 9-elements vector as

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{F.45}$$

The subroutine SYMHEQ treats the matrix $\mathbf{A}_m$ as a vector, in the same way as described previously in expression (F.45) for the case of a real (3,3) matrix. The input vector **A** is then rearranged so that the correspondent matrix $\mathbf{A}_m$ becomes symmetric, and the result is overwritten in the same input vector **A**. Let us follow the loops L1 and L2 (using the notation reported in Table F.20), to understand the logic of this rearrangement:

$k = 0$
L1: $m = 2$
$k = k + n = 0 + 3 = 3$
$s = m = 2$
L2: N = 1
$\mathbf{A}(4) = \mathbf{A}(2) \longrightarrow \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{12} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{pmatrix}$
$s = s + n = 2 + 3 = 5$
L1: $m = 3$
$k = k + n = 3 + 3 = 6$
$s = m = 3$
L2: N = 1
$\mathbf{A}(7) = \mathbf{A}(3) \longrightarrow \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{12} & a_{22} & a_{23} & a_{13} & a_{32} & a_{33} \end{pmatrix}$
$s = s + n = 3 + 3 = 6$
L2: N = 2
$\mathbf{A}(8) = \mathbf{A}(6) \longrightarrow \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{12} & a_{22} & a_{23} & a_{13} & a_{23} & a_{33} \end{pmatrix}$

$s = s + n = 6 + 6 = 12$

Therefore, the output of the subroutine is

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{12} & a_{22} & a_{23} & a_{13} & a_{23} & a_{33} \end{pmatrix} \tag{F.46}$$

which can be read more easily if reshaped in a (3,3) matrix as follows:

$$\mathbf{A}_m = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \tag{F.47}$$

which is a symmetric matrix (i.e. $a_{ij} = a_{ji}$ with $i = 1, 2, 3$ and $j = 1, 2, 3$) constructed on the base of the original matrix upper triangular elements (the original (input) matrix form is reported in (F.44)).

F.10 Subroutine SYMHER (libxa.f)

Input : 2·NSIZE·NSIZE vector **A** (representing a (NSIZE,NSIZE) complex-elements square matrix) with float type elements, and integer NSIZE (which is equal to the number or rows = number of columns of the complex-elements square matrix associated to the input vector **A**).
The subroutine modifies the vector **A**, by making the correspondent matrix Hermitian.

		CRYSTAL version: SVN_DEV1218
line	**SUBROUTINE SYMHER: defined in libxa.f**	Complex case of subroutine SYMHEQ (see Section F.9)
18	SUBROUTINE SYMHER(A,NSIZE)	NSIZE $\equiv n$ (number of rows and columns of the matrix $\mathbf{A}_m$)
	USE NUMBERS	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	DIMENSION A(*)	
	N2=NSIZE+NSIZE	N2 = 2·NSIZE = $2n$
	MM=0	MM $\equiv k = 0$
	DO M=2,NSIZE	BEGIN Loop on m = 2,n (L1)
	MM=MM+N2	$k = k + 2n$
	NN=M+M	NN $\equiv s = 2m$
	LL=MM	LL $\equiv r = k$
	DO N=1,M-1	BEGIN Loop on N = 1,$m-1$ (L2)
	LL=LL+2	$r = r + 2$
	A(LL-1)=A(NN-1)	$\mathbf{A}(r-1) = \mathbf{A}(s-1)$
	A(LL)=-A(NN)	$\mathbf{A}(r) = -\mathbf{A}(s)$
	NN=NN+N2	$s = s + 2n$
	ENDDO	END Loop on N = 1,$m-1$
	ENDDO	END Loop on m = 2,n
	RETURN	
36	END	

Table F.22: Code and description of subroutine SYMHER (libxa.f)

Example

Suppose that matrix **A** is a complex (3,3) matrix → $n = 3$
The matrix can be written as the sum of two matrices, one with real elements and the other one with the coefficients associated to the complex part, in the following way:

$$\mathbf{A}_m = \begin{pmatrix} a_{11} + i\,a_{c,11} & a_{12} + i\,a_{c,12} & a_{13} + i\,a_{c,13} \\ a_{21} + i\,a_{c,21} & a_{22} + i\,a_{c,22} & a_{23} + i\,a_{c,23} \\ a_{31} + i\,a_{c,31} & a_{32} + i\,a_{c,32} & a_{33} + i\,a_{c,33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} + i \begin{pmatrix} a_{c,11} & a_{c,12} & a_{c,13} \\ a_{c,21} & a_{c,22} & a_{c,23} \\ a_{c,31} & a_{c,32} & a_{c,33} \end{pmatrix} \quad \text{(F.48)}$$

and it can be rearranged as a 18-elements vector as

$$\begin{aligned} \mathbf{A} = (&a_{11} \quad a_{c,11} \quad a_{12} \quad a_{c,12} \quad a_{13} \quad a_{c,13} \quad a_{21} \quad a_{c,21} \quad a_{22} \quad a_{c,22} \quad a_{23} \quad a_{c,23} \\ &a_{31} \quad a_{c,31} \quad a_{32} \quad a_{c,32} \quad a_{33} \quad a_{c,33}) \end{aligned} \quad \text{(F.49)}$$

The subroutine SYMHER treats the matrix $\mathbf{A}_m$ as a vector, in the same way as described previously in expression (F.49) for the case of a complex (3,3) matrix. The input vector **A** is then rearranged so that the correspondent matrix $\mathbf{A}_m$ becomes Hermitian, and the result is overwritten in the same input vector **A**. Let us follow the loops L1 and L2 (using the notation reported in Table F.22), to understand the logic of this rearrangement:

$k = 0$
L1: $m = 2$
$k = k + 2n = 0 + 2 \cdot 3 = 6$
$s = 2m = 2 \cdot 2 = 4$
$r = k = 6$
L2: N = 1
$r = r + 2 = 6 + 2 = 8$
$\mathbf{A}(7) = \mathbf{A}(3)$
$\mathbf{A}(8) = -\mathbf{A}(4)$
→ $\mathbf{A} = (a_{11} \quad a_{c,11} \quad a_{12} \quad a_{c,12} \quad a_{13} \quad a_{c,13} \quad a_{12} \quad -a_{c,12} \quad a_{22} \quad a_{c,22} \quad a_{23} \quad a_{c,23} \quad a_{31} \quad a_{c,31} \quad a_{32} \quad a_{c,32} \quad a_{33} \quad a_{c,33})$

$s = s + 2n = 4 + 6 = 8$
L1: $m = 3$
$k = k + 2n = 6 + 6 = 12$
$s = 2m = 2 \cdot 3 = 6$
$r = k = 12$
L2: N = 1
$r = r + 2 = 12 + 2 = 14$
$\mathbf{A}(13) = \mathbf{A}(5)$
$\mathbf{A}(14) = -\mathbf{A}(6)$
$\rightarrow \mathbf{A} = (a_{11} \quad a_{c,11} \quad a_{12} \quad a_{c,12} \quad a_{13} \quad a_{c,13} \quad a_{12} \quad -a_{c,12} \quad a_{22} \quad a_{c,22} \quad a_{23} \quad a_{c,23} \quad a_{13} \quad -a_{c,13} \quad a_{32} \quad a_{c,32} \quad a_{33} \quad a_{c,33})$
$s = s + 2n = 6 + 6 = 12$
L2: N = 2
$r = r + 2 = 14 + 2 = 16$
$\mathbf{A}(15) = \mathbf{A}(11)$
$\mathbf{A}(16) = -\mathbf{A}(12)$
$\rightarrow \mathbf{A} = (a_{11} \quad a_{c,11} \quad a_{12} \quad a_{c,12} \quad a_{13} \quad a_{c,13} \quad a_{12} \quad -a_{c,12} \quad a_{22} \quad a_{c,22} \quad a_{23} \quad a_{c,23} \quad a_{13} \quad -a_{c,13} \quad a_{23} \quad -a_{c,23} \quad a_{33} \quad a_{c,33})$
$s = s + 2n = 12 + 6 = 18$

Therefore, the output of the subroutine is

$\mathbf{A} = (a_{11} \quad a_{c,11} \quad a_{12} \quad a_{c,12} \quad a_{13} \quad a_{c,13} \quad a_{12} \quad -a_{c,12} \quad a_{22} \quad a_{c,22} \quad a_{23} \quad a_{c,23} \quad a_{13} \quad -a_{c,13} \quad a_{23} \quad -a_{c,23} \quad a_{33} \quad a_{c,33})$

which can be read more easily if reshaped in a (3,3) matrix as follows:

$$\mathbf{A}_m = \begin{pmatrix} a_{11} + i\,a_{c,11} & a_{12} + i\,a_{c,12} & a_{13} + i\,a_{c,13} \\ a_{12} - i\,a_{c,12} & a_{22} + i\,a_{c,22} & a_{23} + i\,a_{c,23} \\ a_{13} - i\,a_{c,13} & a_{23} - i\,a_{c,23} & a_{33} + i\,a_{c,33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} + i \begin{pmatrix} a_{c,11} & a_{c,12} & a_{c,13} \\ -a_{c,12} & a_{c,22} & a_{c,23} \\ -a_{c,13} & -a_{c,23} & a_{c,33} \end{pmatrix}$$

which is a Hermitian matrix (i.e. $a_{ij} = a^*_{ji}$ with $i = 1, 2, 3$ and $j = 1, 2, 3$) constructed on the base of the original matrix upper triangular elements (the original (input) matrix form is reported in (F.48)).

F.11 Subroutine GENERATE_SAED (geometry.f)

Input : logical variable LOGIK for printing options (the value of this input variable now does not change anything, but once upon a time if the input variable LOGIK = true, the output matrix SAED(1:9,1:NSAED) will be printed in the output file: see line 3465 that is now commented).
The subroutine modifies the matrix SAED(1:9,1:NSAED), saving in it the NSAED symmetrized orthonormal symmetry allowed elastic distortions, each of which is represented by a 9-element vector (regarded as a (NDIM,NDIM) matrix) containing the components of the symmetry allowed elastic distortions.

line	**SUBROUTINE GENERATE_SAED: defined in geometry.f** CRYSTAL version: SVN_DEV1218	
3381	SUBROUTINE GENERATE_SAED(LOGIK)	
	CVRS TO GENERATE THE SYMMETRY ALLOWED ELASTIC DISTORTIONS - 22/03/2002	
	C..modified for crystallographic cell deformations and partial	
	C..optimization - MicCat (20/04/2006)	
	USE NUMBERS	
	USE PARINF_MODULE	
	USE XYVDIM_MODULE	
	USE PARAL1_MODULE	
	USE SAED_MODULE	
	USE MEMORY_OPT	
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	
	LOGICAL LOGIK,LOGIL,LOGIM	NOTE: LOGIK input variable is used to define LOGIL variable but
	DIMENSION TMPV(9),UMPV(9),VRSV(9)	LOGIL is not used in this the subroutine
	PARAMETER (TOL=1E-4_FLOAT)	
	CALL EXTGRP	EXTGRP: EXTEND THE GROUP BY ADDING THE INVERSION OPERATOR (both4.f)
	CALL MULTIP	MULTIP: DETERMINE POINT GROUP MULTIPLICATION TABLE (libx5_com.f)
	NDIM=INF(10)	NDIM = INF(10) = dimension of the system
	Clau NSAED=0	
3399	IF(NDIM.EQ.0)RETURN	Molecular case: if NDIM == 0, RETURN
	C.. ELASTIC DISTORTION CORRESPONDING TO ISOTROPIC VOLUME CHANGE	
	! Mcat mods (inf(195))	
	MVF=INF(50)	MVF = INF(50) = number of symmetry operations
	LOGIL=IAMEQ0.AND.LOGIK	LOGIL variable not used in this subroutine (lines 3465 and 3474 are commented)
3404	LOGIM=.FALSE.	INF(195) = 1 → crystallographic deformation (a,b,c,α,β,γ)
3405	IF(INF(195).EQ.1)GOTO 22	If INF(195) = 1 (crystallographic deformation), then LOGIM = .TRUE., IVRS = 0
3406	TMPV(1:9)=0._FLOAT	Initialize TMPV(1:9) as zeros vector
	JVRS=1	JVRS $\equiv j_v = 1$
	DO I=1,NDIM	BEGIN Loop on dimensions I = 1,NDIM
	TMPV(JVRS)=1._FLOAT	Initialize to 1 diagonal elements of TMPV(1:9) vector (seen as a (3,3) matrix), eq. (F.50)
	JVRS=JVRS+4	$j_v = j_v + 4$ select diagonal elements with positions (NDIM,NDIM)
3411	ENDDO	END Loop on dimensions I = 1,NDIM
3412	GOTO 1	GOTO 1 : Normalize elastic distortion ****
3413	22 LOGIM=.TRUE.	△ Case INF(195) = 1 : LOGIM = true
3414	IVRS=0	Case INF(195) = 1 : IVRS $\equiv i_v = 0$
3415	C.. GENERATE THE UNSYMMETRIZED ELASTIC DISTORTIONS	
3416	2 IF(IVRS.EQ.NDIM)GOTO 10 ! Mcat - it was return	◇◇ If i_v == NDIM, then GOTO 10 △△
3417	JVRS=IVRS*3	$j_v = 3i_v$
3418	IVRS=IVRS+1	$i_v = i_v + 1$
3419	KVRS=0	$k_v = 0$
3420	3 IF(KVRS.EQ.IVRS)GOTO 2	◇◇◇ If k_v == i_v, then GOTO 2 ◇◇
3421	LVRS=KVRS*3	$l_v = 3k_v$
3422	KVRS=KVRS+1	$k_v = k_v + 1$
3423	VRSV(1:9)=0._FLOAT	Initialize VRSV(1:9) as zeros vector
3424	VRSV(IVRS+LVRS)=1._FLOAT	VRSV($i_v + l_v$) = 1
3425	VRSV(KVRS+JVRS)=1._FLOAT	VRSV($k_v + j_v$) = 1
3426	C SYMMETRIZE THIS ELASTIC DISTORTION	
3427	TMPV(1:9)=VRSV(1:9)	TMPV(1:9) = VRSV(1:9)
3428	DO I=2,MVF	BEGIN Loop on number of symmetry operations I = 2,MVF
	UMPV(1:9)=0._FLOAT	Initialize matrix for temporary storing the product (F.51)-(F.52)
3430	CALL MXMB(XYV(1,I),1,3,VRSV,1,3,UMPV,1,3,NDIM,NDIM,NDIM)	
3431	CALL MXMB(UMPV,1,3,XYV(1,I),3,1,TMPV,1,3,NDIM,NDIM,NDIM)	
3432	ENDDO	END Loop on number of symmetry operations I = 2,MVF
3433	C... SCHMIDT ORTHOGONALIZE PREVIOUS ELASTIC DISTORTIONS	
3434	DO J=1,NSAED	BEGIN Loop on symmetry allowed elastic distortions J = 1,NSAED
	SUM=0._FLOAT	Schmidt orthogonalization process begins, see eq. (F.53)
	DO I=1,9	....
	SUM=TMPV(I)*SAED(I,J)+SUM	....
	ENDDO	....
	TMPV(1:9)=TMPV(1:9)-SAED(1:9,J)*SUM	Schmidt orthogonalization process ends, see eq. (F.53)
3440	ENDDO	END Loop on symmetry allowed elastic distortions J = 1,NSAED

```
3443  C... NORMALIZE ELASTIC DISTORTION          **** Normalize elastic distortion
3444  1 SUM=0._FLOAT                             s = 0
      DO I=1,9
      SUM=TMPV(I)**2+SUM                         s = Σ_{i=1}^{9} TMPV(i)^2
      ENDDO
3448  IF(SUM.LT.TOL)GOTO 3                       If s < 10^{-4} then GOTO 3 ◇◇◇
3449  NSAED=NSAED+1                              NSAED = NSAED + 1
3450  SUM=SQRT(1._FLOAT/SUM)                     s ← √(1/s)
3451  SAED(1:9,NSAED)=TMPV(1:9)*SUM              SAED(1:9,NSAED) = TMPV(1:9)·s : normalization (see eq. (F.54))
      !Mcat ini
3453  IF(LOGIM)GOTO 3                            If LOGIM = TRUE, then GOTO 3 ◇◇◇
3454  GOTO 22                                    GOTO 22 △
3455  10 CONTINUE                                △△ CONTINUE
3456  INF(120)=NSAED                             INF(120) = NSAED = number of symmetrized distortions
3457  IF(INF(63).EQ.2.AND.INF(195).EQ.1)THEN     If space group INF(63) = 2 and crystallographic deformation (INF(195) = 1)
      TMPV(1:9)=SAED(1:9,3)
      SAED(1:9,3)=SAED(1:9,4)                    Exchange SAED(1:9,3) ↔ SAED(1:9,4)
      SAED(1:9,4)=TMPV(1:9)
3461  ENDIF                                      Endif
      !Mcat end

3465  !!ale IF(LOGIL)THEN
3466  DO K=1,NSAED ! loop Mcat                   BEGIN Loop on symmetry allowed elastic distortions K = 1,NSAED
      WRITE(IOUT,9999)K
      LVRS=0
      DO I=1,NDIM
      WRITE(IOUT,8888)(SAED(LVRS+J,K),J=1,NDIM)  Print the matrix SAED(1:9,1:NSAED) in the output file (see Fig. F.2)
      LVRS=LVRS+3
      ENDDO                                      END Loop on symmetry allowed elastic distortions K = 1,NSAED
3473  ENDDO ! end loop Mcat
3474  !!ale ENDIF

      ! IF(LOGIM)GOTO 3 removed Mcat
      ! GOTO 22 removed Mcat 20 Apr 06
3481  9999 FORMAT(/' SYMMETRY ALLOWED ELASTIC DISTORTION',I2)
3482  8888 FORMAT(3F11.7)
      !                                          NFIXC = number of constraints relating pairs of cell deformations
                                                 NFIXC taken from input file in subroutine CONOPT_, libopt.f library
3484  if(nfixc.eq.0)goto 11                      If NFIXC = 0, no modification of SAED(1:9,1:NSAED) matrix, GOTO 11 □
      do k=1,nfixc                               BEGIN Loop on fixed symmetrized cell deformations K = 1,NFIXC
      if(ifixc1(k).eq.0)goto 12
      sum=0._float
      do l=1,9
      saed(l,ifixc1(k))=coefc(ifixc1(k))*saed(l,ifixc1(k))+
      *coefc(ifixc2(k))*saed(l,ifixc2(k))
      sum=saed(l,ifixc1(k))**2+sum
      enddo
      sum=sqrt(1._float/sum)
      saed(1:9,ifixc1(k))=saed(1:9,ifixc1(k))*sum
      ifixc1(k)=0
      12 if(ifixc2(k).eq.4.and.inf(63).eq.2)imon=1
      enddo                                      END Loop on fixed symmetrized cell deformations K = 1,NFIXC
      do k=1,nfixc                               BEGIN Loop on fixed symmetrized cell deformations K = 1,NFIXC
      do i=ifixc2(k),nsaed-1
      saed(1:9,i)=saed(1:9,i+1)
      enddo
      do l=1,nfixc                               BEGIN Loop on fixed symmetrized cell deformations L = 1,NFIXC
      if(ifixc2(l).gt.ifixc2(k)) ifixc2(l)=ifixc2(l)-1
      enddo                                      END Loop on fixed symmetrized cell deformations L = 1,NFIXC
      nsaed=nsaed-1
      enddo                                      END Loop on fixed symmetrized cell deformations K = 1,NFIXC
      do i=1,nsaed                               BEGIN Loop on symmetry allowed elastic distortions I = 1,NSAED
      write(iout,99)i
      write(iout,8888)(saed(l,i),l=1,9)          Print the matrix SAED(1:9,1:NSAED) in the output file
      enddo                                      END Loop on symmetry allowed elastic distortions I = 1,NSAED
      11 continue                                □ CONTINUE
      inf(120)=nsaed ! added to take into account Mcat nfixc INF(120) = NSAED
      RETURN
      99 FORMAT(/' ACTIVE SYMMETRY ALLOWED ELASTIC DISTORTION',I2)
      END
```

TMPV is a vector with 9 elements, which has to be interpreted as a (3,3) matrix. It is initialized in lines 3406-3411 as a (NDIM,NDIM) unit matrix. The values of its elements depend on the dimensionality of the system and are reported in equation (F.50).

$$
\begin{aligned}
&\text{NDIM} = 0 : \text{subroutine returns} \\
&\text{NDIM} = 1 : \text{TMPV} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&\text{NDIM} = 2 : \text{TMPV} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&\text{NDIM} = 3 : \text{TMPV} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{F.50}
$$

This corresponds to the identity matrix for a (NDIM,NDIM) system and corresponds to the first (i.e. with index NSAED = 1) elastic distortion, which is then normalized in lines 3444-3451 and saved in vector SAED(1:9,1) of the matrix SAED(1:9,1:NSAED) at line 3451.

In lines 3416-3425 each unsymmetrized elastic distortion is created, and its elements are saved in the VRSV(1:9) vector as well as in the TMPV(1:9) vector. These unsymmetrized elastic distortion are independent of the space group of the system, so that they corresponds to a same set of matrices for all the possible systems of a give dimension NDIM. At the same time, the set depends on the dimensionality NDIM of the system considered. Then, once the dimension NDIM of the system is fixed, the set of unsymmetrized elastic distortion is given by the set of matrices reported in Table F.24.

NDIM	NSAED	**UNSYMMETRIZED ELASTIC DISTORTIONS**			
0	0	no elastic distortions (line 3399: IF(NDIM.EQ.0)RETURN)			
1	2 if INF(195) = 0 1 if INF(195) = 1	VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_10) only if INF(195) = 0
		VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_11)
2	4 if INF(195) = 0 3 if INF(195) = 1	VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_20) only if INF(195) = 0
		VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_21)
		VRSV = TMPV = $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_22)
		VRSV = TMPV = $\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_23)
3	7 if INF(195) = 0 6 if INF(195) = 1	VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	(UED_30) only if INF(195) = 0
		VRSV = TMPV = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_31)
		VRSV = TMPV = $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_32)
		VRSV = TMPV = $\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	(UED_33)
		VRSV = TMPV = $\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	(UED_34)
		VRSV = TMPV = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$	(UED_35)
		VRSV = TMPV = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\rightarrow$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	(UED_36)

Table F.24: Unsymmetrized elastic distortion for systems with dimension NDIM.

In lines 3428-3432 the unsymmetrized elastic distortions (represented by the (3,3) matrix **D** which corresponds to the 9-elements vector VRSV(1:9)) are symmetrized, i.e. transformed into the (3,3) matrices

```
****   24 SYMMOPS - TRANSLATORS IN FRACTIONAL UNITS
**** MATRICES AND TRANSLATORS IN THE CRYSTALLOGRAPHIC REFERENCE FRAME
  V INV                    ROTATION MATRICES                      TRANSLATORS
  1   1  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
  2   2 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
  3   4  0.00 -1.00  0.00  1.00 -1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
  4   3 -1.00  1.00  0.00 -1.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
  5   6  1.00 -1.00  0.00  1.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
  6   5  0.00  1.00  0.00 -1.00  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
  7   7  1.00 -1.00  0.00  0.00 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
  8   8 -1.00  0.00  0.00 -1.00  1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
  9   9  0.00  1.00  0.00  1.00  0.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
 10  10  0.00 -1.00  0.00 -1.00  0.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 11  11  1.00  0.00  0.00  1.00 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 12  12 -1.00  1.00  0.00 -0.00  1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 13  13 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
 14  14  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 15  16  0.00  1.00  0.00 -1.00  1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
 16  15  1.00 -1.00  0.00  1.00  0.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.00
 17  18 -1.00  1.00  0.00 -1.00  0.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 18  17  0.00 -1.00  0.00  1.00 -1.00  0.00  0.00  0.00 -1.00  0.00  0.00  0.50
 19  19 -1.00  1.00  0.00 -0.00  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
 20  20  1.00  0.00  0.00  1.00 -1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
 21  21  0.00 -1.00  0.00 -1.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.00
 22  22  0.00  1.00  0.00  1.00  0.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
 23  23 -1.00  0.00  0.00 -1.00  1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
 24  24  1.00 -1.00  0.00  0.00 -1.00  0.00  0.00  0.00  1.00  0.00  0.00  0.50
```

Figure F.1: Example of the output file block in which the matrix XYV(1:9,I), I = 2,...,MVF (with MVF number of symmetry operations) is printed out. The nine elements of the (3,3) matrix representing the symmetry operation are printed row-by-row in the black box highlighted in the figure, for the 24 symmetry operations (case of 3D graphite, space group: 194, $P63/mmc$). Printed by subroutine XYVSHORT, both4.f library (fractional units).

labeled by **E** (which correspond to the 9-elements vectors TMPV(1:9)) through the products performed in lines 3430-3431 that can be formalized as

$$\mathbf{X}_i \cdot {}^t\mathbf{D} \cdot \mathbf{X}_i = \mathbf{E} \qquad \text{where } i = 2, ..., \text{MVF} \tag{F.51}$$

$$\text{XYV(1:9,I)} \cdot {}^t\text{VRSV(1:9)} \cdot \text{XYV(1:9,I)} \rightarrow \text{TMPV(1:9)} \qquad \text{where I} = 2, ..., \text{MVF} \tag{F.52}$$

where XYV(1:9,I) matrices (correspondent to the (3,3) matrices $\mathbf{X}_i$) are the rotation matrices associated to each I-th and i-th symmetry operations [with (i, I) = 2,...,MVF where MVF are the number of symmetry operations for the space group of the system]. The counter I starts from 2 since for I = 1 the vector XYV(1:9,1) is a (NDIM,NDIM) identity matrix, which left the unsymmetrized elastic distortion **D** unvaried. The product (F.51) has to be performed for each NSAED-th elastic distortion represented by the (3,3) matrix **D** or by the 9-elements vector VRSV(1:9) reported in Table F.24. The matrix XYV(1:9,I), I = 2,...,MVF is printed in the output of the CRYSTAL code as reported in Figure F.1 in the case of graphite bulk (3D system).

In lines 3434-3440 the Schmidt orthogonalization process is carried out. Let $\mathcal{B} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v_n})$ be a basis set in a n-dimensional euclidean vector space. Starting from $\mathcal{B}$ it is possible to construct an orthonormal basis set $\mathcal{B}' = (\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e_n})$ in the same space with the so called Gram-Schmidt orhonormalization process:

$$\mathbf{e}_k = \text{vers}(\mathbf{v}_k - (\mathbf{v}_k \cdot \mathbf{e}_1)\mathbf{e}_1) - ... - (\mathbf{v}_k \cdot \mathbf{e}_{k-1})\mathbf{e}_{k-1})) \qquad k = 1, 2, ..., n \tag{F.53}$$

where $\text{vers}(\mathbf{v}) = \mathbf{v}/\|\mathbf{v}\|$ is the versor of the vector $\mathbf{v}$. In our case, the vectors of the basis set $\mathcal{B}$ are the TMPV(1:9) vectors, while the vectors of the orthonormalized basis set $\mathcal{B}'$ are the SAED(1:9,n) vectors, where $n = 1, ..., \text{NSAED}$. In this framework, the index J which runs over the NSAED in line 3434 corresponds to the index k of equation (F.53). In lines 3434-3440 only the orthogonalization is performed, the versors are computed (and then the orthonormalization of the basis vectors SAED(1:9,n) is completed) in lines 3444-3451. Indeed, in lines 3444-3451 the vector TMPV(1:9) with elastic distortions is normalized and saved in matrix SAED(1:9,1:NSAED) through the formula:

$$\text{SAED(1:9,}n\text{)} = \text{TMPV(1:9)} \cdot \frac{1}{\sqrt{\sum_{i=1}^{9} \text{TMPV}^2(i)}} \qquad n = 1, ..., \text{NSAED} \tag{F.54}$$

where NSAED is the number of allowed elastic distortions.

In lines 3466-3482 the matrix SAED(1:9,1:NSAED) is printed in the output file, in the format reported in Figure F.2 for the case of a 3D system (silicon bulk) without symmetry (space group P_1).

```
SYMMETRY ALLOWED ELASTIC DISTORTION 1
 0.5773503  0.0000000  0.0000000
 0.0000000  0.5773503  0.0000000
 0.0000000  0.0000000  0.5773503

SYMMETRY ALLOWED ELASTIC DISTORTION 2
 0.8164966  0.0000000  0.0000000
 0.0000000 -0.4082483  0.0000000
 0.0000000  0.0000000 -0.4082483

SYMMETRY ALLOWED ELASTIC DISTORTION 3
 0.0000000  0.7071068  0.0000000
 0.7071068  0.0000000  0.0000000
 0.0000000  0.0000000  0.0000000

SYMMETRY ALLOWED ELASTIC DISTORTION 4
 0.0000000  0.0000000  0.0000000
 0.0000000  0.7071068  0.0000000
 0.0000000  0.0000000 -0.7071068

SYMMETRY ALLOWED ELASTIC DISTORTION 5
 0.0000000  0.0000000  0.7071068
 0.0000000  0.0000000  0.0000000
 0.7071068  0.0000000  0.0000000

SYMMETRY ALLOWED ELASTIC DISTORTION 6
 0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000  0.7071068
 0.0000000  0.7071068  0.0000000
```

```
TEST10 - SILICON BULK
CRYSTAL
0 0 0
1
5.420000  5.420000  5.420000   90.000  90.000  90.000
2
 14    1.25000000E-01  1.25000000E-01  1.25000000E-01
 14   -1.25000000E-01 -1.25000000E-01 -1.25000000E-01
TESTGEOM
END
```

Figure F.2: Example of the output file block in which the matrix SAED(1:9,1:NSAED) containing the symmetry allowed elastic distortions is printed (left panel), and geometry block of the input file for the 3D system (silicon bulk) considered for the example.

Example

For example, in the case of a NDIM = 3 dimensional crystal system, the symmetrized elastic distortions SAED(1:9,1:NSAED) are created line-by-line in the following way:

Case INF(195) = 0 : standard cell deformations (isotropic, constant volume)

lines	operation
3397	NDIM = 3
3404	LOGIM = FALSE
3406-3411	Creation of (NDIM,NDIM) unit matrix rearranged in a 9-elements vector TMPV(1:9), see eq.s (UED_10), (UED_20), (UED_30)
3412	GOTO 1 →
→ 3444-3451	Normalize of the first elastic distortion (identity) TMPV(1:9) vector and save it in the element SAED(1:9,1), see eq. (F.54)
3449	NSAED = NSAED +1 = 0 + 1 = 1
3454	GOTO 22 →
→ 3413	LOGIM = TRUE
3414	$i_v = 0$
3416	$i_v = 0 \neq$ NDIM, continue
3417	$j_v = 3i_v = 0$
3418	$i_v = i_v + 1 = 1$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 1$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(1) = 1
3425	VRSV(1) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_11), (UED_21), (UED_31)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED > 1), see eq. (F.54)
3449	NSAED = NSAED +1 = 1 + 1 = 2
3453	GOTO 3 →
→ 3420	$k_v = 1 = i_v = 1$, GOTO 2 →
→ 3416	$i_v = 1 \neq$ NDIM, continue
3417	$j_v = 3i_v = 3$
3418	$i_v = i_v + 1 = 2$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 2$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(2) = 1
3425	VRSV(4) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_22), (UED_32)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)

3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 2 + 1 = 3
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 1 \neq i_v = 2$, continue
3421	$l_v = 3k_v = 3$
3422	$k_v = k_v + 1 = 2$
3423	VRSV(1:9) = 0
3424	VRSV(5) = 1
3425	VRSV(5) = 1
3427	TMPV(1:9) = VRSV(1:9) $\rightarrow$ eq.s (UED_23), (UED_33)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED >1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 3 + 1 = 4
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 2 = i_v = 2$, GOTO 2 $\rightarrow$
$\rightarrow$ 3416	$i_v = 2 \neq$ NDIM, continue
3417	$j_v = 3i_v = 6$
3418	$i_v = i_v + 1 = 3$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(3) = 1
3425	VRSV(7) = 1
3427	TMPV(1:9) = VRSV(1:9) $\rightarrow$ eq.s (UED_34)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED >1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 4 + 1 = 5
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 1 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 3$
3422	$k_v = k_v + 1 = 2$
3423	VRSV(1:9) = 0
3424	VRSV(6) = 1
3425	VRSV(8) = 1
3427	TMPV(1:9) = VRSV(1:9) $\rightarrow$ (UED_35)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED >1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 5 + 1 = 6
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 2 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 6$
3422	$k_v = k_v + 1 = 3$
3423	VRSV(1:9) = 0
3424	VRSV(9) = 1
3425	VRSV(9) = 1
3427	TMPV(1:9) = VRSV(1:9) $\rightarrow$ eq.s (UED_36)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED >1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3448	$s = \sum_{i=1}^{9}$ TMPV$(i)^2 < 10^{-4}$, (not increment NSAED, not normalize and not save in SAED(1:9,NSAED)) GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 3 = i_v = 3$, GOTO 2 $\rightarrow$
$\rightarrow$ 3416	$i_v = 3 =$ NDIM, GOTO 10 $\rightarrow$
$\rightarrow$ 3455	continue
3456	INF(120) = NSAED = 6

Case INF(195) = 1 : crystallographic deformation (a,b,c,α,β,γ)

lines	operation
3397	NDIM = 3
3404	LOGIM = FALSE
3405	INF(195) = 1, GOTO 22 →
→ 3413	LOGIM = TRUE
3414	$i_v = 0$
3416	$i_v = 0 \neq$ NDIM, continue
3417	$j_v = 3i_v = 0$
3418	$i_v = i_v + 1 = 1$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 1$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(1) = 1
3425	VRSV(1) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_11), (UED_21), (UED_31)
3428-3432	Symmetrize the first elastic distortion (with NSAED = 1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the first (NSAED = 1) elastic distortions, see eq. (F.53)
3444-3451	Normalize the first (NSAED = 1) elastic distortion and save it in the element SAED(1:9,1), see eq. (F.54)
3449	NSAED = NSAED +1 = 0 + 1 = 1
3453	GOTO 3 →
→ 3420	$k_v = 1 = i_v = 1$, GOTO 2 →
→ 3416	$i_v = 1 \neq$ NDIM, continue
3417	$j_v = 3i_v = 3$
3418	$i_v = i_v + 1 = 2$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 2$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(2) = 1
3425	VRSV(4) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_22), (UED_32)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED > 1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 1 + 1 = 2
3453	GOTO 3 →
→ 3420	$k_v = 1 \neq i_v = 2$, continue
3421	$l_v = 3k_v = 3$
3422	$k_v = k_v + 1 = 2$
3423	VRSV(1:9) = 0
3424	VRSV(5) = 1
3425	VRSV(5) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_23), (UED_33)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED > 1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 2 + 1 = 3
3453	GOTO 3 →
→ 3420	$k_v = 2 = i_v = 2$, GOTO 2 →
→ 3416	$i_v = 2 \neq$ NDIM, continue
3417	$j_v = 3i_v = 6$
3418	$i_v = i_v + 1 = 3$
3419	$k_v = 0$
3420	$k_v = 0 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 0$
3422	$k_v = k_v + 1 = 1$
3423	VRSV(1:9) = 0
3424	VRSV(3) = 1
3425	VRSV(7) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_34)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED > 1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 3 + 1 = 4
3453	GOTO 3 →
→ 3420	$k_v = 1 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 3$
3422	$k_v = k_v + 1 = 2$
3423	VRSV(1:9) = 0
3424	VRSV(6) = 1
3425	VRSV(8) = 1
3427	TMPV(1:9) = VRSV(1:9) → eq.s (UED_35)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED > 1), see eq. (F.51)

3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 4 + 1 = 5
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 2 \neq i_v = 3$, continue
3421	$l_v = 3k_v = 6$
3422	$k_v = k_v + 1 = 3$
3423	VRSV(1:9) = 0
3424	VRSV(9) = 1
3425	VRSV(9) = 1
3427	TMPV(1:9) = VRSV(1:9) $\rightarrow$ eq.s (UED_36)
3428-3432	Symmetrize the NSAED-th elastic distortion (with NSAED >1), see eq. (F.51)
3434-3440	Schmidt orthogonalize the previous 1,...,NSAED elastic distortions, see eq. (F.53)
3444-3451	Normalize the NSAED-th elastic distortion and save it in the element SAED(1:9,NSAED) (with NSAED >1), see eq. (F.54)
3449	NSAED = NSAED + 1 = 5 + 1 = 6
3453	GOTO 3 $\rightarrow$
$\rightarrow$ 3420	$k_v = 3 = i_v = 3$, GOTO 2 $\rightarrow$
$\rightarrow$ 3416	$i_v = 3 =$ NDIM, GOTO 10 $\rightarrow$
$\rightarrow$ 3455	continue
3456	INF(120) = NSAED = 6

Let us make an example for the calculation of symmetry allowed elastic distortions SAED(1:9,1:NSAED) in a simple case. Consider the silicon bulk 3D system with space group P_1 ($\rightarrow$ number of symmetry operators MVF = 1). In this case the rotation matrix to be considered for the calculation is only the identity matrix

$$\mathbf{X}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{in the code:} \quad \text{XYV(1:9,I)} = (1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1) \tag{F.55}$$

since no other symmetry operations are defined with P_1 space group. The symmetrization of the elastic distortions, performed in lines 3428-3432, is therefore skipped in this case. The set of symmetry allowed distortions SAED(1:9,1:NSAED) for the case with P_1 space group (only one symmetry operator MVF = 1) is computed and reported in the following, for the case of standard cell deformations (isotropic, constant volume : INF(195) = 0) and the case of crystallographic deformation (INF(195) = 1).

Case INF(195) = 0 : standard cell deformations (isotropic, constant volume)

lines	operation	
3406-3411	TMPV = (1 0 0 0 1 0 0 0 1)	unit matrix
3412	GOTO 1$\rightarrow$	
$\rightarrow$ 3444-3451	NSAED = 1 SAED(1:9,1) = (0.5773503 0 0 0 0.5773503 0 0 0 0.5773503)	normalization
3413-3432	TMPV(1:9) = VRSV(1:9) = (1 0 0 0 0 0 0 0 0)	unsymmetrized saed
3434-3440	TMPV(1:9) = (0.6666666 0 0 0 $-$0.3333334 0 0 0 $-$0.3333334)	Schmidt orthogonalization
3444-3451	NSAED = 2 SAED(1:9,2) = (0.8164966 0 0 0 $-$0.4082483 0 0 0 $-$0.4082483)	normalization
3453	GOTO 3 $\rightarrow$	
$\rightarrow$ 3420-3432	TMPV(1:9) = VRSV(1:9) = (0 1 0 1 0 0 0 0 0)	unsymmetrized saed
3434-3440	TMPV(1:9) = (0 1 0 1 0 0 0 0 0)	Schmidt orthogonalization
3444-3451	NSAED = 3 SAED(1:9,3) = (0 0.7071068 0 0.7071068 0 0 0 0 0)	normalization
3453	GOTO 3 $\rightarrow$	
$\rightarrow$ 3420-3432	TMPV(1:9) = VRSV(1:9) = (0 0 0 0 1 0 0 0 0)	unsymmetrized saed
3434-3440	TMPV(1:9) = (0.6666666 0 0 0 $-$0.3333334 0 0 0 $-$0.3333334)	Schmidt orthogonalization

Finally, in lines 3484-3512 the matrix SAED(1:9,1:NSAED) is modified in the case optimization with constrained symmetrized cell deformation (3D only) is performed, in which there are NFIXC $\neq 0$ symmetrized cell deformations to be fixed, taken from input (using the keyword FIXDEF, see subroutine CONOPT_, libopt.f library). NFIXC is the number of constraints relating pairs of cell deformations, IFIXC1(i) and IFIXC2(i), i = 1,...,NFIXC are the integer sequence number of the two constrained symmetrized cell deformations, while COEFC(IFIXC1(i)) and COEFC(IFIXC2(i)), i = 1,...,NFIXC are the real coefficients multiplying the two cell deformations in the linear combination constraint, taken from the input file.

F.12 Subroutine SMAT (libxj.f)

Input : logical variable BOLTZ_ACTIVE, that is true if a calculation of electronic transport properties is performed (see boltzatorb.f90 module).

SUBROUTINE SMAT: defined in libxj.f CRYSTAL version: SVN_DEV1286

```
SUBROUTINE SMAT(boltzao_active)
C*** THIS ROUTINE WAS DERIVED FROM BMAT ROUTINE
USE NUMBERS
USE PARINF_MODULE
USE RETIC_MODULE
USE XYVDIM_MODULE
USE POLARI_MODULE
USE EXPO_MODULE
USE PARAL1_MODULE
USE MEMORY_USE
USE MOM_MODULE, ONLY: MOM_INPUT,BTOBTRUE
IMPLICIT REAL(FLOAT) (A-H,O-Z)
LOGICAL :: BOLTZAO_ACTIVE
INTEGER :: NREC_PROC                    (n_x, n_y, n_z) = shrinking factors along (x, y, z)
MVF=INF(2)                              MVF = n = number of symmetry operators
NDF=INF(7)                              NDF = m = number of atomic orbitals
! NOCC=INF(148)                         This line is a comment
INF64=INF(64)                           INF64 = 0 closed shell / = 1 open shell
IS1IS2=IS1*IS2                          IS1IS2 = s_12 = n_x · n_y
IS123=IS1*IS2*IS3                       IS123 = s_123 = n_x · n_y · n_z
NDFJJJ=NDF*NDF                          NDFJJJ = m^2
NDFVRS=NDFJJJ+NDFJJJ                    NDFVRS = 2m^2
NBINI=1                                 NBINI=1
NBFI=NDF                                NBFI = m
MBANDS=NBFI-NBINI+1                     MBANDS = n_b = m
IO10=IUNIT(8)                           IO10 = fort.8 [ units where the eigenvectors related to irreducible k points are read ]
! IO10=IUNIT(10)                        This is a comment [ probably it is the old values for IO10 ]
IO70=IUNIT(70)                          IO70 = fort.70 [ units where the eigenvectors related to reducible k points are written ]
IF(BTOBTRUE)THEN                        Option activated for band to band transition calculations
IO10=IUNIT(153)                         [ BTOBTRUE = false by default ]
ENDIF                                   INF(79) = n_g = number of direct lattice vector
CALL CRYALLOC(EX,2,INF(79),'SMAT','EX') allocation of the (2, n_g) matrix EX [ see Sections F.7 and F.8 ]
IS10=IS1*16                             IS10 = 16 · n_x
IS20=IS2*16                             IS20 = 16 · n_y
IS30=IS3*16                             IS30 = 16 · n_z
NORDER(1:IS1IS2*IS3*2)=0                NORDER(1 : 2 · n_x · n_y · n_z) = 0
KK=0                                    Initialization of KK index [ KK = 0 ]
DO IUNR=1,INF64+1                       BEGIN loop IUNR = 1 closed shell / IUNR = 1,2 open shell
IF(INF64.EQ.0)THEN                      If closed shell → NOCC1 = number of electrons n_e in the reference cell
NOCC1=INF(9)                            INF(9) = number of electrons n_e in the reference cell
ELSE                                    Else if open shell →
IF(IUNR.EQ.1)THEN                       if IUNR = 1 →
NOCC1=INF(9)+INF(95)                    → NOCC1 = n_e + (n_α − n_β) [ INF(95) = (n_α − n_β) ]
ELSE                                    else if IUNR = 2 →
NOCC1=INF(9)-INF(95)                    → NOCC1 = n_e − (n_α − n_β)
ENDIF                                   endif
ENDIF                                   Endif
NOCC=NOCC1/2                            Number of fully occupied bands : NOCC = n_s = NOCC1/2
DO 1010 K=1,NKF                         BEGIN loop over irreducible k points
KK=KK+1                                 Update the index KK → KK + 1 [ index of the k point ]
JR1=JJ(1,K)                             JR1 = k_x (reciprocal space coordinates)
JR2=JJ(2,K)                             JR2 = k_y (reciprocal space coordinates)
JR3=JJ(3,K)                             JR3 = k_z (reciprocal space coordinates)
IF(LATVRS(K).NE.0)GOTO 889              LATVRS(K) = 0 / 1 if the K-th k point is complex / real
C..... COMPLEX K-POINT .....            LATVRS(K) = 0 [ see Section F.6 for LATVRS(K) definition ]
JPROC=0                                 All the processes initialize JPROC = 0
IF(.NOT.JDONE(KK))THEN                  If the process owns the management of the KK-th k point
CALL RREAD(IO10,A,NDFVRS)               that process reads 2m^2 real numbers from fort.8 file
JPROC=IAM                               that process save in JPROC its index = IAM [ for all the other processes JPROC = 0 ]
ENDIF                                   Endif
CALL IGSUM(JPROC,1) ! MPI_ALLREDUCE (MPI_SUM)  Sums JPROC values from all processes and distributes the result back to all processes
CALL BROADCAST(A,NDFVRS,JPROC)          Broadcast data in A(1:2m^2) from process JPROC to all other processes
DO 776 MV=1,MVF                                                    BEGIN loop on the symmetry operators
MR1=MOD(IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10,IS1)  see formula F.57
MR2=MOD(IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20,IS2)  see formula F.58
MR3=MOD(IRR(3,1,MV)*JR1+IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30,IS3)  see formula F.59
NREC=MR1+1+MR2*IS1+MR3*IS1IS2+(IUNR-1)*IS123                       see formula F.60
IF(BOLTZAO_ACTIVE) ENE_INFO(NREC)= K    if BOLTZAO_ACTIVE → map the NREC-th reducible to the K-th irreducible k point
IF(NORDER(NREC).NE.0)GOTO 776           if the NREC-th k point (MR1, MR2, MR3) has already been generated → cycle 776
CALL EXPU(MR1,MR2,MR3)                  modify the matrix EX(1:n_g) [ see Section F.7 ]
```

```
NORDER(NREC)=2                                  → the NREC-th k point will be seen as already generated in the next cycles
CALL ESTROF(A,AR,NINV(MV),NBINI,NBFI,MBANDS)    Rotate eigenvector A using NINV(MV) and save the result in AR [ see Section F.13 ]
IF(BOLTZAO_ACTIVE)THEN                                  If BOLTZAO_ACTIVE = true
IF(MOD(NREC-1,NPROC).EQ.IAM) THEN                       If the process index = mod(nrec - 1, nproc)
NREC_PROC=INT((NREC-1)/NPROC)+1                         select the process record view on the file
WRITE(UNIT=IO70,REC=NREC_PROC)(AR(J),J=1,NDF*NDF*2)     write 2m^2 numbers on file fort.70 in the corresponding record
ENDIF                                                   Endif
ELSE                                                    Else if BOLTZAO_ACTIVE = false
WRITE(UNIT=IO70,REC=NREC)(AR(J),J=1,NDF*NOCC*2)         write on file fort.70 (no parallelism among processes)
ENDIF                                                   Endif
776 CONTINUE                                            END loop on the symmetry operators
! IF(MVF.EQ.INF(1))GOTO 777                             This line is a comment
JR1=MOD(IS1-JR1,IS1)                                    see formula F.61
JR2=MOD(IS2-JR2,IS2)                                    see formula F.62
JR3=MOD(IS3-JR3,IS3)                                    see formula F.63
DO 778 MV=1,MVF                                         BEGIN loop on the symmetry operators
MR1=MOD(IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10,IS1)   see formula F.64
MR2=MOD(IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20,IS2)   see formula F.65
MR3=MOD(IRR(3,1,MV)*JR1+IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30,IS3)   see formula F.66
NREC=MR1+1+MR2*IS1+MR3*IS1IS2+(IUNR-1)*IS123            see formula F.67
IF(BOLTZAO_ACTIVE) ENE_INFO(NREC)=K                     if BOLTZAO_ACTIVE → ENE_INFO(NREC) = K
IF(NORDER(NREC).NE.0)GOTO 778                           if NORDER(NREC) != 0 → go to 778 (cycle)
CALL EXPU(MR1,MR2,MR3)                                  modify the matrix EX(1:n_g) [ see Section F.7 ]
NORDER(NREC)=2                                          NORDER(NREC) = 2
CALL ESTROE(A,AR,NINV(MV),NBINI,NBFI,MBANDS)    Rotate eigenvector A using NINV(MV) and save the result in AR [ see Section F.14 ]
IF(BOLTZAO_ACTIVE)THEN                                  If BOLTZAO_ACTIVE = true
IF(MOD(NREC-1,NPROC).EQ.IAM) THEN                       If the process index = mod(nrec - 1, nproc)
NREC_PROC=INT((NREC-1)/NPROC)+1                         select the processor index
WRITE(UNIT=IO70,REC=NREC_PROC)(AR(J),J=1,NDF*NDF*2)     write 2m^2 numbers on file fort.70 in the corresponding record
ENDIF                                                   Endif
ELSE                                                    Else if BOLTZAO_ACTIVE = false
WRITE(UNIT=IO70,REC=NREC)(AR(J),J=1,NDF*NOCC*2)         write on file fort.70 (no parallelism among processes)
ENDIF                                                   Endif
778 CONTINUE                                            END loop on the symmetry operators
777 GOTO 1010
C..... REAL K-POINT .....                       LATVRS(K) ≠ 0 [ see Section F.6 for LATVRS(K) definition ]
889 CONTINUE
JPROC=0                                                 All the processes initialize JPROC = 0
IF(.NOT.JDONE(KK))THEN                          If the process owns the management of the KK-th k point
CALL RREAD(IO10,A,NDFJJJ)                       that process reads m^2 real numbers from fort.8 file
JPROC=IAM                                       that process save in JPROC its index = IAM [ for all the other processes JPROC = 0 ]
ENDIF                                           Endif
CALL IGSUM(JPROC,1) ! MPI_ALLREDUCE (MPI_SUM)   Sums JPROC values from all processes and distributes the result back to all processes
CALL BROADCAST(A,NDFJJJ,JPROC)                  Broadcast data in A(1:m^2) from process JPROC to all other processes
DO 886 MV=1,MVF                                         BEGIN loop on the symmetry operators
MR1=MOD(IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10,IS1)   see formula F.57
MR2=MOD(IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20,IS2)   see formula F.58
MR3=MOD(IRR(3,1,MV)*JR1+IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30,IS3)   see formula F.59
NREC=MR1+1+MR2*IS1+MR3*IS1IS2+(IUNR-1)*IS123            see formula F.60
IF(BOLTZAO_ACTIVE) ENE_INFO(NREC)=K             if BOLTZAO_ACTIVE → map the NREC-th reducible to the K-th irreducible k point
IF(NORDER(NREC).NE.0)GOTO 886                   if the NREC-th k point (MR1, MR2, MR3) has already been generated → cycle 886
CALL EXPT(MR1,MR2,MR3)                          modify the matrix EX(1:n_g) [ see Section F.8 ]
NORDER(NREC)=1                                  → the NREC-th k point will be seen as already generated in the next cycles
CALL ESTROG(A,AR,NINV(MV),NBINI,NBFI,MBANDS)    Rotate eigenvector A using NINV(MV) and save the result in AR [ see Section F.15 ]
IF(BOLTZAO_ACTIVE)THEN                                  If BOLTZAO_ACTIVE = true
IF(MOD(NREC-1,NPROC).EQ.IAM) THEN                       If the process index = mod(nrec - 1, nproc)
NREC_PROC=INT((NREC-1)/NPROC)+1                         select the process record view on the file
WRITE(UNIT=IO70,REC=NREC_PROC)(AR(J),J=1,NDF*NDF)       write 2m^2 numbers on file fort.70 in the corresponding record
ENDIF                                                   Endif
ELSE                                                    Else if BOLTZAO_ACTIVE = false
WRITE(UNIT=IO70,REC=NREC)(AR(J),J=1,NDF*NOCC)           write on file fort.70 (no parallelism among processes)
ENDIF                                                   Endif
886 CONTINUE                                    END loop on number of symmetry operators
1010 CONTINUE                                   END loop over irreducible k points
ENDDO ! IUNR                                    END loop IUNR
CALL CRYDEALLOC(EX,'SMAT','EX')                 Deallocation of the (2, n_g) matrix EX
REWIND IO10                                     Position the file associated with the specified unit IO10 to its initial point
RETURN
END
```

Note : The Fock/KS eigenvectors are computed at a number of $\boldsymbol{k}$ points in reciprocal space defined by the shrinking factor IS and they are written unformatted in file fort.10 (in the basis of symmetry adapted Bloch functions) and in file fort.8 (in the basis of AO).

$$\mathrm{IRR}(i,j,\mu) \equiv A^{\mu}_{ij} \tag{F.56}$$

with $i = 1,2,3$ and $j = 1,2,3$ and $\mu = 1,...,n$ where n is the number of symmetry operators

$$\mathrm{MR1} \equiv r_1 = \mod\left(k_x\, A^{\mu}_{11} + k_y\, A^{\mu}_{12} + k_z\, A^{\mu}_{13} + 16\, n_x\,,\, n_x\right) \tag{F.57}$$

$$\mathrm{MR2} \equiv r_2 = \mod\left(k_x\, A^{\mu}_{21} + k_y\, A^{\mu}_{22} + k_z\, A^{\mu}_{23} + 16\, n_y\,,\, n_y\right) \tag{F.58}$$

$$\mathrm{MR3} \equiv r_3 = \mod\left(k_x\, A^{\mu}_{31} + k_y\, A^{\mu}_{32} + k_z\, A^{\mu}_{33} + 16\, n_z\,,\, n_z\right) \tag{F.59}$$

$$\mathrm{nrec} = r_1 + 1 + n_x\, r_2 + n_x\, n_y\, r_3 + n_x\, n_y\, n_z\, (\mathrm{IUNR} - 1) \tag{F.60}$$

$$\tilde{k}_x = \mod\left(n_x - k_x\,,\, n_x\right) = n_x - k_x - n_x\, \mathrm{int}\left(\frac{n_x - k_x}{n_x}\right) = n_x - k_x - n_x\, \mathrm{int}\left(1 - \frac{k_x}{n_x}\right) \tag{F.61}$$

$$\tilde{k}_y = \mod\left(n_y - k_y\,,\, n_y\right) = n_y - k_y - n_y\, \mathrm{int}\left(\frac{n_y - k_y}{n_y}\right) = n_y - k_y - n_y\, \mathrm{int}\left(1 - \frac{k_y}{n_y}\right) \tag{F.62}$$

$$\tilde{k}_z = \mod\left(n_z - k_z\,,\, n_z\right) = n_z - k_z - n_z\, \mathrm{int}\left(\frac{n_z - k_z}{n_z}\right) = n_z - k_z - n_z\, \mathrm{int}\left(1 - \frac{k_z}{n_z}\right) \tag{F.63}$$

$$\mathrm{MR1} \equiv \tilde{r}_1 = \mod\left(\tilde{k}_x\, A^{\mu}_{11} + \tilde{k}_y\, A^{\mu}_{12} + \tilde{k}_z\, A^{\mu}_{13} + 16\, n_x\,,\, n_x\right) \tag{F.64}$$

$$\mathrm{MR2} \equiv \tilde{r}_2 = \mod\left(\tilde{k}_x\, A^{\mu}_{21} + \tilde{k}_y\, A^{\mu}_{22} + \tilde{k}_z\, A^{\mu}_{23} + 16\, n_y\,,\, n_y\right) \tag{F.65}$$

$$\mathrm{MR3} \equiv \tilde{r}_3 = \mod\left(\tilde{k}_x\, A^{\mu}_{31} + \tilde{k}_y\, A^{\mu}_{32} + \tilde{k}_z\, A^{\mu}_{33} + 16\, n_z\,,\, n_z\right) \tag{F.66}$$

$$\mathrm{nrec} = \tilde{r}_1 + 1 + n_x\, \tilde{r}_2 + n_x\, n_y\, \tilde{r}_3 + n_x\, n_y\, n_z\, (\mathrm{IUNR} - 1) \tag{F.67}$$

Input file si.d12 (CRYSTAL executable)

```
TEST08 - SILICON BULK: STO-3G
CRYSTAL
0 0 0
227
5.42
1
14 .125 .125 .125
END
14 3
1 0 3 2. 0.
1 1 3 8. 0.
1 1 3 4. 0.
99 0
END
SHRINK
4 4
TOLDEE
7
END
```

Input file si.d3 (PROPERTIES executable)

```
NEWK
4 4
1 0
BOLTZTRA
TRANGE
300 300 100
MURANGE
-5.0 10.0 0.01
TDFRANGE
-5.0 10.0 0.01
END
END
```

Table F.29: Input file si.d12 (left panel) used to obtain the ground state electronic wavefunction and input file si.d3 (right panel) for electronic transport properties calculation used in the example given in the next pages.

The subroutine SMAT modifies the vectors A($1 : n$) and AR($1 : n$) [where $n = 2n_{ao}^2$ in the case of complex $\boldsymbol{k}$ points and $n = n_{ao}^2$ in the case of real $\boldsymbol{k}$ points] that are both global variables declared in polari_module.f90 module, allocated in subroutine allocate_smat_servi and deallocated in subroutine free_smat_servi of the module polari_module.f90. The vectors A($1 : n$) and AR($1 : n$) contain, respectively, the eigenvector associated to an irreducible $\boldsymbol{k}$ point (read from files fort.8 units) and the eigenvector associated to a reducible $\boldsymbol{k}$ point (written on files fort.70 units and generated by using symmetry operators on the eigenvector of an irreducible $\boldsymbol{k}$ point). If BOLTZ_ACTIVE is true, then the vector ENE_INFO will be also modified by this subroutine.

Example 1. Closed shell system (see input files in Table F.29).

Silicon Bulk (space group 227 - lattice parameters $a = b = c = 5.42$ Å)
SHRINK 4 4 → NKF $= n_{\boldsymbol{k}} = 8$ $\boldsymbol{k}$ points in the Irreducible Brillouin zone

- Calculation with $n_{\boldsymbol{k}} = n_p = 8$ processors
  iam = 0 – jdone = T T T T T T T F
  iam = 1 – jdone = T T T T T T F T
  iam = 2 – jdone = T T T T T F T T
  iam = 3 – jdone = T T T T F T T T
  iam = 4 – jdone = T T T F T T T T
  iam = 5 – jdone = T T F T T T T T
  iam = 6 – jdone = T F T T T T T T
  iam = 7 – jdone = F T T T T T T T
  - (Process ↔ $\boldsymbol{k}$ point) association
    iam = jproc = 0 ↔ ik = 8
    iam = jproc = 1 ↔ ik = 7
    iam = jproc = 2 ↔ ik = 6
    iam = jproc = 3 ↔ ik = 5
    iam = jproc = 4 ↔ ik = 4
    iam = jproc = 5 ↔ ik = 3
    iam = jproc = 6 ↔ ik = 2
    iam = jproc = 7 ↔ ik = 1

- Calculation with $n_{\boldsymbol{k}} > n_p = 4$ processors
  iam = 0 – jdone = T T T F T T T F
  iam = 1 – jdone = T T F T T T F T
  iam = 2 – jdone = T F T T T F T T
  iam = 3 – jdone = F T T T F T T T
  - (Process ↔ $\boldsymbol{k}$ point) association
    iam = jproc = 0 ↔ ik = 4,8
    iam = jproc = 1 ↔ ik = 3,7
    iam = jproc = 2 ↔ ik = 2,6
    iam = jproc = 3 ↔ ik = 1,5

- Calculation with $n_{\boldsymbol{k}} < n_p = 14$ processors
  iam = 0 – jdone = T T T T T T T T
  iam = 1 – jdone = T T T T T T T T
  iam = 2 – jdone = T T T T T T T T
  iam = 3 – jdone = T T T T T T T T
  iam = 4 – jdone = T T T T T T T T
  iam = 5 – jdone = T T T T T T T T
  iam = 6 – jdone = T T T T T T T F
  iam = 7 – jdone = T T T T T T F T
  iam = 8 – jdone = T T T T T F T T
  iam = 9 – jdone = T T T T F T T T
  iam = 10 – jdone = T T T F T T T T
  iam = 11 – jdone = T T F T T T T T
  iam = 12 – jdone = T F T T T T T T
  iam = 13 – jdone = F T T T T T T T
  - (Process ↔ $\boldsymbol{k}$ point) association
    jproc = 6 ↔ ik = 8
    jproc = 7 ↔ ik = 7
    jproc = 8 ↔ ik = 6
    jproc = 9 ↔ ik = 5
    jproc = 10 ↔ ik = 4
    jproc = 11 ↔ ik = 3
    jproc = 12 ↔ ik = 2
    jproc = 13 ↔ ik = 1

- Calculation with $n_{\boldsymbol{k}} > n_p = 3$ processors
  iam = 0 – jdone = T T F T T F T T
  iam = 1 – jdone = T F T T F T T F
  iam = 2 – jdone = F T T F T T F T
  - (Process ↔ $\boldsymbol{k}$ point) association
    iam = jproc = 0 ↔ ik = 3,6
    iam = jproc = 1 ↔ ik = 2,5,8
    iam = jproc = 2 ↔ ik = 1,4,7

For the case of open shell systems, the size of the vector jdone is doubled, and the second part of the vector is replicated (so that both the α and β contributions are taken into account), see the Example 2 in the following.

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
1	2	1 0 0	2
1	4	1 1 0	6
1	5	2 1 0	7
1	6	3 1 0	8
1	8	3 2 1	28
2	2	0 1 0	5
2	4	1 1 0	6
2	5	1 2 0	10
2	6	1 3 0	14
2	8	1 2 3	58
3	2	3 3 3	64
3	4	3 3 0	16
3	5	2 2 3	59
3	6	1 1 2	38
3	8	1 2 3	58
4	2	0 0 1	17
4	4	3 3 0	16
4	5	3 3 1	32
4	6	3 3 2	48
4	8	3 2 1	28
5	2	0 1 0	5
5	4	0 1 1	21
5	5	0 2 1	25
5	6	0 3 1	29
5	8	1 3 2	46
6	2	0 0 1	17
6	4	1 0 1	18
6	5	1 0 2	34
6	6	1 0 3	50
6	8	2 1 3	55
7	2	1 0 0	2
7	4	0 3 3	61
7	5	1 3 3	62
7	6	2 3 3	63
7	8	1 3 2	46
8	2	1 0 0	2
8	4	1 0 1	18
8	5	2 0 1	19
8	6	3 0 1	20
8	8	2 3 1	31
9	2	0 0 1	17
9	4	0 1 1	21
9	5	0 1 2	37
9	6	0 1 3	53
9	8	3 1 2	40
10	2	3 3 3	64
10	4	3 0 3	52
10	5	2 3 2	47
10	6	1 2 1	26
10	8	2 3 1	31
11	2	3 3 3	64
11	4	0 3 3	61
11	5	3 2 2	44
11	6	2 1 1	23
11	8	3 1 2	40
12	2	0 1 0	5
12	4	3 0 3	52
12	5	3 1 3	56
12	6	3 2 3	60
12	8	2 1 3	55
13	2	0 3 0	13
13	4	3 3 0	16
13	5	3 2 0	12
13	6	3 1 0	8
13	8	2 1 3	55
14	2	3 0 0	4
14	4	3 3 0	16
14	5	2 3 0	15
14	6	1 3 0	14
14	8	2 3 1	31
15	2	1 1 1	22
15	4	1 1 0	6
15	5	2 2 1	27
15	6	3 3 2	48
15	8	2 3 1	31
16	2	0 0 3	49
16	4	1 1 0	6
16	5	1 1 3	54
16	6	1 1 2	38
16	8	2 1 3	55
17	2	3 0 0	4
17	4	3 0 3	52
17	5	2 0 3	51
17	6	1 0 3	50
17	8	1 3 2	46
18	2	0 0 3	49
18	4	0 3 3	61
18	5	0 3 2	45
18	6	0 3 1	29
18	8	3 2 1	28
19	2	0 3 0	13
19	4	1 0 1	18
19	5	1 3 1	30
19	6	1 2 1	26
19	8	1 3 2	46
20	2	0 3 0	13
20	4	0 3 3	61
20	5	0 2 3	57
20	6	0 1 3	53
20	8	1 2 3	58
21	2	0 0 3	49
21	4	3 0 3	52
21	5	3 0 2	36
21	6	3 0 1	20
21	8	3 1 2	40
22	2	1 1 1	22
22	4	0 1 1	21
22	5	1 2 2	42
22	6	2 3 3	63
22	8	1 2 3	58
23	2	1 1 1	22
23	4	1 0 1	18
23	5	2 1 2	39
23	6	3 2 3	60
23	8	3 1 2	40
24	2	3 0 0	4
24	4	0 1 1	21
24	5	3 1 1	24
24	6	2 1 1	23
24	8	3 2 1	28
25	2	3 0 0	4
25	4	3 3 0	16
25	5	2 3 0	15
25	6	1 3 0	14
25	8	1 2 3	58
26	2	0 3 0	13
26	4	3 3 0	16
26	5	3 2 0	12
26	6	3 1 0	8
26	8	3 2 1	28
27	2	1 1 1	22
27	4	1 1 0	6
27	5	2 2 1	27
27	6	3 3 2	48
27	8	3 2 1	28

Table F.30: Indexes related to coordinates of the complex reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red} \in \mathcal{C}$ created from the $(i_{irr,\boldsymbol{k}})$-th irreducible ones through the (i_{sym})-th symmetry operators and integer nrec, which gives the record number to be written in the direct access file fort.70. The indexes of $\boldsymbol{k}_{red}$ divided by the shrink integers $n_x = n_y = n_z$ along the correspondent direction gives the Cartesian coordinates of the reciprocal space point.

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
28	2	0 0 3	49
28	4	1 1 0	6
28	5	1 1 3	54
28	6	1 1 2	38
28	8	1 2 3	58
29	2	0 3 0	13
29	4	0 3 3	61
29	5	0 2 3	57
29	6	0 1 3	53
29	8	3 1 2	40
30	2	0 0 3	49
30	4	3 0 3	52
30	5	3 0 2	36
30	6	3 0 1	20
30	8	2 3 1	31
31	2	3 0 0	4
31	4	0 1 1	21
31	5	3 1 1	24
31	6	2 1 1	23
31	8	3 1 2	40
32	2	3 0 0	4
32	4	3 0 3	52
32	5	2 0 3	51
32	6	1 0 3	50
32	8	2 1 3	55
33	2	0 0 3	49
33	4	0 3 3	61
33	5	0 3 2	45
33	6	0 3 1	29
33	8	1 3 2	46
34	2	1 1 1	22
34	4	1 0 1	18
34	5	2 1 2	39
34	6	3 2 3	60
34	8	2 1 3	55
35	2	1 1 1	22
35	4	0 1 1	21
35	5	1 2 2	42
35	6	2 3 3	63
35	8	1 3 2	46
36	2	0 3 0	13
36	4	1 0 1	18
36	5	1 3 1	30
36	6	1 2 1	26
36	8	2 3 1	31

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
37	2	0 1 0	5
37	4	1 1 0	6
37	5	1 2 0	10
37	6	1 3 0	14
37	8	2 3 1	31
38	2	1 0 0	2
38	4	1 1 0	6
38	5	2 1 0	7
38	6	3 1 0	8
38	8	2 1 3	55
39	2	3 3 3	64
39	4	3 3 0	16
39	5	2 2 3	59
39	6	1 1 2	38
39	8	2 1 3	55
40	2	0 0 1	17
40	4	3 3 0	16
40	5	3 3 1	32
40	6	3 3 2	48
40	8	2 3 1	31
41	2	1 0 0	2
41	4	1 0 1	18
41	5	2 0 1	19
41	6	3 0 1	20
41	8	3 1 2	40
42	2	0 0 1	17
42	4	0 1 1	21
42	5	0 1 2	37
42	6	0 1 3	53
42	8	1 2 3	58
43	2	0 1 0	5
43	4	3 0 3	52
43	5	3 1 3	56
43	6	3 2 3	60
43	8	3 1 2	40
44	2	0 1 0	5
44	4	0 1 1	21
44	5	0 2 1	25
44	6	0 3 1	29
44	8	3 2 1	28
45	2	0 0 1	17
45	4	1 0 1	18
45	5	1 0 2	34
45	6	1 0 3	50
45	8	1 3 2	46

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
46	2	3 3 3	64
46	4	0 3 3	61
46	5	3 2 2	44
46	6	2 1 1	23
46	8	3 2 1	28
47	2	3 3 3	64
47	4	3 0 3	52
47	5	2 3 2	47
47	6	1 2 1	26
47	8	1 3 2	46
48	2	1 0 0	2
48	4	0 3 3	61
48	5	1 3 3	62
48	6	2 3 3	63
48	8	1 2 3	58

Table F.31: (continuation of Table F.30) Indexes related to coordinates of the complex reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red} \in \mathcal{C}$ created from the $(i_{irr,\boldsymbol{k}})$-th irreducible ones through the (i_{sym})-th symmetry operators and integer nrec, which gives the record number to be written in the direct access file fort.70. The indexes of $\boldsymbol{k}_{red}$ divided by the shrink integers $n_x = n_y = n_z$ along the correspondent direction gives the Cartesian coordinates of the reciprocal space point.

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
1	1	0 0 0	1
1	3	2 0 0	3
1	7	2 2 0	11
2	1	0 0 0	1
2	3	0 2 0	9
2	7	2 2 0	11
3	1	0 0 0	1
3	3	2 2 2	43
3	7	2 2 0	11
4	1	0 0 0	1
4	3	0 0 2	33
4	7	2 2 0	11
5	1	0 0 0	1
5	3	0 2 0	9
5	7	0 2 2	41
6	1	0 0 0	1
6	3	0 0 2	33
6	7	2 0 2	35
7	1	0 0 0	1
7	3	2 0 0	3
7	7	0 2 2	41
8	1	0 0 0	1
8	3	2 0 0	3
8	7	2 0 2	35
9	1	0 0 0	1
9	3	0 0 2	33
9	7	0 2 2	41
10	1	0 0 0	1
10	3	2 2 2	43
10	7	2 0 2	35
11	1	0 0 0	1
11	3	2 2 2	43
11	7	0 2 2	41
12	1	0 0 0	1
12	3	0 2 0	9
12	7	2 0 2	35
13	1	0 0 0	1
13	3	0 2 0	9
13	7	2 2 0	11
14	1	0 0 0	1
14	3	2 0 0	3
14	7	2 2 0	11
15	1	0 0 0	1
15	3	2 2 2	43
15	7	2 2 0	11
16	1	0 0 0	1
16	3	0 0 2	33
16	7	2 2 0	11
17	1	0 0 0	1
17	3	2 0 0	3
17	7	2 0 2	35
18	1	0 0 0	1
18	3	0 0 2	33
18	7	0 2 2	41
19	1	0 0 0	1
19	3	0 2 0	9
19	7	2 0 2	35
20	1	0 0 0	1
20	3	0 2 0	9
20	7	0 2 2	41
21	1	0 0 0	1
21	3	0 0 2	33
21	7	2 0 2	35
22	1	0 0 0	1
22	3	2 2 2	43
22	7	0 2 2	41
23	1	0 0 0	1
23	3	2 2 2	43
23	7	2 0 2	35
24	1	0 0 0	1
24	3	2 0 0	3
24	7	0 2 2	41
25	1	0 0 0	1
25	3	2 0 0	3
25	7	2 2 0	11
26	1	0 0 0	1
26	3	0 2 0	9
26	7	2 2 0	11
27	1	0 0 0	1
27	3	2 2 2	43
27	7	2 2 0	11
28	1	0 0 0	1
28	3	0 0 2	33
28	7	2 2 0	11
29	1	0 0 0	1
29	3	0 2 0	9
29	7	0 2 2	41
30	1	0 0 0	1
30	3	0 0 2	33
30	7	2 0 2	35
31	1	0 0 0	1
31	3	2 0 0	3
31	7	0 2 2	41
32	1	0 0 0	1
32	3	2 0 0	3
32	7	2 0 2	35
33	1	0 0 0	1
33	3	0 0 2	33
33	7	0 2 2	41
34	1	0 0 0	1
34	3	2 2 2	43
34	7	2 0 2	35
35	1	0 0 0	1
35	3	2 2 2	43
35	7	0 2 2	41
36	1	0 0 0	1
36	3	0 2 0	9
36	7	2 0 2	35
37	1	0 0 0	1
37	3	0 2 0	9
37	7	2 2 0	11
38	1	0 0 0	1
38	3	2 0 0	3
38	7	2 2 0	11
39	1	0 0 0	1
39	3	2 2 2	43
39	7	2 2 0	11
40	1	0 0 0	1
40	3	0 0 2	33
40	7	2 2 0	11
41	1	0 0 0	1
41	3	2 0 0	3
41	7	2 0 2	35
42	1	0 0 0	1
42	3	0 0 2	33
42	7	0 2 2	41

Table F.32: Indexes related to coordinates of the real reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red} \in \mathcal{R}$ created from the $(i_{irr,\boldsymbol{k}})$-th irreducible ones through the (i_{sym})-th symmetry operators and integer nrec, which gives the record number to be written in the direct access file fort.70. The indexes of $\boldsymbol{k}_{red}$ divided by the shrink integers $n_x = n_y = n_z$ along the correspondent direction gives the Cartesian coordinates of the reciprocal space point.

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
43	1	0 0 0	1
43	3	0 2 0	9
43	7	2 0 2	35
44	1	0 0 0	1
44	3	0 2 0	9
44	7	0 2 2	41
45	1	0 0 0	1
45	3	0 0 2	33
45	7	2 0 2	35

i_{sym}	$i_{irr,\boldsymbol{k}}$	$\boldsymbol{k}_{red}$	nrec
46	1	0 0 0	1
46	3	2 2 2	43
46	7	0 2 2	41
47	1	0 0 0	1
47	3	2 2 2	43
47	7	2 0 2	35
48	1	0 0 0	1
48	3	2 0 0	3
48	7	0 2 2	41

Table F.33: (continuation of Table F.32) Indexes related to coordinates of the real reducible $\boldsymbol{k}$ points $\boldsymbol{k}_{red} \in \mathcal{R}$ created from the $(i_{irr,\boldsymbol{k}})$-th irreducible ones through the (i_{sym})-th symmetry operators and integer nrec, which gives the record number to be written in the direct access file fort.70. The indexes of $\boldsymbol{k}_{red}$ divided by the shrink integers $n_x = n_y = n_z$ along the correspondent direction gives the Cartesian coordinates of the reciprocal space point.

Example 2. Open shell system.

$KMnF_3$ Bulk (space group 221 - lattice parameters $a = b = c = 4.19$ Å)
SHRINK 4 4 $\rightarrow$ NKF $= n_{\boldsymbol{k}} = 10$ $\boldsymbol{k}$ points in the Irreducible Brillouin zone

- Calculation with $n_{\boldsymbol{k}} > n_p = 8$ processors
  iam = 0 – jdone = T T T T T T T F T T | T T T T T F T T T T
  iam = 1 – jdone = T T T T T T F T T T | T T T T F T T T T T
  iam = 2 – jdone = T T T T T F T T T T | T T T F T T T T T T
  iam = 3 – jdone = T T T T F T T T T T | T T F T T T T T T T
  iam = 4 – jdone = T T T F T T T T T T | T F T T T T T T T F
  iam = 5 – jdone = T T F T T T T T T T | F T T T T T T T F T
  iam = 6 – jdone = T F T T T T T T T F | T T T T T T T F T T
  iam = 7 – jdone = F T T T T T T T F T | T T T T T T F T T T

  – (Process $\leftrightarrow$ $\boldsymbol{k}$ point) association
    iam = jproc = 0 $\leftrightarrow$ ik = 6β,8α
    iam = jproc = 1 $\leftrightarrow$ ik = 5β,7α
    iam = jproc = 2 $\leftrightarrow$ ik = 4β,6α
    iam = jproc = 3 $\leftrightarrow$ ik = 3β,5α
    iam = jproc = 4 $\leftrightarrow$ ik = 2β,4α,10β
    iam = jproc = 5 $\leftrightarrow$ ik = 1β,3α,9β
    iam = jproc = 6 $\leftrightarrow$ ik = 2α,8β,10α
    iam = jproc = 7 $\leftrightarrow$ ik = 1α,7β,9α

Therefore, in the case of open shell systems, the number of irreducible $\boldsymbol{k}$ points is basically doubled in the jdone vector, so that the α and β contributions are both taken into account. In the previous example there are 10 irreducible $\boldsymbol{k}$ points, but they are doubled so that in the vector jdone there are 20 elements, the first 10 and the last 10 values related to the α and the β components, respectively. In this way, the 20 elements (10 α-$\boldsymbol{k}$ and 10 β-$\boldsymbol{k}$ points) are divided between processors using the methodology illustrated in the example.

F.13 Subroutine ESTROF (both4.f)

Input : the real variables vectors A($1 : 2n_{ao}^2$) and AR($1 : 2n_{ao}^2$) where n_{ao} is the number of atomic orbitals in the system, the integer MV that contains the index of the symmetry operator to be applied on the input vector A, the integer NBINI that contains the index of the first band to be considered, the integer NBFI that contains the index of the last band to be considered and the integer MBANDS that contains the number of bands to be considered.
(in SMAT subroutine, this last input integer is defined as MBANDS = NBFI - NBINI + 1)
The subroutine modifies the elements of the vector AR in the range ($1 :$ MBANDS $\cdot\, n_{ao}$), i.e. it modifies AR($1 :$ MBANDS $\cdot\, n_{ao}$) by rotating these elements of the vector according to the symmetry operator of the space group classified with index MV as given in the third input argument of the subroutine.

line	**SUBROUTINE ESTROF: defined in both4.f**	CRYSTAL version: SVN_DEV1286
6206	SUBROUTINE ESTROF(A,AR,MV,NBINI,NBFI,MBANDS)	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE MEMORY_SCREEN	
	USE ROTMATRIX	
	USE BASATO_MODULE	
	USE EXPO_MODULE	MV = i_s (= index of the symmetry operator) as input argument
	USE SOC	NBINI = $i_{beg,b}$ = initial band index as input argument
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	NBFI = $i_{end,b}$ = final band index as input argument
	DIMENSION A(*),AR(*)	MBANDS = m_b = total number of bands (states) as input argument
	NDF=INF(7)	INF(7) = n_{ao} = number of Atomic Orbitals (AOs)
	INDBAS=(NBINI-1)*NDF	INDBAS = initial index for basis set selection = $(i_{beg,b} - 1)\, n_{ao}$
	NDF=NDF+NDF	NDF = $2n_{ao}$
	IF(LSOC) INDBAS=(NBINI-1)*NDF	if LSOC $\rightarrow$ INDBAS = $2\,(i_{beg,b} - 1)\, n_{ao}$
	MVF=INF(2)	MVF = INF(2) = n_s = number of symmetry operators
	AR(1:MBANDS*NDF)=0._FLOAT	AR(1:MBANDS·NDF) = $\lambda_i = 0$ with $i = 1, ..., m_b \cdot n_{ao}$
	DO LAVRS=1,INF(24)	BEGIN loop on the number of atoms n_{at} per unit cell
	I=MGNAV(LAVRS,MV)	index of the associated $\boldsymbol{g}$ vector
	ECO=EX(1,I)	cosin factor cos(α) of the I-th $\boldsymbol{g}$ vector for a given $\boldsymbol{k}$ point [see Sections F.7 - F.8]
	ESI=EX(2,I)	sin factor sin(α) of the I-th $\boldsymbol{g}$ vector for a given $\boldsymbol{k}$ point [see Sections F.7 - F.8]
	DO LA=NSHPRI(LAVRS),NSHPRI(LAVRS+1)-1	BEGIN loop over shells associated to the LAVRS-th atom
	ICA=NDQ(LAV(LA,MV))+INDBAS	ICA = cumulative number of AOs + INDBAS (CRYSTAL list order)
	INF3=LAT(LA)*MVF+MV	INF3 = LAT(LA) $\cdot\, n_s + i_s$
	ICO=NDQ(LA)	ICO = cumulative number of AOs (CRYSTAL list order)
	DO I=MINZ(INF3)+1,MINZ(INF3+1)	BEGIN loop on point group operators
	VSI=TTO(I)	TTO(I) = T_{ij} vectorized
	VCO=ECO*VSI	VCO = $T_{ij} \cos(\boldsymbol{k} \cdot \boldsymbol{g}) \equiv T_{ij} \cos(\alpha)$
	VSI=ESI*VSI	VSI = $T_{ij} \sin(\boldsymbol{k} \cdot \boldsymbol{g}) \equiv T_{ij} \sin(\alpha)$
	ICA1=(MMO(I)+ICA)*2	ICA1 = pointer to the index of the element of A vector to be used in the operation
	ICA2=(MMOM(I)+ICO)*2	ICA2 = pointer to the index of the element of AR vector to be used in the operation
	DO IND=NBINI,NBFI	BEGIN loop on selected band indexes (IND = $i_{beg,b}$, ..., $i_{end,b}$)
	A1=A(ICA1-1)	Real part of the eigenvector $\Re(\mathbf{C}_{irr})$ associated to the irreducible $\boldsymbol{k}$ point
	A2=A(ICA1)	Imaginary part of the eigenvector $\Im(\mathbf{C}_{irr})$ associated to the irreducible $\boldsymbol{k}$ point
6240	AR(ICA2-1)=A1*VCO-A2*VSI+AR(ICA2-1)	AR(ICA2-1) = AR(ICA2-1) + A1 $\cdot\, T_{ij} \cos(\alpha)$ - A2 $\cdot\, T_{ij} \sin(\alpha)$ [see eq. (F.73)]
6241	AR(ICA2)=A1*VSI+A2*VCO+AR(ICA2)	AR(ICA2) = AR(ICA2) + A1 $\cdot\, T_{ij} \sin(\alpha)$ + A2 $\cdot\, T_{ij} \cos(\alpha)$ [see eq. (F.74)]
	ICA1=ICA1+NDF	Increment the index ICA1 by $2n_{ao}$
	ICA2=ICA2+NDF	Increment the index ICA2 by $2n_{ao}$
	ENDDO	END loop on selected band indexes
	ENDDO	END loop on point group operators
	ENDDO	END loop over shells associated to the LAVRS-th atom
	ENDDO	END loop on number of atoms n_{at}
	RETURN	
6249	END	

Table F.35: Code and description of subroutine ESTROF (both4.f)

In the following, a description of the vectors and matrices appearing in the subroutine is given.
NSHPRI(I), I = 1, ..., (n_{at} + 1) is a vector containing the cumulative number of shells (input order), so that

$$\text{NSHPRI}(1) = 1 \qquad \text{and} \qquad \text{NSHPRI(J)} = \sum_{\text{I=1}}^{\text{J>1}} \text{NSHPRI(I)} \tag{F.68}$$

LAT(I), I = 1, ..., n_h (where n_h is the number of shells) is the type of the I-th shell in input order, where

the value of the elements depends on the type of shell and it can be 0 for shell type S, 1 for shell type SP, 2 for shell type P, 3 for shell type D and 4 for shell type F ($0 \rightarrow$ S; $1 \rightarrow$ SP, $2 \rightarrow$ P; $3 \rightarrow$ D; $4 \rightarrow$ F).

NDQ(I), I = 1, ..., $(n_h + 1)$ [where n_h is the number of shells] is the cumulative number of atomic orbitals (in input order), so that

$$\text{NDQ}(1) = 0 \qquad \text{and} \qquad \text{NDQ(J)} = \sum_{\text{I}=1}^{\text{J}>1} \text{NDQ(I)} \tag{F.69}$$

MGNAV is a $n_{at} \times n_s$ matrix (number of atoms, number of point symmetry operators) that shows which lattice vector brings back the μ-th atom in the zero-cell when the $\hat{R}^{-1}$ operator is applied to it. An example of the MGNAV matrix is given in Figure F.3.

NAF	MVF											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	6	2	6	1	1	6	2	1	2	6
2	1	2	2	4	1	4	4	1	2	1	4	2

Figure F.3: MGNAV matrix for graphite, with NAF = 2 atoms in the unit cell and MVF = 12 operators.

LAV is a $n_h \times n_s$ matrix (number of shells, number of point symmetry operators) that shows on which shell goes a shell by the effect of a symmetry operator $\hat{R}^{-1}$ (shells equivalent by translation do have the same label; then for example in graphite shells labels are numbers from 1 to 4).
An example of the LAV matrix is given in Figure F.4.

LAF	MVF											
	1	2	3	4	5	6	7	8	9	10	11	12
1	1	3	1	1	3	3	1	1	1	3	3	3
2	2	4	2	2	4	4	2	2	2	4	4	4
3	3	1	3	3	1	1	3	3	3	1	1	1
4	4	2	4	4	2	2	4	4	4	2	2	2

Figure F.4: LAV matrix for the graphite example with LAF = 4 shells and MVF = 12 operators.

The vectors TTO, MMO, MMOM and MINZ are initialized in subroutine GSYM11, that defines the symmetry operator matrices for all shell types and all operators, such that the total number of matrices is given by the total number of symmetry operators MVF = inf(2) times the total number of shell types (MXTP + 1, where MXTP = inf(175) is the label of the shell type with the maximum number of atomic orbitals). These matrices are stored in compact form in the TTO vector (null terms are disregarded). Three pointers, MMO, MMOM and MINZ are defined: they point, respectively, the row and the column indices of non-null elements and the starting point of the point group operators. MMO and MMOM have the same size as TTO and give for each TTO element the corresponding row and column indices in the original representation matrices, respectively. MINZ is a MVF $\times$ (MXTP + 1) array and gives the starting point for each operator and for each shell type in the TTO global array.

The two operations in lines 6240-6241 can be understood thinking that the eigenvectors $\mathbf{C}$, that is given in input by the vector A$(1: m_b \cdot n_{ao})$, contains both the real and imaginary part. If the $(i-1)$-th element of the eigenvector A is an element contained in the real part of the eigenvector, the next one i.e. the i-th element belongs to the imaginary part of the eigenvector A. Therefore, in line 6240 a real element of the rotated eigenvector AR$(1: m_b \cdot n_{ao})$ is calculated, while in the next line 6241 an element of its imaginary part is computed, so that at the end both the real and the imaginary part of the eigenvector A$(1: m_b \cdot n_{ao})$ will be rotated according the the symmetry operator with index MV $= i_s$ and stored in the vector AR$(1: m_b \cdot n_{ao})$. In order to rotate the eigenvector $\mathbf{C}$, it has to be multiplied by the matrix representing the symmetry operator $\mathbf{T} = \{T_{ij}\}$ and by a phase $\exp(i\boldsymbol{k} \cdot \boldsymbol{g})$ where $\boldsymbol{k}$ is the reciprocal space

point and $\boldsymbol{g}$ is the direct lattice vector. The phase factor can be also written as

$$e^{i\boldsymbol{k}\cdot\boldsymbol{g}} = \cos(\boldsymbol{k}\cdot\boldsymbol{g}) + i\sin(\boldsymbol{k}\cdot\boldsymbol{g}) \tag{F.70}$$

Moreover, considering the product of two complex numbers $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$, given by

$$z_1\, z_2 = (x_1 + iy_1)(x_2 + iy_2) = x_1\, x_2 - y_1\, y_2 + i(x_1\, y_2 + x_2\, y_1) \tag{F.71}$$

it is easy to demonstrate that the real part $\Re(z_1 z_2)$ and the imaginary part $\Im(z_1 z_2)$ of the product can be written as

$$\Re(z_1 z_2) = \Re(z_1)\Re(z_2) - \Im(z_1)\Im(z_2) \qquad \text{and} \qquad \Im(z_1 z_2) = \Re(z_1)\Im(z_2) + \Re(z_2)\Im(z_1) \tag{F.72}$$

so that the real part and the imaginary part of the rotated eigenvector $\mathbf{C}\,\mathbf{T}\exp(i\boldsymbol{k}\cdot\boldsymbol{g})$ that is stored in the vector $\mathrm{AR}(1:\, m_b \cdot n_{ao})$ can be computed as

$$\Re(\mathbf{C}\,\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\Re(\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) - \Im(\mathbf{C})\,\Im(\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\mathbf{T}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) - \Im(\mathbf{C})\,\mathbf{T}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) \tag{F.73}$$

$$\Im(\mathbf{C}\,\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\Im(\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) + \Im(\mathbf{C})\,\Re(\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\mathbf{T}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) + \Im(\mathbf{C})\,\mathbf{T}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) \tag{F.74}$$

where the matrix $\mathbf{T}$ is a real matrix and can be therefore associated to the phase part by applying the formulae (F.72). Therefore, the following identities can be defined between the notation adopted in the previous equations and the operations in the code in lines 6240-6241

$$[\Re(\mathbf{C})]_{ij} = \mathrm{A1} = \mathrm{A(ICA1 - 1)} \qquad \text{and} \qquad [\Im(\mathbf{C})]_{ij} = \mathrm{A2} = \mathrm{A(ICA1)} \tag{F.75}$$

$$[\Re(\mathbf{C}\,\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}})]_{kl} = \mathrm{AR(ICA2 - 1)} \qquad \text{and} \qquad [\Im(\mathbf{C}\,\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}})]_{kl} = \mathrm{AR(ICA2)} \tag{F.76}$$

$$T_{mn}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) = \mathrm{VCO} \qquad \text{and} \qquad T_{mn}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) = \mathrm{VSI} \tag{F.77}$$

F.14 Subroutine ESTROE (both4.f)

Input : the real variables vectors A($1 : 2n_{ao}^2$) and AR($1 : 2n_{ao}^2$) where n_{ao} is the number of atomic orbitals in the system, the integer MV that contains the index of the symmetry operator to be applied on the input vector A, the integer NBINI that contains the index of the first band to be considered, the integer NBFI that contains the index of the last band to be considered and the integer MBANDS that contains the number of bands to be considered.
(in SMAT subroutine, this last input integer is defined as MBANDS = NBFI - NBINI + 1)
The subroutine modifies the elements of the vector AR in the range ($1 :$ MBANDS $\cdot\, n_{ao}$), i.e. it modifies AR($1 :$ MBANDS $\cdot\, n_{ao}$) by rotating these elements of the vector according to the symmetry operator of the space group classified with index MV as given in the third input argument of the subroutine.

line	**SUBROUTINE ESTROE: defined in both4.f**	CRYSTAL version: SVN_DEV1286
6158	SUBROUTINE ESTROE(A,AR,MV,NBINI,NBFI,MBANDS)	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE MEMORY_SCREEN	
	USE ROTMATRIX	
	USE BASATO_MODULE	
	USE EXPO_MODULE	MV = i_s (= index of the symmetry operator) as input argument
	USE SOC	NBINI = $i_{beg,b}$ = initial band index as input argument
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	NBFI = $i_{end,b}$ = final band index as input argument
	DIMENSION A(*),AR(*)	MBANDS = m_b = total number of bands (states) as input argument
	NDF=INF(7)	INF(7) = n_{ao} = number of Atomic Orbitals (AOs)
	INDBAS=(NBINI-1)*NDF	INDBAS = initial index for basis set selection = $(i_{beg,b} - 1)\, n_{ao}$
	NDF=NDF+NDF	NDF = $2n_{ao}$
	IF(LSOC) INDBAS=(NBINI-1)*NDF	if LSOC $\rightarrow$ INDBAS = $2\,(i_{beg,b} - 1)\, n_{ao}$
	MVF=INF(2)	MVF = INF(2) = n_s = number of symmetry operators
	AR(1:MBANDS*NDF)=0._FLOAT	AR(1:MBANDS·NDF) = $\lambda_i = 0$ with $i = 1, ..., m_b \cdot n_{ao}$
	DO LAVRS=1,INF(24)	BEGIN loop on the number of atoms n_{at} per unit cell
	I=MGNAV(LAVRS,MV)	index of the associated $\boldsymbol{g}$ vector
	ECO=EX(1,I)	cosin factor cos(α) of the I-th $\boldsymbol{g}$ vector for a given $\boldsymbol{k}$ point [see Sections F.7 - F.8]
	ESI=EX(2,I)	sin factor sin(α) of the I-th $\boldsymbol{g}$ vector for a given $\boldsymbol{k}$ point [see Sections F.7 - F.8]
	DO LA=NSHPRI(LAVRS),NSHPRI(LAVRS+1)-1	BEGIN loop over shells associated to the LAVRS-th atom
	ICA=NDQ(LAV(LA,MV))+INDBAS	ICA = cumulative number of AOs + INDBAS (CRYSTAL list order)
	INF3=LAT(LA)*MVF+MV	INF3 = LAT(LA) $\cdot\, n_s + i_s$
	ICO=NDQ(LA)	ICO = cumulative number of AOs (CRYSTAL list order)
	DO I=MINZ(INF3)+1,MINZ(INF3+1)	BEGIN loop on point group operators
	VSI=TTO(I)	TTO(I) = T_{ij} vectorized
	VCO=ECO*VSI	VCO = $T_{ij} \cos(\boldsymbol{k} \cdot \boldsymbol{g}) \equiv T_{ij} \cos(\alpha)$
	VSI=ESI*VSI	VSI = $T_{ij} \sin(\boldsymbol{k} \cdot \boldsymbol{g}) \equiv T_{ij} \sin(\alpha)$
	ICA1=(MMO(I)+ICA)*2	ICA1 = pointer to the index of the element of A vector to be used in the operation
	ICA2=(MMOM(I)+ICO)*2	ICA2 = pointer to the index of the element of AR vector to be used in the operation
	DO IND=NBINI,NBFI	BEGIN loop on selected band indexes (IND = $i_{beg,b}$, ..., $i_{end,b}$)
	A1=A(ICA1-1)	Real part of the eigenvector $\Re(\mathbf{C}_{irr})$ associated to the irreducible $\boldsymbol{k}$ point
	A2=A(ICA1)	Imaginary part of the eigenvector $\Im(\mathbf{C}_{irr})$ associated to the irreducible $\boldsymbol{k}$ point
6192	AR(ICA2-1)=A1*VCO+A2*VSI+AR(ICA2-1)	AR(ICA2-1) = AR(ICA2-1) + A1 $\cdot\, T_{ij} \cos(\alpha)$ + A2 $\cdot\, T_{ij} \sin(\alpha)$ [see eq. (F.80)]
6193	AR(ICA2)=A1*VSI-A2*VCO+AR(ICA2)	AR(ICA2) = AR(ICA2) + A1 $\cdot\, T_{ij} \sin(\alpha)$ - A2 $\cdot\, T_{ij} \cos(\alpha)$ [see eq. (F.81)]
	ICA1=ICA1+NDF	Increment the index ICA1 by $2n_{ao}$
	ICA2=ICA2+NDF	Increment the index ICA2 by $2n_{ao}$
	ENDDO	END loop on selected band indexes
	ENDDO	END loop on point group operators
	ENDDO	END loop over shells associated to the LAVRS-th atom
	ENDDO	END loop on number of atoms n_{at}
	RETURN	
6201	END	

Table F.37: Code and description of subroutine ESTROE (both4.f)

In order to fully understand the operation involved in the subroutine, see the comments on subroutine ESTROF in Section F.13, where the meaning of the vectors and matrices also appearing in the subroutine ESTROE is fully explained.
The difference between subroutine ESTROF (Section F.13) and the subroutine ESTROE is that, in the last one, the product of a complex number z_1 by the complex conjugate of another complex number z_2^* is considered, so that, instead of the product (F.72), the following product has to be taken into account

$$z_1\, z_2^* = (x_1 + iy_1)(x_2 - iy_2) = x_1\, x_2 + y_1\, y_2 + i(x_2\, y_1 - x_1\, y_2) \tag{F.78}$$

Therefore, it is easy to demonstrate that the real part $\Re(z_1 z_2^*)$ and the imaginary part $\Im(z_1 z_2^*)$ of the product can be written as

$$\Re(z_1 z_2^*) = \Re(z_1)\Re(z_2) + \Im(z_1)\Im(z_2) \qquad \text{and} \qquad \Im(z_1 z_2^*) = \Re(z_2)\Im(z_1) - \Re(z_1)\Im(z_2) \tag{F.79}$$

so that the real part and the imaginary part of the rotated eigenvector $\mathbf{C}\,\mathbf{T}\exp(-i\boldsymbol{k}\cdot\boldsymbol{g})$ that is stored in the vector $\mathrm{AR}(1:\, m_b \cdot n_{ao})$ can be computed as

$$\Re(\mathbf{C}\,\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\Re(\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) + \Im(\mathbf{C})\,\Im(\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\mathbf{T}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) + \Im(\mathbf{C})\,\mathbf{T}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) \tag{F.80}$$

$$\Im(\mathbf{C}\,\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\Im(\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) - \Im(\mathbf{C})\,\Re(\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}}) = \Re(\mathbf{C})\,\mathbf{T}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) - \Im(\mathbf{C})\,\mathbf{T}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) \tag{F.81}$$

where the matrix $\mathbf{T}$ is a real matrix and can be therefore associated to the phase part by applying the formulae (F.72). Therefore, the following identities can be defined between the notation adopted in the previous equations and the operations in the code in lines 6192-6193

$$[\Re(\mathbf{C})]_{ij} = \mathrm{A1} = \mathrm{A(ICA1 - 1)} \qquad \text{and} \qquad [\Im(\mathbf{C})]_{ij} = \mathrm{A2} = \mathrm{A(ICA1)} \tag{F.82}$$

$$[\Re(\mathbf{C}\,\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}})]_{kl} = \mathrm{AR(ICA2 - 1)} \qquad \text{and} \qquad [\Im(\mathbf{C}\,\mathbf{T}\,e^{-i\,\boldsymbol{k}\cdot\boldsymbol{g}})]_{kl} = \mathrm{AR(ICA2)} \tag{F.83}$$

$$T_{mn}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) = \mathrm{VCO} \qquad \text{and} \qquad T_{mn}\,\sin(\boldsymbol{k}\cdot\boldsymbol{g}) = \mathrm{VSI} \tag{F.84}$$

F.15 Subroutine ESTROG (both4.f)

Input : the real variables vectors A($1 : n_{ao}^2$) and AR($1 : n_{ao}^2$) where n_{ao} is the number of atomic orbitals in the system, the integer MV that contains the index of the symmetry operator to be applied on the input vector A, the integer NBINI that contains the index of the first band to be considered, the integer NBFI that contains the index of the last band to be considered and the integer MBANDS that contains the number of bands to be considered.
(in SMAT subroutine, this last input integer is defined as MBANDS = NBFI - NBINI + 1)
The subroutine modifies the elements of the vector AR in the range ($1 :$ MBANDS $\cdot\, n_{ao}$), i.e. it modifies AR($1 :$ MBANDS $\cdot\, n_{ao}$) by rotating these elements of the vector according to the symmetry operator of the space group classified with index MV as given in the third input argument of the subroutine.

line	**SUBROUTINE ESTROG: defined in both4.f**	CRYSTAL version: SVN_DEV1286
6325	SUBROUTINE ESTROG(A,AR,MV,NBINI,NBFI,MBANDS)	
	USE NUMBERS	
	USE PARAME_MODULE	
	USE PARINF_MODULE	
	USE MEMORY_SCREEN	
	USE ROTMATRIX	MV = i_s (= index of the symmetry operator) as input argument
	USE BASATO_MODULE	NBINI = $i_{beg,b}$ = initial band index as input argument
	USE EXPO_MODULE	NBFI = $i_{end,b}$ = final band index as input argument
	IMPLICIT REAL(FLOAT) (A-H,O-Z)	MBANDS = m_b = total number of bands (states) as input argument
	DIMENSION A(*),AR(*)	MGNAV(LAVRS,MV) = index of the associated $\boldsymbol{g}$ vector
	NDF=INF(7)	INF(7) = n_{ao} = number of Atomic Orbitals (AOs)
	MVF=INF(2)	MVF = INF(2) = n_s = number of symmetry operators
	INDBAS=(NBINI-1)*NDF	INDBAS = initial index for basis set selection = $(i_{beg,b} - 1)\, n_{ao}$
	AR(1:MBANDS*NDF)=0._FLOAT	AR(1:MBANDS·NDF) = $\lambda_i = 0$ with $i = 1, ..., m_b \cdot n_{ao}$
	DO LAVRS=1,INF(24)	BEGIN loop on the number of atoms n_{at} per unit cell
	ECO=EX(1,MGNAV(LAVRS,MV))	cosin factor cos(α) of the I-th $\boldsymbol{g}$ vector for a given $\boldsymbol{k}$ point [see Sections F.7 - F.8]
	DO LA=NSHPRI(LAVRS),NSHPRI(LAVRS+1)-1	BEGIN loop over shells associated to the LAVRS-th atom
	ICA=NDQ(LAV(LA,MV))+INDBAS	ICA = cumulative number of AOs + INDBAS (CRYSTAL list order)
	ICO=NDQ(LA)	ICO = cumulative number of AOs (CRYSTAL list order)
	INF3=LAT(LA)*MVF+MV	INF3 = LAT(LA) $\cdot\, n_s + i_s$
	DO I=MINZ(INF3)+1,MINZ(INF3+1)	BEGIN loop on point group operators
	VCO=TTO(I)*ECO	VCO = $T_{ij} \cos(\boldsymbol{k} \cdot \boldsymbol{g}) \equiv T_{ij} \cos(\alpha)$
	ICA1=MMO(I)+ICA	ICA1 = pointer to the index of the element of A vector to be used in the operation
	ICA2=MMOM(I)+ICO	ICA2 = pointer to the index of the element of AR vector to be used in the operation
	DO IND=NBINI,NBFI	BEGIN loop on selected band indexes (IND = $i_{beg,b}$, ..., $i_{end,b}$)
6350	AR(ICA2)=A(ICA1)*VCO+AR(ICA2)	AR(ICA2) = AR(ICA2) + A(ICA1) $\cdot\, T_{ij} \cos(\alpha)$ [see eq. (F.74)]
	ICA1=ICA1+NDF	Increment the index ICA1 by n_{ao}
	ICA2=ICA2+NDF	Increment the index ICA2 by n_{ao}
	ENDDO	END loop on selected band indexes
	ENDDO	END loop on point group operators
	ENDDO	END loop over shells associated to the LAVRS-th atom
	ENDDO	END loop on number of atoms n_{at}
	RETURN	
6358	END	

Table F.39: Code and description of subroutine ESTROG (both4.f)

In order to fully understand the operation involved in the subroutine, see the comments on subroutine ESTROF in Section F.13, where the meaning of the vectors and matrices also appearing in the subroutine ESTROG is fully explained.
The difference between subroutine ESTROF (Section F.13) and the subroutine ESTROG is that the last one involves only real quantities, being associated to operations on real reciprocal space points, so that the exponential (F.70) becomes

$$\text{Real reciprocal space point } \boldsymbol{k} \qquad \rightarrow \qquad e^{i\boldsymbol{k}\cdot\boldsymbol{g}} = \cos(\boldsymbol{k} \cdot \boldsymbol{g}) \in \mathcal{R} \tag{F.85}$$

and the correspondent eigenvector $\mathbf{C}$ is real ($\mathbf{C} \in \mathcal{R}$). Therefore, the rotation of the eigenvector $\mathbf{C}$ through the symmetry operator with index MV whose matrix representation is given by $\mathbf{T}$ can be computed as

$$\mathbf{C}\,\mathbf{T}\, e^{i\boldsymbol{k}\cdot\boldsymbol{g}} = \mathbf{C}\,\mathbf{T} \cos(\boldsymbol{k} \cdot \boldsymbol{g}) \quad \in \mathcal{R} \tag{F.86}$$

and the result can be stored in the vector AR($1 :\ m_b \cdot n_{ao}$).
Therefore, the following identities can be defined between the notation adopted in the previous equations

and in the operations in the code in line 6350

$$(\mathbf{C})_{ij} \equiv [\Re(\mathbf{C})]_{ij} = \text{A(ICA1)} \tag{F.87}$$

$$(\mathbf{C}\,\mathbf{T}\,e^{i\,\boldsymbol{k}\cdot\boldsymbol{g}})_{kl} = \text{AR(ICA2)} \tag{F.88}$$

$$T_{mn}\,\cos(\boldsymbol{k}\cdot\boldsymbol{g}) = \text{VCO} \tag{F.89}$$

F.16 Subroutine SMAT_SINGLEK (libxj.f)

Input : the vectors E_IRR($1:n$) and E_RED($1:n$) [where $n = 2n_{ao}^2$ in the case of complex $\boldsymbol{k}$ points and $n = n_{ao}^2$ in the case of real $\boldsymbol{k}$ points], the integer IRC that is defined equal to zero for complex $\boldsymbol{k}$ points or different from zero (equal to one) for real $\boldsymbol{k}$ points, the integers (IK1, IK2, IK3) containing the indexes of the reducible $\boldsymbol{k}$ point whose associated eigenvector has to be computed, the index ISIGMA related to closed shell case (ISIGMA = 1) or open shell case (ISIGMA = 2), the logical variable BOLTZ_ACTIVE that is true if a calculation of electronic transport properties is performed (see boltzatorb.f90 module), the integer OUTF that specifies the index of the output file unit on which information are written and the optional real variable TIME_SMAT that contains (if present) the total CPU time required for the execution of the subroutine.
The vectors E_IRR($1:n$) and E_RED($1:n$) contain, respectively, the eigenvector associated to an irreducible $\boldsymbol{k}$ point and to the reducible $\boldsymbol{k}$ point generated using a particular symmetry operator on the eigenvector of a particular irreducible $\boldsymbol{k}$ point.
The subroutine modifies the vectors E_IRR($1:n$) and E_RED($1:n$), together with the integer IRC that is defined equal to zero or one if the irreducible $\boldsymbol{k}$ point that generates the reducible $\boldsymbol{k}$ point with indexes (IK1, IK2, IK3) is a complex or a real $\boldsymbol{k}$ point, respectively. If the optional real variable TIME_SMAT is present, the subroutine modifies it so that its value will store the total CPU time required for the execution of the subroutine. Moreover, if BOLTZ_ACTIVE is true, then the vector ENE_INFO will be also modified by this subroutine.

SUBROUTINE SMAT: defined in libxj.f CRYSTAL version: SVN_DEV1286

```
SUBROUTINE SMAT_SINGLEK(E_IRR, E_RED, IRC, IK1, IK2, IK3, ISIGMA,
*                       BOLTZ_ACTIVE, OUTF, TIME_SMAT)

USE NUMBERS                                          Declaration of modules used in the subroutine
USE PARINF_MODULE
USE RETIC_MODULE
USE XYVDIM_MODULE
USE POLARI_MODULE
USE EXPO_MODULE
USE PARAL1_MODULE
USE MEMORY_USE
USE TIMER_MODULE
USE MOM_MODULE, ONLY: MOM_INPUT, BTOBTRUE

IMPLICIT NONE
INTEGER , INTENT(IN) :: IK1                          Declaration of input, output and input-output variables
INTEGER , INTENT(IN) :: IK2
INTEGER , INTENT(IN) :: IK3
INTEGER , INTENT(IN) :: ISIGMA
LOGICAL , INTENT(IN) :: BOLTZ_ACTIVE
INTEGER , INTENT(IN) :: OUTF
REAL(FLOAT), INTENT(INOUT) :: E_IRR(*), E_RED(*)
INTEGER , INTENT(INOUT) :: IRC
REAL(FLOAT), INTENT(INOUT), OPTIONAL :: TIME_SMAT

INTEGER :: JR1, JR2, JR3, MV, MVF, NDF               Declaration of local variables
INTEGER :: IS1IS2, IS123, NDFJJJ, NDFVRS
INTEGER :: NBINI, NBFI, MBANDS
INTEGER :: IS10, IS20, IS30
INTEGER :: MR1, MR2, MR3, NREC
INTEGER :: ID1, ID2, ID3
INTEGER :: JPROC, K, KCUM, ISPIN
INTEGER :: IND_IRR, IND_SYM
INTEGER :: NDATA, IO08
LOGICAL :: KFOUND
LOGICAL :: USE_ESTROE, USE_ESTROF
REAL(FLOAT) :: DCPU, TOTAL_CPU
REAL(FLOAT) :: DELAPSE, TOTAL_ELAPSED
REAL(FLOAT) :: TIME_BEG, TIME_END
CHARACTER(LEN = 12) :: NOMZ = 'SMAT_ONEK'

CALL CPU_TIME(TIME_BEG)      Initial CPU time TIME_BEG to compute the total CPU time spent in this subroutine

MVF = INF(2)                 ! NUMBER OF SYMMETRY OPERATORS
NDF = INF(7)                 ! NUMBER OF ATOMIC ORBITALS ( N.AOs )
! INF64 = INF(64)            ! = 0 CLOSED SHELL / = 1 OPEN SHELL
IS1IS2 = IS1*IS2             ! IS1, IS2, IS3 = (n_x, n_y, n_z) = SHRINKING FACTORS ALONG KX, KY, KZ → IS1IS2 = s_12 = n_x · n_y
```

```
IS123 = IS1*IS2*IS3                    IS123 = s_123 = n_x · n_y · n_z
NDFJJJ = NDF*NDF                       ! NDFJJJ = NAO**2 = m^2
NDFVRS = NDFJJJ+NDFJJJ                 ! NDFVRS = 2*NAO**2 = 2m^2
NBINI = 1                              ! INITIAL BAND INDEX = 1
NBFI = NDF                             ! FINAL BAND INDEX = m
MBANDS = NBFI-NBINI+1                  ! NUMBER OF TOTAL BANDS CONSIDERED ( = N.AOs ) = n_b = m
IO08 = IUNIT(8)                        ! UNIT WHERE THE IRREDUCIBLE EIGENVECTORS ARE READ IO08 = fort.8
IF(BTOBTRUE) THEN                      ! LOGICAL RELATED TO THE MAXIMUM OVERLAP METHOD (MOM)
IO08 = IUNIT(153)                      [ Option activated for band to band transition calculations ]
ENDIF
IS10 = IS1*16                          IS10 = 16 · n_x
IS20 = IS2*16                          IS20 = 16 · n_y
IS30 = IS3*16                          IS30 = 16 · n_z
KFOUND = .FALSE.                       Logical variable initialized to false
USE_ESTROF = .FALSE.                   Logical variable initialized to false
USE_ESTROE = .FALSE.                   Logical variable initialized to false

! FOUND THE INDEX IND_IRR OF THE IRREDUCIBLE K POINT THAT GENERATES THE (IK1, IK2, IK3) REDUCIBLE K POINT
! LOOP OVER IRREDUCIBLE K POINTS BEGIN
LOOP_K_IRR : DO K = 1, NKF             BEGIN loop over irreducible k points
JR1 = JJ(1, K)                         ! KX RECIPROCAL COORDINATE OF THE IRR K POINT JR1 = k_x
JR2 = JJ(2, K)                         ! KY RECIPROCAL COORDINATE OF THE IRR K POINT JR2 = k_y
JR3 = JJ(3, K)                         ! KZ RECIPROCAL COORDINATE OF THE IRR K POINT JR3 = k_z
IF(LATVRS(K) .EQ. 0) THEN ! ———————————— COMPLEX K POINT CASE BEGIN LATVRS(K) = 0 [ see Section F.6 ]
DO MV = 1, MVF ! LOOP OVER SYMMETRY OPERATOR BEGIN   BEGIN loop on the symmetry operators
ID1 = IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10   see formula F.57
MR1 = MOD(ID1, IS1)                    see formula F.57
ID2 = IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20   see formula F.58
MR2 = MOD(ID2, IS2)                    see formula F.58
ID3 = IRR(3,1,MV)*JR1+ IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30  see formula F.59
MR3 = MOD(ID3, IS3)                    see formula F.59
IF(MR1 == IK1 .AND. MR2 == IK2 .AND. MR3 == IK3 ) THEN   If indexes (MR1, MR2, MR3) are equal to input indexes (IK1, IK2, IK3) →
KFOUND = .TRUE.                        → set KFOUND = true
USE_ESTROF = .TRUE.                    → set USE_ESTROF = true
IRC = LATVRS(K)                        → IRC = LATVRS(K) = 0
IND_IRR = K                            → save the index of irreducible k point in IND_IRR
IND_SYM = MV                           → save the index of the symmetry operator in IND_SYM
NREC = MR1 + 1 + MR2*IS1 + MR3*IS1IS2 + (ISIGMA-1)*IS123   see formula F.60
EXIT LOOP_K_IRR ! THE LOOP OVER IRREDUCIBLE K POINTS   EXIT loop over irreducible k points
ENDIF                                  Endif
ENDDO ! LOOP OVER SYMMETRY OPERATOR END   END loop on the symmetry operators
IF(.NOT.KFOUND) THEN                   If in the previous loop the input k point (IK1, IK2, IK3) is not found
JR1 = MOD(IS1-JR1, IS1)                see formula F.61
JR2 = MOD(IS2-JR2, IS2)                see formula F.62
JR3 = MOD(IS3-JR3, IS3)                see formula F.63
DO MV = 1, MVF ! LOOP OVER SYMMETRY OPERATOR BEGIN   BEGIN loop on the symmetry operators
ID1 = IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10   see formula F.64
MR1 = MOD(ID1, IS1)                    see formula F.64
ID2 = IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20   see formula F.65
MR2 = MOD(ID2, IS2)                    see formula F.65
ID3 = IRR(3,1,MV)*JR1+IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30   see formula F.66
MR3 = MOD(ID3, IS3)                    see formula F.66
IF(MR1 == IK1 .AND. MR2 == IK2 .AND. MR3 == IK3 ) THEN   If indexes (MR1, MR2, MR3) are equal to input indexes (IK1, IK2, IK3) →
KFOUND = .TRUE.                        → set KFOUND = true
USE_ESTROE = .TRUE.                    → set USE_ESTROE = true
IRC = LATVRS(K)                        → IRC = LATVRS(K) = 0
IND_IRR = K                            → save the index of irreducible k point in IND_IRR
IND_SYM = MV                           → save the index of the symmetry operator in IND_SYM
NREC = MR1 + 1 + MR2*IS1 + MR3*IS1IS2 + (ISIGMA-1)*IS123   see formula F.67
EXIT LOOP_K_IRR ! THE LOOP OVER IRREDUCIBLE K POINTS   EXIT loop over irreducible k points
ENDIF                                  Endif
ENDDO ! LOOP OVER SYMMETRY OPERATOR END   END loop on the symmetry operators
ENDIF ! NOT KFOUND PREVIOUSLY          Endif
ENDIF ! ———————————— COMPLEX K POINT CASE END
IF(LATVRS(K) .NE. 0) THEN ! ———————————— REAL K POINT CASE BEGIN
DO MV = 1, MVF ! LOOP OVER SYMMETRY OPERATOR BEGIN   BEGIN loop on the symmetry operators
ID1 = IRR(1,1,MV)*JR1+IRR(1,2,MV)*JR2+IRR(1,3,MV)*JR3+IS10   see formula F.57
MR1 = MOD(ID1, IS1)                    see formula F.57
ID2 = IRR(2,1,MV)*JR1+IRR(2,2,MV)*JR2+IRR(2,3,MV)*JR3+IS20   see formula F.58
MR2 = MOD(ID2, IS2)                    see formula F.58
ID3 = IRR(3,1,MV)*JR1+IRR(3,2,MV)*JR2+IRR(3,3,MV)*JR3+IS30   see formula F.59
MR3 = MOD(ID3, IS3)                    see formula F.59
IF(MR1 == IK1 .AND. MR2 == IK2 .AND. MR3 == IK3 ) THEN   If indexes (MR1, MR2, MR3) are equal to input indexes (IK1, IK2, IK3) →
KFOUND = .TRUE.                        → set KFOUND = true
IRC = LATVRS(K)                        → IRC = LATVRS(K) ≠ 0
IND_IRR = K                            → save the index of irreducible k point in IND_IRR
IND_SYM = MV                           save the index of the symmetry operator in IND_SYM
NREC = MR1 + 1 + MR2*IS1 + MR3*IS1IS2 + (ISIGMA-1)*IS123   see formula F.60
```

```
EXIT LOOP_K_IRR ! THE LOOP OVER IRREDUCIBLE K POINTS    EXIT loop over irreducible k points
ENDIF                                                   Endif
ENDDO ! LOOP OVER SYMMETRY OPERATOR END                 END loop on the symmetry operators
ENDIF ! ------------------------------------------------ REAL K POINT CASE END
ENDDO LOOP_K_IRR                                        END loop over irreducible k points
! LOOP OVER IRREDUCIBLE K POINTS END

IF_KFOUND : IF(.NOT.KFOUND) THEN ! REDUCIBLE K POINT NOT FOUND   If the input reducible k point (IK1, IK2, IK3) is not found →
IF(IAMEQ0) THEN                                                  → write using only the process with index zero
WRITE(OUTF, 14) IK1, IK2, IK3                                    → in the output file that the input k point is not found
ENDIF ! IAMEQ0
ELSE IF_KFOUND ! REDUCIBLE K POINT FOUND                         Else if the input reducible k point (IK1, IK2, IK3) is found →
IF(BOLTZ_ACTIVE) ENE_INFO(NREC) = IND_IRR   → if BOLTZAO_ACTIVE → map the NREC-th reducible to the K-th irreducible k point
IF(IAMEQ0) THEN                                                  → write using only the process with index zero
IF(LATVRS(IND_IRR) .EQ. 0) THEN                                  → [ complex k point case ]
WRITE(OUTF, 10) 'C', MR1,MR2,MR3, 'C', JR1,JR2,JR3, IND_IRR      → write indexes of reducible and irreducible k points
ELSE                                                             → [ real k point case ]
WRITE(OUTF, 10) 'R', MR1,MR2,MR3, 'R', JR1,JR2,JR3, IND_IRR      → write indexes of reducible and irreducible k points
ENDIF
WRITE(OUTF, 12) IND_SYM, ISIGMA   → write the index of the symmetry operator found and the input index ISIGMA [ spin state ]
ENDIF ! IAMEQ0

! READ THE FILE IO08 WITH EIGENVECTORS BEGIN
! ISIGMA = 1 CLOSED SHELL – ISIGMA = 2 OPEN SHELL
KCUM = 0                                                Initialization of KCUM index [ KCUM = 0 ]
LOOP_READ_EIG : DO ISPIN = 1, ISIGMA                    BEGIN loop over spin state
DO K = 1, NKF                                           BEGIN loop over irreducible k points
KCUM = KCUM + 1                                         Update the index KCUM → KCUM + 1 [ index of the k point ]
IF(LATVRS(K) .EQ. 0) THEN                               If LATVRS(K) = 0 → complex k point →
NDATA = NDFVRS                                          → NDATA = 2m^2
ELSE IF(LATVRS(K) .NE. 0) THEN                          Else if LATVRS(K) ≠ 0 → real k point →
NDATA = NDFJJJ                                          → NDATA = m^2
ENDIF                                                   Endif
JPROC = 0                                               All the processes initialize JPROC = 0
IF(.NOT.JDONE(KCUM)) THEN ! PARALLEL READ :             If the process owns the management of the KK-th k point
CALL RREAD(IO08, E_IRR, NDATA)                          that process reads 2m^2 real numbers from fort.8 file
JPROC = IAM                                             that process save in JPROC its index = IAM [ for all the other processes JPROC = 0 ]
ENDIF                                                   Endif
IF(K == IND_IRR .AND. ISPIN == ISIGMA) EXIT LOOP_READ_EIG   If the eigenvector correct irreducible k point is saved in E_IRR → EXIT
ENDDO                                                   END loop over irreducible k points
ENDDO LOOP_READ_EIG                                     END loop over spin state
! READ THE FILE IO08 WITH EIGENVECTORS END

! BROADCAST EIGENVECTOR THAT HAS BEEN READ BY THE JPROC-TH PROCESS
CALL IGSUM(JPROC, 1) ! MPI_ALLREDUCE (MPI_SUM)  Sums JPROC values from all processes and distributes the result back to all processes
CALL BROADCAST(E_IRR, NDATA, JPROC)             Broadcast data in A(1:NDATA) from process JPROC to all other processes
REWIND IO08                                     Position the file associated with the specified unit IO08 to its initial point

IF(LATVRS(IND_IRR) .EQ. 0) THEN ! ---------------------- COMPLEX K POINT CASE BEGIN LATVRS(K) = 0 [ see Section F.6 ]
! COMPUTE e**(ikG) AND SAVE IN EX MATRIX        n_g = INF(79) = number of direct lattice vector
CALL EXPU(MR1, MR2, MR3)                        modify the matrix EX(1:n_g) [ see Section F.7 ]
IF(USE_ESTROF) THEN                             Rotate eigenvector A using NINV(MV) and save the result in AR [ see Section F.13 ]
! COMPUTE THE EIGENVECTOR E_RED ASSOCIATED TO THE NREC-TH REDUCIBLE K POINT
! USING THE EIGENVECTOR E_IRR OF THE IND_IRR-TH IRREDUCIBLE K POINT AND THE MV-TH SYMMETRY OPERATOR
CALL ESTROF(E_IRR, E_RED, NINV(IND_SYM), NBINI, NBFI, MBANDS)    see Section F.13
ELSE IF(USE_ESTROE) THEN                        Rotate eigenvector A using NINV(MV) and save the result in AR [ see Section F.14 ]
! COMPUTE THE EIGENVECTOR E_RED ASSOCIATED TO THE NREC-TH REDUCIBLE K POINT
! USING THE EIGENVECTOR E_IRR OF THE IND_IRR-TH IRREDUCIBLE K POINT AND THE MV-TH SYMMETRY OPERATOR
CALL ESTROE(E_IRR, E_RED, NINV(IND_SYM), NBINI, NBFI, MBANDS)    see Section F.14
ENDIF
ENDIF ! ------------------------------------------------ COMPLEX K POINT CASE END
IF(LATVRS(IND_IRR) .NE. 0) THEN ! ---------------------- REAL K POINT CASE BEGIN
! COMPUTE e**(ikG) AND SAVE IN EX MATRIX        n_g = INF(79) = number of direct lattice vector
CALL EXPT(MR1, MR2, MR3)                        modify the matrix EX(1:n_g) [ see Section F.8 ]
! COMPUTE THE EIGENVECTOR E_RED ASSOCIATED TO THE NREC-TH REDUCIBLE K POINT
! USING THE EIGENVECTOR E_IRR OF THE IND_IRR-TH IRREDUCIBLE K POINT AND THE MV-TH SYMMETRY OPERATOR
CALL ESTROG(E_IRR, E_RED, NINV(IND_SYM), NBINI, NBFI, MBANDS)    Rotate eigenvector [ see Section F.14 ]
ENDIF ! ------------------------------------------------ REAL K POINT CASE END
ENDIF IF_KFOUND

! WRITE TIMINGS IN THE OUTPUT FILE                  Write timings [ TELAPSE and TCPU ] using subroutine GET_TIME_DATA
CALL GET_TIME_DATA(DCPU, TOTAL_CPU, DELAPSE, TOTAL_ELAPSED)   subroutine GET_TIME_DATA defined in machdep.F library
IF(IAMEQ0) WRITE(OUTF, 100) NOMZ, TOTAL_ELAPSED, TOTAL_CPU

CALL CPU_TIME(TIME_END)                           Final CPU time TIME_END to compute the total CPU time spent in this subroutine
IF(PRESENT(TIME_SMAT)) TIME_SMAT = TIME_END - TIME_BEG  Update input optional float variable TIME_SMAT
IF(IAMEQ0) WRITE(OUTF, 200) TIME_END - TIME_BEG   Write in the output file the total CPU time spent in this subroutine
```

```
RETURN                                                          Return the function
10 FORMAT(/1X, 79('-')/                                         Format statements
*            1X, '>>>> REDUCIBLE K POINT : ', 3(I0,2X),
*            T64,'GENERATED BY >>>> '
*            /1X, ' IRREDUCIBLE K POINT (', A1, ') : ', 3(I0,2X),
*            T64,'INDEX : ', I0)
12 FORMAT( 1X, ' SYMMETRY OPERATOR : ', I0,
*            T64,'SIGMA : ', I0,
*            /1X, 79('-'))
14 FORMAT(/1X, 79('-')/
*            1X,'>>>> REDUCIBLE K POINT (', A1, ') : ', 3(I0,2X),
*            ' NOT FOUND TO BE GENERATED ',
*            /1X, 79('-'))
100 FORMAT(1X,30('T'),1X,A,T45,'TELAPSE',F12.2,T64,' TCPU',F12.2)
200 FORMAT(1X,30('T'),' TIME ELAPSED IN SMAT_ONEK : ',ES16.9,' SEC')

END SUBROUTINE SMAT_SINGLEK
```

Appendix G

Fort.98 unit information

properties.f90		
subroutine f90main3		
leggi = iopt(1:6).eq.'RDFMWF'		
leggi = true if the keyword RDFMWF is used		
leggi = false if the keyword RDFMWF is not used		
call init_properties_io(leggi) $\Longrightarrow$	**libxs.f**	
	subroutine init_properties_io(read_formatted)	
	check of fort.98 or fort.9 file existence	
	if exists, open it	
	otherwise return error in output	
	if(read_formatted) *cerca e apre la fort.98*	
	else *cerca e apre la fort.9*	
	endif	
if(leggi) call convrt $\Longrightarrow$	**libxc_prop.f**	
	subroutine convrt	
	print header (see Fig. G.1)	
	call inp12f $\Longrightarrow$	**libxc_prop.f**
		subroutine inp12f
		read the fort.98 file
		allocate vectors
		broadcast information to all processors
	call resprt(qtot) $\Longrightarrow$	**props1.f**
		subroutine resprt(qtot)
		print information (see Fig. G.2)
	call basprt $\Longrightarrow$	**libx1_com.f**
		subroutine basprt
		print information (see Fig. G.3)
	call genprt $\Longrightarrow$	**both4.f**
		subroutine genprt
		print information (see Fig. G.4)
	if(iameq0) call outo3b(iunit(9)) $\Longrightarrow$	**libxc_com.f**
		subroutine outo3b(io09)
		write formatted information on fort.9 file
	Deallocation of vectors	
	call free_basold $\Longrightarrow$	**both4.f**
		subroutine free_basold
		(if allocated) deallocate vectors

libxc_com.f		
subroutine out12f		
write the fort.98 file		

Table G.1: The first part of the table describes the hierarchy of subroutines invoked if the keyword RDFMWF is used. The keyword RDFMWF, entered in the first record of the properties input deck, reads formatted data from file fort.98 and writes the corresponding unformatted data in file fort.9. In the second part of the table, the hierarchy of subroutines calling for the printing of file fort.98 is outlined. Plain text is part of the source code, descriptions about some parts of the source code are reported in italics, the names of the libraries and the subroutines involved are written in bold.

```
********************************************************************************
*                                                                              *
* *                                                                            *
*               FORMATTED WAVE FUNCTION DATA FROM FILE fort.98                 *
*               CONVERTED TO BINARY AND WRITTEN IN FILE fort.9                 *
*                                                                              *
********************************************************************************
```

Figure G.1: Header of the conversion from fort.98 to fort.9 file, printed in the output file. This part of the output file is written by the subroutine convrt (libxc_prop.f). The input file mgo.d3 used to generate this output is reported in Figure G.5, right panel.

```
********************************************************************************
TEST
CRYSTAL - PROPERTIES - TYPE OF CALCULATION :  RESTRICTED CLOSED SHELL
********************************************************************************
KOHN-SHAM HAMILTONIAN

(EXCHANGE)[CORRELATION] FUNCTIONAL:(PERDEW-BURKE-ERNZERHOF)[PERDEW-BURKE-ERNZERHOF]


DIRECT LATTICE VECTOR COMPONENTS (ANGSTROM)
        0.00000    2.10500    2.10500
        2.10500    0.00000    2.10500
        2.10500    2.10500    0.00000

LATTICE PARAMETERS  (ANGSTROM AND DEGREES) - PRIMITIVE CELL
      A          B          C          ALPHA      BETA     GAMMA        VOLUME
   2.97692    2.97692    2.97692     60.0000   60.0000   60.0000      18.65462
********************************************************************************
N. OF ATOMS PER CELL         2 COULOMB OVERLAP TOL          (T1) 10**   -6
NUMBER OF SHELLS             6 COULOMB PENETRATION TOL      (T2) 10**   -6
NUMBER OF AO                18 EXCHANGE OVERLAP TOL         (T3) 10**   20
N. OF ELECTRONS PER CELL    20 EXCHANGE PSEUDO OVP (F(G))   (T4) 10**   20
CORE ELECTRONS PER CELL     12 EXCHANGE PSEUDO OVP (P(G))   (T5) 10**   20
N. OF SYMMETRY OPERATORS    48 POLE ORDER IN MONO ZONE                   4
********************************************************************************
  ATOM N.AT.  SHELL    X(A)      Y(A)      Z(A)      EXAD       N.ELECT.
********************************************************************************
   1   12 MG    3     0.000     0.000     0.000   4.000E-01      10.104
   2    8 O     3     2.105     2.105     2.105   2.100E-01       9.896
********************************************************************************
DE(K)                   0.000E+00  ENERGY LEVEL SHIFTING        0.00000
TOTAL ENERGY -2.7528465266093E+02  CONVERGENCE ON ENER       -3.763E-10
KIN. ENERGY   2.7408771812165E+02  VIR. COEFF.           1.04558917E+00
N. OF SCF CYCLES                7  FERMI ENERGY              -0.111E+00
WEIGHT OF F(I) IN F(I+1)      30
SHRINK. FACT.(MONKH.)     8  8  8  SHRINKING FACTOR(GILAT NET)        8
NUMBER OF K POINTS IN THE IBZ  29  CELL VOLUME (A.U.)        125.887585
********************************************************************************
```

Figure G.2: Part of the output file written by the subroutine resprt (props1.f). The input file mgo.d3 used to generate this output is reported in Figure G.5, right panel.

```
*******************************************************************************
LOCAL ATOMIC FUNCTIONS BASIS SET
*******************************************************************************
  ATOM   X(AU)   Y(AU)   Z(AU)  N. TYPE  EXPONENT  S COEF   P COEF   D/F/G COEF
*******************************************************************************
  1 MG   0.000   0.000   0.000
                                1 S
                                       6.837E+04 1.633E+00 0.000E+00 0.000E+00
                                       9.699E+03 3.219E+00 0.000E+00 0.000E+00
                                       2.041E+03 5.820E+00 0.000E+00 0.000E+00
                                       5.299E+02 9.594E+00 0.000E+00 0.000E+00
                                       1.592E+02 1.315E+01 0.000E+00 0.000E+00
                                       5.468E+01 1.281E+01 0.000E+00 0.000E+00
                                       2.124E+01 6.873E+00 0.000E+00 0.000E+00
                                       8.746E+00 1.323E+00 0.000E+00 0.000E+00
                           2-     5 SP
                                       1.568E+02-4.649E-01 1.485E+01 0.000E+00
                                       3.103E+01-1.743E+00 1.632E+01 0.000E+00
                                       9.645E+00-7.355E-01 1.240E+01 0.000E+00
                                       3.711E+00 1.307E+00 6.127E+00 0.000E+00
                                       1.612E+00 1.375E+00 2.351E+00 0.000E+00
                                       6.429E-01 3.702E-01 4.648E-01 0.000E+00
                           6-     9 SP
                                       4.000E-01 8.718E-01 1.103E+00 0.000E+00
  2 O    3.978   3.978   3.978
                               10 S
                                       4.000E+03 1.243E+00 0.000E+00 0.000E+00
                                       1.356E+03 2.928E+00 0.000E+00 0.000E+00
                                       2.485E+02 5.767E+00 0.000E+00 0.000E+00
                                       6.953E+01 6.948E+00 0.000E+00 0.000E+00
                                       2.389E+01 6.681E+00 0.000E+00 0.000E+00
                                       9.276E+00 3.521E+00 0.000E+00 0.000E+00
                                       3.820E+00 6.897E-01 0.000E+00 0.000E+00
                                       1.235E+00 1.428E-01 0.000E+00 0.000E+00
                          11-    14 SP
                                       5.219E+01-4.094E-01 5.793E+00 0.000E+00
                                       1.033E+01-1.250E+00 5.862E+00 0.000E+00
                                       3.210E+00-2.363E-01 3.933E+00 0.000E+00
                                       1.235E+00 1.066E+00 2.039E+00 0.000E+00
                                       5.364E-01 6.396E-01 5.711E-01 0.000E+00
                          15-    18 SP
                                       2.100E-01 5.377E-01 4.928E-01 0.000E+00
```

Figure G.3: Part of the output file written by the subroutine basprt (libx1_com.f). The input file mgo.d3 used to generate this output is reported in Figure G.5, right panel.

```
*******************************************************************************
N. OF ATOMS PER CELL         2  COULOMB OVERLAP TOL         (T1) 10**   -6
NUMBER OF SHELLS             6  COULOMB PENETRATION TOL     (T2) 10**   -6
NUMBER OF AO                18  EXCHANGE OVERLAP TOL        (T3) 10**   20
N. OF ELECTRONS PER CELL    20  EXCHANGE PSEUDO OVP (F(G))  (T4) 10**   20
CORE ELECTRONS PER CELL     12  EXCHANGE PSEUDO OVP (P(G))  (T5) 10**   20
N. OF SYMMETRY OPERATORS    48  POLE ORDER IN MONO ZONE                  4
*******************************************************************************
TYPE OF CALCULATION :  RESTRICTED CLOSED SHELL
KOHN-SHAM HAMILTONIAN

(EXCHANGE)[CORRELATION] FUNCTIONAL:(PERDEW-BURKE-ERNZERHOF)[PERDEW-BURKE-ERNZERHOF]
```

Figure G.4: Part of the output file written by the subroutine genprt (both4.f). The input file mgo.d3 used to generate this output is reported in Figure G.5, right panel.

```
TEST11 - MGO BULK
CRYSTAL
0 0 0
 225
4.21
2
 12 0.    0.    0.
  8 0.5   0.5   0.5
END
12  3
 0  0  8  2.  1.
 68371.875      0.0002226
  9699.34009    0.0018982
  2041.176786   0.0110451
   529.862906   0.0500627
   159.186000   0.169123
    54.6848     0.367031
    21.2357     0.400410
     8.74604    0.14987
0 1 6 8. 1.
156.795  -0.00624  0.00772
31.0339  -0.07882  0.06427
9.6453   -0.07992  0.2104
3.7109    0.29063  0.34314
1.61164   0.57164  0.3735
0.64294   0.30664  0.23286
0 1 1 0. 1.
0.4       1.       1.
8 3
0 0 8 2. 1.
4000.    0.00144
1355.58  0.00764
248.545  0.05370
69.5339  0.16818
23.8868  0.36039
9.27593  0.38612
3.82034  0.14712
1.23514  0.07105
0 1 5 8. 1.
52.1878 -0.00873  0.00922
10.3293 -0.08979  0.07068
3.21034 -0.04079  0.20433
1.23514  0.37666  0.34958
0.536420 0.42248  0.27774
0 1 1 0. 1.
0.210000 1.  1.
99   0
END
DFT
PBE
END
SHRINK
8 8
FMIXING
30
PPAN
END
```

```
RDFMWF
BAND
MGO - Path: Gamma-X-W-L-Gamma
4 8 60 7 14 1 0
 0 0 0  4 0 4
 4 0 4  4 2 6
 4 2 6  4 4 4
 4 4 4  0 0 0
END
```

Figure G.5: Input files mgo.d12 (left panel) and mgo.d3 (right panel) used to convert formatted file fort.98 to binary fort.9 file (keyword RDFMWF) and to compute the band structure of MgO crystal.

Variable	Description
MVF	number of symmetry operators extended to inversion
INF(2)	number of symmetry operators
NAF	number of atoms per unit cell
LAF	number of shells
NPRIM	number of primitives GTO
INF(5)	maximum number of star of direct lattice vectors
INF(79)	number of direct lattice vectors **G** available
INF(64)	= 0 (closed shell), = 1 open shell
INF(228)	number of elements of FG_IRR array (RO)
INF(19)	dimension of the vector F(**G**) or P(**G**) (**G** IRR)
MVLAF	total number of shell couple sets C_s (after shell selection based on overlap criteria, the shell couples are reduced according to hermiticity, and the remaining shells are classified in couple sets C_s containing symmetry-related shells)
MVLAF1	MVLAF+1
INF(39)	maximum number of irreducible **G** vectors for a shell couple set type C_{st} (shell couple set type remaining after shell selection based on overlap criteria, selection based on hermiticity, classification by symmetry related couple sets and classification in type of couple sets by the couple behavior under symmetry operators effect)
INF(133)	number of different sets of irreducible **G** vectors
INF(37)	maximum number of **G** vectors allocated on each shell couple type C_t + 1 = INF(145) + 1 (after shell selection based on overlap criteria, the remaining shell couples are classified in shell couples types C_t on the base of the distance vector joining them)
INF(145)	maximum number of **G** vectors allocated on each shell couple type C_t (after shell selection based on overlap criteria, the remaining shell couples are classified in shell couples types C_t on the base of the distance vector joining them)
NNGIDMF	INF(39)·INF(133)
MAX_G_COUPLES	total number of couples of shells after the shell selection based on overlap criteria

Definitions
Star : a star is the set of direct **G** vectors with the same Euclidean norm $\|\mathbf{G}\|$
FG_IRR : irreducible Fock matrix in the direct space (arranged and stored as a vector in the code)
PG_IRR : irreducible density matrix in the direct space (arranged and stored as a vector in the code)

Table G.2: List of important variables and definitions used in Table G.3.

Variable	Description	Type	Format	Number of data
TITOLO		CHARACTER	A80	1
"LIMINF LIMTOL LIMPAR"		CHARACTER		
LIMINF LIMTOL LIMPAR		INTEGERS	3I10	3
"INF"		CHARACTER	A6/	
INF(I),I=1,LIMINFO	array containing INF integers (see INF manual)	INTEGERS	(8I10)	LIMINFO
"TOL"		CHARACTER	A6/	
ITOL(I),I=1,LIMTOLO	array containing tolerances integers	INTEGERS	(8I10)	LIMTOLO
"PAR"		CHARACTER	A6,1P/	
PAR(I),I=1,LIMPARO	array containing PAR reals (see PAR manual)	REALS	(4E20.13)	LIMPARO
"XYVGVE"		CHARACTER	A6,1P/	
XYV(9*MVF)	rotation matrices	REALS	(4E20.13)	9·MVF
TRASV(3*MVF)	translators	REALS	(4E20.13)	3·MVF
PARET(9)	direct lattice vectors cartesian components (Bohr)	REALS	(4E20.13)	9
W1R(9)	transformation matrix from primitive to crystallographic cell (see Section G.0.1)	REALS	(4E20.13)	9
"BASATO"		CHARACTER	A6,1P/	
AZNUC(I), I=1,NAF	array containing initial nuclear charges for each atom (input/output order)	REALS	(4E20.13)	NAF
XA(I,J), I=1,3, J=1,NAF	atomic positions (cartesian, Bohr) (input/output order)	REALS	(4E20.13)	3·NAF
CHE(I), I=1,LAF	array containing initial formal electron charges attributed to the shell (output order)	REALS	(4E20.13)	LAF

EXAD(I), I=1,LAF	array containing the last (minor) exponent of the primitive GTF (exponent of the most diffuse GTF in each shell of each basis set, output order)	REALS	(4E20.13)	LAF
XL(I,J), I=1,3, J=1,LAF	atomic positions (cartesian, Bohr) of the atom associated to the J-th shell (output order)	REALS	(4E20.13)	3·LAF
EXX(I), I=1,NPRIM	array containing the exponents of the primitive GTF (output order)	REALS	(4E20.13)	NPRIM
C1(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of S shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
C2(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of P shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
C3(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of D/F/G shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
CMAX(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of the shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
C2W(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of P shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
C3W(I), I=1,NPRIM	array in which each element is a product of normalization factors for the wavefunction and the linear coefficients for the expansion of D/F/G shells in primitive GFT (see Ref. 2)	REALS	(4E20.13)	NPRIM
*** IF SOME ATOMS HAVE CORE PSEUDO-POTENTIALS				
"INFPOT"		CHARACTER	A6/	
II,JJ,ITYPSE	II is the total number of primitive GTF contained in all the pseudo-potential basis sets used in input JJ = NANGECP = 6 (JJ has a value of 6·ITYPSE). NANGECP is an integer defined in the code as the number of angular symmetry + 1 ITYPSE is the number of type of pseudo-potentials basis sets used in input	INTEGERS	(8I10)	3
NPOT(I), I=1,II	array containing the modulus of the value of n_{kl} (see eq. 3.18 Ref. 3) for each GTF (modulus of integers reported in columns labeled with N in the table with pseudo-potential information in the output file) (output order, row-by-row) (see Section G.0.2)	INTEGERS	(8I10)	II
NBTYP(I), I=1,JJ	per each pseudo-potential basis set, number of the primitive GTF of each pseudo-potential shell (output order, see table with pseudo-potential information in the output file)	INTEGERS	(8I10)	JJ

NSOM(I), I=1,ITYPSE+1	NSOM(1) = 0 NSOM(I) with I > 1 : cumulative number of primitive GTF each input pseudo-potential basis set (output order, see table with pseudo-potential information in the output file)	INTEGERS	(8I10)	ITYPSE+1
APOT(I), I=1,II	array containing the exponents of each primitive GTF for each input pseudo-potential basis set (α_k in eq. 3.17 and α_{kl} in eq. 3.18, Ref. 3) (output order, row-by-row, see table with pseudo-potential information in the output file)	REALS	1P,(4E20.13)	II
CPOT(I), I=1,II	array containing the coefficients of each primitive GTF for each input pseudo- potential basis set (C_k in eq. 3.17 and C_{kl} in eq. 3.18, Ref. 3) (output order, row-by-row, see table with pseudo-potential information in the output file)	REALS	1P,(4E20.13)	II
*** ENDIF SOME ATOMS HAVE CORE PSEUDO-POTENTIALS				
*** IF KEYWORDS MOLEBSSE, MOLSPLIT OR ATOMBSSE ARE USED				
"MOLBAR"		CHARACTER	A6,1P/	
XBAR(I,J), I=1,3, J=1,NAF	coordinates of the barycentre of the molecular fragments (which coincides with the atomic seeds for each fragment defined in the input file) (input/output order)	REALS	(4E20.13)	3·NAF
N1MOL(I), I=1,NAF	array containing two indices per each fragment, which define the initial and final atomic indices for the range of atoms contained in each fragment (input/output order)	INTEGERS	(8I10)	NAF
*** ENDIF KEYWORDS MOLEBSSE, MOLSPLIT OR ATOMBSSE ARE USED				
"NINV"		CHARACTER	A6/	
NINV(I), I=1,INF(2)	index INV of the inverse matrix (output order, see SYMMOPS part in the output file)	INTEGERS	(8I10)	INF(2)
"BASATO"		CHARACTER	A6/	
NAT(I), I=1,NAF	atomic number	INTEGERS	(8I10)	NAF
NSHPRI(I), I=1,NAF+1	NSHPRI(1) = 1 NSHPRI(J) = $\sum_{I=1}^{J>1}$ NSHPRI(I) cumulative number of shells (input order)	INTEGERS	(8I10)	(NAF+1)
IPSEUD(I), I=1,NAF	= 0 (all atoms are all electron) = 1 (some atoms have core pseudo-potentials)	INTEGERS	(8I10)	NAF
LAA(I), I=1,LAF+1	LAA(1) = 1 LAA(J) = $\sum_{I=1}^{J>1}$ LAA(I) cumulative number of gaussians (input order)	INTEGERS	(8I10)	(LAF+1)
LAN(I), I=1,LAF	number of gaussians of the I-th shell (input order)	INTEGERS	(8I10)	LAF
LAT(I), I=1,LAF	type of the I-th shell 0=S; 1=SP, 2=P; 3=D; 4=F (input order)	INTEGERS	(8I10)	LAF
LATAO(I), I=1,LAF	number of atomic orbitals of the I-th shell (input order)	INTEGERS	(8I10)	LAF
NDQ(I), I=1,LAF+1	NDQ(1) = 0 NDQ(J) = $\sum_{I=1}^{J>1}$ NDQ(I) cumulative number of atomic orbitals (input order)	INTEGERS	(8I10)	(LAF+1)
LATOAT(I), I=1,LAF	array containing the indices of the atom to which the I-th shell is associated (input order)	INTEGERS	(8I10)	LAF
"SPINOR"		CHARACTER	A6/	
ISPIN(I), I=1,NAF	spin of the I-th atom at convergence (final SCF atomic spins, output order)	INTEGERS	(8I10)	NAF

ICHMOD(I),I=1,NAF	array containing integers different from zero in the positions of the array corresponding to the label of the atoms (labels in output order) whose electronic configuration has been modified with the keyword CHEMOD (see Section G.0.3)	INTEGERS	(8I10)	NAF
*** IF TYPE OF RUN IS != SCF (= TEST OF INPUT OR = RESTART)				
"BASOLD"		CHARACTER	A6,1P/	
PAROLD(9)	direct lattice vectors cartesian components (Bohr)	REALS	(4E20.13)	9
TRASVO(3*INF(2))	translators	REALS	(4E20.13)	3·INF(2)
EXAOLD(I), I=1,LAF	last (minor) exponent of the primitive GTF (exponent of the most diffuse GTF in each shell of each basis set, input order)	REALS	(4E20.13)	LAF
XOLD(I,J), I=1,3, J=1,NAF	atomic positions (cartesian, Bohr, input order)	REALS	(4E20.13)	3·NAF
XLOLD(I,J), I=1,3, J=1,LAF	atomic positions (cartesian, Bohr) of the atom associated to the J-th shell (input order)	REALS	(4E20.13)	3·LAF
HMODUS(I), I=1,INF(5)+1	array containing the square of the Euclidean norm of the **G** vectors in the I-th star, i.e. the values of $\|\mathbf{G}\|^2$ for the direct vectors in the I-th star HMODUS(1) is always equal to zero HMODUS(INF(5)+1) = 1.0000E+025	REALS	(4E20.13)	(INF(5)+1)
XGOLD(I,J), I=1,3, J=1,INF(79)	cartesian coordinates of the J-th direct lattice vector **G** (see Section 3.4 Ref. 1)	REALS	(4E20.13)	3·INF(79)
*** ENDIF TYPE OF RUN IS != SCF (= TEST OF INPUT OR RESTART)				
*** IF NUMBER OF ATOMS "GHOSTIZED" != 0				
"IGHOST"		CHARACTER	A6/	
IGHOST(I), I=1,NAF	array with elements equal to zero (if the atom is not transformed into ghost atom) or equal to the atomic number of the atom (if the atom is transformed into ghost atom) (atomic indices in output order)	INTEGERS	(8I10)	NAF
*** ENDIF NUMBER OF ATOMS "GHOSTIZED" != 0				
CALL RWRITF(IO98,QTOT,NAF)	write NAF atomic charges at convergence (final SCF atomic charges, output order) (see Section G.0.19)	REALS	1P,4E21.13	NAF
CALL RWRITF(IO98,FG,NTUF)	write FG_IRR vector (see Section G.0.19) NTUF=(INF(64)+1)·INF(228)	REALS	1P,4E21.13	NTUF
CALL RWRITF(IO98,PG,NTUT)	write PG_IRR vector (see Section G.0.19) NTUT=(INF(64)+1)·INF(19)	REALS	1P,4E21.13	NTUT
"NCF"		CHARACTER	A6/	
NCF(I), I=1,MVLAF1	array of indexes which point to the positions of the first couple for each shell couple set C_s listed in LA3 and LA4 arrays (see Section G.0.4)	INTEGERS	(8I10)	MVLAF1
"NSTATG"		CHARACTER	A6/	
NSTATG(I), I=1,MVLAF1	NSTATG(1) = 0 array containing the starting point of each couple set C_s in the irreducible Fock and density matrices (equal starting points for both Fock and density matrices) (see Section G.0.5)	INTEGERS	(8I10)	MVLAF1
"NSTAFG"		CHARACTER	A6/	
NSTAFG(I), I=1,MVLAF1	NSTATG(1) = 0 array containing the cumulative number of non-zero elements for each couple set C_s in the irreducible Fock matrix (see Section G.0.6)	INTEGERS	(8I10)	MVLAF1

"LG"		CHARACTER	A6/	
LG(J,I), J=1,3, I=1,INF(79)	Bravais lattice vector indices associated to each I-th direct lattice vector **G** (see Section 3.4 Ref. 1)	INTEGERS	(8I10)	3·INF(79)
"IDIME"		CHARACTER	A6/	
IDIME(I), I=1,MVLAF	array containing the number of **G**-stars to be considered for each couple set C_s in the irreducible density matrix (see Section G.0.7)	INTEGERS	(8I10)	MVLAF
"IDIMF"		CHARACTER	A6/	
IDIMF(I), I=1,MVLAF	arrays containing the number of **G**-stars to be considered for each couple set C_s in the irreducible Fock matrix (see Section G.0.7)	INTEGERS	(8I10)	MVLAF
"LA3"		CHARACTER	A6/	
LA3(I), I=1,LADIM	array of indexes with labels of the first shell of each shell couple C_2 LADIM = NCF(MVLAF1) (see Section G.0.8)	INTEGERS	(8I10)	LADIM
"LA4"		CHARACTER	A6/	
LA4(I), I=1,LADIM	array of indexes with labels of the second shell of each shell couple C_2 LADIM = NCF(MVLAF1) (see Section G.0.9)	INTEGERS	(8I10)	LADIM
"IROF"		CHARACTER	A6/	
IROF(I), I=1,MVLAF	array of indexes that indicate the starting point in the NNGI sequence at which the list of the irreducible **G** vectors for a couple set C_s begins (see Section G.0.10)	INTEGERS	(8I10)	MVLAF
"NNGI"		CHARACTER	A6/	
NNGI(I), I=1,NNGIDMF	array containing the list of the irreducible **G** vectors for each shell couple set C_{st} (see Section G.0.11)	INTEGERS	(8I10)	NNGIDMF
"NSHGI"		CHARACTER	A6/	
NSHGI(I), I=1,NNGIDMF	array containing the indexes corresponding to the number of irreducible **G** vectors for each star of each shell couple set C_{st} (see Section G.0.12)	INTEGERS	(8I10)	NNGIDMF
"NNNC"		CHARACTER	A6/	
NNNC(I), I=1,MAX_G_COUPLES	NNNC(1) = 0 array with indexes pointing to the starting position in the $\mathbf{P}_{irr}$ matrix for each C_1 shell couple. Note that the indexes in the vector points to the starting positions in the $\mathbf{P}_{irr}$ matrix for each C_1 shell couple, but are referred to the construction of the reducible $\mathbf{P}_{red}$ matrix, so the indexes in the vector can assume non-increasing values (see Section G.0.13)	INTEGERS	(8I10)	MAX_G_COUPLES
"JPOINT"		CHARACTER	A6/	
JPOINT(I), I=1,MAX_G_COUPLES	array that relates each couple in C_1 to the index of the subset C_s it belongs in the symmetry classification (see Section G.0.14)	INTEGERS	(8I10)	MAX_G_COUPLES
"LA34V"		CHARACTER	A6/	
LA34V(I), I=1,MAX_G_COUPLES	array containing, for every couple of shells C_1, the starting point in NQGSHG for the associated **G** vectors (see Section G.0.15)	INTEGERS	(8I10)	MAX_G_COUPLES
"LA34F"		CHARACTER	A6/	

LA34F	number of type of independent couples of shells C_t resulting from the distance vector classification performed over the selected shell couples C_1 (see Section G.0.16)	INTEGERS	(8I10)	1 INTEGERS
"NGSHG"		CHARACTER	A6/	
NGSHG(I), I=1,LA34F*INF(37)	array of indices which permit to find the starting point of a star of a given shell couple type C_t (see Section G.0.17)	INTEGERS	(8I10)	LA34F·INF(37)
"NQGSHG"		CHARACTER	A6/	
NQGSHG(I), I=1,LA34F*INF(145)	array containing the ordered list of all the **G** vectors for each shell couple type C_t and each star (see Section G.0.18)	INTEGERS	(8I10)	LA34F·INF(145)
REWIND IO98 (FORT.98) CLOSE(IO98)				

Table G.3: The first column contains variable name as written in the CRYSTAL code, the second column reports the description of the variable meaning, the third column records the type of variable (Fortran style), the fourth column describes the format in which the variables are written in fort.98 file (Fortran style: A characters, I integers, E real numbers in exponential notation), and the last column lists the length of each variable (i.e. the number of data numbers contained in each variable). All the variables not explicitly defined in this table are reported and described in Table G.2.

G.0.1 W1R

Matrix $\mathbf{W} \equiv$ W1R is the conversion matrix from primitive to conventional (crystallographic) cell, defined as

$$\mathbf{W} = \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{pmatrix} \qquad \text{(primitive} \rightarrow \text{crystallographic cell) conversion matrix} \tag{G.1}$$

It is printed in the output file (in row order) through the subroutine COOPRT, as reported in the following:

line	**Lines 1184 and 1287-1288 of SUBROUTINE COOPRT: defined in both4.f**	
1184	WRITE(IOUT,204)((W1R(I,J),J=1,3),I=1,3)	Writing matrix W1R in the output file
1287	204 FORMAT(/' TRANSFORMATION MATRIX PRIMITIVE-CRYSTALLOGRAPHIC CELL'/	
1288	*9F8.4/)	

so that we can find in the output file the components of the matrix $\mathbf{W}$ in equation (G.1) printed as

$$\begin{pmatrix} W_{11} & W_{12} & W_{13} & W_{21} & W_{22} & W_{23} & W_{31} & W_{32} & W_{33} \end{pmatrix} \tag{G.2}$$

If $\mathbf{a}_1^c$, $\mathbf{a}_2^c$, $\mathbf{a}_3^c$ are the direct lattice vectors of the conventional cell, and $\mathbf{a}_1^p$, $\mathbf{a}_2^p$, $\mathbf{a}_3^p$ are the direct lattice vectors of the primitive cell, then the conversion matrix W1R = $\mathbf{W}$ has to be applied on the primitive direct lattice vectors to obtain the conventional direct lattice vectors, as follows

$$\begin{pmatrix} \mathbf{a}_1^c \\ \mathbf{a}_2^c \\ \mathbf{a}_3^c \end{pmatrix} = {}^{\mathbf{t}}\mathbf{W} \begin{pmatrix} \mathbf{a}_1^p \\ \mathbf{a}_2^p \\ \mathbf{a}_3^p \end{pmatrix} \qquad \rightarrow \qquad \mathbf{a}_i^c = \sum_{j=1}^{3} W_{ij}\, \mathbf{a}_j^p \quad \text{with } i = 1,2,3 \tag{G.3}$$

where $\mathbf{a}_i^c = (a_{ix}^c, a_{iy}^c, a_{iz}^c)$ and $\mathbf{a}_i^p = (a_{ix}^p, a_{iy}^p, a_{iz}^p)$, with $i = 1,2,3$. Equation (G.3) can be rewritten more explicitly in components as

$$\begin{pmatrix} a_{1x}^c & a_{1y}^c & a_{1z}^c \\ a_{2x}^c & a_{2y}^c & a_{2z}^c \\ a_{3x}^c & a_{3y}^c & a_{3z}^c \end{pmatrix} = \begin{pmatrix} W_{11} & W_{21} & W_{31} \\ W_{12} & W_{22} & W_{32} \\ W_{13} & W_{23} & W_{33} \end{pmatrix} \begin{pmatrix} a_{1x}^p & a_{1y}^p & a_{1z}^p \\ a_{2x}^p & a_{2y}^p & a_{2z}^p \\ a_{3x}^p & a_{3y}^p & a_{3z}^p \end{pmatrix} \tag{G.4}$$

In the same way as in (G.3), the fractional coordinates $\mathbf{x}^c = (x_1^c, x_2^c, x_3^c)$ of a nucleus in the conventional cell can be obtained from the fractional coordinates $\mathbf{x}^p = (x_1^p, x_2^p, x_3^p)$ of the same nucleus in the primitive cell through the equation

$$\begin{pmatrix} x_1^c \mathbf{a}_1^c \\ x_2^c \mathbf{a}_2^c \\ x_3^c \mathbf{a}_3^c \end{pmatrix} = {}^{\mathbf{t}}\mathbf{W} \begin{pmatrix} x_1^p \mathbf{a}_1^p \\ x_2^p \mathbf{a}_2^p \\ x_3^p \mathbf{a}_3^p \end{pmatrix} \qquad \rightarrow \qquad x_i^c \mathbf{a}_i^c = \sum_{j=1}^{3} W_{ij}\, x_j^p\, \mathbf{a}_j^p \quad \text{with } i = 1,2,3 \tag{G.5}$$

G.0.2 NPOT

Close to the nucleus, the term with $l(l+1)/r^2$ (from the angular part of the Schrödinger equation) is dominant. This would explain why terms with a negative NPOT such as -2 appear: r^{-2}. See Ref. [4].

G.0.3 ICHMOD

ICHMOD is an array with dimension equal to the number of atoms per unit cell, and it containing integers in the position of the array corresponding to the label of the atoms (labels that are the indexed of the atoms given in output order) whose electronic configuration has been modified with the keyword CHEMOD in the input file. If the element i of the array is zero, then the electronic configuration of the i-th atom has not been modified in input. If the element i of the array is k, then the electronic configuration of the i-th atom is the k-th electronic configuration which has been modified in input with the keyword CHEMOD. Thus, the values different from zero in ICHMOD array indentify the order in which the atomic labels of the atoms whose electronic configuration has to be modified are given in the input file.

G.0.4 NCF

After a first classification that reduces the number of shell couples according to the overlap criteria, the selected shell couples in the unit cell are classified as a function of the vector joining them. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples are reorganized in symmetry related subsets. The labels of the first and second shells of these kind of couples are stored in the LA3 and LA4 vectors, respectively. The starting point of each set in LA3 and LA4 (each set is formed by a certain number of couples, the first couple can generate by symmetry all the other couples) is given by the NCF array.
See Section 3.3 Ref. [1].

G.0.5 NSTATG

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets C_s. Each subset C_s is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The vector NSTATG contains the starting point of each couple subset C_s in the irreducible Fock $\mathbf{F}_{irr}$ and density $\mathbf{P}_{irr}$ matrices. In order to simplify the code, a fix allocation and the same starting points of each couple subset C_s for both the irreducible matrices are chosen. The allocation of both irreducible Fock and density matrices corresponds to the size of the irreducible density matrix (the highest criterium is used): so, the irreducible Fock matrix will contain a lot of null terms.
See Section 3.7 Ref. [1].

G.0.6 NSTATFG

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified

as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets C_s. Each subset C_s is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The vector NSTATG contains the starting point of each couple subset C_s in the irreducible Fock $\mathbf{F}_{irr}$ and density $\mathbf{P}_{irr}$ matrices. In order to simplify the code, a fix allocation and the same starting points of each couple subset C_s for both the irreducible matrices are chosen. The allocation of both irreducible Fock and density matrices corresponds to the size of the irreducible density matrix (the highest criterium is used): so, the irreducible Fock matrix will contain a lot of null terms. The vector NSTATFG contains the cumulative number of non-zero elements for each couple subset C_s in the irreducible Fock $\mathbf{F}_{irr}$ matrix.
See Section 3.7 Ref. [1].

G.0.7 IDIME, IDIMF, IDMCOU

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets C_s. Each subset C_s is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The couple sets are then classified in type of couple sets C_{st}: couples with the same C_{st} behave in the same way under symmetry operators effect. For each C_{st} a mother couple set is defined, and corresponds to the first couple set of that C_{st} in the symmetry order. Note that couple sets are classified under geometrical properties only. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of $\mathbf{G}$ for each couple C_t will be considered, forming $(\lambda_i, \lambda_j, \mathbf{G})$ sets. Starting from the classification C_{st} of the couple of shells, the previously selected vectors $\mathbf{G}$ are classified by symmetry, in order to find the irreducible ones for each shell couple set C_{st}. IDIME, IDIMF, IDMCOU are three arrays containing, respectively, the number of $\mathbf{G}$-stars to be considered for each couple subset C_s in the irreducible density $\mathbf{P}_{irr}$, irreducible Fock $\mathbf{F}_{irr}$ and overlap $\mathbf{S}$ matrices.
See Section 3.7 Ref. [1].

G.0.8 LA3

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The labels of the *first* shells of each couple in C_2 are stored in LA3 vector.
See Section 3.3 Ref. [1].

G.0.9 LA4

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The labels of the *second* shells of each couple in C_2 are stored in LA4 vector.
See Section 3.3 Ref. [1].

G.0.10 IROF

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The couple sets are then classified in type of couple sets C_{st}: couples with the same C_{st} behave in the same way under symmetry operators effect. For each C_{st} a mother couple set is defined, and corresponds to the first couple set of that C_{st} in the symmetry order. Note that couple sets are classified under geometrical properties only. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of **G** for each couple C_t will be considered, forming $(\lambda_i, \lambda_j, \mathbf{G})$ sets. Starting from the classification C_{st} of the couple of shells, the previously selected vectors **G** are classified by symmetry, in order to find the irreducible ones for each shell couple set C_{st}. We need, now, to identify the **G** vectors, for each shell couple set C_{st}, able to generate all the star of **G** vectors of all the other symmetry related shell couple sets. The vector NNGI contains the list of the irreducible **G** vectors for each shell couple set C_{st}. The vector IROF indicates the starting point in the NNGI sequence at which the list of the irreducible **G** vectors for a couple set C_s begins.
See Section 3.7 Ref. [1].

G.0.11 NNGI

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The couple sets are then classified in type of couple sets C_{st}: couples with the same C_{st} behave in the same way under symmetry operators effect. For each C_{st} a mother couple set is defined, and corresponds to the first couple set of that C_{st} in the symmetry order. Note that couple sets are classified under geometrical properties only. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of **G** for each couple C_t will be considered, forming $(\lambda_i, \lambda_j, \mathbf{G})$ sets. Starting from the classification C_{st} of the couple of shells, the previously selected vectors **G** are classified by symmetry, in order to find the irreducible ones for each shell couple set C_{st}. We need, now, to identify the **G** vectors, for each shell couple set C_{st}, able to generate all the star of **G** vectors of all the other symmetry related shell couple sets. The vector NNGI contains the list of the irreducible **G** vectors for each shell couple set C_{st}.
See Section 3.6 Ref. [1].

G.0.12 NSHGI

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The couple sets are then classified in type of couple sets C_{st}: couples with the same C_{st} behave in the same way under symmetry operators effect. For each C_{st} a mother couple set is defined, and corresponds to the first couple set of that C_{st} in the symmetry order. Note that couple sets are classified under geometrical properties only. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis

of the chosen thresholds, only a certain number of **G** for each couple C_t will be considered, forming (λ_i, λ_j,**G**) sets. Starting from the classification C_{st} of the couple of shells, the previously selected vectors **G** are classified by symmetry, in order to find the irreducible ones for each shell couple set C_{st}. We need, now, to identify the **G** vectors, for each shell couple set C_{st}, able to generate all the star of **G** vectors of all the other symmetry related shell couple sets. The vector NSHGI contains information on how the irreducible **G** vectors are distributed in each star for each shell couple set C_{st}, i.e. it contains the indices to identify (using the NNGI array) which are the irreducible **G** vectors for a given star of a given couple set C_{st}. The indices in NSHGI array correspond to the number of irreducible **G** vectors for each star of each shell couple set C_{st}.
See Section 3.6 Ref. [1].

G.0.13 NNNC

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets C_s. Each subset C_s is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The couple sets are then classified in type of couple sets C_{st}: couples with the same C_{st} behave in the same way under symmetry operators effect. For each C_{st} a mother couple set is defined, and corresponds to the first couple set of that C_{st} in the symmetry order. Note that couple sets are classified under geometrical properties only. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of **G** for each couple C_t will be considered, forming (λ_i, λ_j,**G**) sets. Starting from the classification C_{st} of the couple of shells, the previously selected vectors **G** are classified by symmetry, in order to find the irreducible ones for each shell couple set C_{st}. The whole density matrix $\mathbf{P}_{red}$ is obtained by applying the point symmetry operators and then the hermiticity to the irreducible matrix $\mathbf{P}_{irr}$. All the couple ($\mu_i, \mu_j, \tilde{\mathbf{g}}$) generated from the irreducible set, with $S^{\tilde{\mathbf{g}}}_{\mu_i \mu_j} > T_{max1}$ (where T_{max1} is the maximum among T_1, T_4 and T_5 TOLINTEG thresholds), are considered, and the NNNC array is built: it points the starting position in the $\mathbf{P}_{irr}$ matrix for each C_1 shell couple, and its size corresponds to the total number of shell couples in the C_1 couple sets. Note that the indexes in NNNC vector points to the starting positions in the $\mathbf{P}_{irr}$ matrix for each C_1 shell couple, but are referred to the construction of the reducible $\mathbf{P}_{red}$ matrix, so the indexes in the vector can assume non-increasing values.
See Section 3.7 Ref. [1].

G.0.14 JPOINT

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the vector joining them. Then, the set of couples, resulting from previous selection, is reduced according to hermiticity. The remaining couples C_2 are reorganized in symmetry related subsets. Each subset is formed by a certain number of couples, the first couple can generate by symmetry all the other couples. The vector JPOINT relates each couple in C_1 to the index of the subset it belongs in the symmetry classification.
See Section 3.3 Ref. [1].

G.0.15 LA34V

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the distance vector joining them. LA34F is the number of type of independent couples C_t of shells (λ_i, λ_j) found by that classification. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold):

on the basis of the chosen thresholds, only a certain number of **G** for each couple C_t will be considered, forming (λ_i, λ_j,**G**) sets. In calculating the integrals, we will deal with the single C_1 couples selected by the overlap criteria. As the classification of the (λ_i, λ_j,**G**) is performed per couple shell type, not for each single couple C_1, we must be able to go directly from the couple (λ_i, λ_j,**G**) to its **G** organization, skipping all passages leading a general couple to the reference couple. On this purpose, LA34X and LA34V vectors have been calculated. For every couple of shells C_1 LA34X shows the number of **G**-stars, while LA34V contains the starting point in NQGSHG for the associated **G** vectors.
See Section 3.5 Ref. [1].

G.0.16 LA34F

After a first classification that reduces the number of shell couples according to the overlap criteria, the selected shell couples C_1 in the unit cell are classified as a function of the vector joining them. LA34F is the number of type of independent couples of shells C_t resulting from the distance vector classification.
See Section 3.2 Ref. [1].

G.0.17 NGSHG

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the distance vector joining them. LA34F is the number of type of independent couples of shells C_t found by that classification. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of **G** for each couple will be considered. For each couple type we need to know the involved **G** vectors, their organization in stars and their label in the initial definition sequence. This information is stored in two vectors: NQGSHG vector contains the ordered list of all the **G** vectors for each shell couple type C_t and each star, NGSHG vector permits to find the starting point of a star of a given shell couple type C_t.
See Section 3.5 Ref. [1].

G.0.18 NQGSHG

After a first classification that reduces the number of shell couples according to the overlap criteria (C_1: shell couples selected on overlap criteria), the selected shell couples in the unit cell are classified as a function of the distance vector joining them. LA34F is the number of type of independent couples of shells C_t found by that classification. For each couple C_t, there will be a maximum interacting distance, beyond which the considered shells do not interact anymore (overlap less than a threshold): on the basis of the chosen thresholds, only a certain number of **G** for each couple will be considered. For each couple type we need to know the involved **G** vectors, their organization in stars and their label in the initial definition sequence. NQGSHG vector contains the ordered list of all the **G** vectors for each shell couple type C_t and each star. Note that all labels in NQGSHG refer to the labels attributed to each **G** vector in its initial definition (same labels for the order of **G** vectors indices and coordinates in LG(I,J) and XG(I,J), I=1,3, J=1,INF(79) matrices).
See Section 3.5 Ref. [1].

G.0.19 SUBROUTINE RWRITF

line	**SUBROUTINE RWRITF - defined in libxc_com.f**
1086	SUBROUTINE RWRITF(JOUT,A,N)
	USE NUMBERS
	IMPLICIT REAL(FLOAT) (A-H,O-Z)
	DIMENSION A(N)
	WRITE(JOUT,100)A
	100 FORMAT(1P,4E21.13)
	RETURN
1093	END

References for Appendix G

[1] Raffaella Demichelis and Roberto Dovesi, Discussion about some parts of the CRYSTAL code - Work in progress, version 1.0.1, 8 October, 2008

[2] Jacques Desmarais, NORMS

[3] R. Dovesi et al., CRYSTAL17 User's Manual, 2018

[4] P. J. Hay and W. R. Wadt, Ab initio effective core potentials for molecular calculations. Potentials for the transition metal atoms Sc to Hg, J. Chem. Phys. 82, 270, 1985, and discussion with Klaus Doll

Bibliography

[1] M. D. Towler, A. Zupan, M. Causà, Density functional theory in periodic systems using local Gaussian basis sets, Comput. Phys. Commun. 98 (1996) 181–205.

[2] M. E. Tuckerman, Ab initio molecular dynamics: basic concepts, current trends and novel applications, J. Phys.: Condens. Matter 14 (2002) R1297.

[3] T. D. Kühne, Second generation Car-Parrinello molecular dynamics, Wiley Interdiscip. Rev. Comput. Mol. Sci. 4 (2014) 391–406.

[4] P. Carloni, U. Rothlisberger, M. Parrinello, The Role and Perspective of Ab Initio Molecular Dynamics in the Study of Biological Systems, Acc. Chem. Res. 35 (2002) 455–464.

[5] T. Todorova, A. P. Seitsonen, J. Hutter, I.-F. W. Kuo, C. J. Mundy, Molecular Dynamics Simulation of Liquid Water: Hybrid Density Functionals, J. Phys. Chem. B 110 (2006) 3685–3691.

[6] C. Zhang, D. Donadio, F. Gygi, G. Galli, First Principles Simulations of the Infrared Spectrum of Liquid Water Using Hybrid Density Functionals, J. Chem. Theory Comput. 7 (2011) 1443–1449.

[7] R. A. DiStasio Jr., B. Santra, Z. Li, X. Wu, R. Car, The individual and collective effects of exact exchange and dispersion interactions on the ab initio structure of liquid water, J. Chem. Phys. 141 (2014) 084502.

[8] F. Ambrosio, G. Miceli, A. Pasquarello, Structural, Dynamical, and Electronic Properties of Liquid Water: A Hybrid Functional Study, J. Phys. Chem. B 120 (2016) 7456–7470.

[9] B. Santra, R. A. D. Jr., F. Martelli, R. Car, Local structure analysis in ab initio liquid water, Mol. Phys. 113 (2015) 2829–2841.

[10] E. Sevgen, F. Giberti, H. Sidky, J. K. Whitmer, G. Galli, F. Gygi, J. J. de Pablo, Hierarchical Coupling of First-Principles Molecular Dynamics with Advanced Sampling Methods, J. Chem. Theory Comput. 14 (2018) 2881–2888.

[11] S. Mandal, J. Debnath, B. Meyer, N. N. Nair, Enhanced sampling and free energy calculations with hybrid functionals and plane waves for chemical reactions, J. Chem. Phys. 149 (2018) 144113.

[12] S. Mandal, N. N. Nair, Efficient computation of free energy surfaces of chemical reactions using ab initio molecular dynamics with hybrid functionals and plane waves, J. Comput. Chem. 41 (2020) 1790–1797.

[13] S. Chawla, G. A. Voth, Exact exchange in ab initio molecular dynamics: An efficient plane-wave based algorithm, J. Chem. Phys. 108 (1998) 4697–4700.

[14] S. Mandal, N. N. Nair, Speeding-up ab initio molecular dynamics with hybrid functionals using adaptively compressed exchange operator based multiple timestepping, J. Chem. Phys. 151 (2019) 151102.

[15] M. Guidon, F. Schiffmann, J. Hutter, J. VandeVondele, Ab initio molecular dynamics using hybrid density functionals, J. Chem. Phys. 128 (2008) 214104.

[16] H.-Y. Ko, J. Jia, B. Santra, X. Wu, R. Car, R. A. DiStasio Jr., Enabling Large-Scale Condensed-Phase Hybrid Density Functional Theory Based Ab Initio Molecular Dynamics. I. Theory, Algorithm, and Performance, J. Chem. Theory Comput. 16 (2020) 3757–3785.

[17] J. P. Heremans, M. S. Dresselhaus, L. E. Bell, D. T. Morelli, When thermoelectrics reached the nanoscale, Nat. Nanotech. 8 (2013) 471–473.

[18] B. Champagne, J.-M. André, Determination of ab initio polarizabilities of polymers: Application to polyethylene and polysilane, Int. J. Quantum Chem. 42 (1992) 1009–1024.

[19] P. Otto, F. L. Gu, J. Ladik, Calculation of ab initio dynamic hyperpolarizabilities of polymers, J. Chem. Phys. 110 (1999) 2717–2726.

[20] T.-Y. Wu, On the nature of theories of irreversible processes, Int. J. Theor. Phys. 2 (1969) 325–343.

[21] R. Kubo, Statistical-mechanical theory of irreversible processes. I. General theory and simple applications to magnetic and conduction problems, J. Phys. Soc. Jpn. 12 (1957) 570–586.

[22] G. Sansone, A. Ferretti, L. Maschio, Ab initio electronic transport and thermoelectric properties of solids from full and range-separated hybrid functionals, J. Chem. Phys. 147 (2017) 114101.

[23] R. Orlando, M. Delle Piane, I. J. Bush, P. Ugliengo, M. Ferrabone, R. Dovesi, A new massively parallel version of CRYSTAL for large systems on high performance computing architectures, J. Comput. Chem. 33 (2012) 2276–2284.

[24] N. L. Doltsinis, D. Marx, First principles molecular dynamics involving excited states and nonadiabatic transitions, J. Theor. Comput. Chem. 01 (2002) 319–349.

[25] J. C. Tully, Mixed quantum-classical dynamics: mean-field and surface-hopping in *Classical and Quantum Dynamics in Condensed Phase Simulations*, 1998, pp. 489–514.

[26] G. E. P. Box, M. E. Muller, A note on the generation of random normal deviates, Ann. Math. Stat. 29 (1958) 610–611.

[27] M. Fixman, Classical statistical mechanics of constraints: A theorem and application to polymers, Proc. Natl. Acad. Sci. 71 (1974) 3050–3053.

[28] H. W. Graben, J. R. Ray, Unified treatment of adiabatic ensembles, Phys. Rev. A 43 (1991) 4100–4103.

[29] V. Kuzkin, On angular momentum balance for particle systems with periodic boundary conditions, ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 95 (2014) 1290–1295.

[30] R. P. Feynman, Forces in molecules, Phys. Rev. 56 (1939) 340–343.

[31] M. Levy, Universal variational functionals of electron densities, first-order density matrices, and natural spin-orbitals and solution of the v-representability problem, Proc. Natl. Acad. Sci. 76 (1979) 6062–6065.

[32] E. H. Lieb, Density functionals for Coulomb systems, Int. J. Quantum Chem. 24 (1983) 243–277.

[33] E. Noether, Invariante variationsprobleme, Nachr. Ges. Wiss. Göttingen, Mathematisch-Physikalische Klasse 1918 (1918) 235–257.

[34] H. F. Trotter, On the product of semi-groups of operators, Proc. Amer. Math. Soc. 10 (1959) 545–551.

[35] M. Suzuki, Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems, Commun. Math. Phys. 51 (1976) 183–190.

[36] B. A. Dubrovin, A. T. Fomenko, S. P. Novikov, Modern Geometry - Methods and Applications. Part I. The Geometry of Surfaces, Transformation Groups, and Fields, Springer Science, 1984.

[37] M. E. Tuckerman, B. J. Berne, G. J. Martyna, Reply to comment on: Reversible multiple time scale molecular dynamics, J. Chem. Phys. 99 (1993) 2278–2279.

[38] M. E. Tuckerman, Y. Liu, G. Ciccotti, G. J. Martyna, Non-Hamiltonian molecular dynamics: Generalizing Hamiltonian phase space principles to non-Hamiltonian systems, J. Chem. Phys. 115 (2001) 1678–1702.

[39] W. Swope, H. Andersen, P. Berens, K. Wilson, A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters, J. Chem. Phys. 76 (1982) 637–649.

[40] L. Verlet, Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1967) 98–103.

[41] J. R. Ray, H. Zhang, Correct microcanonical ensemble in molecular dynamics, Phys. Rev. E 59 (1999) 4781–4785.

[42] M. J. Uline, D. W. Siderius, D. S. Corti, On the generalized equipartition theorem in molecular dynamics ensembles and the microcanonical thermodynamics of small systems, J. Chem. Phys. 128 (2008) 124301.

[43] E. M. Pearson, T. Halicioglu, W. A. Tiller, Laplace-transform technique for deriving thermodynamic equations from the classical microcanonical ensemble, Phys. Rev. A 32 (1985) 3030–3039.

[44] L. V. Woodcock, Isothermal molecular dynamics calculations for liquid salts, Chem. Phys. Lett. 10 (1971) 257–261.

[45] H. C. Andersen, Molecular dynamics simulations at constant pressure and/or temperature, J. Chem. Phys. 72 (1980) 2384–2393.

[46] S. Nosé, A unified formulation of the constant temperature molecular dynamics methods, J. Chem. Phys. 81 (1984) 511–519.

[47] S. Nosé, Constant Temperature Molecular Dynamics Methods, Prog. Theor. Phys. Supplement 103 (1991) 1–46.

[48] S. Nosé, A molecular dynamics method for simulations in the canonical ensemble, Mol. Phys. 52 (1984) 255–268.

[49] J. M. Haile, S. Gupta, Extensions of the molecular dynamics simulation method. II. isothermal systems, J. Chem. Phys. 79 (1983) 3067–3076.

[50] D. J. Evans, W. G. Hoover, B. H. Failor, B. Moran, A. J. C. Ladd, Nonequilibrium molecular dynamics via Gauss's principle of least constraint, Phys. Rev. A 28 (1983) 1016–1021.

[51] S. C. Harvey, R. K.-Z. Tan, T. E. Cheatham III, The flying ice cube: Velocity rescaling in molecular dynamics leads to violation of energy equipartition, J. Comput. Chem. 19 (1998) 726–740.

[52] E. Braun, S. M. Moosavi, B. Smit, Anomalous effects of velocity rescaling algorithms: The flying ice cube effect revisited, J. Chem. Theory Comput. 14 (2018) 5262–5272.

[53] G. Bussi, D. Donadio, M. Parrinello, Canonical sampling through velocity rescaling, J. Chem. Phys. 126 (2007) 014101.

[54] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, J. R. Haak, Molecular dynamics with coupling to an external bath, J. Chem. Phys. 81 (1984) 3684–3690.

[55] I. Fukuda, S. Queyroy, H. Nakamura, A robust, symmetric operator-composition integrator for the Berendsen temperature-control molecular dynamics equation, J. Phys. Soc. Japan 89 (2020) 064004.

[56] D. S. Kleinerman, C. Czaplewski, A. Liwo, H. A. Scheraga, Implementations of Nosé–Hoover and Nosé–Poincaré thermostats in mesoscopic dynamic simulations with the united-residue model of a polypeptide chain, J. Chem. Phys. 128 (2008) 245103.

[57] T. Morishita, Fluctuation formulas in molecular-dynamics simulations with the weak coupling heat bath, J. Chem. Phys. 113 (2000) 2976–2982.

[58] W. G. Hoover, Canonical dynamics: Equilibrium phase-space distributions, Phys. Rev. A 31 (1985) 1695–1697.

[59] D. J. Evans, B. L. Holian, The Nosé-Hoover thermostat, J. Chem. Phys. 83 (1985) 4069–4074.

[60] S. G. Itoh, T. Morishita, H. Okumura, Decomposition-order effects of time integrator on ensemble averages for the Nosé-Hoover thermostat, J. Chem. Phys. 139 (2013) 064103.

[61] H. Ishida, A. Kidera, Constant temperature molecular dynamics of a protein in water by high-order decomposition of the Liouville operator, J. Chem. Phys. 109 (1998) 3276–3284.

[62] G. S. Ezra, Reversible measure-preserving integrators for non-Hamiltonian systems, J. Chem. Phys. 125 (2006) 034104.

[63] K. Cho, J. D. Joannopoulos, L. Kleinman, Constant-temperature molecular dynamics with momentum conservation, Phys. Rev. E 47 (1993) 3145–3151.

[64] G. J. Martyna, M. L. Klein, M. Tuckerman, Nosé.Hoover chains: The canonical ensemble via continuous dynamics, J. Chem. Phys. 97 (1992) 2635–2643.

[65] D. Kusnezov, A. Bulgac, W. Bauer, Canonical ensembles from chaos, Ann. Phys. 204 (1990) 155–185.

[66] I. P. Hamilton, Modified Nosé-Hoover equation for a one-dimensional oscillator: Enforcement of the virial theorem, Phys. Rev. A 42 (1990) 7467–7470.

[67] R. G. Winkler, Extended-phase-space isothermal molecular dynamics: Canonical harmonic oscillator, Phys. Rev. A 45 (1992) 2250–2255.

[68] M. Suzuki, General Nonsymmetric Higher-Order Decomposition of Exponential Operators and Symplectic Integrators, J. Phys. Soc. Jpn. 61 (1992) 3015–3019.

[69] M. Suzuki, Decomposition formulas of exponential operators and Lie exponentials with some applications to quantum mechanics and statistical physics, J. Math. Phys. 26 (1985) 601–612.

[70] M. Suzuki, General theory of fractal path integrals with applications to many-body theories and statistical physics, J. Math. Phys. 32 (1991) 400–407.

[71] H. Yoshida, Construction of higher order symplectic integrators, Phys. Lett. A 150 (1990) 262–268.

[72] H. Goldstein, C. Poole, J. Safko, Classical Mechanics, 3rd ed., Am. J. Phys. 70 (2002) 782–783.

[73] S. Nosé, An extension of the canonical ensemble molecular dynamics method, Mol. Phys. 57 (1986) 187–191.

[74] M. Parrinello, A. Rahman, Crystal Structure and Pair Potentials: A Molecular-Dynamics Study, Phys. Rev. Lett. 45 (1980) 1196–1199.

[75] S. Nosé, M. Klein, Constant pressure molecular dynamics for molecular systems, Mol. Phys. 50 (1983) 1055–1076.

[76] M. Ferrario, J. Ryckaert, Constant pressure-constant temperature molecular dynamics for rigid and partially rigid molecular systems, Mol. Phys. 54 (1985) 587–603.

[77] G. C. Simone Melchionna, B. L. Holian, Hoover NPT dynamics for systems varying in shape and size, Mol. Phys. 78 (1993) 533–544.

[78] M. Ferrario, Thermodynamic Constraints, in Computer Simulation in Chemical Physics (p. 153-171) by M. P. Allen, and D. J. Tildesley, 1st Edition, Springer, Dordrecht, 1993.

[79] G. J. Martyna, D. J. Tobias, M. L. Klein, Constant pressure molecular dynamics algorithms, J. Chem. Phys. 101 (1994) 4177–4189.

[80] J. Jellinek, Dynamics for nonconservative systems: ergodicity beyond the microcanonical ensemble, J. Phys. Chem. 92 (1988) 3163–3173.

[81] J. Jellinek, R. S. Berry, Generalization of Nosé's isothermal molecular dynamics, Phys. Rev. A 38 (1988) 3069–3072.

[82] E. Forest, R. D. Ruth, Fourth-order symplectic integration, Phys. D: Nonlinear Phenom. 43 (1990) 105–117.

[83] M. Tuckerman, B. J. Berne, G. J. Martyna, Reversible multiple time scale molecular dynamics, J. Chem. Phys. 97 (1992) 1990–2001.

[84] A. Sergi, M. Ferrario, D. Costa, Reversible integrators for basic extended system molecular dynamics, Mol. Phys. 97 (1999) 825–832.

[85] N. Wiener, Generalized harmonic analysis, Acta Math. 55 (1930) 117 – 258.

[86] A. Khintchine, Korrelationstheorie der stationären stochastischen Prozesse, Math. Ann. 109 (1934) 604–615.

[87] K. Wendler, M. Brehm, F. Malberg, B. Kirchner, L. Delle Site, Short Time Dynamics of Ionic Liquids in AIMD-Based Power Spectra, J. Chem. Theory Comput. 8 (2012) 1570–1579.

[88] J. D. Bernal, R. H. Fowler, A Theory of Water and Ionic Solution, with Particular Reference to Hydrogen and Hydroxyl Ions, J. Chem. Phys. 1 (2004) 515–548.

[89] L. Pauling, The Structure and Entropy of Ice and of Other Crystals with Some Randomness of Atomic Arrangement, J. Am. Chem. Soc. 57 (1935) 2680–2684.

[90] Howe, R., The possible ordered structures of ice Ih, J. Phys. Colloques 48 (1987) C1–599–C1–604.

[91] C. Pisani, S. Casassa, P. Ugliengo, Proton-ordered ice structures at zero pressure. a quantum-mechanical investigation, Chem. Phys. Lett. 253 (1996) 201–208.

[92] C. Ribaldone, S. Casassa, Born-Oppenheimer Molecular Dynamics with Linear Combination of Atomic Orbitals and Hybrid Functionals for Condensed Matter Simulations Made Possible. Theory and Performance for the Microcanonical and Canonical Ensembles, J. Chem. Theory Comput. accepted manuscript.

[93] J. P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, Phys. Rev. Lett. 77 (1996) 3865–3868.

[94] S. Grimme, Semiempirical gga-type density functional constructed with a long-range dispersion correction, J. Comput. Chem. 27 (2006) 1787–1799.

[95] S. Grimme, J. Antony, S. Ehrlich, H. Krieg, A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, J. Chem. Phys. 132 (2010) 154104.

[96] M. Sprik, J. Hutter, M. Parrinello, Ab initio molecular dynamics simulation of liquid water: Comparison of three gradient-corrected density functionals, J. Chem. Phys. 105 (1996) 1142–1152.

[97] E. Schwegler, J. C. Grossman, F. Gygi, G. Galli, Towards an assessment of the accuracy of density functional theory for first principles simulations of water. II, J. Chem. Phys. 121 (2004) 5400–5409.

[98] R. Mills, Self-diffusion in normal and heavy water in the range 1-45°, J. Phys. Chem. 77 (1973) 685–688.

[99] K. Krynicki, C. D. Green, D. W. Sawyer, Pressure and temperature dependence of self-diffusion in water, Faraday Discuss. Chem. Soc. 66 (1978) 199–208.

[100] P. L. Silvestrelli, M. Parrinello, Structural, electronic, and bonding properties of liquid water from first principles, J. Chem. Phys. 111 (1999) 3572–3580.

[101] A. K. Soper, The Radial Distribution Functions of Water as Derived from Radiation Total Scattering Experiments: Is There Anything We Can Say for Sure?, ISRN Physical Chemistry 2013 (2013) 1–67.

[102] A. D. Becke, Density functional thermochemistry. III. The role of exact exchange, J. Chem. Phys. 98 (1993) 5648–5652.

[103] C. Adamo, V. Barone, Toward reliable density functional methods without adjustable parameters: the PBE0 model, J. Chem. Phys. 110 (1999) 6158–6170.

[104] C. Gatti, V. R. Saunders, C. Roetti, Crystal field effects on the topological properties of the electron density in molecular crystals: the case of urea, J. Chem. Phys. 101 (1994) 10686–10696.

[105] A. D. Becke, A multicenter numerical integration scheme for polyatomic molecules, J. Chem. Phys. 88 (1988) 2547–2553.

[106] M. D. Towler, A. Zupan, M. Causà, Density functional theory in periodic systems using local gaussian basis sets, Comput. Phys. Commun. 98 (1996) 181–205.

[107] H. J. Monkhorst, J. D. Pack, Special points for Brillouin-zone integrations, Phys. Rev. B 13 (1976) 5188–5192.

[108] C. G. Broyden, The convergence of a class of double-rank minimization algorithms 1. General considerations, IMA J. Appl. Math. 6 (1970) 76–90.

[109] C. G. Broyden, The convergence of a class of double-rank minimization algorithms 2. The new algorithm, IMA J. Appl. Math. 6 (1970) 222–231.

[110] R. Fletcher, A new approach to variable metric algorithms, Comput. J. 13 (1970) 317–322.

[111] D. Goldfarb, A family of variable-metric methods derived by variational means, Math. Comput. 24 (1970) 23–26.

[112] D. Shanno, Conditioning of quasi-Newton methods for function minimization, Math. Comput. 24 (1970) 647–656.

[113] A. Erba, J. K. Desmarais, S. Casassa, B. Civalleri, L. Donà, I. J. Bush, B. Searle, L. Maschio, L. Edith-Daga, A. Cossard, C. Ribaldone, E. Ascrizzi, N. L. Marana, J.-P. Flament, B. Kirtman, CRYSTAL23: A Program for Computational Solid State Physics and Chemistry, J. Chem. Theory Comput. 19 (2023) 6891–6932.

[114] R. Dovesi, V. R. Saunders, C. Roetti, R. Orlando, C. M. Zicovich-Wilson, F. Pascale, B. Civalleri, K. Doll, N. M. Harrison, I. J. Bush, P. D'Arco, M. Llunell, M. Causá, Y. Nöel, L. Maschio, A. Erba, M. Rérat, S. Casassa, B. G. Searle, J. Desmarais, online crystal23 user's manual, www.crystal.unito.it/include/manuals/crystal23.pdf.

[115] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, Numerical Recipes in FORTRAN 77: The Art of Scientific Computing, 2nd Edition, Cambridge University Press, 1992.

[116] E. Polak, in: Computational Methods in Optimization: A Unified Approach, Vol. 77 of Mathematics in Science and Engineering, Elsevier, 1971, pp. iv–xvii, 1–329.

[117] D. Jacobs, The state of the art in numerical analysis, Math. Comput. 32 (1977) 1325.

[118] E. Polak, in: Computational Methods in Optimization: A Unified Approach, Vol. 77 of Mathematics in Science and Engineering, Elsevier, 1971, p. 56ff.

[119] H. B. Schlegel, Optimization of equilibrium geometries and transition structures, J. Comput. Chem. 3 (1982) 214–218.

[120] R. Dovesi, A. Erba, R. Orlando, C. Zicovich-Wilson, B. Civalleri, L. Maschio, M. Rérat, S. Casassa, J. Baima, S. Salustro, B. Kirtman, Quantum-mechanical condensed matter simulations with crystal, Wiley Interdiscip. Rev. Comput. Mol. Sci. 8 (2018) 1–36.

[121] R. Dovesi, V. R. Saunders, C. Roetti, R. Orlando, C. M. Zicovich-Wilson, F. Pascale, B. Civalleri, K. Doll, N. M. Harrison, I. J. Bush, P. D'Arco, M. Llunel, M. Causà, Y. Noel, L. Maschio, A. Erba, M. Rérat, S. Casassa, online crystal17 User's Manual, https://www.crystal.unito.it/manuals/crystal17.pdf (2018).

[122] J. R. Beeler, Radiation effects computer experiments, Defects in Solids, Elsevier, Oxford, 1983, p. 27.

[123] H. Jónsson, G. Mills, W. Jacobsen, Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions, in *Classical and Quantum Dynamics in Condensed Phase Simulations*, World Scientific, 1998.

[124] J. H. A. Hagelaar, E. Bitzek, C. F. J. Flipse, P. Gumbsch, Atomistic simulations of the formation and destruction of nanoindentation contacts in tungsten, Phys. Rev. B 73 (2006) 045425.

[125] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, P. Gumbsch, Structural relaxation made simple, Phys. Rev. Lett. 97 (2006) 170201.

[126] J. Guénolé, W. G. Nöhring, A. Vaid, F. Houllé, Z. Xie, A. Prakash, E. Bitzek, Assessment and optimization of the fast inertial relaxation engine (fire) for energy minimization in atomistic simulations and its implementation in lammps, Comput. Mater. Sci. 175 (2020) 109584.

[127] C. Ribaldone, S. Casassa, Fast inertial relaxation engine in the CRYSTAL code, AIP Advances 12 (2022) 015323.

[128] P. Koskinen, S. Malola, H. Häkkinen, Self-passivating edge reconstructions of graphene, Phys. Rev. Lett. 101 (2008) 115502.

[129] J. A. Flores-Livas, A. Sanna, E. K. Gross, High temperature superconductivity in sulfur and selenium hydrides at high pressure, Eur. Phys. J. B 89 (2016) 1–7.

[130] M. Walter, M. Moseler, Ligand-protected gold alloy clusters: doping the superatom, J. Phys. Chem. C 113 (2009) 15834–15837.

[131] J. A. Flores-Livas, M. Amsler, T. J. Lenosky, L. Lehtovaara, S. Botti, M. A. L. Marques, S. Goedecker, High-pressure structures of disilane and their superconducting properties, Phys. Rev. Lett. 108 (2012) 117004.

[132] F. Shuang, P. Xiao, R. Shi, F. Ke, Y. Bai, Influence of integration formulations on the performance of the fast inertial relaxation engine (fire) method, Comput. Mater. Sci. 156 (2019) 135–141.

[133] D. Sheppard, R. Terrell, G. Henkelman, Optimization methods for finding minimum energy paths, J. Chem. Phys. 128 (2008) 134106.

[134] B. Schaefer, S. Alireza Ghasemi, S. Roy, S. Goedecker, Stabilized quasi- Newton optimization of noisy potential energy surfaces, J. Chem. Phys. 142 (2015) 034112.

[135] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1995) 1–19.

[136] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, E. Lindahl, GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers, SoftwareX 1-2 (2015) 19–25.

[137] J. Stadler, R. Mikulla, H. Trebin, IMD: a software package for molecular dynamics studies on parallel computers, Int. J. Mod. Phys. C 08 (1997) 1131–1140.

[138] W. Smith, C. Yong, P. Rodger, DL_POLY: application to molecular simulation, Mol. Simul. 28 (2002) 385–471.

[139] S. Chill, M. Welborn, R. Terrell, L. Zhang, J. Berthet, A. Pedersen, H. Jónsson, G. Henkelman, EON: software for long time simulations of atomic scale systems, Model. Simul. Mat. Sci. Eng. 22 (2014) 055002.

[140] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dulak, J. Friis, M. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K. W. Jacobsen, The atomic simulation environment-a python library for working with atoms, J. Phys. Condens. Matter 29 (2017) 273002.

[141] R. Sure, S. Grimme, Corrected small basis set Hartree-Fock method for large systems, J. Comput. Chem. 34 (2013) 1672–1685.

[142] J. Brandenburg, S. Grimme, Dispersion corrected Hartree-Fock and density functional theory for organic crystal structure prediction, Top. Curr. Chem. 345 (2013) 1–23.

[143] M. Cutini, B. Civalleri, M. Corno, R. Orlando, J. Brandenburg, L. Maschio, P. Ugliengo, Assessment of different quantum mechanical methods for the prediction of structure and cohesive energy of molecular crystals, J. Chem. Theory Comput. 12 (2016) 3340–3352.

[144] B. Civalleri, C. M. Zicovich-Wilson, L. Valenzano, P. Ugliengo, B3LYP augmented with an empirical dispersion term (B3LYP-D*) as applied to molecular crystals, CrystEngComm 10 (2008) 405–410.

[145] A. V. Krukau, O. A. Vydrov, A. F. Izmaylov, G. E. Scuseria, Influence of the exchange screening parameter on the performance of screened hybrid functionals, J. Chem. Phys. 125 (2006) 224106.

[146] T. Stedman, L. M. Woods, Transport theory within a generalized Boltzmann equation for multiband wave packets, Phys. Rev. Res. 2 (2020) 033086.

[147] J. A. Pople, R. Krishnan, H. B. Schlegel, J. S. Binkley, Derivative studies in Hartree-Fock and Møller-Plesset theories, Int. J. Quantum Chem. 16 (S13) (1979) 225–241.

[148] P. Steneteg, I. A. Abrikosov, V. Weber, A. M. N. Niklasson, Wavefunction extended lagrangian Born-Oppenheimer molecular dynamics, Phys. Rev. B 82 (2010) 075110.

[149] X. Pan, R. Van, E. Epifanovsky, J. Liu, J. Pu, K. Nam, Y. Shao, Accelerating ab initio quantum mechanical and molecular mechanical (QM/MM) molecular dynamics simulations with multiple time step integration and a recalibrated semiempirical QM/MM Hamiltonian, J. Phys. Chem. B 126 (2022) 4226–4235.

[150] H.-S. Lee, M. E. Tuckerman, Dynamical properties of liquid water from ab initio molecular dynamics performed in the complete basis set limit, J. Chem. Phys. 126 (2007) 164501.

[151] J. C. Maxwell, V. Illustrations of the dynamical theory of gases. – Part I. On the motions and collisions of perfectly elastic spheres, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 19 (1860) 19–32.

[152] D. C. Wallace, G. K. Straub, Ensemble corrections for the molecular-dynamics ensemble, Phys. Rev. A (1983) 2201–2205.

[153] M. P. Allen, D. J. Tildesley, Computer simulation of liquids, 2nd Edition, Oxford University Press, 2017.