Adaptive residual refinement in an RBF finite difference scheme for 2D time-dependent problems

Adaptive residual refinement in a RBF finite difference scheme for 2D time-dependent problems

G. Garmanjani^1, **M. Esmaeilbeigi**^{1}, **R. Cavoretto**^{2,3}
^{1}Department of Mathematics, Malayer University, Malayer 65719-95863, Iran
^{2}Department of Mathematics "G. Peano", University of Torino, via Carlo Alberto 10, 10123 Torino, Italy
^{3}Member of the INdAM Research group GNCS

Abstract

The current study introduces a fast and accurate method based on a novel adaptive meshfree technique to solve time-dependent partial differential equations with solutions representing high gradients or quick changes in several local areas of the domain. Utilizing uniform grids for these problems is prohibitive computationally since the solution reaches the singularity. This study aims to suggest an adaptive strategy to produce a suitable and cost-effective irregular node refinement. For this purpose, a dynamic algorithm is proposed that finds areas with quick changes and applies a local node adaptive approach merely in those nearly singular areas. Additionally, within this algorithm, unlike Kansa technique, the radial basis function collocation technique was mixed with a finite difference scheme. According to this approach, in place of using the adaptive algorithm on the complete domain of the problem, it can be used only on time steps. Therefore, we need only to solve small systems of linear equations on each time step instead of large systems on the entire domain. Besides performing a stability analysis of the numerical scheme, the new algorithm is tested on parabolic (heat equation) and hyperbolic (wave equation) PDEs over regular and irregular two-dimensional domains. The attained results prove the accuracy and effectiveness of the proposed technique. Especially, our computational method is able to reduce the nodes in the domain with no impairment in terms of accuracy, thus turning out to be effective in the localization of oscillations owing to sharp gradients in the solution.

Keywords: partial differential equations, meshless methods, RBF collocation, adaptive distribution
2010 MSC: 65M70, 65M06, 65M50, 65Y20

1. Introduction

Time-dependent PDEs (partial differential equations) [44] can explain phenomena in different areas, like economics, biology, engineering, physics, etc. Supported by the development of the computer, the numerical methods [3] have become effective techniques to study these types of problems. RBF (radial basis function) approaches were admired for their ease and simple operation in multivariate approximation of scattered data [2, 8, 16, 32, 49]. These positive features made the RBFs particularly suitable for solving mathematically time-dependent PDEs (see [1, 27, 29, 30, 31, 37, 43, 45, 48]). Comparing the latter with low-order approaches like finite differences (FDs), finite elements, and finite volumes, RBF-based techniques provide several benefits such as no need for a triangulation or mesh, dimension independence, simple operation, and no stair casing or polygonizing for boundaries. Furthermore, high-order or spectral convergence is attained depending on how the RBFs are selected [9, 21].

Lots of evolutionary problems involving time-dependent PDEs have solutions with steep transitions like sharp wave fronts or boundary layers. Previously, it was mostly recognized that adaptive mesh-based approaches are able to solve the sharp transitions to a suitable level of accuracy with no need to use numerous mesh points [23]. Such approaches, by continuous relocation of mesh points to track properties to the calculated solution, offer an ideal adaptive tactic to solve these types of problems. A strategy of adaptive refinement aims to create an optimal point distribution providing the necessary precision and the least freedom level. Within an adaptive algorithm, a larger number of nodes are generally added only on the domain parts in which more details or steep variations are present, whereas simultaneously a reduced (but adequate) amount of nodes is maintained in smooth areas. Indeed, it is known that node positions have an important role since further nodes are needed in a very localized area, as well as for the classical problem of interpolation stability, determined by the Lebesgue constant and demonstrated over the Gibbs and Runge phenomena [6, 35].

For time-dependent problems, adaptive approaches are of particular interest as the fact of facing dynamic problems may lead to occur quick movement of the solution in some specific regions. Usually, two methods exist for modification of grids in time: the former consists in transferring the grid with the fronts [25], the latter aims to localize the refinement simply where required. Here, at all time steps, the refined grid may have to reflect the problem dynamics [5, 10]. In addition, some scientists prefer a hybrid approach of local mesh refinement and moving meshes [4].

These attractive properties motivated several scientists in various research fields, and thus significant effort was performed to integrate an adaptive algorithm with RBF-based approaches. At first, some adaptive methods based on greedy algorithms were proposed in [40]. Later, in the work [47] adaptive RBF algorithms were used to choose the location of the collocation points, while in [52] a dynamic adaptive method for time-based PDEs was introduced. Moreover, in the paper [28] a dynamic node adaptive approach for almost singular problems on large domains was developed. Different RBF schemes based on residual subsampling methods were then established for solving interpolation and boundary value problems in [19, 26]. Similar adaptive techniques were also proposed to solve elliptic PDEs [13, 14, 15]. More lately, various scientists suggested several adaptive approaches for solving time-dependent PDEs (see e.g. [18, 20, 38, 41, 42] and references therein).

In this study, we solve time-dependent PDEs whose solution shows fast changes or high gradients in several local areas of the domain. More precisely, a novel adaptive meshfree technique is proposed with a RBF-FD approximation scheme used to solve the time-dependent PDEs through regular or irregular 2D domains. The main goal of the current study is to develop a dynamic algorithm detecting areas with fast changes and applying a local node adaptive algorithm only in the nearly singular areas. Though numerous adaptive techniques were introduced for detecting nearly singular regions, owing to the ill-conditioning and to the large domain of the PDE problem, the node adaptive algorithms present in literature might meet any difficulty in finding these almost singular areas. In the present algorithm, a node adaptive strategy works only on local nearly singular areas. Additionally, the RBF collocation technique has been mixed with a FD scheme. On the basis of this approach, instead of using the adaptive refinement scheme on the whole domain, our new algorithm is used only on time steps. In addition to this, we apply our adaptive approach only in any time step, unlike other available methods that make use of it on the entire domain. In addition, it should be noted that our adaptive algorithm has a reiterative approach, i.e. it adds new nodes to initial refinement in an iterative method that needs to solve a linear system at any iteration. In this case, we can thus solve small linear systems on each time step instead of large systems on the entire domain. The main feature of the proposed method is the low-dimensional linear system that must be solved in each step. This technique indeed first detects automatically areas with fast changes, and then applies a local node adaptive algorithm only in those nearly singular regions. Accordingly, this advantage overwhelms possible problems that occur in other adaptive algorithms.

The article is organized as follows. Section 2 provides the key theoretical foundations regarding RBF approximation; moreover, the RBF collocation technique with a FD-based discretization is proposed for time-dependent PDEs. Stability analysis of the used collocation technique is provided in section 3. In section 4, the adaptive refinement scheme is described. The results of our extensive numerical tests are provided in section 5. Ultimately, section 6 presents the conclusions.

2. RBF approximation

Given that the numerical solution of time-dependent PDEs utilizing RBFs is characterized by concepts of scattered data interpolation, in this section we first provide an overview of the key theoretical aspects

RBF	$\phi_\epsilon(r)$
Gaussian C^∞ (GA)	$e^{-\epsilon^2 r^2}$
MultiQuadric C^∞ (MQ)	$(1+\epsilon^2 r^2)^{1/2}$
Inverse MultiQuadric C^∞ (IMQ)	$(1+\epsilon^2 r^2)^{-1/2}$
Thin Plate Spline $C^{\nu+1}$ (TPS)	$(-1)^{\nu+1} r^{2\nu} \log r$
Matérn C^2 (M2)	$e^{-\epsilon r}(\epsilon r+1)$
Wendland C^2 (W2)	$(1-\epsilon r)_+^4\,(4\epsilon r+1)$

Table 1: Some examples of well-known RBFs. Note that $(\cdot)_+$ identifies the truncated power function, and $\nu \in \mathbb{N}$.

related to RBF interpolation, and then we extend them to the case of RBF collocation.

2.1. RBF interpolation by conditionally positive definite functions

Within scattered data interpolation by RBFs, the reconstruction of a function $u(\boldsymbol{x})$ defined at a set of distinct *data points* or *centers* or *nodes* $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ of a domain $\Omega \subseteq \mathbb{R}^d$ may be expressed as a linear combination of RBFs, which in general can also provide for adding a polynomial term. Thus, the resulting RBF interpolant $s_{u,X} : \Omega \to \mathbb{R}$ assumes the form

$$s_{u,X}(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \phi_\epsilon(||\boldsymbol{x}-\boldsymbol{x}_j||) + \sum_{k=1}^{Q} \beta_k p_k(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^d, \tag{1}$$

in which α_j and β_k denote real coefficients to be computed, $|| \cdot ||$ identifies the Euclidean distance, and $\phi_\epsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}$ defines a RBF affected by a *shape parameter* $\epsilon > 0$ such that $\phi_\epsilon(||\boldsymbol{x}-\boldsymbol{z}||) = \phi(\epsilon||\boldsymbol{x}-\boldsymbol{z}||)$, for all $\boldsymbol{x}, \boldsymbol{z} \in \Omega$. For easiness, ϕ_ϵ is referred as ϕ in the following. Table 1 shows a few of the most known RBFs together with their smoothness orders [33].

Moreover, in (1) $p_1, \dots, p_Q$ create a basis for the Q-dimensional space $\pi_{m-1}(\mathbb{R}^d)$ of polynomials of overall degree $\leq m-1$ in d variables. In this case, the interpolation conditions

$$s_{u,X}(\boldsymbol{x}_i) = u(\boldsymbol{x}_i), \qquad i = 1, \dots, N,$$

are accomplished by the following constrains

$$\sum_{j=1}^{N} \alpha_j p_k(\boldsymbol{x}_j) = 0, \qquad k = 1, \dots, Q. \tag{2}$$

The expansion (1) is therefore obtained by solving the linear system

$$\begin{pmatrix} A & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{pmatrix}, \tag{3}$$

in which the matrix $A \in \mathbb{R}^{N\times N}$ is formed by the entries $A_{ij} = \phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||)$, $i,j = 1,\dots,N$, $P \in \mathbb{R}^{N\times Q}$ is given by $P_{jk} = (p_k(\boldsymbol{x}_j))$, $j = 1,\dots,N$, $k = 1,\dots,Q$, $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_N)^T$, $\boldsymbol{\beta} = (\beta_1,\dots,\beta_Q)^T$, $\boldsymbol{u} = (u_1,\dots,u_N)^T$, $\boldsymbol{0}$ is a zero vector of length Q, and O is a $Q\times Q$ zero matrix. The system (3) is clearly solvable when the interpolation matrix A is non-singular (or invertible) [50].

Definition 2.1. *The points $X = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N\} \subseteq \mathbb{R}^d$ with $N \geq Q = \dim \pi_m(\mathbb{R}^d)$ are named $\pi_m(\mathbb{R}^d)$-unisolvent when the zero polynomial is the merely polynomial from $\pi_m(\mathbb{R}^d)$ vanishing on all of them.*

Theorem 2.1. *Supposing that ϕ represents conditionally positive definite of order m and X is a $\pi_{m-1}(\mathbb{R}^d)$-unisolvent set of nodes. Then the system given in (3) is uniquely solvable.*

The use of the polynomial $\sum_{k=1}^{Q} \beta_k p_k(\boldsymbol{x})$ in (1) is generally needed if ϕ is conditionally positive definite, namely when ϕ possesses a polynomial progress to infinity as in the case of MQ and TPS. These functions (in $\mathbb{R}^d$, for any d) have a global support and are conditionally positive definite of order $m = 1$ and $m = \nu+1$, with $\nu \in \mathbb{N}$, respectively. Consequently, adding a polynomial term of such orders in (1) – along with constrains (2) – is essential to ensure existence and uniqueness of the solution in (3). In contrast, the addition of a polynomial is not typically vital when we work with positive definite RBFs like GA, IMQ, and M2 functions (see Table 1). This fact is true in $\mathbb{R}^d$ for any d. Furthermore, as all the functions have a global support, they generate a dense (or full) interpolation matrix and occasionally – for some specific selections of ϵ – it may be very ill-conditioned [33].

To enhance the condition number of the matrix, compactly supported RBFs (CSRBFs) might also be used, though they disappear over a user-determined threshold distance σ. Consequently, only the entries of the interpolation matrix referring to the nodes lying closer than σ to a certain CSRBF center assume a non-null value. This leads to a sparse matrix. Practically, the use of CSRBFs waned since it has been clear that, to achieve a good level of precision, most nodes in the point set need to be covered by the overlap distance σ, so resulting in a full matrix again [46]. A well-known class of CSRBFs is represented by Wendland functions such as the W2 in Table 1, whose support is $[0, \sigma]$, with $\sigma = 1/\epsilon$. Such a function is positive definite in $\mathbb{R}^d$ for $d \leq 3$ [51].

For any linear partial differential operator $\mathcal{L}$, within a similar representation as (1), $\mathcal{L}u$ may be approximated by [24]

$$\mathcal{L}u(\boldsymbol{x}) \simeq \sum_{j=1}^{N} \alpha_j \mathcal{L}\phi(||\boldsymbol{x} - \boldsymbol{x}_j||) + \sum_{k=1}^{Q} \beta_k \mathcal{L}p_k(\boldsymbol{x}).$$

In our technique, the RBF ϕ we use is the TPS. The reason is that in [36] Franke demonstrated that TPS and MQ yield the most precise results for the multivariate approximation of scattered data. Nevertheless, while the TPS method is parameter free and is characterized by a sound mathematical theory [7], the common RBF methods (including MQ as well) depend on the value of ϵ and so far no mathematical theory has been found to select the optimum value. Therefore, though ϵ influences the accuracy of the numerical method, such a choice can be done by using experimental tuning parameters or costly optimization approaches that enable us to determine the optimal shape parameter [11, 34, 39]. Moreover, we observe that the TPS is conditionally positive definite of order $m = \nu + 1$ [50]. Given that ϕ is $C^{2\nu-1}$ continuous, a higher-order TPS has to be utilized for higher-order partial differential operators. To prevent the problems at $x = 0$ ($\log(0) = -\infty$), we make use of $\phi(\boldsymbol{x}) = (-1)^3 \|\boldsymbol{x}\|_2^3 \log \|\boldsymbol{x}\|_2^{\|\boldsymbol{x}\|_2}$ for $k = 2$.

2.2. RBF collocation based on FDs for time-dependent PDEs

Here, a meshfree collocation technique combined with a RBF-FD scheme is provided for solving parabolic and hyperbolic PDEs.

2.2.1. Parabolic PDEs

Considering a spatial domain Ω and a linear operator $\mathcal{L}$ that perform over a smooth function on Ω, we assume that the operator $\mathcal{L}$ always acts w.r.t. the spatial variable even when a time variable t exists. On the time interval $[0, T]$, we look for a scalar function $u : \Omega \times [0, T] \to \mathbb{R}$ fulfilling the time-dependent PDE

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial t} - \mathcal{L}u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, \tag{4}$$

together with the following initial and boundary conditions

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{5}$$

$$\mathcal{B}u(\boldsymbol{x}, t) = g(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \partial\Omega, \quad t > 0, \tag{6}$$

where f, g, and u_0 are the known functions, and $\mathcal{B}$ represents a boundary operator of Dirichlet, Neumann or mixed type, while $\partial\Omega$ identifies the boundary of the spatial domain Ω. When considering Dirichlet boundary conditions, the time derivative of the PDE (4) may be discretized by a common FD formula using the following θ-weighted method

$$\frac{u^{n+1}(\boldsymbol{x}) - u^n(\boldsymbol{x})}{\delta t} = \theta\left(\mathcal{L}u^{n+1}(\boldsymbol{x}) + f^{n+1}(\boldsymbol{x})\right) + (1-\theta)\left(\mathcal{L}u^n(\boldsymbol{x}) + f^n(\boldsymbol{x})\right), \tag{7}$$

where $0 \le \theta \le 1$, $u^{n+1}(\boldsymbol{x}) = u(\boldsymbol{x}, t^{n+1})$, $t^{n+1} = t^n + \delta t$, and δt is the time step size. The rearrangement of (7) gives

$$u^{n+1}(\boldsymbol{x}) + \eta\mathcal{L}u^{n+1}(\boldsymbol{x}) = u^n(\boldsymbol{x}) + \zeta\mathcal{L}u^n(\boldsymbol{x}) + z^{n+1}(\boldsymbol{x}), \tag{8}$$

where $\eta = -\theta\delta t$, $\zeta = (1-\theta)\delta t$, and $z^{n+1} = \delta t(\theta f^{n+1}(\boldsymbol{x}) + (1-\theta)f^n(\boldsymbol{x}))$.

Because we use TPS RBFs for the two-dimensional domain, supposing a total of $(N-6)$ collocation points, $u^n(\boldsymbol{x})$ can be estimated by

$$u^n(x,y) \simeq \sum_{j=1}^{N-6} \lambda_j^n \varphi(r_j) + \lambda_{N-5}^n x^2 + \lambda_{N-4}^n y^2 + \lambda_{N-3}^n xy + \lambda_{N-2}^n x + \lambda_{N-1}^n y + \lambda_N^n. \tag{9}$$

To get the coefficients $(\lambda_1, \lambda_2, \ldots, \lambda_{N-1}, \lambda_N)$, the collocation technique is utilized using (9) in every points $\boldsymbol{x}_i = (x_i, y_i),\ i = 1, 2, \ldots, N-6$. Thus, we have

$$u^n(x_i,y_i) \simeq \sum_{j=1}^{N-6} \lambda_j^n \varphi(r_{ij}) + \lambda_{N-5}^n x_i^2 + \lambda_{N-4}^n y_i^2 + \lambda_{N-3}^n x_i y_i + \lambda_{N-2}^n x_i + \lambda_{N-1}^n y_i + \lambda_N^n, \tag{10}$$

where $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}$. The additional conditions in (2) are thus expressed as follows

$$\sum_{j=1}^{N-6} \lambda_j^n x_j^2 = \sum_{j=1}^{N-6} \lambda_j^n y_j^2 = \sum_{j=1}^{N-6} \lambda_j^n x_j y_j = \sum_{j=1}^{N-6} \lambda_j^n x_j = \sum_{j=1}^{N-6} \lambda_j^n y_j = \sum_{j=1}^{N-6} \lambda_j^n = 0. \tag{11}$$

Writing (10) together with (11) in a matrix form, we have

$$\boldsymbol{u}^n = A\boldsymbol{\lambda}^n, \tag{12}$$

where $\boldsymbol{u}^n = \left(u_1^n,\ \ldots,\ u_{N-6}^n,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)^T$, and $\boldsymbol{\lambda}^n = (\lambda_1^n,\ \ldots,\ \lambda_N^n)^T$. Then, setting $\varphi_{ij} = \varphi(r_{ij})$, $i,j = 1,\ldots,N-6$, the matrix A is given by

$$A = \begin{pmatrix}
\varphi_{11} & \cdots & \varphi_{1(N-6)} & x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\varphi_{(N-6)1} & \cdots & \varphi_{(N-6)(N-6)} & x_{N-6}^2 & y_{N-6}^2 & x_{N-6}y_{N-6} & x_{N-6} & y_{N-6} & 1 \\
x_1^2 & \cdots & x_{N-6}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
y_1^2 & \cdots & y_{N-6}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_1y_1 & \cdots & x_{N-6}y_{N-6} & 0 & 0 & 0 & 0 & 0 & 0 \\
x_1 & \cdots & x_{N-6} & 0 & 0 & 0 & 0 & 0 & 0 \\
y_1 & \cdots & y_{N-6} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & \cdots & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}_{N\times N}.$$

Assuming that B and I are the indexes of boundary and internal points, respectively, and $N-6$ represents the overall number of centers, namely $N = N_I + N_B + 6$, then the $N \times N$ matrix A may be decomposed into three matrices A_I, A_B, and A_E, i.e.

$$A = A_I + A_B + A_E,$$

where

$$\begin{aligned}
A_I &= \left[a_{ij} \text{ for } (i \in I, 1 \le j \le N) \text{ and } 0 \text{ elsewhere}\right], \\
A_B &= \left[a_{ij} \text{ for } (i \in B, 1 \le j \le N) \text{ and } 0 \text{ elsewhere}\right], \\
A_E &= \left[a_{ij} \text{ for } (N-5 \le i \le N, 1 \le j \le N) \text{ and } 0 \text{ elsewhere}\right].
\end{aligned}$$

Utilizing the notation $\mathcal{L}A$ to designate the matrix that has the same size as A, with entries of $\widehat{a}_{ij} = \mathcal{L}a_{ij}$, $1 \leq i, j \leq N$, and using (9) in (8) along with (6), the resultant system can be written in the matrix form

$$C\boldsymbol{\lambda}^{n+1} = D\boldsymbol{\lambda}^n + \boldsymbol{v}^{n+1}, \tag{13}$$

where

$$\begin{aligned}
&C = A + \eta\mathcal{L}A_I,\\
&D = A_I + \zeta\mathcal{L}A_I,\\
&\boldsymbol{v}^{n+1} = \left[z_i^{n+1} \text{ for } (i \in I), \quad g_j^{n+1} \text{ for } (j \in B) \text{ and } 0 \text{ elsewhere}\right]^T,\\
&\boldsymbol{\lambda}^n = \left(\lambda_1^n, \, \ldots, \, \lambda_N^n\right)^T.
\end{aligned}$$

The system (13) is attained by integrating (8), which refers to the internal points, and (6) that applies to the boundary points. Hence, utilizing the condition (5), we can calculate $\boldsymbol{\lambda}^{n+1}$ by solving the system (13). Then, by replacing such values of $\boldsymbol{\lambda}^n$ in (12), the approximated solution of the PDE at time level n is attained.

2.2.2. Hyperbolic PDEs

Considering a spatial domain Ω and a linear operator $\mathcal{L}$ that performs over a smooth function on Ω, we assume that the operator $\mathcal{L}$ always acts w.r.t. the spatial variable even when a time variable t exists. On the time interval $[0, T]$, we look for a scalar function $u : \Omega \times [0, T] \to \mathbb{R}$ fulfilling the time-dependent PDE

$$\frac{\partial^2 u(\boldsymbol{x}, t)}{\partial t^2} - \mathcal{L}u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, \tag{14}$$

in conjunction with the subsequent initial and boundary conditions,

$$u(\boldsymbol{x}, 0) = h(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{15}$$

$$u_t(\boldsymbol{x}, 0) = k(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{16}$$

$$\mathcal{B}u(\boldsymbol{x}, t) = g(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \partial\Omega, \quad t > 0, \tag{17}$$

where f, g, h and k are the known functions, and $\mathcal{B}$ represents a boundary operator of Dirichlet, Neumann or mixed type, while $\partial\Omega$ identifies the boundary of the spatial domain Ω. When considering Dirichlet boundary conditions, the discretization of the time derivative of the partial differential equation (14) can be achieved through a conventional finite difference formula utilizing the θ-weighted approach.

$$\frac{u^{n+1}(\boldsymbol{x}) - 2u^n(\boldsymbol{x}) + u^{n-1}(\boldsymbol{x})}{(\delta t)^2} = \theta\left(\mathcal{L}u^{n+1}(\boldsymbol{x}) + f^{n+1}(\boldsymbol{x})\right) + (1-\theta)\left(\mathcal{L}u^n(\boldsymbol{x}) + f^n(\boldsymbol{x})\right), \tag{18}$$

where $0 \leq \theta \leq 1$, $u^{n+1}(\boldsymbol{x}) = u(\boldsymbol{x}, t^{n+1})$, $t^{n+1} = t^n + \delta t$, $u^{n-1}(\boldsymbol{x}) = u(\boldsymbol{x}, t^{n-1})$, $t^{n-1} = t^n - \delta t$, and δt is the time step size. The rearrangement of (18) gives

$$u^{n+1}(\boldsymbol{x}) + \eta\mathcal{L}u^{n+1}(\boldsymbol{x}) = 2u^n(\boldsymbol{x}) - u^{n-1}(\boldsymbol{x}) + \zeta\mathcal{L}u^n(\boldsymbol{x}) + z^{n+1}(\boldsymbol{x}), \tag{19}$$

where $\eta = -\theta(\delta t)^2$, $\zeta = (1-\theta)(\delta t)^2$, and $z^{n+1} = (\delta t)^2(\theta f^{n+1}(\boldsymbol{x}) + (1-\theta)f^n(\boldsymbol{x}))$.

Utilizing the notation $\mathcal{L}A$ to designate the matrix that has the same size as A, with entries of $\widehat{a}_{ij} = \mathcal{L}a_{ij}$, $1 \leq i, j \leq N$, and using (9) in (19) along with (17), the resultant system can be written in the matrix form

$$C\boldsymbol{\lambda}^{n+1} = D\boldsymbol{\lambda}^n - E\boldsymbol{\lambda}^{n-1} + \boldsymbol{v}^{n+1}, \tag{20}$$

where

$$\begin{aligned}
&C = A + \eta \mathcal{L} A_I, \\
&D = 2A_I + \zeta \mathcal{L} A_I, \\
&E = A_I, \\
&\boldsymbol{v}^{n+1} = \left[z_i^{n+1} \text{ for } (i \in I), \;\; g_j^{n+1} \text{ for } (j \in B) \text{ and } 0 \text{ elsewhere} \right]^T, \\
&\boldsymbol{\lambda}^n = (\lambda_1^n, \; \ldots, \; \lambda_N^n)^T.
\end{aligned}$$

The system (20) is attained by integrating (19), which refers to the internal points, and (17) that applies to the boundary points. Hence, utilizing the condition (16), we can calculate $\boldsymbol{\lambda}^{n+1}$ by solving the system (20). Then, by replacing such values of $\boldsymbol{\lambda}^n$ in (12), the approximated solution of the PDE at time level n is attained.

At $n = 0$ Eq. (19) has the following form

$$u^1(\boldsymbol{x}) + \eta \mathcal{L} u^1(\boldsymbol{x}) = 2u^0(\boldsymbol{x}) - u^{-1}(\boldsymbol{x}) + \zeta \mathcal{L} u^0(\boldsymbol{x}) + z^1(\boldsymbol{x}), \tag{21}$$

To approximate $u^{-1}(\boldsymbol{x})$ in the internal points, the initial velocity is used. For this, we discretize the initial velocity as

$$\frac{u^1(\boldsymbol{x}) - u^{-1}(\boldsymbol{x})}{2\delta t} = k(\boldsymbol{x}) \tag{22}$$

Writing (21) together with (22) we have

$$2u^1(\boldsymbol{x}) + \eta \mathcal{L} u^1(\boldsymbol{x}) = 2u^0(\boldsymbol{x}) + 2\delta t k(\boldsymbol{x}) + \zeta \mathcal{L} u^0(\boldsymbol{x}) + z^1(\boldsymbol{x}). \tag{23}$$

Though the system of equations is effective for any value of $\theta \in [0, 1]$, we will utilize $\theta = 1/2$ (the prominent Crank-Nicholson method).

3. Stability analysis

Starting from the results in [22], we analyze numerical stability of the proposed method, which is applied to the time-dependent equation (4). Now, introducing a perturbation in (13), we define the error

$$\boldsymbol{e}^n = \boldsymbol{u}^n - \tilde{\boldsymbol{u}}^n,$$

where $\boldsymbol{u}^n$ represents the discrete exact solution and $\tilde{\boldsymbol{u}}^n$ denotes the approximated numerical solution. Hence, the equation for the error $\boldsymbol{e}^{n+1}$ can be written as follows

$$\boldsymbol{e}^{n+1} = K \boldsymbol{e}^n, \tag{24}$$

where the amplification matrix K takes the form

$$K = AC^{-1}DA^{-1}.$$

If the error $\boldsymbol{e}^n \to 0$, as $n \to \infty$, then the numerical scheme is stable. In particular, stability is assured when the spectral radius ρ of the matrix K is less than or equal to 1, i.e. $\rho(K) \leq 1$. If we substitute K in (24) we have

$$CA^{-1}\boldsymbol{e}^{n+1} = DA^{-1}\boldsymbol{e}^n. \tag{25}$$

Moreover, if we assume Dirichlet boundary conditions, equation (25) becomes

$$P\boldsymbol{e}^{n+1} = Q\boldsymbol{e}^n, \tag{26}$$

where

$$P = [I - \theta \delta t M]\,,$$
$$Q = [I + (1-\theta)\delta t M]\,,$$

$I \in \mathbb{R}^{N\times N}$ is the identity matrix and the matrix $M = \mathcal{L}A_I A^{-1}$.

From (26) it follows that stability is guaranteed provided that all the eigenvalues of the matrix $P^{-1}Q$ are less than one. This fact happens if

$$\left| \frac{1+(1-\theta)\delta t \lambda_M}{1-\theta\delta t\lambda_M} \right| \leq 1, \tag{27}$$

with λ_M denoting an eigenvalue of matrix M. The following generalized eigenvalue problem is solved to determine such eigenvalues

$$\mathcal{L}A_I \boldsymbol{s} = \lambda_M A \boldsymbol{s}.$$

For Crank-Nicholson scheme, namely assuming $\theta = 1/2$, the condition (27) is always true provided that $\lambda_M \leq 0$. This also holds for $\theta = 1$. In both cases we state that the numerical method is unconditionally stable. When considering the explicit method, i.e. for $\theta = 0$, the stability condition assumes the form

$$|1 + \delta t \lambda_M| \leq 1.$$

Hence the related technique is stable when

$$\delta t \leq -\frac{2}{\lambda_M} \text{ and } \lambda_M \leq 0.$$

4. Adaptive node refinement algorithm

Now, we can present the algorithm for solving time-dependent PDEs, whose solution offers fast changes or high gradients in several local areas of the domain. Our major goal is to develop an adaptive algorithm first to find the areas with quick changes and then applies a local node adaptive approach merely in those almost singular areas. In this algorithm, we consider a numerical scheme characterized by the use of collocation points and TPS RBFs. Since our technique is characterized by FDs, we utilize the adaptive algorithm at any time step. Comparing the latter with the conventional algorithms applied on the whole domain of the problem, the novel algorithm is used in any time step, i.e. we solve small linear systems of equations in the adaptive approach on each time step instead of large linear systems as in Kansa collocation method. The step by step process is showed in Algorithm 1, where the domain Ω is assumed to a square region in order to make the description simpler. At the final step of this recursive procedure (Algorithm 1), the remaining area will be small enough. Hence, the ultimate phase of the adaptive algorithm can be completed as detailed in Algorithm 2.

In the algorithms presented in this section, we compute the residuals of the Root Mean Square (RMS) error in $N = N_I + N_B$ points:

$$\text{ResRMS} = \sqrt{\frac{1}{N}\left(\sum_{i=1}^{N_I}\left(\mathcal{L}\tilde{u}\left(x_i, y_i, T\right) - f\left(x_i, y_i, T\right)\right)^2 + \sum_{i=1}^{N_B}\left(\tilde{u}\left(x_i{}^*, y_i{}^*, T\right) - g\left(x_i{}^*, y_i{}^*, T\right)\right)^2\right)}, \tag{28}$$

where $(x_i, y_i)_{i=1}^{N_I}$ are interior points and $(x_i{}^*, y_i{}^*)_{i=1}^{N_B}$ are boundary points. The functions f and g in (28) derive from PDE problem, while $\tilde{u}$ denotes the approximate solution.

In order to describe the algorithm in a more detailed way, we distinguish three different positions in which a nearly singular area can be located in the domain, namely close to the center, close to the boundary, or in an intermediate region between center and boundary.

Algorithm 1 The step by step adaptive algorithm

```
Input: the domain $\Omega = [a,b] \times [a,b]$, the real numbers $\delta$ and Tol
Output: adaptive centers
1: the remaining area = $\Omega$.
2: centers = $\emptyset$.
3: adaptive centers = $\emptyset$.
4: the covering area = $\emptyset$.
5: if $\frac{b-a}{2\delta} = \left\lfloor \frac{b-a}{2\delta} \right\rfloor$ then
6:     $n = \frac{b-a}{2\delta}$.
7: else
8:     $n = \left\lfloor \frac{b-a}{2\delta} \right\rfloor + 1$.
9: end if
10: for $i = 1 : n$ do
11:     $\Delta_i$ = a boundary layer of the remaining area with width $\delta$ in all direction at the $i$-th step.
12:     the covering area = the covering area $\cup$ $\Delta_i$.
13:     the remaining area = $\Omega$ - (the covering area).
14:     centers = a coarse uniform grid on $\Delta_i$.
15:     adaptive centers = centers $\cup$ adaptive centers.
16:     Solving the PDE problem by RBF technique on adaptive centers.
17:     ResRMS = the residuals' RMS error at the evaluation points located on $\Delta_i$.
18:     while ResRMS > Tol do
19:         R = compute the residuals at points halfway between the adaptive centers on $\Delta_i$.
20:         Tol1 = mean value of R.
21:         P = intermediate points with residual higher than Tol1.
22:         adaptive centers = P $\cup$ adaptive centers.
23:         Solving the PDE problem by RBF process on adaptive centers.
24:         ResRMS = the residuals' RMS error at the evaluation points located on $\Delta_i$.
25:     end while
26: end for
```

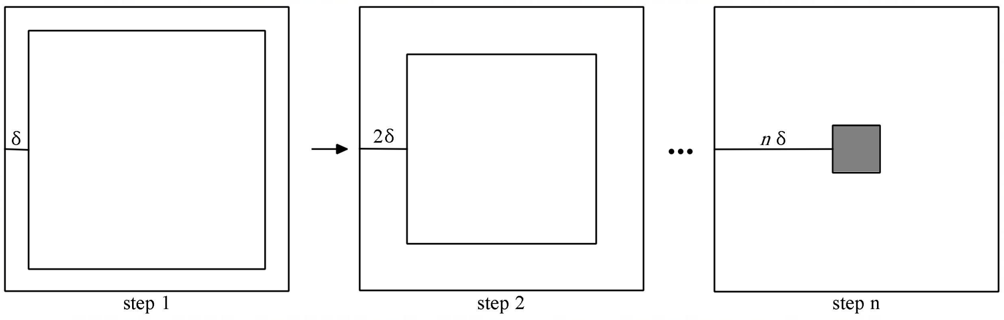


Figure 1: Some steps of Algorithm 1 when quick changes are located close to the center of the domain.

Firstly, in Fig. 1 we show the case in which the nearly singular area is located in the center of Ω. Here, the ultimate step of the algorithm starts with no need to apply the adaptive strategy in the previous steps. The final stage is started if the remaining region is characterized by a small enough area. So the adaptive procedure is carried out just in the small area with quick changes.

In Fig. 2 we then illustrate a few of the algorithm steps if a nearly singular area is localized in the

Algorithm 2 Final step of the adaptive algorithm

```
1: centers = a coarse uniform grid on the remaining area.
2: adaptive centers = centers $\cup$ adaptive centers.
3: Solving the PDE problem by RBF technique on $\Omega$.
4: ResRMS = the residuals' RMS error at the evaluation points located in the remaining area.
5: **if** ResRMS $\leq$ Tol **then**
6:     The final solution is achieved.
7: **else**
8:     **while** ResRMS $>$ Tol **do**
9:         R = compute the residuals at points halfway between the adaptive centers on the remaining area.
10:        Tol1 = mean value of R.
11:        P = intermediate points with residual higher than Tol1.
12:        adaptive centers = P $\cup$ adaptive centers.
13:        Solving the PDE problem by RBF technique on $\Omega$.
14:        ResRMS = the residuals' RMS error at the evaluation points located in the remaining area.
15:    **end while**
16:    The final solution is obtained.
17: **end if**
```

boundary. Here, since all boundary conditions must be satisfied, the adaptive refinement process in the first step is carried out over the entire boundary layer. In the next steps, the algorithm goes on without using any adaptive policy until the remaining region is small enough and the final stage of the computational procedure is executed.

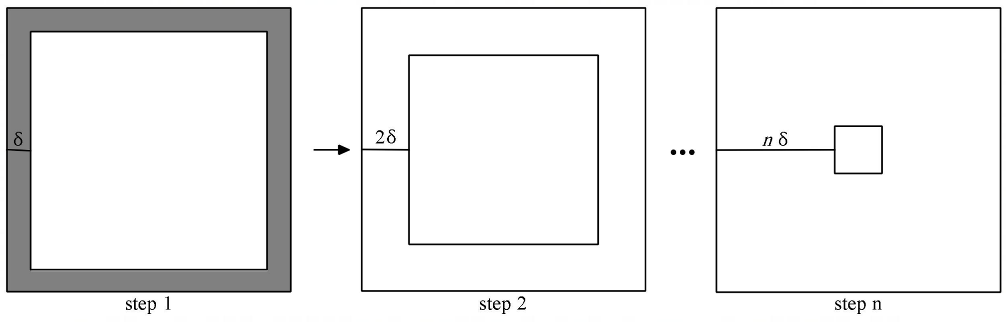


Figure 2: Some steps of Algorithm 1 when quick changes are located close to the boundary of the domain.

Finally, in Fig. 3 we give a graphical representation of some algorithm steps when quick changes are located in an intermediate region between the center and boundary of the domain. The algorithm proceeds until the nearly singular regions is identified and after that the node adaptive strategy is applied. Then, the final step of the algorithm is initiated with no further request of applying the adaptive strategy. Note that hachure present in the mentioned figures represents the region in which the node adaptive approach was carried out.

According to Figs. 1–3, prior to utilizing our adaptive approach, the algorithm discovers the areas with quick changes and then the node adaptive method is conducted just on small regions with quick variations. Hence, probable problems of other algorithms are reduced by this dynamic capability of finding the almost singular area in large domains. Moreover, by this capability, we can use specific or more sophisticated methods in the nearly singular area, whereas in other parts of the domain basic or simpler methods are sufficient for our purposes. We observe that when more than one nearly singular area exists in the computational domain, the procedure can be employed to identify these "critical" areas applying the node adaptive approach until the entire calculation domain is covered. This capability of the suggested technique is well described

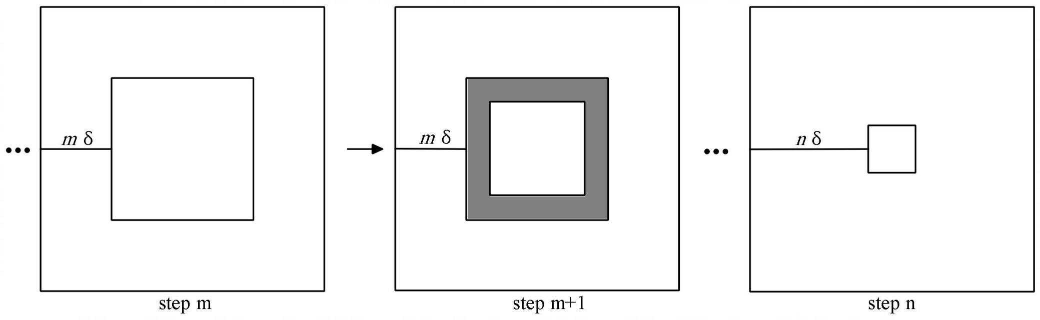


Figure 3: Some steps of Algorithm 1 when quick changes are located in an intermediate area between center and boundary of the domain.

by the numerical experiments provided in section 5.

Remark 4.1. *The adaptive algorithm is only utilized for the first time step when the position of high gradients or fast changes of the problem solution in the domain has no change at the advanced time. In such a case, relative nodal positions will remain unaffected. Therefore, in the adaptive approach only one linear system must be solved (using the LU decomposition technique) instead of numerous linear systems at any time step. This allows us a significant saving in terms of time and memory space. Consequently, we need to solve a novel linear system by applying the LU factorization only once, as the coefficient matrix is not changed at various time steps.*

Remark 4.2. *In the pursuit of attaining the desired level of accuracy through the utilization of a uniform point arrangement, it becomes imperative to significantly increase the number of collocation points. The majority of computational operations employed in the uniform approach are dedicated to solving the system of linear equations. The arithmetic complexity associated with solving this system is denoted as $O(N^3)$, which signifies a substantial volume of computational operations. Due to the large value of N, the magnitude of this computational operation is very significant. Conversely, the adaptive algorithm presented in this paper detects regions with rapid changes and exclusively applies a local node adaptive algorithm within these nearly singular areas. Consequently, the computational operations employed in Algorithm 1 to generate adapted points solely involve solving small localized linear systems and conducting limited comparisons. By employing this adapted approach, the number of collocation points is significantly reduced, and the system of linear equations possesses much smaller dimensions in comparison to the uniform approach. The computational time reported in next section clearly demonstrates the advantage of the adapted method over the uniform method. In summary, it can be concluded that in the uniform approach, in order to achieve the desired level of accuracy, the point arrangement throughout the entire problem-solving area must be chosen with much greater precision. Although the uniform arrangement approach does not incur any costs associated with point arrangement, the computational operations required to solve the system of linear equations are so extensive that the cost of generating the adaptive point arrangement is comparatively negligible. On the other hand, the modified approach will have the ability to significantly reduce the number of collocation points through the solution of a small system of linear equations and limited comparisons, which will save considerable time in solving the final system of linear equations. In addition, in the adapted approach, we are dealing with a smaller system of equations, and the possibility of encountering computational problems should be greatly reduced.*

5. Numerical experiments

This section aims to show the results obtained by applying the adaptive algorithm explained previously for solving the time-dependent PDEs. For this purpose, we take two difficult benchmark problems in two

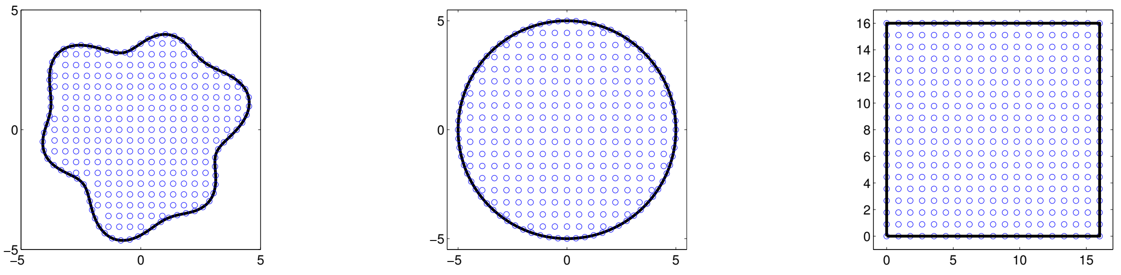


Figure 4: Example of regular and irregular domains for uniform points.

dimensions (parabolic and hyperbolic PDEs). At first, we test our scheme for the subsequent heat equation

$$u_t(x,y,t) - u_{xx}(x,y,t) - u_{yy}(x,y,t) = f(x,y,t), \quad (x,y) \in \Omega, \quad t \in [0,T], \tag{29}$$

along with the initial and Dirichlet boundary conditions

$$u(x,y,0) = u_0(x,y), \quad (x,y) \in \Omega,$$

$$u(x,y,t) = g(x,y,t), \quad (x,y) \in \partial\Omega, \quad t \in [0,T],$$

where f, g, and u_0 are selected based on the analytical solution. Next, we apply the new algorithm for the following wave equation

$$u_{tt}(x,y,t) - u_{xx}(x,y,t) - u_{yy}(x,y,t) = f(x,y,t), \quad (x,y) \in \Omega, \quad t \in [0,T], \tag{30}$$

along with the initial and Dirichlet boundary conditions

$$\begin{aligned} u(x,y,0) &= h(x,y), \quad (x,y) \in \Omega, \\ u_t(x,y,0) &= k(x,y), \quad (x,y) \in \Omega, \end{aligned}$$

$$u(x,y,t) = g(x,y,t), \quad (x,y) \in \partial\Omega, \quad t \in [0,T],$$

where f, g, h, and k are selected based on the analytical solution. In the current research, we take into account three kinds of regular and irregular domains involving convex (square and circular) and non-convex domains, see Fig. 4.

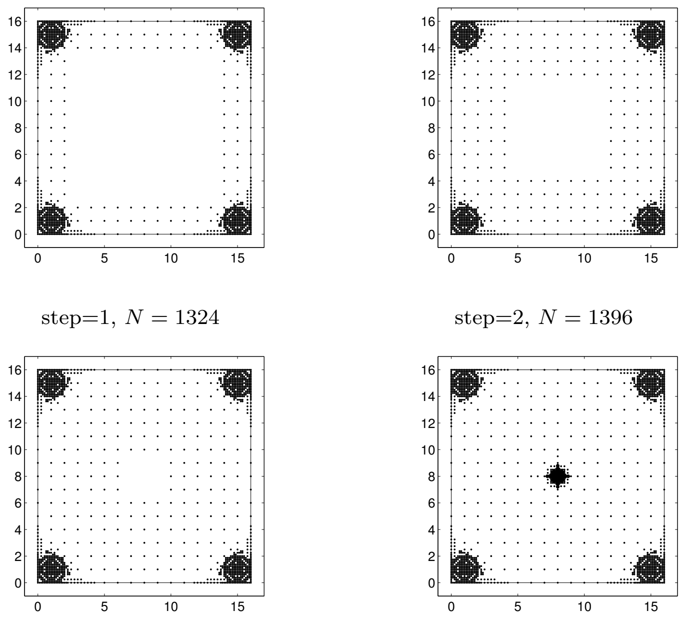
The values of Tol and δ in the algorithm given in section 3 have the main role in the detection of the ultimate node distribution and accordingly in the solution accuracy. Indeed, the choice of Tol and δ depends on the examined problems. In general, however, we expect that smaller values of such parameters will cause more precise results since a denser distribution is formed in a greatly localized region. Note that in this section N identifies the overall number of nodes in the ultimate node distribution. All numerical results showed in this section were attained using Matlab on a laptop (Intel Core i5, 2.6 GHz processor).

In order to analyze the accuracy of our numerical scheme and assess the approximation error, we define the maximum absolute (L_∞) and the RMS errors

$$L_\infty = \max_{1\le i\le M} |u(x_i,y_i,T) - \tilde{u}(x_i,y_i,T)|,$$

$$\text{RMS} = \sqrt{\frac{1}{M}\sum_{i=1}^{M} \left|u\left(x_i,y_i,T\right) - \tilde{u}\left(x_i,y_i,T\right)\right|^2},$$

where T represents the maximum time level, i.e. $T = t_{\max}$, $u\left(x,y,T\right)$ is the exact solution, $\tilde{u}\left(x,y,T\right)$ denotes the approximate solution and M refers to the overall number of evaluation points. In all our numerical tests we compute the L_∞ and RMS errors using a uniform distribution consisting of $M = 8100$ (90×90) points.



step=1, $N = 1324$ step=2, $N = 1396$

step=3, $N = 1436$ step=4, $N = 1849$

Figure 5: Different steps of the adaptive algorithm in Example 5.1.

Though the approximation method expressed in section 2 is usually effective for any values of $\theta \in [0, 1]$, in the following we discuss the case $\theta = 1/2$, which characterizes the famous Crank-Nicholson method. Moreover, we perform all our tests using the TPS RBF.

The current analysis is devoted to analyze the efficiency computing the CPU times (in seconds) of our numerical method, as well as to verify its accuracy and stability.

5.1. Parabolic PDE

5.1.1. Regular domain

In this subsection, we present numerical results attained from tests performed for the heat equation (29) on the square domain.

Example 5.1. *Consider the 2D heat equation* (29) *with the following exact solution*

$$u(x, y, t) = \Big(2e^{-100(x-8)^2-100(y-8)^2} + 2e^{-10(x-1)^2-10(y-1)^2} + 2e^{-10(x-15)^2-10(y-15)^2} + 2e^{-10(x-1)^2-10(y-15)^2} + 2e^{-10(x-15)^2-10(y-1)^2}\Big) e^{-0.2\pi t}. \tag{31}$$

This solution is a function with various areas characterized by fast changes, and an adaptive refinement strategy needs to be applied in each of these nearly singular areas. Our adaptive algorithm is used for solving this problem in the domain $\Omega = [0, 16] \times [0, 16]$. Further, in all cases, we consider $T = 1$ and $\delta t = 0.1$ as time parameters.

Fig. 5 depicts the nodes distributions in various steps of the algorithm, while Fig. 6 represents the exact solution profile and the ultimate adaptive distribution gained by applying the adaptive algorithm.

The results reported in Table 2 show that our adaptive distribution (algorithm) achieves better accuracy with a smaller number of nodes than uniform distribution with many more nodes.

Moreover, Fig. 7 displays the absolute errors calculated at the evaluation points on $\Omega = [0, 16] \times [0, 16]$.

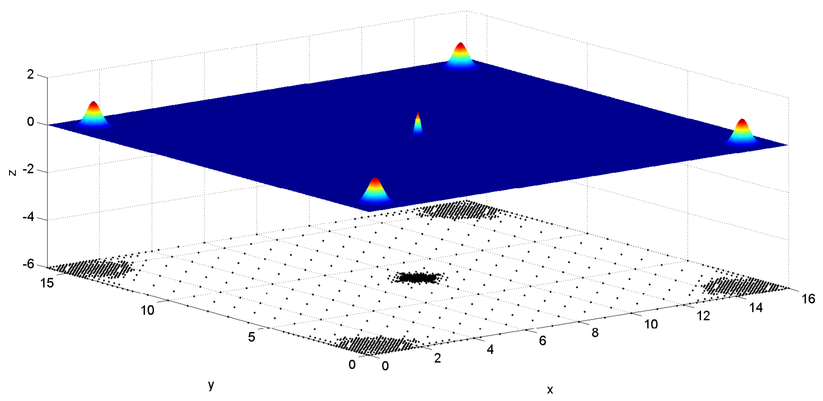


Figure 6: Profile of the final adaptive distribution in Example 5.1 and the exact solution (31) on $\Omega = [0, 16] \times [0, 16]$ with $\delta t = 0.1$ and $T = 1$.

Table 2: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.1 on $\Omega = [0, 16] \times [0, 16]$ with $\delta t = 0.1$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa(A)$	Time(s)
Uniform distribution	6889	3.16×10^{-1}	5.68×10^{0}	_	1.43×10^{12}	347.55
Adaptive distribution	1849	1.02×10^{-4}	1.57×10^{-3}	2	1.10×10^{12}	318.81

As previously stated, if the location of high gradients corresponding to the solution of the problem over the domain is not changed at the advance time, our adaptive algorithm needs to be used only in the first time step. In such a case, relative nodal locations will not change. Therefore, we have to solve only one linear system instead of a large number of linear systems on each time step. In particular, in Table 3 we observe that the algorithm is only used in the first step and this adaptive distribution will not change in the advance time steps. Note that the value T used here (and in the next tables) represents the maximum time level T expressed in seconds (s).

According to this work, a quite uniform behavior can be summarized as follows: numerical stability (condition number) of the two approaches (uniform vs adaptive distribution) is almost the same, with typically a slight advantage for the adaptive technique, however, we observe a considerable decrease of the

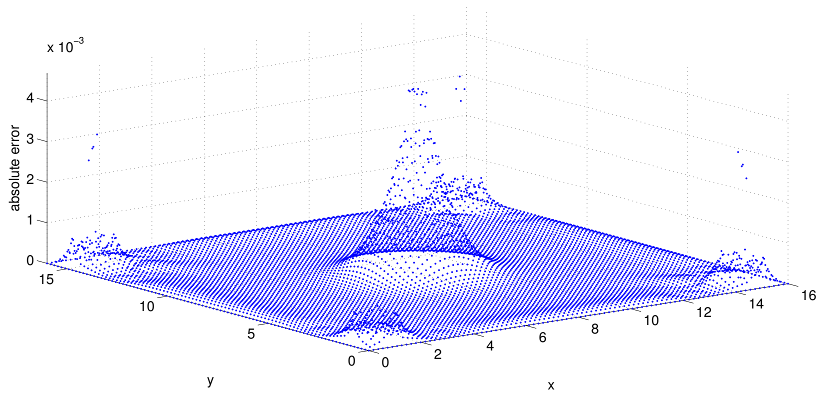


Figure 7: Absolute errors on $\Omega = [0, 16] \times [0, 16]$ in Example 5.1 with $N = 1849$, $\delta t = 0.1$, and $T = 1$.

Table 3: Accuracy, condition number, and CPU time for adaptive distribution on $\Omega = [0, 16] \times [0, 16]$ in Example 5.1 with $\delta t = 0.1$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	2.25×10^{-4}	3.65×10^{-3}	2	3.24×10^{12}	32.53
2	1.41×10^{-4}	2.11×10^{-3}	2	3.24×10^{12}	45.62
3	8.50×10^{-5}	1.16×10^{-3}	2	3.24×10^{12}	51.05
4	5.25×10^{-5}	6.27×10^{-4}	2	3.24×10^{12}	56.36
5	3.43×10^{-5}	3.39×10^{-4}	2	3.24×10^{12}	65.22
10	1.11×10^{-5}	1.96×10^{-5}	2	3.25×10^{12}	76.64
15	7.13×10^{-6}	1.60×10^{-5}	2	3.25×10^{12}	101.22
20	4.84×10^{-6}	1.19×10^{-5}	2	3.23×10^{12}	131.75
25	3.30×10^{-6}	8.61×10^{-6}	2	3.23×10^{12}	153.97

Table 4: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.2 on the non-convex domain $\Omega \subseteq [-5, 5] \times [-5, 5]$ with $\delta t = 0.01$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa(A)$	Time(s)
Uniform distribution	7329	9.58×10^{-2}	6.48×10^{-1}	_	3.88×10^{11}	3870.13
Adaptive distribution	841	6.53×10^{-4}	3.39×10^{-3}	1	2.25×10^{11}	708.39

CPU time and a remarkable increase in accuracy when the adaptive approach is compared with the uniform one.

5.1.2. Irregular domain

This subsection presents numerical results attained from experiments performed to examine the behavior of our adaptive technique for the heat equation on convex (circular) and non-convex domains, see Fig. 4.

Example 5.2. *Consider the 2D heat equation with the subsequent exact solution*

$$u(x, y, t) = \left(e^{-100(x)^2-100(y)^2}\right) e^{-0.2\pi t}. \tag{32}$$

Now, to show the applicability of our technique on an irregular shaped domain, this problem is solved on a non-convex domain $\Omega \subseteq [-5, 5] \times [-5, 5]$, whose area is revealed in Fig. 4 (left). Specially, the boundary of this irregular domain Ω is defined as follows

$$\partial\Omega = \{(r, \theta) \mid r(\theta) = 4 + \frac{4}{10}(\sin(6\theta) + \sin(3\theta))\}.$$

The exact solution (32) is a flat function with an interior point-wise steep peak at center. Moreover, in this example, we take as time parameters the constants $\delta t = 0.01$ and $T = 1$.

In Fig. 8 we report distributions of nodes in various steps of the algorithm on the non-convex domain. Fig. 9 gives a graphical representation of the exact solution and the conclusive adaptive distribution attained via the adaptive scheme.

Numerical results highlight that, using an adaptive distribution of nodes generated by our algorithm, we can get a better accuracy than an uniform distribution (even if in the latter case a very large number of nodes is considered), see Table 4. Moreover, Fig. 10 displays the absolute errors calculated at the evaluation points on the irregular non-convex domain.

In Table 5, the dynamic algorithm is only used in the first step and this adaptive distribution will not change in the advance time steps.

Even in this second case, our adaptive method (w.r.t. the uniform one) behaves in a quite uniform way: the conditioning is rather similar, whereas the efficiency expressed in CPU time and the accuracy obtained computing the RMS error show that our adaptive approach performs much better than the uniform one.

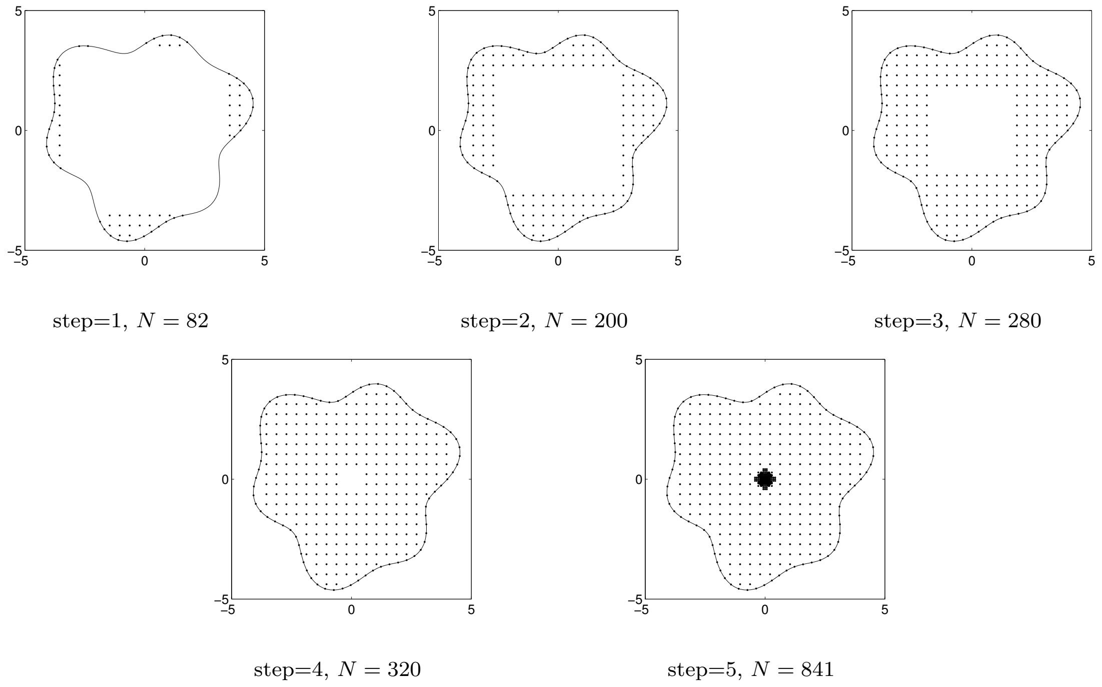


Figure 8: Different steps of the adaptive algorithm in Example 5.2.

Table 5: Accuracy, condition number, and CPU time for adaptive distribution on the non-convex domain $\Omega \subseteq [-5,5] \times [-5,5]$ in Example 5.2 with $\delta t = 0.01$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	1.00×10^{-4}	6.35×10^{-4}	1	1.44×10^{12}	65.36
2	9.93×10^{-5}	4.63×10^{-4}	1	1.43×10^{12}	113.63
3	8.53×10^{-5}	3.18×10^{-4}	1	1.43×10^{12}	167.46
4	6.81×10^{-5}	2.17×10^{-4}	1	1.43×10^{12}	255.08
5	5.18×10^{-5}	1.47×10^{-4}	1	1.43×10^{12}	323.56
10	9.58×10^{-6}	2.11×10^{-5}	1	1.43×10^{12}	578.15
15	1.47×10^{-6}	3.02×10^{-6}	1	1.43×10^{12}	941.51
20	2.14×10^{-7}	4.32×10^{-7}	1	1.43×10^{12}	1396.67
25	3.09×10^{-8}	6.19×10^{-8}	1	1.43×10^{12}	1528.38

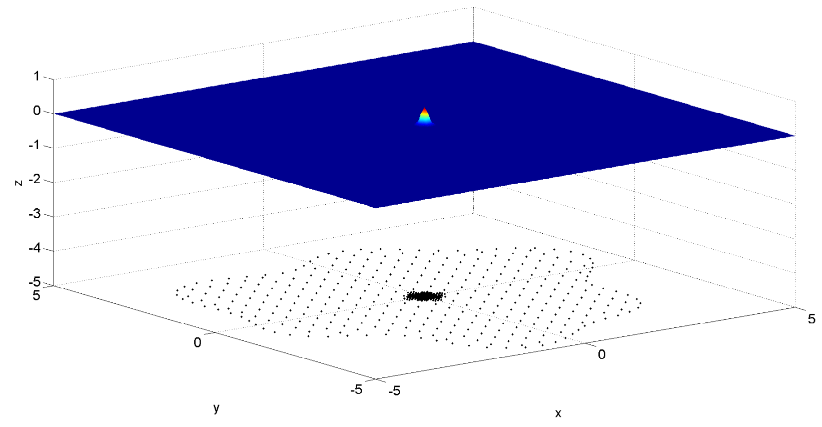


Figure 9: Profile of the final adaptive distribution in Example 5.2 and the exact solution (32) on the non-convex domain $\Omega \subseteq [-5,5] \times [-5,5]$ with $\delta t = 0.01$ and $T = 1$.

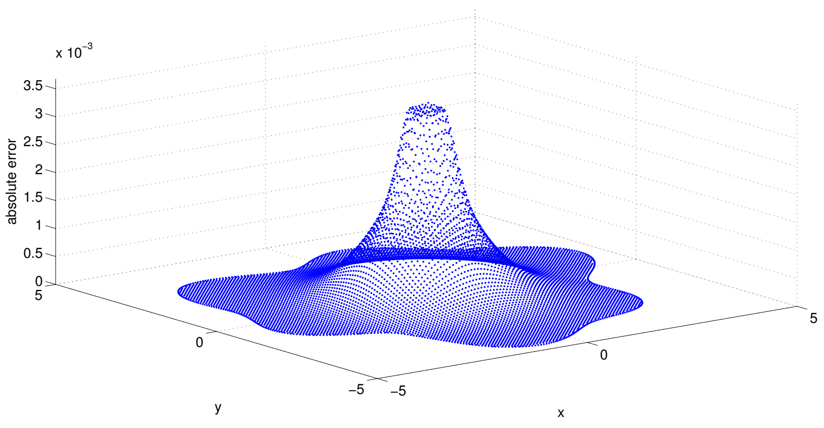


Figure 10: Absolute error on the non-convex domain $\Omega \subseteq [-5,5] \times [-5,5]$ in Example 5.2 with $N = 841$, $\delta t = 0.01$, and $T = 1$.

Example 5.3. *Consider the 2D heat equation* (29) *where f and g are selected based on the following analytical solution*

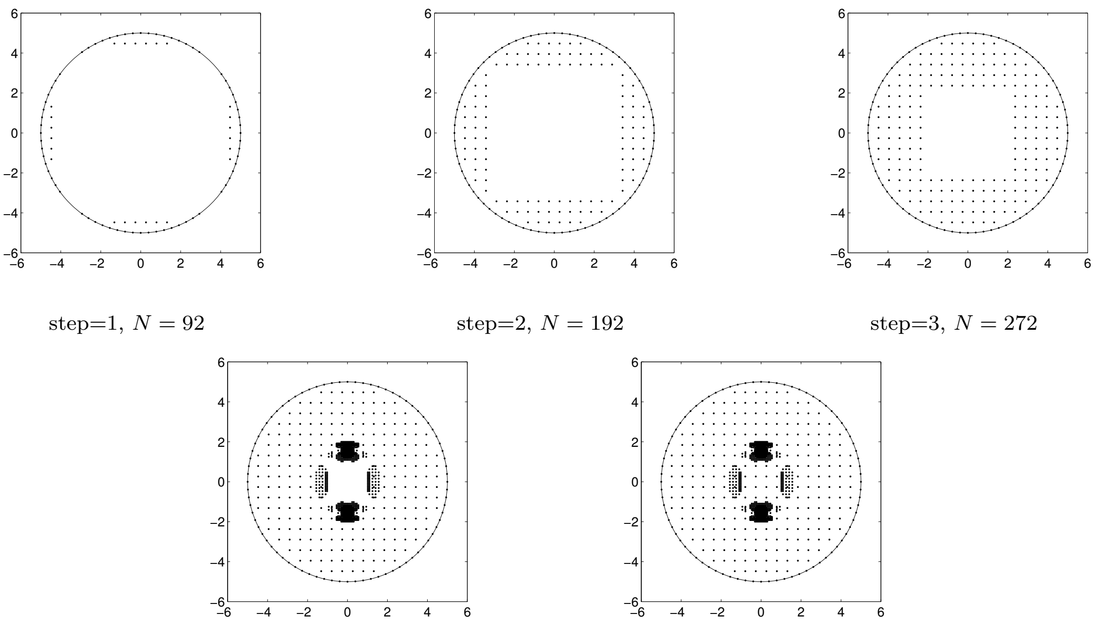
$$u(x,y.t) = \left(e^{-100(x)^2-100(y+1.5)^2} + e^{-100(x)^2-100(y-1.5)^2} \right) e^{-0.2\pi t}. \tag{33}$$

In this last example, the heat problem is solved on the circular domain $\Omega = [-5,5] \times [-5,5]$ showed in Fig. 4 (center). Here, we use as time parameters the constant values $\delta t = 0.1$ and $T = 1$.

As in previous cases, in Fig. 11 we plot the node distributions obtained in the various steps of the algorithm, while in Fig. 12 we show the exact solution together with the final adaptive distribution of nodes derived from application of our adaptive algorithm. The experimental results demonstrate that the adaptive approach with a smaller number of nodes has a greater efficacy and accuracy than the uniform one that involves a denser node distribution, see Table 6. Moreover, Fig. 13 demonstrates the absolute errors calculated at the evaluation points on the circular domain.

In Table 7, the dynamic algorithm is only used in the first step and this adaptive distribution will not change in the advance time steps.

Finally, we conclude this third example observing that RMS error, CPU time and conditioning have a behavior similar to previous cases.



step=1, $N = 92$ step=2, $N = 192$ step=3, $N = 272$

step=4, $N = 1833$ step=5, $N = 1849$

Figure 11: Different steps of the adaptive algorithm in Example 5.3.

Table 6: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.3 on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ with $\delta t = 0.1$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa(A)$	Time(s)
Uniform distribution	7094	1.25×10^{-2}	1.80×10^{-1}	-	6.42×10^{10}	459.29
Adaptive distribution	1849	3.69×10^{-5}	2.92×10^{-4}	1	5.68×10^{10}	344.77

Table 7: Accuracy, condition number, and CPU time for adaptive distribution on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ in Example 5.3 with $\delta t = 0.1$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	5.99×10^{-5}	3.88×10^{-4}	1	5.73×10^{12}	33.25
2	4.16×10^{-5}	2.53×10^{-4}	1	5.73×10^{12}	36.17
3	2.82×10^{-5}	1.39×10^{-4}	1	5.73×10^{12}	47.45
4	2.00×10^{-5}	8.37×10^{-5}	1	5.73×10^{12}	57.79
5	1.50×10^{-5}	5.25×10^{-5}	1	5.73×10^{12}	64.40
10	4.76×10^{-6}	1.13×10^{-5}	1	5.75×10^{12}	73.77
15	1.54×10^{-6}	6.09×10^{-6}	1	5.54×10^{12}	110.05
20	5.11×10^{-7}	3.69×10^{-6}	1	5.71×10^{12}	140.12
25	2.05×10^{-7}	2.63×10^{-6}	1	5.73×10^{12}	175.21

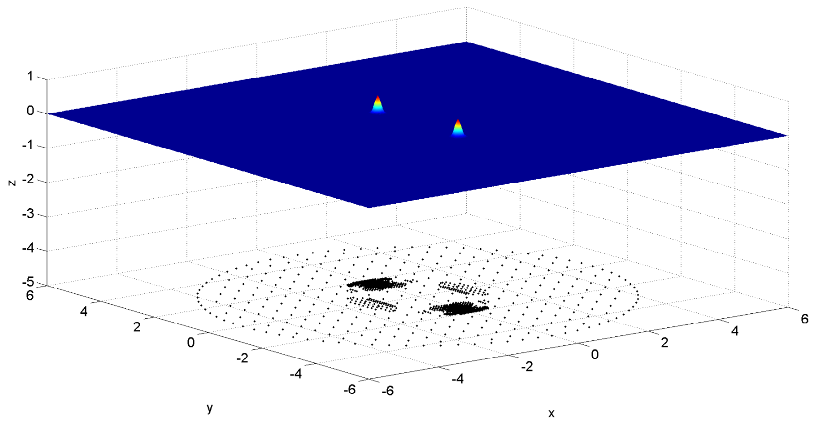


Figure 12: Profile of the final adaptive distribution in Example 5.3 and the exact solution (33) on the circular domain $\Omega = [-5,5]\times[-5,5]$ with $\delta t = 0.1$ and $T = 1$.

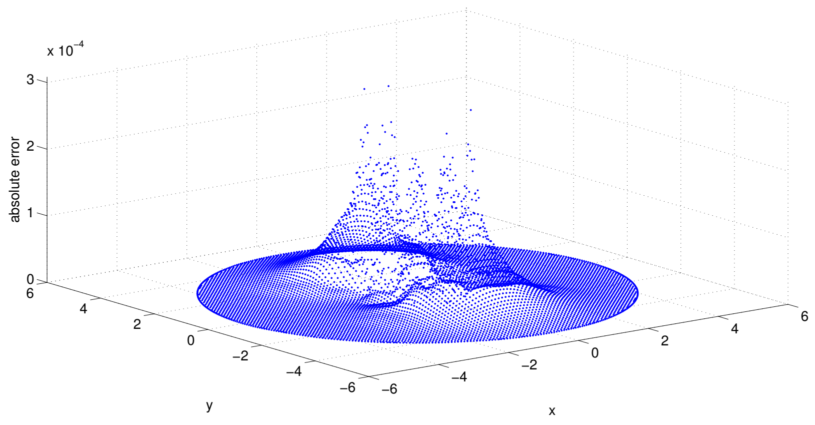


Figure 13: Absolute errors on the circular domain $\Omega = [-5,5]\times[-5,5]$ in Example 5.3 with $N = 1849$, $\delta t = 0.1$, and $T = 1$.

5.2. Hyperbolic PDE

5.2.1. Regular domain

In the following section, we present numerical findings obtained from experiments conducted on the square domain for the wave equation (30).

Example 5.4. *Consider the 2D wave equation* (30) *with the subsequent exact solution*

$$u(x,y,t) = \Big(2e^{-100(x-8)^2-100(y-8)^2} + 2e^{-10(x-1)^2-10(y-1)^2} + 2e^{-10(x-15)^2-10(y-15)^2} + 2e^{-10(x-1)^2-10(y-15)^2} + 2e^{-10(x-15)^2-10(y-1)^2}\Big)e^{-0.2\pi t}. \tag{34}$$

This solution is a function with various areas characterized by fast changes, and an adaptive refinement strategy needs to be applied in each of these nearly singular areas. Our adaptive algorithm is used for solving this problem in the domain $\Omega = [0,16]\times[0,16]$. Further, in all cases, we consider $T = 1$ and $\delta t = 0.1$ as time parameters.

Fig. 14 illustrates the exact solution profile and the ultimate adaptive distribution gained by applying the adaptive algorithm. The outcomes presented in Table 8 demonstrate that our adaptive distribution al-

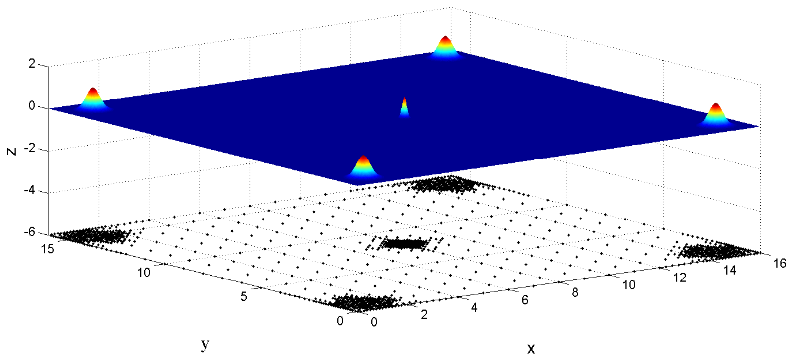


Figure 14: Profile of the final adaptive distribution in Example 5.4 and the exact solution (34) on $\Omega = [0, 16] \times [0, 16]$ with $\delta t = 0.1$ and $T = 1$.

Table 8: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.4 on $\Omega = [0, 16] \times [0, 16]$ with $\delta t = 0.1$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa(A)$	Time(s)
Uniform distribution	6889	3.76×10^{-1}	7.13×10^{0}	_	1.56×10^{12}	505.14
Adaptive distribution	1849	5.15×10^{-4}	7.98×10^{-3}	2	1.10×10^{12}	478.74

gorithm attains superior accuracy with a reduced number of nodes in comparison to the uniform distribution approach, which employs a significantly larger number of nodes.

In addition, Figure 15 illustrates the absolute errors that have been computed at the evaluation points on $\Omega = [0, 16] \times [0, 16]$.

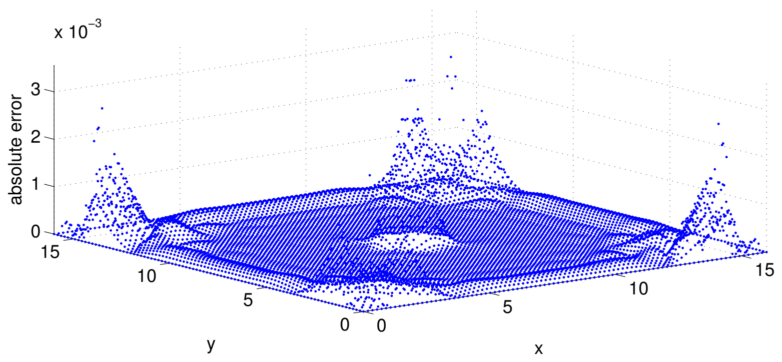


Figure 15: Absolute errors on $\Omega = [0, 16] \times [0, 16]$ in Example 5.4 with $N = 1849$, $\delta t = 0.1$, and $T = 1$.

As previously mentioned, if the position of high gradients corresponding to the solution of the problem across the entire domain remains unchanged at the advance time, our adaptive algorithm will only need to be applied in the initial time step. In such a scenario, the relative nodal positions will remain constant. Consequently, we will only need to solve a single linear system instead of a multitude of linear systems at each time step. Specifically, Table 9 demonstrates that the algorithm is solely utilized in the first step, and this adaptive distribution will remain unaltered in subsequent advance time steps.

Moreover, as expected, these tests point out a similar behavior of error compared to the parabolic PDE case.

Table 9: Accuracy, condition number, and CPU time for adaptive distribution on $\Omega = [0, 16] \times [0, 16]$ in Example 5.4 with $\delta t = 0.1$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	3.07×10^{-4}	3.44×10^{-3}	2	1.51×10^{13}	49.20
2	3.92×10^{-4}	2.91×10^{-3}	2	1.51×10^{13}	67.85
3	4.11×10^{-4}	2.13×10^{-3}	2	1.51×10^{13}	73.08
4	4.54×10^{-4}	1.49×10^{-3}	2	1.51×10^{13}	80.94
5	4.48×10^{-4}	1.19×10^{-3}	2	1.51×10^{13}	91.64
10	4.07×10^{-4}	1.03×10^{-3}	2	1.50×10^{13}	175.94
15	3.53×10^{-4}	7.74×10^{-4}	2	1.51×10^{13}	230.93
20	4.48×10^{-4}	1.18×10^{-3}	2	1.51×10^{13}	193.74
25	2.36×10^{-4}	6.82×10^{-4}	2	1.51×10^{13}	300.05

5.2.2. Irregular domain

This subsection presents the numerical results obtained from experiments conducted to investigate the behavior of our adaptive technique for the wave equation on convex (circular) and non-convex domains, as depicted in Fig. 4.

Example 5.5. *Let us contemplate the 2D wave equation alongside the ensuing exact solution.*

$$u(x, y, t) = \left(e^{-100(x)^2 - 100(y)^2}\right) e^{-0.2\pi t}. \tag{35}$$

In order to demonstrate the effectiveness of our technique on a domain with irregular shape, we have solved the problem on a non-convex domain $\Omega \subseteq [-5, 5] \times [-5, 5]$, the area that is represented in Fig. 4 (left). Specifically, the boundary of this irregular domain Ω is defined as follows:

$$\partial\Omega = \{(r, \theta) \mid r(\theta) = 4 + \frac{4}{10}(\sin(6\theta) + \sin(3\theta))\}.$$

The exact solution (35) is a flat function with an interior point-wise steep peak at center. Moreover, in this example, we take as time parameters the constants $\delta t = 0.01$ and $T = 1$.

Fig. 16 gives a graphical representation of the exact solution and the conclusive adaptive distribution attained via the adaptive scheme. The numerical findings indicate that by utilizing an adaptive node

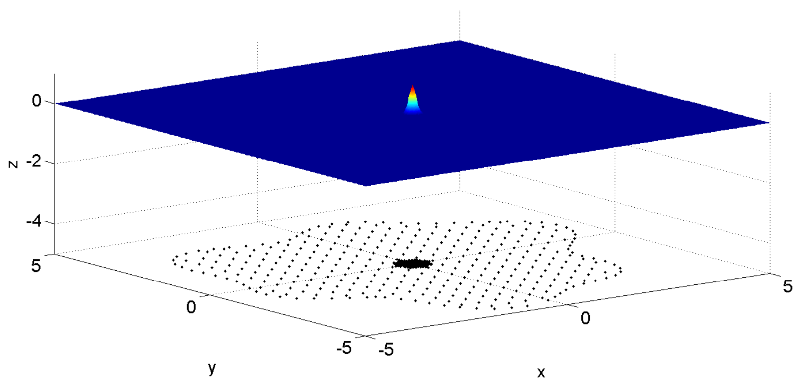


Figure 16: Profile of the final adaptive distribution in Example 5.5 and the exact solution (35) on the non-convex domain $\Omega \subseteq [-5, 5] \times [-5, 5]$ with $\delta t = 0.01$ and $T = 1$.

distribution produced by our algorithm, a higher degree of accuracy can be achieved in comparison to a

Table 10: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.5 on the non-convex domain $\Omega \subseteq [-5,5]\times[-5,5]$ with $\delta t = 0.01$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa(A)$	Time(s)
Uniform distribution	7329	1.11×10^{1}	1.03×10^{2}	_	6.00×10^{12}	3115.35
Adaptive distribution	841	1.18×10^{-4}	1.02×10^{-3}	1	1.78×10^{13}	840.09

Table 11: Accuracy, condition number, and CPU time for adaptive distribution on the non-convex domain $\Omega \subseteq [-5,5]\times[-5,5]$ in Example 5.5 with $\delta t = 0.01$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	6.22×10^{-4}	3.90×10^{-3}	1	1.30×10^{14}	67.95
2	1.12×10^{-3}	3.89×10^{-3}	1	1.34×10^{14}	119.19
3	1.54×10^{-3}	3.14×10^{-3}	1	1.30×10^{14}	180.85
4	1.66×10^{-3}	2.43×10^{-3}	1	1.30×10^{14}	234.15
5	1.16×10^{-3}	1.88×10^{-3}	1	1.28×10^{14}	286.49
10	1.09×10^{-3}	1.96×10^{-3}	1	1.30×10^{14}	577.30
15	1.12×10^{-3}	2.82×10^{-3}	1	1.28×10^{14}	859.68
20	1.17×10^{-3}	2.07×10^{-3}	1	1.30×10^{14}	1132.19
25	1.16×10^{-3}	3.56×10^{-3}	1	1.31×10^{14}	1393.68

uniform distribution, even when the latter involves a substantial number of nodes. This is evidenced in Table 10. Moreover, Fig. 17 displays the absolute errors calculated at the evaluation points on the irregular non-convex domain.

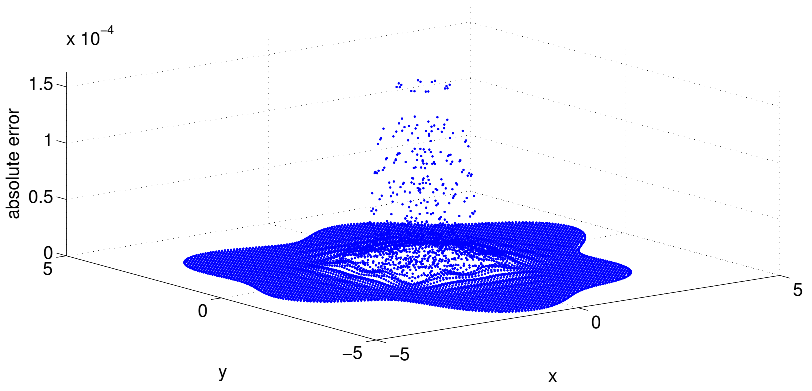


Figure 17: Absolute error on the non-convex domain $\Omega \subseteq [-5,5]\times[-5,5]$ in Example 5.5 with $N = 841$, $\delta t = 0.01$, and $T = 1$.

In Table 11, the dynamic algorithm is only used in the first step and this adaptive distribution will not change in the advance time steps.

Even in this case, our adaptive method (w.r.t. the uniform one) behaves in a quite uniform way: the conditioning is rather similar, whereas the efficiency expressed in CPU time and the accuracy obtained computing the RMS error show that our adaptive scheme performs much better than the uniform one.

Example 5.6. *Consider the 2D wave equation* (30) *where f and g are selected based on the following analytical solution*

$$u(x,y.t) = \left(e^{-100(x)^2-100(y+1.5)^2} + e^{-100(x)^2-100(y-1.5)^2}\right)e^{-0.2\pi t}. \tag{36}$$

Table 12: Comparison of accuracy, condition number, and CPU time between the uniform and adaptive distribution in Example 5.6 on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ with $\delta t = 0.1$ and $T = 1$.

Method	N	RMS	L_∞	δ	$\kappa\,(A)$	Time(s)
Uniform distribution	7094	1.23×10^{-1}	8.12×10^{-1}	_	4.01×10^{11}	365.66
Adaptive distribution	1849	5.26×10^{-5}	4.43×10^{-4}	1	5.76×10^{11}	350.84

In this last example, the heat problem is solved on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ showed in Fig. 4 (center). Here, we use as time parameters the constant values $\delta t = 0.1$ and $T = 1$.

As in previous cases, in Fig. 18 we show the exact solution together with the final adaptive distribution of nodes derived from application of our adaptive algorithm. The experimental results demonstrate that the

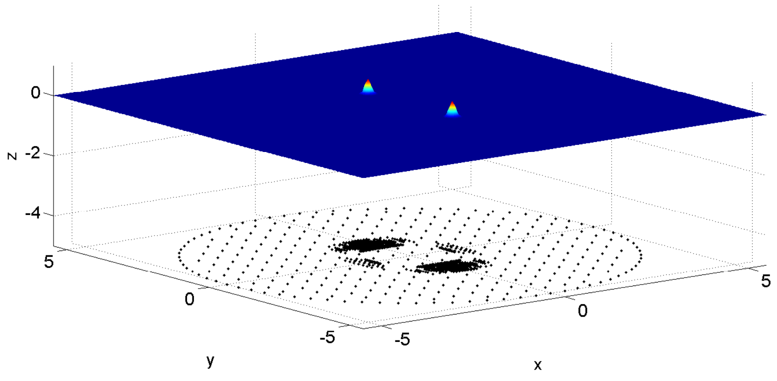


Figure 18: Profile of the final adaptive distribution in Example 5.6 and the exact solution (36) on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ with $\delta t = 0.1$ and $T = 1$.

adaptive approach with a smaller number of nodes has a greater efficacy and accuracy than the uniform one that involves a denser node distribution, see Table 12. Moreover, Fig. 19 demonstrates the absolute errors calculated at the evaluation points on the circular domain.

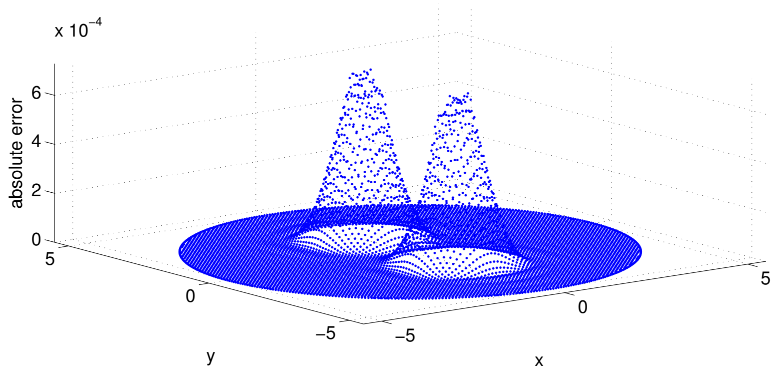


Figure 19: Absolute errors on the circular domain $\Omega = [-5, 5] \times [-5, 5]$ in Example 5.6 with $N = 1849$, $\delta t = 0.1$, and $T = 1$.

In Table 13, the dynamic algorithm is only used in the first step and this adaptive distribution will not change in the advance time steps.

Finally, we conclude this last example observing that RMS error, CPU time and conditioning have a behavior similar to previous cases.

Table 13: Accuracy, condition number, and CPU time for adaptive distribution on the circular domain $\Omega = [-5,5] \times [-5,5]$ in Example 5.6 with $\delta t = 0.1$ and $N = 1849$.

$T(s)$	RMS	L_∞	δ	$\kappa(A)$	Time(s)
1	1.15×10^{-4}	6.94×10^{-4}	1	5.70×10^{11}	35.66
2	2.19×10^{-4}	6.83×10^{-4}	1	5.70×10^{11}	34.88
3	3.12×10^{-4}	7.55×10^{-4}	1	5.70×10^{11}	43.61
4	3.72×10^{-4}	6.77×10^{-4}	1	5.66×10^{11}	51.59
5	3.80×10^{-4}	5.80×10^{-4}	1	5.67×10^{11}	55.84
10	3.03×10^{-4}	6.57×10^{-4}	1	5.70×10^{11}	86.02
15	1.24×10^{-4}	6.82×10^{-4}	1	5.69×10^{11}	113.00
20	2.57×10^{-4}	2.11×10^{-3}	1	5.70×10^{11}	147.60
25	5.46×10^{-4}	3.27×10^{-3}	1	5.68×10^{11}	176.30

6. Conclusions and future work

In this paper, we proposed a novel adaptive meshfree technique to solve the time-dependent PDEs in two-dimensional domains. The dynamic algorithm introduced in this study finds areas with fast changes and applies a local node adaptive approach only in the almost singular areas. Moreover, in this algorithm, unlike Kansa approach, the RBF collocation technique was mixed with a FD scheme. Using this idea, the novel adaptive algorithm is simply applied on time steps rather than on the entire domain (like other available adaptive algorithms). Therefore, this benefit overwhelms possible problems existing in other adaptive algorithms. Moreover, if the position of high gradients or fast changes in the solution does not change at an advanced time, we can only utilize our dynamic algorithm in the first time step. Indeed, in such a condition relative nodal locations will not change. Therefore, the approach requires to solve only one linear system instead of many linear systems on each time step. This causes a remarkable saving of memory space and time. Consequently, we solve a new linear system only once, as the coefficient matrix is not changed in the different time steps. This research shows that the adaptive method can reach a particular accuracy level with a considerable lower computational effort compared to the uniform RBF-FD technique. The numerical study presented in this work demonstrates high accuracy can be acquired for adaptive distributions even if a reduced number of nodes is necessary. On the contrary, numerous nodes are required to obtain relatively high accuracy for the uniform distribution. The considered examples show good efficiency of this technique based on CPU time-saving and convergence rate. Therefore, our adaptive scheme showed that we can find the almost singular areas over the entire domain.

As a future work, we intend to implement a new modified algorithm for the localized RBF-partition of unity method (RBF-PUM), which might also be used with other kinds of kernel bases to solve 3D time-dependent problems. For an overview on RBF-PUMs, see e.g. [12, 17] and references therein.

Acknowledgments

The authors sincerely thank the anonymous referee(s) for the valuable comments and suggestions, which enabled to significantly improve the quality of this paper. The third author acknowledges financial support from the INdAM–GNCS Project "Computational methods for kernel-based approximation and its applications", code CUP_E55F22000270001. Moreover, this work was partially supported by the Department of Mathematics "G. Peano"of the University of Torino through 2020 project "Mathematical methods in computational sciences". This research has been accomplished within the RITA (Research ITalian network on Approximation) and the UMI Group TAA (Approximation Theory and Applications).

Declarations

Conflict of interest The authors declare that they have no any competing interests.

References

[1] M. Ahmad, Meshless analysis of parabolic interface problems, Eng. Anal. Bound. Elem. 94 (2018) 134-152.

[2] M. Ahmad, B. Ullah, Local radial basis function collocation method for stokes equations with interface conditions, Eng. Anal. Bound. Elem. 119 (2020) 246-256.

[3] W.F. Ames, Numerical methods for partial differential equations, Academic Press, 2014.

[4] D.C. Arney, J.E. Flaherty, An adaptive mesh-moving and local refinement method for time-dependent partial differential equations, ACM Trans. Math. Software, 16 (1990) 48-71.

[5] J. Bell, M. Berger, J. Saltzman, M. Welcome, Three-dimensional adaptive mesh refinement for hyperbolic conservation laws, SIAM J. Sci. Comput. 15 (1) (1994) 127-138.

[6] J.P. Boyd, Trouble with Gegenbauer reconstruction for defeating Gibbs phenomenon: Runge phenomenon in the diagonal limit of Gegenbauer polynomial approximations, J. Comput. Phys. 204 (2005) 253-264.

[7] I. Boztosun, A. Charafi, An analysis of the linear advection-diffusion equation using mesh-free and mesh-dependent methods, Eng. Anal. Bound. Elem. 26 (2002) 889-895.

[8] M.D. Buhmann, Radial basis functions: theory and implementations, In: Cambridge monographs on applied and computational mathematics, vol. 12., Cambridge University Press, 2003.

[9] M. Buhmann, N. Dyn, Spectral convergence of multiquadric interpolation, Proc. Edinburgh Math. Soc. 36 (2) (1993) 319-333.

[10] D. Burgarelli, M. Kischinhevsky, R.J. Biezuner, A new adaptive mesh refinement strategy for numerically solving evolutionary PDE's, J. Comput. Appl. Math. 196 (1) (2006) 115-131.

[11] R.E. Carlson, T. Foley, The parameter R^2 in multiquatric interpolation, Comput. Math. Appl. 21 (1991) 29-42.

[12] R. Cavoretto, A numerical algorithm for multidimensional modeling of scattered data points, Comput. Appl. Math. 34 (2015) 65-80.

[13] R. Cavoretto, A. De Rossi, Adaptive meshless refinement schemes for RBF-PUM collocation, Appl. Math. Lett. 90 (2019) 131-138.

[14] R. Cavoretto, A. De Rossi, Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme, Appl. Math. Comput. 369 (2020) 124824.

[15] R. Cavoretto, A. De Rossi, A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs, Comput. Math. Appl. 79 (2020) 3206-3222.

[16] R. Cavoretto, Adaptive radial basis function partition of unity interpolation: A bivariate algorithm for unstructured data, J. Sci. Comput. 87 (2021) 41.

[17] R. Cavoretto, A. De Rossi, W. Erb, Partition of unity methods for signal processing on graphs, J. Fourier Anal. Appl. 27 (2021) 66.

[18] R. Cavoretto, Adaptive LOOCV-based kernel methods for solving time-dependent BVPs, Appl. Math. Comput. 429 (2022) 127228.

[19] R. Cavoretto, A. De Rossi, An adaptive residual sub-sampling algorithm for kernel interpolation based on maximum likelihood estimations, J. Comput. Appl. Math. 418 (2023) 114658.

[20] S. Çayan, B. B. Özhan, M. Sezer, An adaptive approach for solving fourth-order partial differential equations: algorithm and applications to engineering models, Comput. Appl. Math. 41 8 (2022) 1-17.

[21] AH-D Chengi, M.A. Golberg, E.J. Kansa, G. Zammito, Exponential convergence and h–c multiquadric collocation method for partial differential equations, Numer. Methods Partial Differ. Eq. 19 (5) (2003) 571-594.

[22] P.P. Chinchapatnam, K. Djidjeli, P.B. Nair, Unsymmetric and symmetric meshless schemes for the unsteady convection-diffusion equation, Comput. Methods Appl. Mech. Engrg. 195 (2006) 2432-2453.

[23] S.F. Davis, J.E. Flaherty, An Adaptive Finite Element Method for Initial-Boundary Value Problems for Partial Differential Equations, SIAM J. Sci. Stat. Comp. 3 (1) (1982) 6-22.

[24] M. Dehghan, A. Shokri, A numerical method for two-dimentional Schrödinger equation using collocation radial basis functions, Comput. Math. Appl. 54 (2007) 136-146.

[25] V. Dolejsi, Anisotropic hp-adaptive discontinuous Galerkin method for the numerical solution of time dependent PDEs, Appl. Math. Comput. 267 (2015) 682-697.

[26] T.A. Driscoll, A.R.H. Heryudono, Adaptive residual subsampling for radial basis function interpolation collocations problems, Comput. Math. Appl. 53 (2007) 927-939.

[27] M. Esmaeilbeigi, M.M. Hosseini, S.T. Mohyud-Din, A new approach of the radial basis functions method for telegraph equations, Int. J. Phys. Sci. 6 (6) (2011) 1517-1527.

[28] M. Esmaeilbeigi, M.M. Hosseini, Dynamic node adaptive strategy for nearly singular problems on large domains, Eng. Anal. Bound. Elem. 36 (2012) 1311-1321.

[29] M. Esmaeilbeigi, G. Garmanjani, A shift-adaptive meshfree method for solving a class of initial-boundary value problems with moving boundaries in one-dimensional domain, Numer. Methods Partial Differ. Eq. 32 (6) (2016) 1622-1646.

[30] M. Esmaeilbeigi, G. Garmanjani, Gaussian radial basis function interpolant for the different data sites and basis centers, Calcolo, 54 (1) (2017) 155-166.

[31] M. Esmaeilbeigi, M. Paripour, G. Garmanjani, Approximate solution of the fuzzy fractional Bagley-Torvik equation by the RBF collocation method, Comput. Methods Differ. Equ. 6.2 (2018) 186-214.

[32] M. Esmaeilbeigi, O. Chatrabgoun, An efficient method based on RBFs for multilayer data interpolation with application in air pollution data analysis, Comput. Appl. Math. 38.4 (2019) 1-20.

[33] G.E. Fasshauer, Meshfree Approximation Methods with Matlab, World Scientific, Singapore, 2007.

[34] G.E. Fasshauer, Positive definite kernels: Past, present and future, Dolomites Res. Notes Approx. 4 (2011) 21-63.

[35] B. Fornberg, J. Zuev, The Runge phenomenon and spatially variable shape parameters in RBF interpolation, Comput. Math. Appl. 54 (2007) 379-398.

[36] R. Franke, Scattered data interpolation: test of some methods, Math. Comput. 38 (1982) 181-200.

[37] G. Garmanjani, R. Cavoretto, M. Esmaeilbeigi, A RBF partition of unity collocation method based on finite difference for initial–boundary value problems, Comput. Math. Appl. 75 (11) (2018) 4066-4090.

[38] L. Ge, T. Sun, An adaptive hp-version stochastic Galerkin method for constrained optimal control problem governed by random reaction diffusion equations, Comput. Appl. Math. 41 3 (2022) 1-30.

[39] A. Golbabai, E. Mohebianfar, H. Rabiei, On the new variable shape parameter strategies for radial basis functions, Comput. Appl. Math. 34 (2015) 691-704.

[40] Y.C. Hon, R. Schaback, X. Zhou, An adaptive greedy algorithm for solving large RBF collocation problems, Numer. Algorithms, 32 (1) (2003) 13-25.

[41] M. Hussain, S. Haq, A hybrid radial basis functions collocation technique to numerically solve fractional advection-diffusion models, Numer. Methods Partial Differ. Eq. (2020), to appear; https://doi.org/10.1002/num.22472

[42] Z. Jannesari, M. Tatari, An adaptive strategy for solving convection dominated diffusion equation, Comput. Appl. Math. 39 2 (2020) 1-15.

[43] M.N. Khan, I. Hussain, I. Ahmad, H. Ahmad, A local meshless method for the numerical solution of space-dependent inverse heat problems, Math. Methods Appl. Sci. 44 (4) (2021) 3066-3079.

[44] P.G. Lefloch, Hyperbolic systems of conservation laws, Oxford University Press, 2000.

[45] J. Lu, Y. Nie, A collocation method based on localized radial basis functions with reproducibility for nonlocal diffusion models, Comput. Appl. Math. 40 8 (2021) 1-23.

[46] F.M.B. Martinez, Meshless methods for elliptic and free-boudary problems, PhD thesis, 2008.

[47] S.A. Sarra, Adaptive radial basis function methods for time dependent partial differential equations, Appl. Numer. Math. 54 (1) (2005) 79-94.

[48] V. Singh, R.K. Mohanty, Local meshless method for convection dominated steady and unsteady partial differential equations, Eng. Comput. 35 (2019) 803-812.

[49] Z. Ullah, B. Ullah, W. Khan, Proportional topology optimisation with maximum entropy-based meshless method for minimum compliance and stress constrained problems, Eng. Comput. 38 (6) (2022) 5541-5561.

[50] H. Wendland, Scattered Data Approximation, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.

[51] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. comput. Math. 4 (1995) 389-396.

[52] Z. Wu, Dynamically knots setting in meshless method for solving time dependent propagation equation, Comput. Methods. Appl. Mech. Eng. 193 (12-14) (2004) 1221-1229.