A new algorithm for computing moments of complex non-central Wishart distributions

E. Di Nardo*

elvira.dinardo@unibas.it

http://www.unibas.it/utenti/dinardo/Tel: +39 0971205890, Fax: +39 0971205896

G. Guarino**

giuseppe.guarino@aspbasilicata.it

* Dipartimento di Matematica e Informatica, Università degli Studi della Basilicata,Viale dell'Ateneo Lucano n.10, 85100 Potenza, Italy

**Medical School, Università del Sacro Cuore (Rome branch), Largo Agostino Gemelli n.8, 00168 Roma, Italy

Introduction

Abstract: A new algorithm for computing joint moments of complex non-central Wishart distributions W is provided, relied on a symbolic method which is particularly suited to be implemented for multivariate statistical distributions. The joint moments we compute have the form

$$E\left[Tr\left(W\cdot H_1 \right)^{i_1} \cdot Tr\left(W\cdot H_2 \right)^{i_2} \cdot \cdots \cdot Tr\left(W\cdot H_m \right)^{i_m} \right] \qquad (1)$$

with $i = (i_1, i_2, ..., i_m)$ a multi-index of non-negative integers and $H_1, H_2, ..., H_m$ complex matrices. Since the non-central Wishart random matrix results to be the convolution of two different random matrices $W = Wc + A$, one linked to the central distribution and the other involving formal variables, the main idea is to apply a suitable binomial expansion, by using multisets subdivisions, and then insert the moments of these two different random matrices computed in a separate procedure. Again multiset subdivisions are employed to compute also these moments. In particular, the moments of the associated central Wishart distribution are constructed by using a combinatorial device, the necklace, which takes advantage of the cyclic property of the trace. In the literature, the existing algorithms to compute joint moments (1) make use of multivariable derivative of the moment generating function associated to the complex non-central Wishart distribution, which has the following quite complicated expression:

$$f(z_1, z_2, \ldots, z_n) := \frac{\exp\left(-Tr\left(\frac{\Omega\Sigma(H_1 z_1 + \ldots + H_m z_m)}{I - \Sigma(H_1 z_1 + \ldots + H_m z_m)}\right)\right)}{\det\left(I - \Sigma(H_1 z_1 + \ldots + H_m z_m)\right)^n}$$

with I identity matrix, Σ the covariance matrix and Ω the non-centrality matrix.

Application Areas/Subject: Combinatorics & algebraic methods
Keywords: Convolution, multiset subdivision, necklace, joint moment, trace
See Also: Background on multiset subdivisions and multivariate Faà di Bruno's formula, see [2,3].

Initialization

```
> restart
> with(combinat, partition, multinomial, permute);
```

[*partition*, *multinomial*, *permute*] **(2.1)**

Multiset subdivisions

The function makeTab has been extensively discussed in [2]: here we just mention that the procedure makeTab performs multiset subdivisions. Informally, a subdivision is a partition of a multiset. See the subsequent examples .

The list of all partitions of a set with 3 blocks is:

$$\{\{\alpha_1, \alpha_2, \alpha_3\}\}, \{\{\alpha_1, \alpha_2\}, \{\alpha_3\}\}, \{\{\alpha_2\}, \{\alpha_1, \alpha_3\}\}, \{\{\alpha_1\}, \{\alpha_2, \alpha_3\}\}, \{\{\alpha_1\}, \{\alpha_2\}, \{\alpha_3\}\}.$$

Setting $[\alpha_1 = \alpha, \alpha_2 = \alpha, \alpha_3 = \beta]$ we obtain:

$$[[\alpha^2 \beta], [\alpha^2, \beta], [\alpha, \alpha, \beta], [\alpha, \alpha\beta], [\alpha, \alpha\beta]]$$

Compacting the previous output we obtain:

$$[[[\alpha\beta, \alpha], 2], [[\alpha, \alpha, \beta], 1], [[\alpha^2 \beta], 1], [[\alpha^2, \beta], 1]]$$

http://www.maplesoft.com/applications/view.aspx?SID=33039

The Maple routines and examples

The Maple routines

```
> nRep := proc(u) mul(x2!, x = convert(u, multiset)) end proc:

  URv := proc(u, v)
      local U, ou, i, ptr, vI;
      ou := NULL; U := [ ]; vI := indets(v);
      for ptr from nops(u) by -1 to 2 do
          if has(u[ptr], v) then break end if
      end do;
      for i from ptr to nops(u) do
          if not (u[i] = ou or has(u[i], vI)) then
            ou := u[i];
            U := [op(U), [op(u[1..i-1]), u[i]*v, op(u[i+1..-1])]]
          end if
      end do;
      op(U), [op(u), v]
  end proc:

  URV := proc( )
      local U, V, i;
      U := [args[1, 1]]; V := args[2, 1];
      for i to nops(V) do U := [seq(URv(u, V[i]), u = U)] end do;
      seq([x, args[1, 2]*args[2, 2]], x = U)
  end proc:

  URmV := proc( )
      local U, i;
      if nargs = 1 then args else
          U := URV(args[1], args[2]);
          for i from 3 to nargs do
              U := seq(URV(u, args[i]), u = [U])
          end do;
          seq([x[1], x[2]/nRep(x[1])], x = U)
      end if
  end proc:

  comb := proc(V, ptr, Y)
      if ptr = nops(V) + 1 then return Y end if;
      seq(comb(V, ptr + 1, [op(Y), L]), L = V[ptr])
  end proc:

  makeTab := proc( )
      local U;
      U := [seq(`if`(args[i] = 0, NULL, [seq([[seq(alpha[i]^z, z = y)], multinomial(args[i],
      seq(r, r = y))], y = partition(args[i]))]), i = 1..nargs)];
```

```
    if nops(U) = 1 then [seq([x[1], x[2]/nRep(x[1])], x = op(U))]
                   else [seq(URmV(op(x)), x = [comb(U, 1, [ ])])] end if
end proc:
```

Examples

In (3.1.2.1) the multiset $\{\alpha_1, \alpha_1\}$ is considered with α_1 having multiplicity 2.
The subdivisions are:
$\{\alpha_1\}$, $\{\alpha_1\}$, denoted in the output with $[[\alpha_1, \alpha_1], 1]$
$\{\alpha_1, \alpha_1\}$ denoted in the output with $[[\alpha_1^2], 1]$

```
> makeTab(2)
```

$$[[[\alpha_1, \alpha_1], 1], [[\alpha_1^2], 1]] \qquad \textbf{(3.1.2.1)}$$

In (3.1.2.2) the multiset $\{\alpha_1, \alpha_2\}$ is considered with α_1 having multiplicity 1 and α_2 having multiplicity 1.
The subdivisions are:
$\{\alpha_1\}$, $\{\alpha_2\}$, denoted in the output with $[[\alpha_1, \alpha_2], 1]$
$\{\alpha_1, \alpha_2\}$, denoted in the output with $[[\alpha_1\, \alpha_2], 1]$

```
> makeTab(1, 1)
```

$$[[[\alpha_1\, \alpha_2], 1], [[\alpha_1, \alpha_2], 1]] \qquad \textbf{(3.1.2.2)}$$

In (3.1.2.3) the multiset $\{\alpha_1, \alpha_1, \alpha_2\}$ is considered with α_1 having multiplicity 2 and α_2 having multiplicity 1.
The subdivisions are:
$\{\alpha_1, \alpha_2\}$, $\{\alpha_1\}$, denoted in the output with $[[\alpha_1\, \alpha_2, \alpha_1], 2]$
$\{\alpha_1\}$, $\{\alpha_1\}$, $\{\alpha_2\}$, denoted in the output with $[[\alpha_1, \alpha_1, \alpha_2], 1]$
$\{\alpha_1\}$, $\{\alpha_2\}$, denoted in the output with $[[\alpha_1^2, \alpha_2], 1]$
$\{\alpha_1, \alpha_1, \alpha_2\}$, denoted in the output with $[[\alpha_1^2\, \alpha_2], 1]$

```
> makeTab(2, 1)
```

$$[[[\alpha_1\, \alpha_2, \alpha_1], 2], [[\alpha_1, \alpha_1, \alpha_2], 1], [[\alpha_1^2\, \alpha_2], 1], [[\alpha_1^2, \alpha_2], 1]] \qquad \textbf{(3.1.2.3)}$$

MFB Function

The procedure *MF*B computes the moments of the complex central Wishart distribution *Wc* and those of the matrix *A* whose entries are suitable formal variables. In [1], these moments correspond to formulae (4.9) and (4.10) respectively. Here we just mention that in order to compute the i-th coefficient of the composition of two multivariable formal power series, the function *MFB* does not compute nested partial derivatives but makes use of the procedure *makeTab*.

See http://www.maplesoft.com/applications/view.aspx?SID=101396 for details.

The Code

```
> MFB :=proc( )
      option remember;
      local n, vIndets, E;
      n := add(args_i, i = 1 ..nargs);
      if n = 0 then return 1 end if;
      vIndets := [seq(α_i, i = 1 ..nargs)];
      E := add(f_nops(y_1)·y_2·mul(g_seq(degree(x, vIndets_i), i = 1 .. nops(vIndets)), x = y_1), y
       = makeTab(args))
    end proc:
```

Examples

In (4.2.1) the coefficient of order 2 of $f(g(x))$ that is $\left(\frac{d^n}{dx^n} f(g(x))\right)$ is computed

```
> MFB(2);
```

$$f_2 g_1^2 + f_1 g_2 \qquad \textbf{(4.2.1)}$$

In (4.2.2) the coefficient of order 2 of $f(g(x_1, x_2))$ that is $\left(\frac{\partial^2}{\partial x_1 \partial x_2} f(g(x_1, x_2))\right)$ is computed

```
> MFB(1, 1);
```

$$f_1 g_{1,1} + f_2 g_{1,0} g_{0,1} \qquad \textbf{(4.2.2)}$$

In (4.2.3) the coefficient of order 2 of $f(g(x_1, x_2, x_3))$ that is $\left(\frac{\partial^3}{\partial x_1 \partial x_2 \partial x_3} f(g(x_1, x_2, x_3))\right)$ is computed

```
> MFB(1, 1, 1)
```

$$f_1 g_{1,1,1} + f_2 g_{1,1,0} g_{0,0,1} + f_2 g_{1,0,1} g_{0,1,0} + f_2 g_{1,0,0} g_{0,1,1} + f_3 g_{1,0,0} g_{0,1,0} g_{0,0,1} \qquad \textbf{(4.2.3)}$$

Algorithm for computing joint moments of complex non-central Wishart distributions

In this section we introduce the Maple algorithm to perform the symbolic computation of joint moments (1).
The computation is split in more than one procedure, which are explained in details in the following

The mkT function

The function mkT has been extensively discussed in [3]: here we just mention that the procedure mkT gives the list of all subvectors having the same lenght of an assigned vector V, given in input, and such that their summation returns the vector V.

http://www.maplesoft.com/applications/view.aspx?SID=33039 for details.

The Code

```
> mkT := proc(V, n)
    local vE, L, nV;
    nV := nops(V);
    vE := [seq(α_i, i = 1 ..nV)];
    L := seq(`if`(nops(x_1) <= n, x_1, NULL), x = makeTab(op(V)));
    L := seq([seq([seq(degree(y, z), z = vE)], y = x), [0$nV]$(n - nops(x))], x = [L]);
    L := seq(op(permute(x)), x = [L]);
  end:
```

Example

In (5.1.1) the input vector V=[1,1] is split in 2 subvectors, having the same length of V and whose summation returns [1,1].
The print is in orizontal mode

```
> mkT([1, 1], 2);
```

$$[[1, 1], [0, 0]], [[0, 0], [1, 1]], [[1, 0], [0, 1]], [[0, 1], [1, 0]] \qquad \textbf{(5.1.1)}$$

The print is in vertical mode

```
> for L in mkT([2, 1], 2) do print(L); od;
```

$$[[1, 1], [1, 0]]$$
$$[[1, 0], [1, 1]]$$
$$[[2, 1], [0, 0]]$$
$$[[0, 0], [2, 1]]$$
$$[[2, 0], [0, 1]]$$
$$[[0, 1], [2, 0]] \qquad \textbf{(5.1.2)}$$

The leftShift function

Left Shift of a vector: each element of the output vector is obtained by shifting by one position to the left the corresponding element of the input vector, given in arg1.

The Code

```
> leftShift := v -> [ op(v[2..-1]), v[1]];
```

$$leftShift := v \rightarrow [\, op(v_{2..-1}), v_1 \,] \qquad \textbf{(5.2.1)}$$

Example

In (5.2.2) the vector [1,2,3,4] is shifted in [2,3,4,1].

```
> leftShift( [1, 2, 3, 4 ])
```

$$[2, 3, 4, 1] \qquad \textbf{(5.2.2)}$$

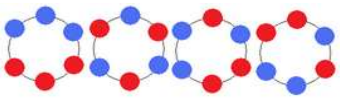
The Mnecklaces function

Let us consider the multiset $\{\underbrace{1,\ldots,1}_{i_1}\underbrace{2,\ldots,2}_{i_2},\ldots,\underbrace{m,\ldots,m}_{i_m}\}$

The procedure Mnecklaces gives in output all necklaces which can be constructed by permuting the elements of the multiset, passed as *arg1*. In combinatorics, a m-ary necklace of length n is an equivalence class of n-character strings over an alphabet of size m. Here n is equal to $i_1 + i_2 + \ldots + i_m$ given the lexicographically smallest string, called the representative, all other elements of the necklace can be obtained by circular rotation of the representative. A necklace represents a structure with n circularly connected beads of up to m different colors.

Depending on the value assigned to arg2, the following outputs are performed:
0: print only the cardinality of necklaces, grouping and enumerating those having the same cardinality;
1: print only the cardinality of necklaces in a list;
2: print the cardinality and the representative of necklaces;
3: print all element for all necklaces. The first is the representative;
4: print only the number of necklaces.

Example: The four necklaces which can be made with 3 red beads and 3 blue beads are the follow:



You can obtain the same result by using:

Mnecklaces([[1, 3], [2, 3]], 4) →4

Instead *Mnecklaces*([[1, 3], [2, 3]], 2) generates:

[6, [1, 1, 1, 2, 2, 2]]
[6, [1, 1, 2, 1, 2, 2]]
[6, [1, 1, 2, 2, 1, 2]]
[2, [1, 2, 1, 2, 1, 2]]

or using colors

Mnecklaces([[*blue*, 3], [*red*, 3]], 2) generates:

[6, [red, red, red, blue, blue, blue]]
[6, [red, red, blue, red, blue, blue]]
[6, [red, red, blue, blue, red, blue]]
[2, [red, blue, red, blue, red, blue]]

The Code

```
> Mnecklaces := proc(V, flag)
    local i, n, u, v, L, U;
    L := [seq(x[1]$x[2], x = V)];
    n := nops(L);
    if n = 0 then return(NULL); fi;
    L := {op(permute(L))};
    v := L[1]; u := [v]; U := [];
    while true do
      L := L minus {v};
      for i from 1 to n - 1 do
        v := leftShift(v);
        if {v} subset L then
          L := L minus {v};
          u := [op(u), v];
        fi;
      od;
      U := [op(U), u];
      if nops(L) = 0 then break; fi;
      v := L[1]; u := [v];
    od;
    # ----- Output Management ----- #
    if flag = 0 then
```

```
    convert([seq(nops(x), x = U)], multiset);
  elif flag = 1 then
    sort([seq(nops(x), x = U)]);
  elif flag = 2 then
    [seq([nops(x), x_1], x = U)];
  elif flag = 3 then
    U;
  else nops([seq([nops(x), x_1], x = U)]); fi;
end:
```

>

Example

In the following, necklaces that can be made with 4 beads, labelled with 1, and 2 beads, labelled with 2.

> *Mnecklaces*([[1, 4], [2, 2]], 0)

[[3, 1], [6, 2]] **(5.3.1)**

In the following, printing of different outputs for necklaces made with 6 beads of 2 different colors, labelled with 1 and 2 respectively.

> Mnecklaces([[1, 3], [2, 3]], 0)

[[2, 1], [6, 3]] **(5.3.2)**

> *Mnecklaces*([[1, 3], [2, 3]], 1)

[2, 6, 6, 6] **(5.3.3)**

> *Mnecklaces*([[1, 3], [2, 3]], 2)

[[6, [1, 1, 1, 2, 2, 2]], [6, [1, 1, 2, 1, 2, 2]], [6, [1, 1, 2, 2, 1, 2]], [2, [1, 2, 1, 2, 1, 2]]] **(5.3.4)**

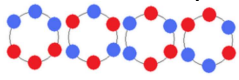
> *Mnecklaces*([[1, 3], [2, 3]], 3)

[[[1, 1, 1, 2, 2, 2], [1, 1, 2, 2, 2, 1], [1, 2, 2, 2, 1, 1], [2, 2, 2, 1, 1, 1], [2, 2, 1, 1, 1, 2], [2, 1, 1, 1, 2, 2]], [[1, 1, 2, 1, 2, 2], [1, 2, 1, 2, 2, 1], [2, 1, 2, 2, 1, 1], [1, 2, 2, 1, 1, 2], [2, 2, 1, 1, 2, 1], [2, 1, 1, 2, 1, 2]], [[1, 1, 2, 2, 1, 2], [1, 2, 2, 1, 2, 1], [2, 2, 1, 2, 1, 1], [2, 1, 2, 1, 1, 2], [1, 2, 1, 1, 2, 2], [2, 1, 1, 2, 2, 1]], [[1, 2, 1, 2, 1, 2], [2, 1, 2, 1, 2, 1]]] **(5.3.5)**

> *Mnecklaces*([[1, 3], [2, 3]], 4)

4 **(5.3.6)**

A different output can be obtained by using colors: 3 red and 3 blue beads.



```
> for L in Mnecklaces([[blue, 3], [red, 3]], 2) do print(L[2]); od;
```

[red, red, red, blue, blue, blue]

[red, red, blue, red, blue, blue]

[red, red, blue, blue, red, blue]

[red, blue, red, blue, red, blue] **(5.3.7)**

The m2v function

The routine m2v transforms a multiset given in args1 as input in a list. A parenthesis is inserted in between different symbols. This because, after having generated a necklace and its representative, is necessary to group equal characters as input to the next step.

The code

```
> m2v := proc(V)
    local u, U, v, i; u := [ ]; U := [ ];
    v := V[1];
    for i from 1 to nops(V) do
      if V[i] = v then u := [op(u), V[i]];
                  else v := V[i];
                       U := [op(U), u];
                       u := [v];
      fi;
    od;
    [op(U), u];
  end:
```

Example

In (5.4.2.1) the input is the multiset [2,2,1,1,1,2,1]. This list is transformed in [2, 2],[1, 1, 1], [2],[1]

```
> m2v([2, 2, 1, 1, 1, 2, 1])
```

[[2, 2], [1, 1, 1], [2], [1]] **(5.4.2.1)**

Some useful sub-functions

cn function

Cn takes as input the output of m2v and associates to each block in the list a product of matrices having the same indexes in the block. Therefore, for grouping with cardinality more than 1, powers of matrices will be given.

cno function

Cno does the same of cn, but for more than one list. Different lists are linked in correspondence with different elements of a summation.

conv & convo function

The procedures conv and convo make use of cn and cno with input lists produced by the procedure Mnecklaces.

The Codes

> $cn := (V, c) \to c / nops(V) * Tr(seq(mul(B_y, y = x), x = m2v(V)))$

$$cn := (V, c) \to \frac{c\, Tr(seq(mul(B_y, y = x), x = m2v(V)))}{nops(V)} \quad \textbf{(5.5.1.1)}$$

> $cno := V \to add(Tr(\Omega, seq(mul(B_y, y = x), x = m2v(v))), v = V)$

$$cno := V \to add(Tr(\Omega, seq(mul(B_y, y = x), x = m2v(v))), v = V) \quad \textbf{(5.5.1.2)}$$

> $conv := M \to add(cn(v_2, v_1), v = Mnecklaces(M, 2))$

$$conv := M \to add(cn(v_2, v_1), v = Mnecklaces(M, 2)) \quad \textbf{(5.5.1.3)}$$

> $convo := M \to add(cno(v), v = Mnecklaces(M, 3))$

$$convo := M \to add(cno(v), v = Mnecklaces(M, 3)) \quad \textbf{(5.5.1.4)}$$

Examples

In (5.5.2.1) the steps are:
$([1, 1, 1, 2, 2, 1, 2], 6) \to [1, 1, 1], [2, 2], [1], [2] \to 6\, Tr(B_1^3, B_2^2, B_1, B_2)$

> $cn([1, 1, 1, 2, 2, 1, 2], 6);$

$$\frac{6}{7}\, Tr(B_1^3, B_2^2, B_1, B_2) \quad \textbf{(5.5.2.1)}$$

In (5.5.2.2) the steps are:
$[[1, 1, 1, 2, 2, 1, 2], [1, 1, 2, 2, 2, 1, 2]] \to [[1, 1, 1], [2, 2], [1], [2]], [[1, 1], [2, 2, 2], [1], [2]] \to Tr(\Omega, B_1^3, B_2^2, B_1, B_2) + Tr(\Omega, B_1^2, B_2^3, B_1, B_2)$

> $cno([[1, 1, 1, 2, 2, 1, 2], [1, 1, 2, 2, 2, 1, 2]])$

$$Tr(\Omega, B_1^3, B_2^2, B_1, B_2) + Tr(\Omega, B_1^2, B_2^3, B_1, B_2) \quad \textbf{(5.5.2.2)}$$

In (5.5.2.3) and (5.5.2.4) the multiset given in input is $[1, 2, 3]$. The produced necklaces are $[123]$ and $[132]$. Then to each necklace, the trace of products of matrices whose indexes are $[123]$and $[132]$ is built. The same is done by *convo*, but with the non-centrality matrix Ω inserted in first position.

> $conv([[1, 1], [2, 1], [3, 1]]);$

$$Tr(B_1, B_2, B_3) + Tr(B_1, B_3, B_2) \quad \textbf{(5.5.2.3)}$$

> $convo([[1, 1], [2, 1], [3, 1]]);$

$$Tr(\Omega, B_1, B_2, B_3) + Tr(\Omega, B_2, B_3, B_1) + Tr(\Omega, B_3, B_1, B_2) + Tr(\Omega, B_1, B_3, B_2) + Tr(\Omega, B_3, B_2, B_1) + Tr(\Omega, B_2, B_1, B_3) \quad \textbf{(5.5.2.4)}$$

If the multiset is empty, a 0 is produced in output.

> $convo([[1,0],[2,0],[3,0]])$

$$0 \qquad \textbf{(5.5.2.5)}$$

> $conv([[1,0],[2,0],[3,0]])$

$$0 \qquad \textbf{(5.5.2.6)}$$

The nCWishart function

The procedure nCWishart computes moments of a complex non-central Wishart distribution, according to the formula (4.8) of [1]. In particular for the binomial expansion the procedure calls mkT. Then the procedure calls *Mnecklaces* to construct all necklaces related to the multiset with molteplicity given by the multi-index *i*. These necklaces are converted in traces of products of matrices indexed by the strings in the necklaces by *conv* and *convo*. The traces are then inserted in the moments of the central Wishart distribution given in formula (4.9) and of the matrix of formal variables in formula (4.10), concurring in the convolution $Wc + A$ of the non-central Wishart distribution.

The Code

```
> nCWishart := proc( )
    local sum, L, V, U, M, n, eV, len;
    sum := 0 : len := add(x, x = args);
    M := mkT([args], 2);
    eV := [seq(B[i] = Σ·H[i], i = 1 ..nargs)];
    V := [seq(f_i = (-1)^i, i = 1 ..len)];
    U := [seq(f_i = n^i, i = 1 ..len)];
    V := [op(V), seq(g_op(m_1) = convo([seq([i, m_1,i], i = 1 ..nargs)]), m = M)];
    U := [op(U), seq(g_op(m_1) = conv([seq([i, m_1,i], i = 1 ..nargs)]), m = M)];
    for L in M do
      sum := sum + (eval(MFB(op(L_1)), V)·eval(MFB(op(L_2)), U)) :
    od:
    eval(simplify(sum), eV);
  end:
>
```

Example

In (5.6.2.1) the joint moment in (1) with $i_1 = 1$

> $nCWishart(1)$

$$-Tr(\Omega, \Sigma H_1) + n\, Tr(\Sigma H_1) \qquad \textbf{(5.6.2.1)}$$

In (5.6.2.2) the joint moment in (1) with $i_1 = 1$ and $i_2 = 1$

> $nCWishart(1, 1)$

$$-Tr(\Omega, \Sigma H_1, \Sigma H_2) - Tr(\Omega, \Sigma H_2, \Sigma H_1) + Tr(\Omega, \Sigma H_1)\, Tr(\Omega, \Sigma H_2) \quad \textbf{(5.6.2.2)}$$
$$+ n\, Tr(\Sigma H_1, \Sigma H_2) + n^2\, Tr(\Sigma H_1)\, Tr(\Sigma H_2) - Tr(\Omega, \Sigma H_1)\, n\, Tr(\Sigma H_2)$$
$$- Tr(\Omega, \Sigma H_2)\, n\, Tr(\Sigma H_1)$$

In (5.6.2.3) the joint moment in (1) with $i_1 = 1$ and $i_2 = 2$

```
> nCWishart(1, 2)
```

$$-n\, Tr(\Sigma H_2)\, Tr(\Omega, \Sigma H_1, \Sigma H_2) - n\, Tr(\Sigma H_2)\, Tr(\Omega, \Sigma H_2, \Sigma H_1) \quad \textbf{(5.6.2.3)}$$
$$+ n\, Tr(\Sigma H_2)\, Tr(\Omega, \Sigma H_1)\, Tr(\Omega, \Sigma H_2) - Tr(\Omega, \Sigma H_2)\, n\, Tr(\Sigma H_1, \Sigma H_2)$$
$$- Tr(\Omega, \Sigma H_2)\, n^2\, Tr(\Sigma H_1)\, Tr(\Sigma H_2) + 2\, Tr(\Omega, \Sigma H_2)\, Tr(\Omega, \Sigma H_1, \Sigma H_2)$$
$$+ 2\, Tr(\Omega, \Sigma H_2)\, Tr(\Omega, \Sigma H_2, \Sigma H_1) - Tr(\Omega, \Sigma H_1)\, Tr(\Omega, \Sigma H_2)^2 - Tr(\Omega,$$
$$\Sigma H_1, \Sigma^2 H_2^2) - Tr(\Omega, \Sigma^2 H_2^2, \Sigma H_1) - Tr(\Omega, \Sigma H_2, \Sigma H_1, \Sigma H_2) + Tr(\Omega,$$
$$\Sigma H_1)\, Tr(\Omega, \Sigma^2 H_2^2) + 2\, n^2\, Tr(\Sigma H_1, \Sigma H_2)\, Tr(\Sigma H_2) + n^3\, Tr(\Sigma H_1)\, Tr(\Sigma H_2)^2$$
$$+ n\, Tr(\Sigma H_1, \Sigma^2 H_2^2) + \frac{1}{2}\, n^2\, Tr(\Sigma H_1)\, Tr(\Sigma^2 H_2^2) - Tr(\Omega,$$
$$\Sigma H_1)\, n^2\, Tr(\Sigma H_2)^2 - \frac{1}{2}\, Tr(\Omega, \Sigma H_1)\, n\, Tr(\Sigma^2 H_2^2) + n\, Tr(\Sigma H_1)\, Tr(\Omega, \Sigma H_2)^2$$
$$- n\, Tr(\Sigma H_1)\, Tr(\Omega, \Sigma^2 H_2^2)$$

```
>
```

Conclusions

The proposed algorithm computes joint moments (1) of a complex non-central Wishart distribution, by using a symbolic representation as convolution of the central Wishart distribution and of a matrix of formal variables. The moments either of the central Wishart distribution either of the matrix of formal variables are computed by using necklaces of multisets having multiplicities given by the vector $i = (i_1, i_2, \ldots, i_m)$

References

[1] Di Nardo E. (2013) On a symbolic representation of non-central Wishart random matrices with applications. Submitted. Available on demand.

[2] Di Nardo E., G. Guarino, D. Senato, Multiset Subdivision, source Maple algorithm located in www.maplesoft.com (http://www.maplesoft.com/applications/view.aspx?SID=33039)

[3] Di Nardo E., G. Guarino, D. Senato (2011), A new algorithm for computing the multivariate Faà di Bruno's formula, Appl. Math. Comp. doi:10.1016/j.amc.2011.01.001 (http://www.maplesoft.com/applications/view.aspx?SID=101396)

Legal Notice: The copyright for this application is owned by the author(s). Neither Maplesoft nor the author are responsible for any errors contained within and are not liable for any damages resulting from the use of this material. This application is intended for non-commercial, non-profit use only. Contact the author for permission if you wish to use this application in for-profit activities