NAG Library Function Document

**nag_gen_real_mat_print_comp (x04cbc)**

1  Purpose

nag_gen_real_mat_print_comp (x04cbc) prints a real matrix.

2  Specification

```c
#include <nag.h>
#include <nagx04.h>

void nag_gen_real_mat_print_comp (Nag_OrderType order,
    Nag_MatrixType matrix, Nag_DiagType diag, Integer m, Integer n,
    const double a[], Integer pda, const char *form, const char *title,
    Nag_LabelType labrow, const char *rlabs[], Nag_LabelType labcol,
    const char *clabs[], Integer ncols, Integer indent, const char *outfile,
    NagError *fail)
```

3  Description

nag_gen_real_mat_print_comp (x04cbc) prints a double matrix, or part of it, using a format specifier supplied by you. The matrix is output to the file specified by `outfile` or, by default, to standard output.

4  References

None.

5  Arguments

1:  *order* – Nag_OrderType  
   *Input*
   
   *On entry:* the `order` argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by `order = Nag_RowMajor`. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   *Constraint:* `order = Nag_RowMajor` or `Nag_ColMajor`.

2:  *matrix* – Nag_MatrixType  
   *Input*
   
   *On entry:* indicates the part of the matrix to be printed.

   `matrix = Nag_GeneralMatrix`
   The whole of the rectangular matrix.

   `matrix = Nag_LowerMatrix`
   The lower triangle of the matrix, or the lower trapezium if the matrix has more rows than columns.

   `matrix = Nag_UpperMatrix`
   The upper triangle of the matrix, or the upper trapezium if the matrix has more columns than rows.

   *Constraint:* `matrix = Nag_GeneralMatrix, Nag_LowerMatrix` or `Nag_UpperMatrix`. 

---

Mark 25
On entry: indicates whether the diagonal elements of the matrix are to be printed.

- **diag = Nag_NonRefDiag**
  The diagonal elements of the matrix are not referenced and not printed.

- **diag = Nag_UnitDiag**
  The diagonal elements of the matrix are not referenced, but are assumed all to be unity, and are printed as such.

- **diag = Nag_NonUnitDiag**
  The diagonal elements of the matrix are referenced and printed.

If **matrix = Nag_GeneralMatrix**, then **diag** must be set to **Nag_NonUnitDiag**.

**Constraints:**

- If **matrix ≠ Nag_GeneralMatrix**, **diag = Nag_NonRefDiag**, **Nag_UnitDiag** or **Nag_NonUnitDiag**;
- If **matrix = Nag_GeneralMatrix**, **diag = Nag_NonUnitDiag**.

4: **m** – Integer

On entry: the number of rows and columns of the matrix, respectively, to be printed.

If either **m** or **n** is less than 1, **nag_gen_real_mat_print_comp (x04cbc)** will exit immediately after printing **title**; no row or column labels are printed.

5: **n** – Integer

**form** – const char *

On entry: a valid C format code. This should be of the form [%][flag][ww:pp][format indicator], where **ww:pp** indicates that up to two digits may be used to specify the field width and precision respectively. Only % and format indicator must be present. flag can be one of −, +, < space > or # and format indicator can be e, E, f, g or G. Thus, possible formats include %f, %+23.15G, %.6e. **form** is used to print elements of the matrix **A**.
In addition, nag_gen_real_mat_print_comp (x04cbc) chooses its own format code when form is NULL or form = 's'.

form = NULL

nag_gen_real_mat_print_comp (x04cbc) will choose a format code such that numbers will be printed with either a %8.4f, a %11.4f or a %13.4e format. The %8.4f code is chosen if the sizes of all the matrix elements to be printed lie between 0.001 and 1.0. The %11.4f code is chosen if the sizes of all the matrix elements to be printed lie between 0.001 and 9999.9999. Otherwise the %13.4e code is chosen.

form = 's'

nag_gen_real_mat_print_comp (x04cbc) will choose a format code such that numbers will be printed to as many significant digits as are necessary to distinguish between neighbouring machine numbers. Thus any two numbers that are stored with different internal representations should look different on output.

Constraint: form must be of the form %[flag][ww:pp]format indicator].

9: title – const char *  

On entry: a title to be printed above the matrix, or name of the matrix.

If title = NULL, no title (and no blank line) will be printed.

If title contains more than ncols characters, the contents of title will be wrapped onto more than one line, with the break after ncols characters.

Any trailing blank characters in title are ignored.

10: labrow – Nag_LabelType  

On entry: indicates the type of labelling to be applied to the rows of the matrix.

labrow = Nag_NoLabels
Prints no row labels.

labrow = Nag_IntegerLabels
Prints integer row labels.

labrow = Nag_CharacterLabels
Prints character labels, which must be supplied in array rlabs.

Constraint: labrow = Nag_NoLabels, Nag_IntegerLabels or Nag_CharacterLabels.

11: rlabs[dim] – const char *  

Note: the dimension, dim, of the array rlabs must be at least

    m when labrow = Nag_CharacterLabels;
            otherwise rlabs may be NULL.

On entry: if labrow = Nag_CharacterLabels, rlabs must contain labels for the rows of the matrix; otherwise rlabs is not referenced and may be NULL.

Labels are right-justified when output, in a field which is as wide as necessary to hold the longest row label. Note that this field width is subtracted from the number of usable columns, ncols.

12: labcol – Nag_LabelType  

On entry: indicates the type of labelling to be applied to the columns of the matrix.

labcol = Nag_NoLabels
Prints no column labels.

labcol = Nag_IntegerLabels
Prints integer column labels.
labcol = Nag_CharacterLabels
Prints character labels, which must be supplied in array clabs.

Constraint: labcol = Nag_NoLabels, Nag_IntegerLabels or Nag_CharacterLabels.

13: clabs[dim] – const char *

Input

Note: the dimension, dim, of the array clabs must be at least
n when labcol = Nag_CharacterLabels;
otherwise clabs may be NULL.

On entry: if labcol = Nag_CharacterLabels, clabs must contain labels for the columns of the matrix; otherwise clabs is not referenced and may be NULL.

Labels are right-justified when output. Any label that is too long for the column width, which is determined by form, is truncated.

14: ncols – Integer

Input

On entry: the maximum output record length. If the number of columns of the matrix is too large to be accommodated in ncols characters, the matrix will be printed in parts, containing the largest possible number of matrix columns, and each part separated by a blank line.

ncols must be large enough to hold at least one column of the matrix using the format specifier in form. If a value less than or equal to 0 or greater than 132 is supplied for ncols, then the value 80 is used instead.

15: indent – Integer

Input

On entry: the number of columns by which the matrix (and any title and labels) should be indented. The effective value of ncols is reduced by indent columns. If a value less than 0 or greater than ncols is supplied for indent, the value 0 is used instead.

16: outfile – const char *

Input

On entry: the name of a file to which output will be directed. If outfile is NULL the output will be directed to standard output.

17: fail – NagError *

Input/Output

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
Memory allocation failed.

NE_BAD_PARAM
On entry, argument ⟨value⟩ had an illegal value.

NE_COL_WIDTH
⟨value⟩ is not wide enough to hold at least one matrix column. ncols = ⟨value⟩ and indent = ⟨value⟩.

NE_INTERNAL_ERROR
An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.
7 Accuracy
Not applicable.

8 Parallelism and Performance
Not applicable.

9 Further Comments
nag_gen_real_mat_print_comp (x04cbc) may be used to print a vector, either as a row or as a column. The following code fragment illustrates possible calls.

```c
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>

double *a = 0;
Integer n = 4;
if (!a = NAG_ALLOC(n, double))
{
    Vprintf("Allocation failure\n");
    return -1;
}
/* Read A from data file */
for (i = 0; i < n; ++i)
    Vscanf("%lf", &a[i]);
/* Print vector A as a row vector */
x04cbc(Nag_RowMajor, Nag_GeneralMatrix, Nag_NonUnitDiag,
1, n, a, n, 0, 0, Nag_NoLabels, 0, Nag_IntegerLabels, 0,
0, 0, 0, NAGERR_DEFAULT);
/* Print vector A as a column vector */
x04cbc(Nag_RowMajor, Nag_GeneralMatrix, Nag_NonUnitDiag,
1, n, a, 1, 0, 0, Nag_NoLabels, 0, Nag_NoLabels, 0,
0, 0, 0, NAGERR_DEFAULT);
```

10 Example
None.