

## NAG C Library Function Document

### nag\_band\_real\_mat\_print\_comp (x04cfc)

#### 1 Purpose

nag\_band\_real\_mat\_print\_comp (x04cfc) prints a real band matrix.

#### 2 Specification

```
void nag_band_real_mat_print_comp (Nag_OrderType order, Integer m, Integer n,
    Integer kl, Integer ku, const double a[], Integer pda, const char *format,
    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\_band\_real\_mat\_print\_comp (x04cfc) prints a real band matrix stored in packed form, using a format specifier supplied by the user. The matrix is output to the file specified by **outfile** or, by default, to standard output.

#### 4 References

None.

#### 5 Parameters

1: **order** – Nag\_OrderType *Input*

*On entry:* the **order** parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

*Constraint:* **order = Nag\_RowMajor or Nag\_ColMajor.**

2: **m** – Integer *Input*  
 3: **n** – Integer *Input*

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

If either **m** or **n** is less than 1, nag\_band\_real\_mat\_print\_comp (x04cfc) will exit immediately after printing **title**; no row or column labels are printed.

4: **kl** – Integer *Input*

*On entry:* the number of sub-diagonals of the band matrix *A*.

*Constraint:* **kl ≥ 0.**

5: **ku** – Integer *Input*

*On entry:* the number of super-diagonals of the band matrix *A*.

*Constraint:* **ku ≥ 0.**

6: **a[dim]** – const double *Input*

**Note:** the dimension, *dim*, of the array **a** must be at least  $\max(1, \mathbf{pda} \times \mathbf{n})$  when **order = Nag\_ColMajor** and at least  $\max(1, \mathbf{pda} \times \mathbf{m})$  when **order = Nag\_RowMajor**.

*On entry:* the  $m$  by  $n$  matrix  $A$ . This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements  $a_{ij}$ , for  $i = 1, \dots, m$  and  $j = \max(1, i - k_l), \dots, \min(n, i + k_u)$ , depends on the **order** parameter as follows:

if **order** = **Nag\_ColMajor**,  $a_{ij}$  is stored as  $\mathbf{a}[(j - 1) \times \mathbf{pda} + \mathbf{kl} + \mathbf{ku} + i - j]$ ;  
 if **order** = **Nag\_RowMajor**,  $a_{ij}$  is stored as  $\mathbf{a}[(i - 1) \times \mathbf{pda} + \mathbf{kl} + j - i]$ .

7: **pda** – Integer *Input*

*On entry:* the stride separating row or column elements (depending on the value of **order**) of the matrix  $A$  in the array **a**.

*Constraint:*  $\mathbf{pda} \geq \mathbf{kl} + \mathbf{ku} + 1$ .

8: **format** – char \* *Input*

*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 2 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. **format** is used to print elements of the matrix  $A$ .

In addition, `nag_band_real_mat_print_comp` (x04fcf) chooses its own format code when **format** is **NULL** or **format** = \*.

If **format** = **NULL**, `nag_band_real_mat_print_comp` (x04fcf) 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.

If **format** = \*, `nag_band_real_mat_print_comp` (x04fcf) 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:* **format** must be of the form  $\%[flag]ww.pp[format indicator]$ .

9: **title** – char \* *Input*

*On entry:* a title to be printed above 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 *Input*

*On entry:* indicates the type of labelling to be applied to the rows of the matrix, as follows:

if **labrow** = **Nag\_NoLabels**, `nag_band_real_mat_print_comp` (x04fcf) prints no row labels;  
 if **labrow** = **Nag\_IntegerLabels**, `nag_band_real_mat_print_comp` (x04fcf) prints integer row labels;  
 if **labrow** = **Nag\_CharacterLabels**, `nag_band_real_mat_print_comp` (x04fcf) prints character labels, which must be supplied in array **rlabs**.

*Constraint:* **labrow** = **Nag\_NoLabels**, **Nag\_IntegerLabels** or **Nag\_CharacterLabels**.

11: **rlabs**[*dim*] – const char \* *Input*

*On entry:* if **labrow** = **Nag\_CharacterLabels**, **rlabs** must be dimensioned at least of length **m** and must contain labels for the rows of the matrix, otherwise **rlabs** should 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

*Input*

*On entry:* indicates the type of labelling to be applied to the columns of the matrix, as follows:

if **labcol** = **Nag\_NoLabels**, nag\_band\_real\_mat\_print\_comp (x04cfc) prints no column labels;

if **labcol** = **Nag\_IntegerLabels**, nag\_band\_real\_mat\_print\_comp (x04cfc) prints integer column labels;

if **labcol** = **Nag\_CharacterLabels**, nag\_band\_real\_mat\_print\_comp (x04cfc) prints character labels, which must be supplied in array **clabs**.

*Constraint:* **labcol** = **Nag\_NoLabels** or **Nag\_IntegerLabels**, or C.

13: **clabs**[*dim*] – const char \*

*Input*

*On entry:* if **labcol** = **Nag\_CharacterLabels**, **clabs** must be dimensioned at least of length **n** and must contain labels for the columns of the matrix, otherwise **clabs** should be **NULL**.

Labels are right-justified when output. Any label that is too long for the column width, which is determined by **format**, 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 **format**. 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** – 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 parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_COL\_WIDTH

$\langle value \rangle$  is not wide enough to hold at least one matrix column. **ncols** =  $\langle value \rangle$ , **indent** =  $\langle value \rangle$ .

### NE\_INVALID\_FORMAT

The string  $\langle value \rangle$ , has not been recognised as a valid format.

### NE\_ALLOC\_FAIL

Memory allocation failed.

**NE\_BAD\_PARAM**

On entry, parameter  $\langle value \rangle$  had an illegal value.

**NE\_NOT\_WRITE\_FILE**

Cannot open file  $\langle value \rangle$  for writing.

**NE\_NOT\_APPEND\_FILE**

Cannot open file  $\langle value \rangle$  for appending.

**NE\_NOT\_CLOSE\_FILE**

Cannot close file  $\langle value \rangle$ .

**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 consult NAG for assistance.

**7 Accuracy**

Not applicable.

**8 Further Comments**

None.

**9 Example**

None.

---