NAG C Library Function Document

nag_gen_complx_mat_print_comp (x04dbc)

1 Purpose

nag_gen_complx_mat_print_comp (x04dbc) prints a complex matrix.

2 Specification

```
void nag_gen_complx_mat_print_comp (Nag_OrderType order, Nag_MatrixType matrix,
     Nag_DiagType diag, Integer m, Integer n, const Complex a[], Integer pda,
     Nag_ComplexFormType cmplxform, 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_gen_complx_mat_print_comp (x04dbc) prints a complex matrix, or part of it, 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: **matrix** – Nag_MatrixType *Input*

On entry: indicates the part of the matrix to be printed, as follows:

if **matrix** = **Nag_GeneralMatrix**, the whole of the rectangular matrix;

if **matrix** = **Nag_LowerMatrix**, the lower triangle of the matrix, or the lower trapezium if the matrix has more rows than columns;

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

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

3: **diag** – Nag_DiagType *Input*

On entry: unless **matrix** = **Nag_GeneralMatrix**, **diag** must specify whether the diagonal elements of the matrix are to be printed, as follows:

if **diag** = **Nag_NonRefDiag**, the diagonal elements of the matrix are not referenced and not printed;

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

if **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** $\neq$ **Nag_GeneralMatrix**, **diag** = **Nag_NonRefDiag**, **Nag_UnitDiag** or **Nag_NonUnitDiag**;
if **matrix** = **Nag_GeneralMatrix**, **diag** = **Nag_NonUnitDiag**.

4: **m** – Integer *Input*
5: **n** – Integer *Input*

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_complx_mat_print_comp (x04dbc) will exit immediately after printing **title**; no row or column labels are printed.

6: **a**[dim] – const Complex *Input*

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

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix A is stored in **a**$[(j-1) \times \textbf{pda} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix A is stored in **a**$[(i-1) \times \textbf{pda} + j - 1]$.

On entry: the matrix to be printed. Only the elements that will be referred to, as specified by parameters **matrix** and **diag**, need be set.

7: **pda** – Integer *Input*

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

Constraints:

if **order** = **Nag_ColMajor**, **pda** $\geq \max(1, \textbf{m})$;
if **order** = **Nag_RowMajor**, **pda** $\geq \max(1, \textbf{n})$.

8: **cmplxform** – Nag_ComplexFormType *Input*

On entry: indicates how the value of **format** is to be used to print matrix elements.

If **cmplxform** = **Nag_AboveForm**, the format code in **format** is assumed to contain a single real edit-descriptor which is to be used to print the real and imaginary parts of each complex number one above the other. Each row of the matrix is separated by a blank line, and any row labels are attached only to the real parts. This option means that about twice as many columns can be fitted into **ncols** characters than if any other **cmplxform** option is used. A typical value of **format** for this **cmplxform** option might be %11.4e.

If **cmplxform** = **Nag_BracketForm**, the format code in **format** is assumed to contain a single edit-descriptor such as %13.4f, * or **NULL**, which is used to print the real and imaginary parts of each complex number separated by a comma, and surrounded by brackets. Thus a matrix element printed with this **cmplxform** option might look like this: $(12.345, -11.323)$.

If **cmplxform** = **Nag_DirectForm**, the format code in **format** is used unaltered to print a complex number. This **cmplxform** option allows the user flexibility to specify exactly how the number is printed. With this option for **cmplxform** and a suitable value for **format** it is possible, for example, to print a complex number in the form $(0.123 + 3.214i)$ or $(0.123\text{e}{-}02, 0.234\text{e}{-}01)$.

Constraint: **cmplxform** = **Nag_AboveForm**, **Nag_BracketForm** or **Nag_DirectForm**.

9: **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, %-11.4G, %.6e. **format** is used in conjunction with parameter **cmplxform**, described above, to print elements of the matrix A.

In addition, nag_gen_complx_mat_print_comp (x04dbc) chooses its own format code when **format** is **NULL** or **format** $= *$.

If **format** = **NULL**, nag_gen_complx_mat_print_comp (x04dbc) 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_gen_complx_mat_print_comp (x04dbc) 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.

More complicated values of **format**, to print a complex number in a desired form, may be used. See the description of parameter **cmplxform** above for more details.

Constraint:

if **cmplxform** = **Nag_AboveForm** or **Nag_BracketForm**, **format** must be of the form %[*flag*]*ww.pp*[*format indicator*].

10: **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.

11: **labrow** – Nag_LabelType *Input*

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

if **labrow** = **Nag_NoLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints no row labels;

if **labrow** = **Nag_IntegerLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints integer row labels;

if **labrow** = **Nag_CharacterLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints character labels, which must be supplied in array **rlabs**.

Constraint: **labrow** = **Nag_NoLabels**, **Nag_IntegerLabels** or **Nag_CharacterLabels**.

12: **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**.

13: **labcol** – Nag_LabelType *Input*

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

if **labcol** = **Nag_NoLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints no column labels;

if **labcol** = **Nag_IntegerLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints integer column labels;

if **labcol** = **Nag_CharacterLabels**, nag_gen_complx_mat_print_comp (x04dbc) prints character labels, which must be supplied in array **clabs**.

Constraint: **labcol** = **Nag_NoLabels**, **Nag_IntegerLabels** or **Nag_CharacterLabels**.

14: **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.

15: **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.

16: **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.

17: **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.

18: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_COL_WIDTH

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

NE_INVALID_FORMAT

The string ⟨*value*⟩, has not been recognised as a valid format.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

NE_NOT_WRITE_FILE

Cannot open file ⟨*value*⟩ for writing.

NE_NOT_APPEND_FILE

Cannot open file ⟨*value*⟩ for appending.

NE_NOT_CLOSE_FILE

Cannot close file ⟨*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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

nag_gen_complx_mat_print_comp (x04dbc) may be used to print a vector, either as a row or as a column. The following code fragment illustrates possible calls.

```
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
Complex *a = 0;
Integer n = 4;
if ( !(a = NAG_ALLOC(n, Complex)) )
  {
    Vprintf("Allocation failure\n");
    return -1;
  }
/* Read A from data file */
for (i = 0; i < n; ++i)
  Vscanf("%lf%lf", &a[i].re, &a[i].im);
/* Print vector A as a row vector */
x04dbc(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 */
x04dbc(Nag_RowMajor, Nag_GeneralMatrix, Nag_NonUnitDiag,
n, 1, a, 1, 0, 0,  Nag_IntegerLabels, 0, Nag_NoLabels, 0,
0, 0, 0, NAGERR_DEFAULT)
```

9 Example

See Section 9 of the document for nag_zhetri (f07mwc).