# NAG Library Function Document

# nag_pack_complx_mat_print (x04dcc)

## 1 Purpose

nag_pack_complx_mat_print (x04dcc) is an easy-to-use function to print a Complex triangular matrix stored in a packed one-dimensional array.

## 2 Specification

```
#include <nag.h>
#include <nagx04.h>
void nag_pack_complx_mat_print (Nag_OrderType order, Nag_UploType uplo,
     Nag_DiagType diag, Integer n, const Complex a[], const char *title,
     const char *outfile, NagError *fail)
```

## 3 Description

nag_pack_complx_mat_print (x04dcc) prints a Complex triangular matrix stored in packed form. It is an easy-to-use driver for nag_pack_complx_mat_print_comp (x04ddc). The function uses default values for the format in which numbers are printed, for labelling the rows and columns, and for output record length.

nag_pack_complx_mat_print (x04dcc) will choose a format code such that numbers will be printed with 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. The chosen code is used to print each complex element of the matrix with the real part above the imaginary part.

The matrix is printed with integer row and column labels, and with a maximum record length of 80.

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 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

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

2: **uplo** – Nag_UploType *Input*

*On entry*: indicates the type of the matrix to be printed

**uplo** = Nag_Lower
The matrix is lower triangular

**uplo** = Nag_Upper
The matrix is upper triangular

*Constraint*: **uplo** = Nag_Lower or Nag_Upper.

3: **diag** – Nag_DiagType *Input*

*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.

*Constraint*: **diag** = Nag_NonRefDiag, Nag_UnitDiag or Nag_NonUnitDiag.

4: **n** – Integer *Input*

*On entry*: the order of the matrix to be printed.

If **n** is less than 1, nag_pack_complx_mat_print (x04dcc) will exit immediately after printing **title**; no row or column labels are printed.

5: **a**[$dim$] – const Complex *Input*

**Note**: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

*On entry*: the matrix to be printed. Note that **a** must have space for the diagonal elements of the matrix, even if these are not stored.

The storage of elements $A_{ij}$ depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
$A_{ij}$ is stored in **a**$[(j-1) \times j/2 + i - 1]$, for $i \leq j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
$A_{ij}$ is stored in **a**$[(2n-j) \times (j-1)/2 + i - 1]$, for $i \geq j$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
$A_{ij}$ is stored in **a**$[(2n-i) \times (i-1)/2 + j - 1]$, for $i \leq j$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
$A_{ij}$ is stored in **a**$[(i-1) \times i/2 + j - 1]$, for $i \geq j$.

If **diag** = Nag_UnitDiag, the diagonal elements of $A$ are assumed to be 1, and are not referenced; the same storage scheme is used whether **diag** = Nag_NonUnitDiag or **diag** = Nag_UnitDiag.

6: **title** – const 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 80 characters, the contents of **title** will be wrapped onto more than one line, with the break after 80 characters.

Any trailing blank characters in **title** are ignored.

7: **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.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

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

**NE_NOT_APPEND_FILE**

Cannot open file ⟨*value*⟩ for appending.

**NE_NOT_CLOSE_FILE**

Cannot close file ⟨*value*⟩.

**NE_NOT_WRITE_FILE**

Cannot open file ⟨*value*⟩ for writing.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag_pack_complx_mat_print (x04dcc) is not threaded in any implementation.

## 9 Further Comments

A call to nag_pack_complx_mat_print (x04dcc) is equivalent to a call to nag_pack_complx_mat_print_comp (x04ddc) with the following argument values:

```
ncols = 80
 indent = 0
 labrow = Nag_IntegerLabels
 labcol = Nag_IntegerLabels
 form = 0
 cmplxform = Nag_AboveForm
```

## 10 Example

None.