NAG Library Routine Document

X04DDF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of **bold italicised** terms and other implementation-dependent details.

1 Purpose

X04DDF prints a complex triangular matrix stored in a packed one-dimensional array.

2 Specification

```
SUBROUTINE X04DDF (UPLO, DIAG, N, A, USEFRM, FORM, TITLE, LABROW, RLABS,      &
                   LABCOL, CLABS, NCOLS, INDENT, IFAIL)

INTEGER              N, NCOLS, INDENT, IFAIL
COMPLEX (KIND=nag_wp) A(*)
CHARACTER(*)         FORM, TITLE, RLABS(*), CLABS(*)
CHARACTER(1)         UPLO, DIAG, USEFRM, LABROW, LABCOL
```

3 Description

X04DDF prints a complex triangular matrix stored in packed form, using a format specifier supplied by you. The matrix must be packed by column. The matrix is output to the unit defined by X04ABF.

4 References

None.

5 Parameters

1: UPLO – CHARACTER(1) *Input*

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

UPLO = 'L'
  The matrix is lower triangular. In this case, the packed array A holds the matrix elements in the following order: $(1, 1), (2, 1), \ldots, (N, 1), (2, 2), (3, 2), \ldots, (N, 2)$, etc.

UPLO = 'U'
  The matrix is upper triangular. In this case, the packed array A holds the matrix elements in the following order: $(1, 1), (1, 2), (2, 2), (1, 3), (2, 3), (3, 3), (1, 4)$, etc.

Constraint: UPLO = 'L' or 'U'.

2: DIAG – CHARACTER(1) *Input*

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

DIAG = 'B'
  The diagonal elements of the matrix are not referenced and not printed.

DIAG = 'U'
  The diagonal elements of the matrix are not referenced, but are assumed all to be unity, and are printed as such.

DIAG = 'N'
  The diagonal elements of the matrix are referenced and printed.

Constraint: DIAG = 'B', 'U' or 'N'.

3:     N – INTEGER   *Input*

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

    If N is less than 1, X04DDF will exit immediately after printing TITLE; no row or column labels are printed.

4:     A(∗) – COMPLEX (KIND=nag_wp) array   *Input*

    Note: the dimension of the array A must be at least $\max(1, N \times (N + 1)/2)$.

    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.

    More precisely,

        if UPLO = 'U', the upper triangle of A must be stored with element A_{ij} in $A(i + j(j - 1)/2)$ for $i \leq j$;

        if UPLO = 'L', the lower triangle of A must be stored with element A_{ij} in $A(i + (2n - j)(j - 1)/2)$ for $i \geq j$.

    If DIAG = 'U', the diagonal elements of A are assumed to be 1, and are not referenced; the same storage scheme is used whether DIAG = 'N' or 'U'.

5:     USEFRM – CHARACTER(1)   *Input*

    On entry: indicates how the value of FORM is to be used to print matrix elements, except in MATLAB mode where USEFRM is ignored.

    USEFRM = 'A'
        The format code in FORM 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 USEFRM option is used. A typical value of FORM for this USEFRM option might be 'E13.4', ' ∗ ' or ' '.

    USEFRM = 'B'
        The format code in FORM is assumed to contain a single edit-descriptor such as 'E13.4', ' ∗ ' or ' ' 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 USEFRM option might look like this: $(12.345, -11.323)$.

    USEFRM = 'D'
        The format code in FORM is used unaltered to print a complex number. This USEFRM option allows you flexibility to specify exactly how the number is printed. With this option for USEFRM and a suitable value for FORM it is possible, for example, to print a complex number in the form $(0.123 + 3.214i)$ or $(0.123E-02, 0.234E-01)$. See Section 9 for an example illustrating this option.

    Constraint: USEFRM = 'A', 'B' or 'D'.

6:     FORM – CHARACTER(∗)   *Input*

    On entry: describes the Fortran format code that is used in conjunction with USEFRM for printing the elements of the matrix A. The format code may be any allowed on the system, whether it is standard Fortran or not. It may or may not be enclosed in brackets.

    In addition, there are the following special codes which force X04DDF to choose its own format code:

    FORM = ' '
        X04DDF will choose a format code such that numbers will be printed with an F8.4, an F11.4 or a 1PE13.4 format. The F8.4 code is chosen if the sizes of the real and imaginary parts of all the matrix elements to be printed lie between 0.001 and 1.0. The F11.4 code is chosen if

the sizes of all the numbers to be printed lie between 0.001 and 9999.9999. Otherwise the 1PE13.4 code is chosen.

FORM = '∗'

X04DDF 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. Whether they do in fact look different will depend on the run-time library of the Fortran compiler in use.

By preceding the desired format code by the string 'MATLAB', X04DDF will print the matrix such that it can be input into MATLAB, and TITLE will be used as the name of the matrix.

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

Examples of valid values for FORM are '(F11.4)', '1P, 2E13.5', 'MATLABF11.4', 'MATLAB∗'.

Constraint: the character length of the format specifier in FORM must be ≤ 80.

7: TITLE – CHARACTER(*) *Input*

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

If TITLE = ' ', 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.

If printing in MATLAB mode, TITLE will be used as the name of the matrix.

8: LABROW – CHARACTER(1) *Input*

On entry: indicates the type of labelling to be applied to the rows of the matrix, except in MATLAB mode where LABROW is ignored.

LABROW = 'N'

Prints no row labels.

LABROW = 'I'

Prints integer row labels.

LABROW = 'C'

Prints character labels, which must be supplied in array RLABS.

Constraint: LABROW = 'N', 'I' or 'C'.

9: RLABS(∗) – CHARACTER(*) array *Input*

Note: the dimension of the array RLABS must be at least N if LABROW = 'C', and at least 1 otherwise.

On entry: if LABROW = 'C', RLABS must contain labels for the rows of the matrix, except in MATLAB mode where RLABS is ignored.

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.

10: LABCOL – CHARACTER(1) *Input*

On entry: indicates the type of labelling to be applied to the columns of the matrix, except in MATLAB mode where LABCOL is ignored.

LABCOL = 'N'

Prints no column labels.

LABCOL = 'I'
>   Prints integer column labels.

LABCOL = 'C'
>   Prints character labels, which must be supplied in array CLABS.

Constraint: LABCOL = 'N', 'I' or 'C'.

11:   CLABS(∗) – CHARACTER(*) array   *Input*

Note: the dimension of the array CLABS must be at least N if LABCOL = 'C', and at least 1 otherwise.

On entry: if LABCOL = 'C', CLABS must contain labels for the columns of the matrix, except in MATLAB mode where CLABS is ignored.

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

12:   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 0 or greater than 132 is supplied for NCOLS, then the value 80 is used instead.

13:   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.

14:   IFAIL – INTEGER   *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value −1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6   Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

>   On entry, UPLO ≠ 'L' or 'U'.

IFAIL = 2

>   On entry, DIAG ≠ 'N', 'U' or 'B'.

IFAIL = 3

> On entry, USEFRM ≠ 'A', 'B' or 'D'.

IFAIL = 4

> On entry, the format specifier in FORM is more than 80 characters long.

IFAIL = 5

> The format specifier in FORM cannot be used to output a number. The specifier probably has too wide a field width or contains an illegal edit descriptor.

IFAIL = 6

> On entry, either LABROW or LABCOL ≠ 'N', 'I' or 'C'.

IFAIL = 7

> The quantity NCOLS – INDENT – *labwid* (where *labwid* is the width needed for the row labels) is not large enough to hold at least one column of the matrix.

7   Accuracy

Not applicable.

8   Further Comments

None.

9   Example

This example program calls X04DDF three times, first to print a 4 by 4 lower triangular matrix, and then twice to print a 4 by 4 upper triangular matrix, various options for labelling and formatting are illustrated.

9.1   Program Text

```
    Program x04ddfe

!     X04DDF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04ddf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nmax = 4, nout = 6
      Integer, Parameter              :: la = (nmax*(nmax+1))/2
      Character (7), Parameter        :: clabs(nmax) = (/              &
                                         'Un     ','Deux   ',          &
                                         'Trois  ','Quatre ' /)
      Character (7), Parameter        :: rlabs(nmax) = (/              &
                                         'Uno    ','Due    ',          &
                                         'Tre    ','Quattro' /)
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: aa
      Integer                         :: i, ifail, indent, ncols
      Character (19)                  :: form
!     .. Local Arrays ..
      Complex (Kind=nag_wp)           :: a(la)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: cmplx, real
!     .. Executable Statements ..
```

```
      Write (nout,*) 'X04DDF Example Program Results'

      Write (nout,*)
      Flush (nout)

!     Generate an array of data

      Do i = 1, la
        aa = real(i,kind=nag_wp)
        a(i) = cmplx(aa,-aa,kind=nag_wp)
      End Do

      ncols = 80
      indent = 0

!     Print order nmax lower triangular matrix with default format and
!     integer row and column labels, and bracketed complex elements

      form = ' '

      ifail = 0
      Call x04ddf('Lower','Non-unit',nmax,a,'Bracketed',form,'Example 1:', &
        'Integer',rlabs,'Integer',clabs,ncols,indent,ifail)

      Write (nout,*)
      Flush (nout)

!     Print order nmax upper triangular matrix with user-supplied format
!     and row and column labels, using the supplied format directly

      form = 'SS,F7.1,SP,F6.1,''i'''

      ifail = 0
      Call x04ddf('Upper','Unit',nmax,a,'Direct',form,'Example 2:', &
        'Character',rlabs,'Character',clabs,ncols,indent,ifail)

      Write (nout,*)
      Flush (nout)

!     Print order nmax upper triangular matrix in MATLAB format
!     Row and column labelling and USEFRM are ignored

      form = 'MATLABF8.2'

      ifail = 0
      Call x04ddf('Upper','Non-unit',nmax,a,' ',form,'A',' ',rlabs,' ',clabs, &
        ncols,indent,ifail)

    End Program x04ddfe
```

9.2   Program Data

None.

9.3   Program Results

```
 X04DDF Example Program Results

 Example 1:
                          1                           2
 1 (      1.0000,     -1.0000)
 2 (      2.0000,     -2.0000) (      5.0000,     -5.0000)
 3 (      3.0000,     -3.0000) (      6.0000,     -6.0000)
 4 (      4.0000,     -4.0000) (      7.0000,     -7.0000)

                          3                           4
 1
 2
 3 (      8.0000,     -8.0000)
 4 (      9.0000,     -9.0000) (     10.0000,    -10.0000)
```

```
Example 2:
                          Un          Deux          Trois         Quatre
        Uno      1.0  +0.0i   2.0  -2.0i    4.0  -4.0i    7.0  -7.0i
        Due                   1.0  +0.0i    5.0  -5.0i    8.0  -8.0i
        Tre                                 1.0  +0.0i    9.0  -9.0i
    Quattro                                               1.0  +0.0i

    A = [
      (     1.00    -1.00i) (     2.00    -2.00i) (     4.00    -4.00i) ...
      (     7.00    -7.00i);
      (     0.00    +0.00i) (     3.00    -3.00i) (     5.00    -5.00i) ...
      (     8.00    -8.00i);
      (     0.00    +0.00i) (     0.00    +0.00i) (     6.00    -6.00i) ...
      (     9.00    -9.00i);
      (     0.00    +0.00i) (     0.00    +0.00i) (     0.00    +0.00i) ...
      (    10.00   -10.00i);
        ];
```

_____