NAG Toolbox for MATLAB

x04df

1 Purpose

x04df prints a ***complex*16*** band matrix stored in a packed two-dimensional array.

2 Syntax

```
[ifail] = x04df(m, n, kl, ku, a, usefrm, form, title, labrow, rlabs, labcol, clabs, ncols, indent)
```

3 Description

x04df prints a ***complex*16*** band matrix stored in a packed two-dimensional array, using a format specifier supplied by you. The matrix is output to the unit defined by x04ab.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **m – int32 scalar**

2: **n – int32 scalar**

The number of rows and columns of the band matrix, respectively, to be printed.

If either **m** or **n** is less than 1, x04df will exit immediately after printing **title**; no row or column labels are printed.

3: **kl – int32 scalar**

The number of subdiagonals of the band matrix A.

Constraint: $\mathbf{kl} \geq 0$.

4: **ku – int32 scalar**

The number of superdiagonals of the band matrix A.

Constraint: $\mathbf{ku} \geq 0$.

5: **a(lda,∗) – complex array**

The first dimension of the array **a** must be at least $\mathbf{kl} + \mathbf{ku} + 1$

The second dimension of the array must be at least $\max(1, \min(\mathbf{m} + \mathbf{ku}, \mathbf{n}))$

The band matrix to be printed.

The matrix is stored in rows 1 to $k_l + k_u + 1$, more precisely, the element A_{ij} must be stored in

$$\mathbf{a}(k_u + 1 + i - j, j) \qquad \text{for } \max(1j - k_u) \leq i \leq \min(mj + k_l).$$

6: **usefrm – string**

Indicates how the value of **form** is to be used to print matrix elements.

usefrm = 'A' (Above)

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*16*** 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' (Bracketed)

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*16*** 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' (Direct)

The format code in **form** is used unaltered to print a ***complex*16*** 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*16*** number in the form $(0.123 + 3.214i)$ or $(0.123\mathrm{D}{-}02, 0.234\mathrm{D}{-}01)$. See Section 9 for an example illustrating this option.

Constraint: **usefrm** = 'A', 'B' or 'D'.

7: **form – string**

A valid Fortran format code. This may be any format code allowed on the system, whether it is standard Fortran or not. **form** is used in conjunction with parameter **usefrm**, to print elements of the matrix A. It may or may not be enclosed in brackets. Examples of valid values for **form** are '$(F11.4)$', '$1P, 2E13.5$'.

In addition, there are two special codes which force x04df to choose its own format code:

form = ' '

x04df 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 = $*$

x04df 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.

More complicated values of **form**, to print a ***complex*16*** number in a desired form, may be used. See the description of parameter **usefrm** for more details.

Constraint: the character length of **form** must be ≤ 80.

8: **title – string**

A title to be printed above 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.

9: **labrow – string**

Indicates the type of labelling to be applied to the rows of the matrix.

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'.

10: **rlabs**(∗) **– string array**

Note: the dimension of the array **rlabs** must be at least **m** if **labrow** = 'C', and at least 1 otherwise.

If **labrow** = 'C', must contain labels for the rows of the matrix.

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**.

11: **labcol – string**

Indicates the type of labelling to be applied to the columns of the matrix.

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'.

12: **clabs**(∗) **– string array**

Note: the dimension of the array **clabs** must be at least **n** if **labcol** = 'C', and at least 1 otherwise.

If **labcol** = 'C', must contain labels for the columns of the matrix.

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

13: **ncols – int32 scalar**

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.

14: **indent – int32 scalar**

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.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

lda

5.4 Output Parameters

1: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **kl** < 0.

ifail $= 2$

On entry, **ku** < 0.

ifail $= 3$

On entry, **lda** $<$ **kl** $+$ **ku** $+ 1$.

ifail $= 4$

On entry, **usefrm** $\neq$ 'A', 'B' or 'D'.

ifail $= 5$

On entry, variable **form** is more than 80 characters long.

ifail $= 6$

The code supplied in **form** cannot be used to output a number. **form** probably has too wide a field width or contains an illegal edit descriptor.

ifail $= 7$

On entry, either **labrow** or **labcol** $\neq$ 'N', 'I' or 'C'.

ifail $= 8$

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

```
m = int32(5);
n = int32(5);
kl = int32(1);
ku = int32(1);
a = [complex(11, -11), complex(12, -12), complex(13, -13), complex(14, -
14), complex(15, -15);
     complex(21, -21), complex(22, -22), complex(23, -23), complex(24, -
24), complex(25, -25);
     complex(31, -31), complex(32, -32), complex(33, -33), complex(34, -
34), complex(35, -35)];
usefrm = 'Bracketed';
format = '                    ';
title = 'Example 1:';
labrow = 'Integer';
rlabs = {'Uno    '};
labcol = 'Integer';
clabs = {'Un     '};
ncols = int32(80);
indent = int32(0);
[ifail] = ...
     x04df(m, n, kl, ku, a, usefrm, format, title, labrow, rlabs, labcol,
clabs, ncols, indent)
```

```
Example 1:
                             1                          2
 1  (    21.0000,   -21.0000) (    12.0000,   -12.0000)
 2  (    31.0000,   -31.0000) (    22.0000,   -22.0000)
 3                            (    32.0000,   -32.0000)
 4
 5

                             3                          4
 1
 2  (    13.0000,   -13.0000)
 3  (    23.0000,   -23.0000) (    14.0000,   -14.0000)
 4  (    33.0000,   -33.0000) (    24.0000,   -24.0000)
 5                            (    34.0000,   -34.0000)

                             5
 1
 2
 3
 4  (    15.0000,   -15.0000)
 5  (    25.0000,   -25.0000)
ifail =
           0
```