NAG Library Routine Document

F01ZBF

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

F01ZBF copies a complex triangular matrix stored in a packed one-dimensional array into an unpacked two-dimensional array, or vice versa.

2 Specification

```
SUBROUTINE F01ZBF (JOB, UPLO, DIAG, N, A, LDA, B, IFAIL)
INTEGER                 N, LDA, IFAIL
COMPLEX (KIND=nag_wp) A(LDA,N), B((N*(N+1))/2)
CHARACTER(1)            JOB, UPLO, DIAG
```

3 Description

F01ZBF unpacks a triangular matrix stored in a vector into a two-dimensional array, or packs a triangular matrix stored in a two-dimensional array into a vector. The matrix is packed by column. This routine is intended for possible use in conjunction with routines from Chapters F06, F07 and F08, where some routines that use triangular matrices store them in the packed form described below.

4 References

None.

5 Arguments

1: JOB – CHARACTER(1) *Input*

On entry: specifies whether the triangular matrix is to be packed or unpacked.

JOB = 'P' (Pack)
The matrix is to be packed into array B.

JOB = 'U' (Unpack)
The matrix is to be unpacked into array A.

Constraint: JOB = 'P' or 'U'.

2: UPLO – CHARACTER(1) *Input*

On entry: specifies the type of the matrix to be copied

UPLO = 'L' (Lower)
The matrix is lower triangular. In this case the packed vector holds, or will hold on exit, the matrix elements in the following order: $(1,1),(2,1),\ldots,(\mathrm{N},1),(2,2),(3,2),\ldots,(\mathrm{N},2)$, etc..

UPLO = 'U' (Upper)
The matrix is upper triangular. In this case the packed vector holds, or will hold on exit, 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'.

3: DIAG – CHARACTER(1) *Input*

On entry: must specify whether the diagonal elements of the matrix are to be copied.

DIAG = 'B' (Blank)
The diagonal elements of the matrix are not referenced and not copied.

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

DIAG = 'N' (Non-unit diagonal)
The diagonal elements of the matrix are referenced and copied.

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

4: N – INTEGER *Input*

On entry: N must specify the number of rows and columns of the triangular matrix.

Constraint: $N > 0$.

5: A(LDA, N) – COMPLEX (KIND=nag_wp) array *Input/Output*

On entry: if JOB = 'P', then the leading N by N part of A must contain the matrix to be copied, stored in unpacked form, in the upper or lower triangle depending on argument UPLO. The opposite triangle of A is not referenced and need not be assigned.

On exit: if JOB = 'U', then the leading N by N part of array A contains the copied matrix, stored in unpacked form, in the upper or lower triangle depending on argument UPLO. The opposite triangle of A is not referenced.

6: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F01ZBF is called.

Constraint: $LDA \geq N$.

7: B($(N \times (N + 1))/2$) – COMPLEX (KIND=nag_wp) array *Input/Output*

On entry: if JOB = 'U', then B must contain the triangular matrix packed by column.

On exit: if JOB = 'P', then B contains the triangular matrix packed by column.

Note that B must have space for the diagonal elements of the matrix, even if these are not stored.

8: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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, JOB $\neq$ 'P' or 'U'.

IFAIL $= 2$

On entry, UPLO $\neq$ 'L' or 'U'.

IFAIL $= 3$

On entry, DIAG $\neq$ 'N', 'U' or 'B'.

IFAIL $= 4$

On entry, N < 1.

IFAIL $= 5$

On entry, LDA $<$ N.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

F01ZBF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example reads in a triangular matrix A, and copies it to the packed matrix B.

10.1 Program Text

```
    Program f01zbfe

!     F01ZBF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f01zbf, nag_wp, x04dbf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lb, lda, lenb, n
      Character (1)                    :: diag, uplo
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F01ZBF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Write (nout,*)
      Flush (nout)
      Read (nin,*) n, uplo, diag
      lda = n
      lenb = (n*(n+1))/2
      Allocate (a(lda,n),b(lenb))
!     Read a triangular matrix of order n
      Do i = 1, n
        Read (nin,*) a(i,1:n)
      End Do

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
!     Print the unpacked matrix
      Call x04dbf(uplo,diag,n,n,a,lda,'B','F5.2','Unpacked Matrix A:','I',     &
        rlabs,'I',clabs,80,0,ifail)

      Write (nout,*)
      Flush (nout)

!     Convert to packed vector form
      ifail = 0
      Call f01zbf('Pack',uplo,diag,n,a,lda,b,ifail)

      lb = n*(n+1)/2

!     Print the packed vector
      ifail = 0
      Call x04dbf('G','X',lb,1,b,lb,'B','F5.2','Packed Vector B:','I',rlabs,   &
        'N',clabs,80,0,ifail)

    End Program f01zbfe
```

10.2 Program Data

```
F01ZBF Example Program Data
 4 'U' 'N'                                        : n, uplo, diag
 (1.1,1.1)  (1.2,1.2)  (1.3,1.3)  (1.4,1.4)     : Unpacked Matrix A
 (0.0,0.0)  (2.2,2.2)  (2.3,2.3)  (2.4,2.4)
 (0.0,0.0)  (0.0,0.0)  (3.3,3.3)  (3.4,3.4)
 (0.0,0.0)  (0.0,0.0)  (0.0,0.0)  (4.4,4.4)
```

10.3 Program Results

```
 F01ZBF Example Program Results

 Unpacked Matrix A:
                  1             2             3             4
 1  ( 1.10, 1.10) ( 1.20, 1.20) ( 1.30, 1.30) ( 1.40, 1.40)
 2                ( 2.20, 2.20) ( 2.30, 2.30) ( 2.40, 2.40)
 3                              ( 3.30, 3.30) ( 3.40, 3.40)
 4                                            ( 4.40, 4.40)

 Packed Vector B:
  1  ( 1.10, 1.10)
  2  ( 1.20, 1.20)
  3  ( 2.20, 2.20)
  4  ( 1.30, 1.30)
  5  ( 2.30, 2.30)
  6  ( 3.30, 3.30)
  7  ( 1.40, 1.40)
  8  ( 2.40, 2.40)
  9  ( 3.40, 3.40)
 10  ( 4.40, 4.40)
```