

# NAG Library Routine Document

## F01VJF (DTPUTF)

**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

F01VJF (DTPUTF) copies a real triangular matrix, stored in a standard packed format array, to a Rectangular Full Packed (RFP) format array.

### 2 Specification

```
SUBROUTINE F01VJF (TRANSR, UPLO, N, AP, AR, INFO)
  INTEGER          N, INFO
  REAL (KIND=nag_wp) AP(N*(N+1)/2), AR(N*(N+1)/2)
  CHARACTER(1)    TRANSR, UPLO
```

The routine may be called by its LAPACK name *dtputf*.

### 3 Description

F01VJF (DTPUTF) copies a real  $n$  by  $n$  triangular matrix,  $A$ , stored in packed format, to RFP format. This routine is intended for possible use in conjunction with routines from Chapters F06, F07 and F16 where some routines that use triangular matrices store them in RFP format. The RFP storage format is described in Section 3.3.3 in the F07 Chapter Introduction and the packed storage format is described in Section 3.3.2 in the F07 Chapter Introduction.

### 4 References

Gustavson F G, Waśniewski J, Dongarra J J and Langou J (2010) Rectangular full packed format for Cholesky's algorithm: factorization, solution, and inversion *ACM Trans. Math. Software* **37**, 2

### 5 Arguments

- 1: TRANSR – CHARACTER(1) *Input*  
*On entry:* specifies whether the normal RFP representation of  $A$  or its transpose is stored.  
 TRANSR = 'N'  
 The RFP representation of the matrix  $A$  is stored.  
 TRANSR = 'T'  
 The transpose of the RFP representation of the matrix  $A$  is stored.  
*Constraint:* TRANSR = 'N' or 'T'.
- 2: UPLO – CHARACTER(1) *Input*  
*On entry:* specifies whether  $A$  is upper or lower triangular.  
 UPLO = 'U'  
 $A$  is upper triangular.  
 UPLO = 'L'  
 $A$  is lower triangular.  
*Constraint:* UPLO = 'U' or 'L'.

- 3: N – INTEGER *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $N \geq 0$ .
- 4:  $AP(N \times (N + 1)/2)$  – REAL (KIND=nag\_wp) array *Input*  
*On entry:* the  $n$  by  $n$  triangular matrix  $A$ , packed by columns.  
 More precisely,  
   if UPLO = 'U', the upper triangle of  $A$  must be stored with element  $A_{ij}$  in  
    $AP(i + j(j - 1)/2)$  for  $i \leq j$ ;  
   if UPLO = 'L', the lower triangle of  $A$  must be stored with element  $A_{ij}$  in  
    $AP(i + (2n - j)(j - 1)/2)$  for  $i \geq j$ .
- 5:  $AR(N \times (N + 1)/2)$  – REAL (KIND=nag\_wp) array *Output*  
*On exit:* the upper or lower  $n$  by  $n$  triangular matrix  $A$  (as specified by UPLO) in either normal or transposed RFP format (as specified by TRANSR). The storage format is described in Section 3.3.3 in the F07 Chapter Introduction.
- 6: INFO – INTEGER *Output*  
*On exit:* INFO = 0 unless the routine detects an error (see Section 6).

## 6 Error Indicators and Warnings

$-999 < \text{INFO} < 0$

If  $\text{INFO} = -i$ , argument  $i$  had an illegal value. An explanatory message is output, and execution of the program is terminated.

$\text{INFO} = -999$

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

F01VJF (DTPPTF) is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example reads in a triangular matrix in packed format and copies it to RFP format.

## 10.1 Program Text

```

Program f01vjfe

!      F01VJF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
Use nag_library, Only: dtpttf, nag_wp, x04cbf
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: incl = 1, indent = 0, ncols = 80,    &
                             nin = 5, nout = 6
Character (1), Parameter   :: diag = 'N', intlabel = 'I',        &
                             matrix = 'G', nolabel = 'N'
Character (4), Parameter   :: form = 'F5.2'
!      .. Local Scalars ..
Integer                    :: ifail, info, k, lar1, lar2, lenap,    &
                             lenar, n, q
Character (47)             :: title
Character (1)              :: transr, uplo
!      .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: ap(:), ar(:)
Character (1)              :: clabs(1), rlabs(1)
!      .. Executable Statements ..
Write (nout,*) 'F01VJF Example Program Results'
!      Skip heading in data file
Read (nin,*)
Write (nout,*)
Flush (nout)
Read (nin,*) n, uplo, transr
lenap = (n*(n+1))/2
lenar = lenap

Allocate (ap(lenap),ar(lenar))

!      Read an order n matrix packed into a 1-D array
Read (nin,*) ap(1:lenap)

!      Print the packed array
title = 'Packed Array AP:'
ifail = 0
Call x04cbf(matrix,diag,lenap,incl,ap,lenap,form,title,intlabel,rlabs,    &
            nolabel,clabs,ncols,indent,ifail)

Write (nout,*)
Flush (nout)

!      Convert to Rectangular Full Packed form
!      The NAG name equivalent of dtpttf is f01vjf
Call dtpttf(transr,uplo,n,ap,ar,info)

!      Print the Rectangular Full Packed array
title = 'RFP Packed Array AR:'
ifail = 0
Call x04cbf(matrix,diag,lenar,incl,ar,lenar,form,title,intlabel,rlabs,    &
            nolabel,clabs,ncols,indent,ifail)

Write (nout,*)
Flush (nout)

!      Print the Rectangular Full Packed array showing how the elements are
!      arranged.
title = 'RFP Packed Array AR (graphical representation):'
k = n/2
q = n - k
If (transr=='N' .Or. transr=='n') Then
    lar1 = 2*k + 1
    lar2 = q

```

```

      Else
        lar1 = q
        lar2 = 2*k + 1
      End If

      ifail = 0
      Call x04cbf(matrix,diag,lar1,lar2,ar,lar1,form,title,intlabel,rlabs,      &
        intlabel,clabs,ncols,indent,ifail)

      End Program f01vjfe

```

## 10.2 Program Data

F01VJF Example Program Data

```

4 'U' 'N' : n, uplo, transr
1.1
1.2
2.2
1.3
2.3
3.3
1.4
2.4
3.4
4.4 : Packed Matrix AP

```

## 10.3 Program Results

F01VJF Example Program Results

Packed Array AP:

```

1 1.10
2 1.20
3 2.20
4 1.30
5 2.30
6 3.30
7 1.40
8 2.40
9 3.40
10 4.40

```

RFP Packed Array AR:

```

1 1.30
2 2.30
3 3.30
4 1.10
5 1.20
6 1.40
7 2.40
8 3.40
9 4.40
10 2.20

```

RFP Packed Array AR (graphical representation):

```

      1 2
1 1.30 1.40
2 2.30 2.40
3 3.30 3.40
4 1.10 4.40
5 1.20 2.20

```

---