NAG Toolbox

nag_matop_ztpttf (f01vk)

1 Purpose

nag_matop_ztpttf (f01vk) copies a complex triangular matrix, stored in a standard packed format array, to a Rectangular Full Packed (RFP) format array.

2 Syntax

```
[ar, info] = nag_matop_ztpttf(transr, uplo, n, ap)
[ar, info] = f01vk(transr, uplo, n, ap)
```

3 Description

nag_matop_ztpttf (f01vk) copies a complex n by n triangular matrix, A, stored in packed format, to RFP format. This function is intended for possible use in conjunction with functions from Chapters F07 and F16 where some functions that use triangular matrices store them in RFP format. The RFP storage format is described in Section 3.2.3 in the F07 Chapter Introduction and the packed storage format is described in Section 3.2.2 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 Parameters

5.1 Compulsory Input Parameters

1: **transr** – CHARACTER(1)

Specifies whether the normal RFP representation of A or its conjugate transpose is stored.

transr = 'N'
The RFP representation of the matrix A is stored.

transr = 'C'
The conjugate transpose of the RFP representation of the matrix A is stored.

Constraint: **transr** = 'N' or 'C'.

2: **uplo** – CHARACTER(1)

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

n, the order of the matrix A.

Constraint: **n** ≥ 0.

4: **ap**$(\mathbf{n} \times (\mathbf{n}+\mathbf{1})/\mathbf{2})$ – COMPLEX (KIND=nag_wp) array

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 $i \geq j$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **ar**$(\mathbf{n} \times (\mathbf{n}+\mathbf{1})/\mathbf{2})$ – COMPLEX (KIND=nag_wp) array

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 3.2.3 in the F07 Chapter Introduction.

2: **info** – INTEGER

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

6 Error Indicators and Warnings

$-999 <$ **info** < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

info $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

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

9.1 Program Text

```
function f01vk_example

fprintf('f01vk example results\n\n');

uplo   = 'u';
transr = 'n';
n      = nag_int(4);
ap     = [1.1 + 1.1i;
          1.2 + 1.2i;
          2.2 + 2.2i;
```

```
          1.3 + 1.3i;
          2.3 + 2.3i;
          3.3 + 3.3i;
          1.4 + 1.4i;
          2.4 + 2.4i;
          3.4 + 3.4i;
          4.4 + 4.4i];
% Print the packed vector
fprintf('\n');
[ifail] = x04db('g', 'x', ap, 'b', 'f5.2', 'Packed array ap:', 'i', ...
                'n', nag_int(80), nag_int(0));
% Convert to Rectangular Full Packed form
[ar, info] = f01vk(transr, uplo, n, ap);
% Print the Rectangular Full Packed array
fprintf('\n');
[ifail] = x04db('g', 'x', ar, 'b', 'f5.2', 'RFP packed array ar:', 'i', ...
                'n', nag_int(80), nag_int(0));

k = nag_int(n/2);
q = n - k;
if transr=='N' || transr=='n'
  lar1 = 2*k + 1;
  lar2 = q;
else
  lar1 = q;
  lar2 = 2*k + 1;
end

ar = reshape(ar,lar1,lar2);

fprintf('\n');
[ifail] = x04db('g', 'x', ar, 'b', 'f5.2', ...
                'RFP Packed Array ar (graphical representation):', 'i', ...
                'i', nag_int(80), nag_int(0), 'm', lar1, 'n', lar2);
```

9.2 Program Results

```
     f01vk example results

 Packed array ap:
  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)

 RFP packed array ar:
  1  ( 1.30, 1.30)
  2  ( 2.30, 2.30)
  3  ( 3.30, 3.30)
  4  ( 1.10,-1.10)
  5  ( 1.20,-1.20)
  6  ( 1.40, 1.40)
  7  ( 2.40, 2.40)
  8  ( 3.40, 3.40)
  9  ( 4.40, 4.40)
 10  ( 2.20,-2.20)

 RFP Packed Array ar (graphical representation):
                1             2
```

```
 1  ( 1.30, 1.30) ( 1.40, 1.40)
 2  ( 2.30, 2.30) ( 2.40, 2.40)
 3  ( 3.30, 3.30) ( 3.40, 3.40)
 4  ( 1.10,-1.10) ( 4.40, 4.40)
 5  ( 1.20,-1.20) ( 2.20,-2.20)
```