NAG Toolbox

nag_matop_complex_band_pack (f01zd)

1 Purpose

nag_matop_complex_band_pack (f01zd) copies a complex band matrix stored in a packed array into an unpacked array, or vice versa.

2 Syntax

```
[a, b, ifail] = nag_matop_complex_band_pack(job, kl, ku, a, b, 'm', m, 'n', n)
[a, b, ifail] = f01zd(job, kl, ku, a, b, 'm', m, 'n', n)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: **m** was made optional.

3 Description

nag_matop_complex_band_pack (f01zd) unpacks a band matrix that is stored in a packed array, or packs a band matrix that is stored in an unpacked array. The band matrix has m rows, n columns, k_l nonzero subdiagonals, and k_u nonzero superdiagonals. This function is intended for possible use in conjunction with functions from Chapters F07 and F08, where functions that use band matrices store them in the packed form described below.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **job** – CHARACTER(1)

Specifies whether the band matrix is to be packed or unpacked.

job = 'P' (Pack)
The band matrix is to be packed into array **b**.

job = 'U' (Unpack)
The band matrix is to be unpacked into array **a**.

Constraint: **job** = 'P' or 'U'.

2: **kl** – INTEGER

k_l, the number of subdiagonals of the band matrix.

Constraint: **kl** ≥ 0.

3: **ku** – INTEGER

k_u, the number of superdiagonals of the band matrix.

Constraint: **ku** ≥ 0.

4: **a**$(lda, \mathbf{n})$ – COMPLEX (KIND=nag_wp) array

lda, the first dimension of the array, must satisfy the constraint $lda \geq \mathbf{m}$.

If **job** = 'P', then the leading m by n part of **a** must contain the band matrix stored in unpacked form. Elements of the array that lie outside the banded part of the matrix are not referenced and need not be assigned.

5: **b**$(ldb, :)$ – COMPLEX (KIND=nag_wp) array

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

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

If **job** = 'U', then **b** must contain the band matrix in packed form, in the leading $(k_l + k_u + 1)$ by $\min(m + k_u, n)$ part of the array. The matrix is packed column by column, with the leading diagonal of the matrix in row $(k_u + 1)$ of **b**, the first superdiagonal starting at position 2 in row k_u, the first subdiagonal starting at position 1 in row $(k_u + 2)$, and so on. Elements of **b** that are not needed to store the band matrix, for instance the leading k_u by k_u triangle, are not referenced and need not be assigned.

5.2 Optional Input Parameters

1: **m** – INTEGER

2: **n** – INTEGER

Default: For **m**, the first dimension of the array **a**. the second dimension of the array **a**.

m and n, the number of rows and columns of the band matrix, respectively.

Constraints:

$\mathbf{m} > 0$;
$\mathbf{n} > 0$.

5.3 Output Parameters

1: **a**$(lda, \mathbf{n})$ – COMPLEX (KIND=nag_wp) array

If **job** = 'U', then the leading m by n part of **a** contains the band matrix stored in unpacked form. Elements of the leading m by n part of **a** that are not within the banded part of the matrix are assigned the value zero.

2: **b**$(ldb, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **b** will be $(\mathbf{kl} + \mathbf{ku} + 1)$.

The second dimension of the array **b** will be $\min(\mathbf{m} + \mathbf{ku}, \mathbf{n})$.

If **job** = 'P', then **b** contains the band matrix stored in packed form. Elements of **b** that are not needed to store the band matrix are not referenced.

3: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **job** $\neq$ 'P' or 'U'.

ifail $= 2$

On entry, **kl** < 0.

ifail $= 3$

On entry, **ku** < 0.

ifail $= 4$

On entry, $lda <$ **m**.

ifail $= 5$

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

ifail $= 6$

On entry, **m** < 1,
or **n** < 1.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

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

9.1 Program Text

```
function f01zd_example

fprintf('f01zd example results\n\n');

job = 'Pack';
kl = nag_int(1);
ku = nag_int(1);
a = [ 1.1 - 1.1i,  1.2 - 1.2i,  0   + 0i,    0   + 0i,    0   + 0i;
      2.1 - 2.1i,  2.2 - 2.2i,  2.3 - 2.3i,  0   + 0i,    0   + 0i;
      0   + 0i,    3.2 - 3.2i,  3.3 - 3.3i,  3.4 - 3.4i,  0   + 0i;
      0   + 0i,    0   + 0i,    4.3 - 4.3i,  4.4 - 4.4i,  4.5 - 4.5i;
      0   + 0i,    0   + 0i,    0   + 0i,    5.4 - 5.4i,  5.5 - 5.5i];
n = size(a,1);
b = complex(zeros(kl+ku+1,n));

[A, ab, ifail] = f01zd(job, kl, ku, a, b);

matrix = 'General';
```

```
diag = ' ';
title1 = 'Unpacked Matrix A:';
[ifail] = x04da(matrix, diag, A, title1);
fprintf('\n');
title1 = 'Packed Matrix AB:';
[ifail] = x04da(matrix, diag, ab, title1);
```

9.2 Program Results

```
     f01zd example results

 Unpacked Matrix A:
             1          2          3          4          5
 1      1.1000     1.2000     0.0000     0.0000     0.0000
       -1.1000    -1.2000     0.0000     0.0000     0.0000

 2      2.1000     2.2000     2.3000     0.0000     0.0000
       -2.1000    -2.2000    -2.3000     0.0000     0.0000

 3      0.0000     3.2000     3.3000     3.4000     0.0000
        0.0000    -3.2000    -3.3000    -3.4000     0.0000

 4      0.0000     0.0000     4.3000     4.4000     4.5000
        0.0000     0.0000    -4.3000    -4.4000    -4.5000

 5      0.0000     0.0000     0.0000     5.4000     5.5000
        0.0000     0.0000     0.0000    -5.4000    -5.5000

 Packed Matrix AB:
             1          2          3          4          5
 1      0.0000     1.2000     2.3000     3.4000     4.5000
        0.0000    -1.2000    -2.3000    -3.4000    -4.5000

 2      1.1000     2.2000     3.3000     4.4000     5.5000
       -1.1000    -2.2000    -3.3000    -4.4000    -5.5000

 3      2.1000     3.2000     4.3000     5.4000     0.0000
       -2.1000    -3.2000    -4.3000    -5.4000     0.0000
```