NAG Toolbox

nag_matop_real_gen_blkdiag_lu (f01lh)

1 Purpose

nag_matop_real_gen_blkdiag_lu (f01lh) factorizes a real almost block diagonal matrix.

2 Syntax

```
[a, pivot, tol, kpivot, ifail] = nag_matop_real_gen_blkdiag_lu(n, blkstr, a, tol, 'nbloks', nbloks, 'lena', lena)
[a, pivot, tol, kpivot, ifail] = f01lh(n, blkstr, a, tol, 'nbloks', nbloks, 'lena', lena)
```

3 Description

nag_matop_real_gen_blkdiag_lu (f01lh) factorizes a real almost block diagonal matrix, A, by row elimination with alternate row and column pivoting such that no 'fill-in' is produced. The code, which is derived from ARCECO described in Diaz *et al.* (1983), uses Level 1 and Level 2 BLAS. No three successive diagonal blocks may have columns in common and therefore the almost block diagonal matrix must have the form shown in the following diagram:

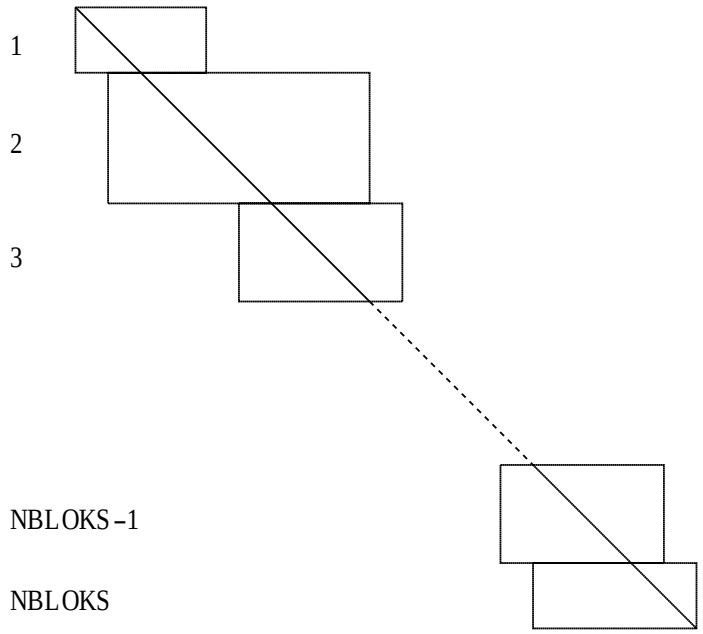

Figure 1

This function may be followed by nag_linsys_real_blkdiag_fac_solve (f04lh), which is designed to solve sets of linear equations $AX = B$ or $A^{\mathrm{T}}X = B$.

4 References

Diaz J C, Fairweather G and Keast P (1983) Fortran packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination *ACM Trans. Math. Software* **9** 358–375

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, the order of the matrix A.

Constraint: $\mathbf{n} > 0$.

2: **blkstr**$(\mathbf{3}, \mathbf{nbloks})$ – INTEGER array

Information which describes the block structure of A as follows:

blkstr$(1, k)$ must contain the number of rows in the kth block, $k = 1, 2, \ldots,$ **nbloks**;

blkstr$(2, k)$ must contain the number of columns in the kth block, $k = 1, 2, \ldots,$ **nbloks**;

blkstr$(3, k)$ must contain the number of columns of overlap between the kth and $(k+1)$th blocks, $k = 1, 2, \ldots,$ **nbloks** $- 1$. **blkstr**$(3,$ **nbloks**$)$ need not be set.

The following conditions delimit the structure of A:

$$\mathbf{blkstr}(1,k), \mathbf{blkstr}(2,k) > 0, \quad k = 1, 2, \ldots, \mathbf{nbloks},$$

$$\mathbf{blkstr}(3,k) \geq 0, \quad k = 1, 2, \ldots, \mathbf{nbloks} - 1,$$

(there must be at least one column and one row in each block and a non-negative number of columns of overlap);

$$\mathbf{blkstr}(3,k-1) + \mathbf{blkstr}(3,k) \leq \mathbf{blkstr}(2,k), \quad k = 2, 3, \ldots, \mathbf{nbloks} - 1,$$

(the total number of columns in overlaps in each block must not exceed the number of columns in that block);

$$\mathbf{blkstr}(2,1) \geq \mathbf{blkstr}(1,1),$$

$$\mathbf{blkstr}(2,1) + \sum_{k=2}^{j} [\mathbf{blkstr}(2,k) - \mathbf{blkstr}(3,k-1)] \geq \sum_{k=1}^{j} \mathbf{blkstr}(1,k),$$
$$j = 2, 3, \ldots, \mathbf{nbloks} - 1,$$

$$\sum_{k=1}^{j} [\mathbf{blkstr}(2,k) - \mathbf{blkstr}(3,k)] \leq \sum_{k=1}^{j} \mathbf{blkstr}(1,k), \quad j = 1, 2, \ldots, \mathbf{nbloks} - 1,$$

(the index of the first column of the overlap between the jth and $(j+1)$th blocks must be $\leq$ the index of the last row of the jth block, and the index of the last column of overlap must be $\geq$ the index of the last row of the jth block);

$$\sum_{k=1}^{\mathbf{nbloks}} \mathbf{blkstr}(1,k) = n,$$

$$\mathbf{blkstr}(2,1) + \sum_{k=2}^{\mathbf{nbloks}} [\mathbf{blkstr}(2,k) - \mathbf{blkstr}(3,k-1)] = nk,$$

(both the number of rows and the number of columns of A must equal n).

3: **a**$(\mathbf{lena})$ – REAL (KIND=nag_wp) array

The elements of the almost block diagonal matrix stored block by block, with each block stored column by column. The sizes of the blocks and the overlaps are defined by the argument **blkstr**.

If a_{rs} is the first element in the kth block, then an arbitrary element a_{ij} in the kth block must be stored in the array element:

$$\mathbf{a}(p_k + (j-r)m_k + (i-s) + 1)$$

where

$$p_k = \sum_{l=1}^{k-1} \mathbf{blkstr}(1,l) \times \mathbf{blkstr}(2,l)$$

is the base address of the kth block, and

$$m_k = \mathbf{blkstr}(1,k)$$

is the number of rows of the kth block.

See Section 9 for comments on scaling.

4: **tol** – REAL (KIND=nag_wp)

A relative tolerance to be used to indicate whether or not the matrix is singular. For a discussion on how **tol** is used see Section 9. If **tol** is non-positive, then **tol** is reset to 10ϵ, where ϵ is the ***machine precision***.

5.2 Optional Input Parameters

1: **nbloks** – INTEGER

Default: the dimension of the array **blkstr**.

n, the total number of blocks of the matrix A.

Constraint: $0 < \mathbf{nbloks} \leq \mathbf{n}$.

2: **lena** – INTEGER

Default: the dimension of the array **a**.

The dimension of the array **a**.

Constraint: $\mathbf{lena} \geq \sum_{k=1}^{\mathbf{nbloks}} [\mathbf{blkstr}(1,k) \times \mathbf{blkstr}(2,k)]$.

5.3 Output Parameters

1: **a**(**lena**) – REAL (KIND=nag_wp) array

The factorized form of the matrix.

2: **pivot**(**n**) – INTEGER array

Details of the interchanges.

3: **tol** – REAL (KIND=nag_wp)

Unchanged unless $\mathbf{tol} \leq 0.0$ on entry, in which case it is set to 10ϵ.

4: **kpivot** – INTEGER

If $\mathbf{ifail} = 2$, **kpivot** contains the value k, where k is the first position on the diagonal of the matrix A where too small a pivot was detected. Otherwise **kpivot** is set to 0.

5: **ifail** – INTEGER

$\mathbf{ifail} = 0$ 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, **n** < 1,
or **nbloks** < 1,
or **n** $<$ **nbloks**,
or **lena** is too small,
or illegal values detected in **blkstr**.

ifail $= 2$ (*warning*)

The factorization has been completed, but a small pivot has been detected.

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

The accuracy of nag_matop_real_gen_blkdiag_lu (f01lh) depends on the conditioning of the matrix A.

8 Further Comments

Singularity or near singularity in A is determined by the argument **tol**. If the absolute value of any pivot is less than **tol** $\times a_{\max}$, where $a_{\max}$ is the maximum absolute value of an element of A, then A is said to be singular. The position on the diagonal of A of the first of any such pivots is indicated by the argument **kpivot**. The factorization, and the test for near singularity, will be more accurate if before entry A is scaled so that the ∞-norms of the rows and columns of A are all of approximately the same order of magnitude. (The ∞-norm is the maximum absolute value of any element in the row or column.)

9 Example

This example solves the set of linear equations $Ax = b$ where

$$A = \begin{pmatrix}
-1.00 & -0.98 & -0.79 & -0.15 & & & & & & & & & & & & & & & & \\
-1.00 & -0.25 & -0.87 & 0.35 & & & & & & & & & & & & & & & & \\
 & 0.78 & 0.31 & -0.85 & 0.89 & -0.69 & -0.98 & -0.76 & & & & & & & & & & & & \\
 & -0.82 & 0.12 & -0.01 & 0.75 & 0.32 & -1.00 & -0.53 & & & & & & & & & & & & \\
 & -0.83 & -0.98 & -0.58 & 0.04 & 0.87 & 0.38 & -1.00 & & & & & & & & & & & & \\
 & -0.21 & -0.93 & -0.84 & 0.37 & -0.94 & -0.96 & -1.00 & & & & & & & & & & & & \\
 & & & & -0.99 & -0.91 & -0.28 & -0.90 & 0.78 & -0.93 & -0.76 & 0.48 & & & & & & & & \\
 & & & & -0.87 & -0.14 & -1.00 & -0.59 & -0.99 & 0.21 & -0.73 & -0.48 & & & & & & & & \\
 & & & & -0.93 & -0.91 & 0.10 & -0.89 & -0.68 & -0.09 & -0.58 & -0.21 & & & & & & & & \\
 & & & & 0.85 & -0.39 & 0.79 & -0.71 & 0.39 & -0.99 & -0.12 & -0.75 & & & & & & & & \\
 & & & & 0.17 & -1.37 & 1.29 & -1.59 & 1.10 & -1.63 & -1.01 & -0.27 & & & & & & & & \\
 & & & & & & & & & & 0.08 & 0.61 & 0.54 & -0.41 & 0.16 & -0.46 & & & & \\
 & & & & & & & & & & -0.67 & 0.56 & -0.99 & 0.16 & -0.16 & 0.98 & & & & \\
 & & & & & & & & & & -0.24 & -0.41 & 0.40 & -0.93 & 0.70 & 0.43 & & & & \\
 & & & & & & & & & & & & & 0.71 & -0.97 & -0.60 & -0.30 & 0.18 & \\
 & & & & & & & & & & & & & -0.47 & -0.98 & -0.73 & 0.07 & 0.04 & \\
 & & & & & & & & & & & & & -0.25 & -0.92 & -0.52 & -0.46 & -0.58 & \\
 & & & & & & & & & & & & & -0.89 & -0.94 & -0.54 & -1.00 & -0.36 &
\end{pmatrix}$$

and

$$b = \begin{pmatrix} -2.92 \\ -1.17 \\ -1.30 \\ -1.17 \\ -2.10 \\ -4.51 \\ -1.71 \\ -4.59 \\ -4.19 \\ -0.93 \\ -3.31 \\ 0.52 \\ -0.12 \\ -0.05 \\ -0.98 \\ -2.07 \\ -2.73 \\ -1.95 \end{pmatrix}$$

The exact solution is

$$x = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)^{\mathrm{T}}.$$

9.1 Program Text

```
     function f01lh_example

fprintf('f01lh example results\n\n');

% Block structure of A
n = nag_int(18);
blkstr = [nag_int(2),4,5,3,4;
                  4, 7,8,6,5;
                  3, 4,2,3,0];
a1 = [-1.00  -0.98  -0.79  -0.15;
      -1.00   0.25  -0.87   0.35];
a2 = [ 0.78   0.31  -0.85   0.89  -0.69  -0.98 -0.76;
      -0.82   0.12  -0.01   0.75   0.32  -1.00 -0.53;
      -0.83  -0.98  -0.58   0.04   0.87   0.38 -1.00;
      -0.21  -0.93  -0.84   0.37  -0.94  -0.96 -1.00];
a3 = [-0.99  -0.91  -0.28   0.90   0.78  -0.93  -0.76   0.48;
      -0.87  -0.14  -1.00  -0.59  -0.99   0.21  -0.73  -0.48;
      -0.93  -0.91   0.10  -0.89  -0.68  -0.09  -0.58  -0.21;
       0.85  -0.39   0.79  -0.71   0.39  -0.99  -0.12  -0.75;
       0.17  -1.37   1.29  -1.59   1.10  -1.63  -1.01  -0.27];
a4 = [ 0.08   0.61   0.54  -0.41   0.16  -0.46;
      -0.67   0.56  -0.99   0.16  -0.16   0.98;
      -0.24  -0.41   0.40  -0.93   0.70   0.43];
a5 = [ 0.71  -0.97  -0.60  -0.30   0.18;
      -0.47  -0.98  -0.73   0.07   0.04;
      -0.25  -0.92  -0.52  -0.46  -0.58;
       0.89  -0.94  -0.54  -1.00  -0.36];
% Flatten A
a = [reshape(a1,[ 8,1]);
     reshape(a2,[28,1]);
     reshape(a3,[40,1]);
     reshape(a4,[18,1]);
     reshape(a5,[20,1])];

% Right hand side
b  = [-2.92;  -1.27;  -1.30;  -1.17;  -2.10;  -4.51;  -1.71;  -4.59;
      -4.19;  -0.93;  -3.31;   0.52;  -0.12;  -0.05;  -0.98;  -2.07;
      -2.73;  -1.95];

% Factorize A
tol = 0;
[AF, pivot, tol, index, ifail] = ...
f01lh(n, blkstr, a, tol);

% Solve system
trans = 'N';
[x, ifail] = f04lh( ...
                   trans, blkstr, AF, pivot, b);
disp('Component solution');
disp(x);
```

9.2 Program Results

```
     f01lh example results

Component solution
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```