NAG Toolbox for MATLAB

f01lh

1 Purpose

f01lh factorizes a real almost block diagonal matrix.

2 Syntax

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

3 Description

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:

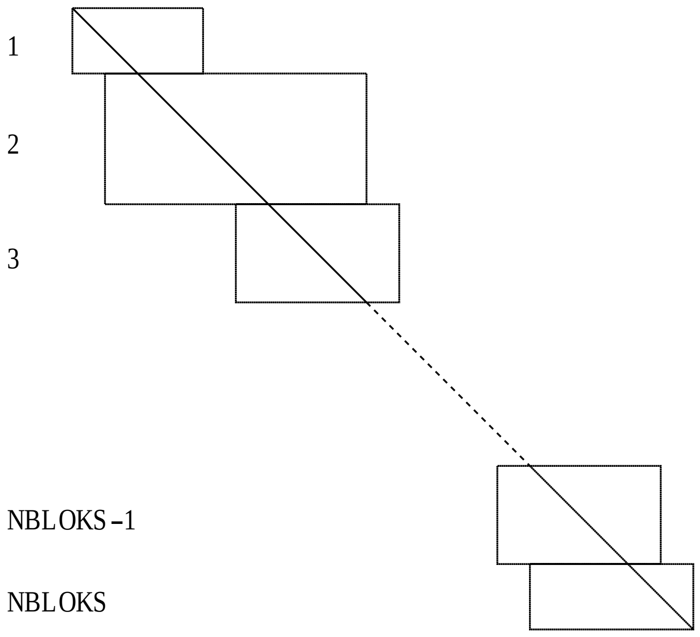

Figure 1

This function may be followed by 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 – int32 scalar**

n, the order of the matrix A.

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

2: **blkstr(3,nbloks) – int32 array**

Information which describes the block structure of A as follows:

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

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

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

The following conditions delimit the structure of A:

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

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

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

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

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

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

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

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

(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(lena) – double 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 parameter **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 8 for comments on scaling.

4: **tol – double scalar**

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 8. If **tol** is nonpositive, then **tol** is reset to 10ϵ, where ϵ is the ***machine precision*** .

5.2 Optional Input Parameters

1: **nbloks – int32 scalar**

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 – int32 scalar**

Default: 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 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **a(lena) – double array**

The factorized form of the matrix.

2: **pivot(n) – int32 array**

Details of the interchanges.

3: **tol – double scalar**

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

4: **kpivot – int32 scalar**

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 – int32 scalar**

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

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

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

7 Accuracy

The accuracy of f01lh depends on the conditioning of the matrix A.

8 Further Comments

Singularity or near singularity in A is determined by the parameter **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 parameter **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

```
n = int32(18);
blkstr = [int32(2), int32(4), int32(5), int32(3), int32(4);
          int32(4), int32(7), int32(8), int32(6), int32(5);
          int32(3), int32(4), int32(2), int32(3), int32(0)];
a = [-1;
          -1;
          -0.98;
          0.25;
          -0.79;
          -0.87;
          -0.15;
          0.35;
          0.78;
          -0.82;
          -0.83;
          -0.21;
          0.31;
          0.12;
          -0.98;
          -0.93;
          -0.85;
          -0.01;
          -0.58;
          -0.84;
          0.89;
          0.75;
          0.04;
          0.37;
          -0.6899999999999999;
          0.32;
          0.87;
```

```
      -0.9399999999999999;
      -0.98;
      -1;
      0.38;
      -0.96;
      -0.76;
      -0.53;
      -1;
      -1;
      -0.99;
      -0.87;
      -0.93;
      0.85;
      0.17;
      -0.91;
      -0.14;
      -0.91;
      -0.39;
      -1.37;
      -0.28;
      -1;
      0.1;
      0.79;
      1.29;
      0.9;
      -0.59;
      -0.89;
      -0.71;
      -1.59;
      0.78;
      -0.99;
      -0.68;
      0.39;
      1.1;
      -0.93;
      0.21;
      -0.09;
      -0.99;
      -1.63;
      -0.76;
      -0.73;
      -0.58;
      -0.12;
      -1.01;
      0.48;
      -0.48;
      -0.21;
      -0.75;
      -0.27;
      0.08;
      -0.67;
      -0.24;
      0.61;
      0.5600000000000001;
      -0.41;
      0.54;
      -0.99;
      0.4;
      -0.41;
      0.16;
      -0.93;
      0.16;
      -0.16;
      0.7;
      -0.46;
      0.98;
      0.43;
      0.71;
      -0.47;
      -0.25;
```

```
     0.89;
     -0.97;
     -0.98;
     -0.92;
     -0.9399999999999999;
     -0.6;
     -0.73;
     -0.52;
     -0.54;
     -0.3;
     0.07000000000000001;
     -0.46;
     -1;
     0.18;
     0.04;
     -0.58;
     -0.36];
tol = 0;
[aOut, pivot, tolOut, index, ifail] = f01lh(n, blkstr, a, tol)
```

```
aOut =
     array elided
pivot =
           1
           2
           3
           3
           6
           5
           5
           2
           5
           4
           8
           2
           2
           6
           4
           2
           4
           4
tolOut =
   1.1113e-15
index =
           0
ifail =
           0
```