F01LHF – NAG Fortran Library Routine Document

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

F01LHF factorizes a real almost block diagonal matrix.

2 Specification

```
 SUBROUTINE F01LHF(N, NBLOKS, BLKSTR, A, LENA, PIVOT, TOL, INDEX,
1                  IFAIL)
 INTEGER           N, NBLOKS, BLKSTR(3,NBLOKS), LENA, PIVOT(N),
1                  INDEX, IFAIL
 real              A(LENA), TOL
```

3 Description

The routine 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 [1], 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:

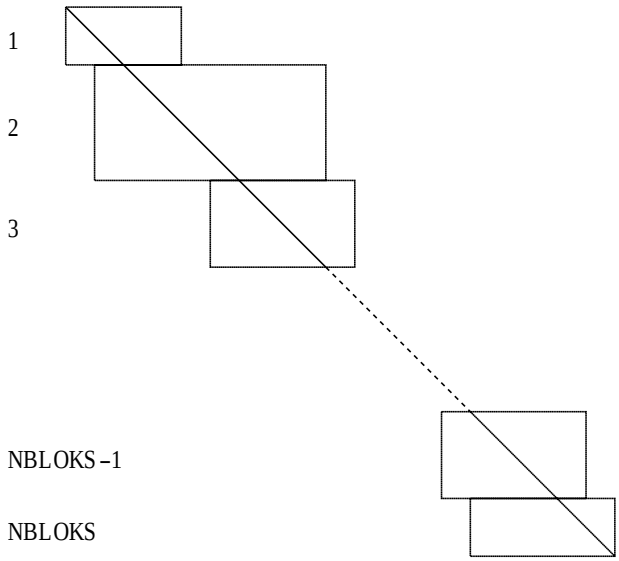

Figure 1

This routine may be followed by F04LHF, which is designed to solve sets of linear equations $AX = B$ or $A^TX = B$.

4 References

[1] 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

1: N — INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: N > 0.

2: NBLOKS — INTEGER *Input*

On entry: the total number of blocks of the matrix A.

Constraint: $0 <$ NBLOKS $\leq N$.

3: BLKSTR(3,NBLOKS) — INTEGER array *Input*

On entry: 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:

$$\begin{array}{ll} \text{BLKSTR}(1,k),\ \text{BLKSTR}(2,k) > 0, & k = 1, 2, \ldots, \text{NBLOKS}, \\ \text{BLKSTR}(3,k) \geq 0, & k = 1, 2, \ldots, \text{NBLOKS} - 1, \end{array}$$

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

$$\text{BLKSTR}(3,k-1) + \text{BLKSTR}(3,k) \leq \text{BLKSTR}(2,k), \quad k = 2, 3, \ldots, \text{NBLOKS} - 1$$

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

$$\begin{array}{lll} \text{BLKSTR}(2,1) & \geq \text{BLKSTR}(1,1), & \\ \text{BLKSTR}(2,1) + \displaystyle\sum_{k=2}^{j} [\text{BLKSTR}(2,k) - \text{BLKSTR}(3,k-1)] & \geq \displaystyle\sum_{k=1}^{j} \text{BLKSTR}(1,k), & \\ & & j = 2, 3, \ldots, \text{NBLOKS} - 1 \\ \displaystyle\sum_{k=1}^{j} [\text{BLKSTR}(2,k) - \text{BLKSTR}(3,k)] & \leq \displaystyle\sum_{k=1}^{j} \text{BLKSTR}(1,k), & j = 1, 2, \ldots, \text{NBLOKS} - 1, \end{array}$$

(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);

$$\begin{array}{l} \displaystyle\sum_{k=1}^{\text{NBLOKS}} \text{BLKSTR}(1,k) = n, \\ \text{BLKSTR}(2,1) + \displaystyle\sum_{k=2}^{\text{NBLOKS}} [\text{BLKSTR}(2,k) - \text{BLKSTR}(3,k-1)] = n \end{array}$$

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

4: A(LENA) — ***real*** array *Input/Output*

On entry: 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:

$$\mathrm{A}(p_k + (j-r)m_k + (i-s) + 1)$$

where

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

is the base address of the kth block, and

$$m_k = \mathrm{BLKSTR}(1,k)$$

which is the number of rows of the kth block.

See Section 8 for comments on scaling.

On exit: the factorized form of the matrix.

5: LENA — INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which F01LHF is called.

Constraint: $\mathrm{LENA} \geq \sum_{k=1}^{\mathrm{NBLOKS}} [\mathrm{BLKSTR}(1,k) \times \mathrm{BLKSTR}(2,k)]$.

6: PIVOT(N) — INTEGER array *Output*

On exit: details of the interchanges.

7: TOL — ***real*** *Input*

On entry: 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 non-positive, then TOL is reset to 10ϵ, where ϵ is the ***machine precision***.

8: INDEX — INTEGER *Output*

On exit: if IFAIL = 2, INDEX 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 INDEX is set to 0.

9: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

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 this routine 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 INDEX. The factorization, and the test for near singularity, will be more accurate if before entry A is scaled so that the infinity-norms of the rows and columns of A are all of approximately the same order of magnitude. (The infinity-norm is the maximum absolute value of any element in the row or column.)

9 Example

To solve 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

$$x = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T.$$

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F01LHF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NBLMAX, NMAX, IRMAX, LENA, LDB
      PARAMETER        (NBLMAX=10,NMAX=20,IRMAX=5,LENA=200,LDB=NMAX)
*     .. Local Scalars ..
      real             TOL
      INTEGER          I, IFAIL, INDEX, IR, J, K, N, NBASEK, NBLOKS
*     .. Local Arrays ..
      real             A(LENA), B(LDB,IRMAX)
      INTEGER          BLKSTR(3,NBLMAX), PIVOT(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F01LHF, F04LHF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F01LHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NBLOKS
      WRITE (NOUT,*)
      IF (NBLOKS.LE.NBLMAX) THEN
         NBASEK = 0
         N = 0
         DO 40 I = 1, NBLOKS
            READ (NIN,*) (BLKSTR(J,I),J=1,3)
            DO 20 K = 1, BLKSTR(1,I)
               IF (NBASEK+BLKSTR(2,I)*BLKSTR(1,I).GT.LENA) THEN
                  WRITE (NOUT,*)
     +              ' Array A is too small for this problem'
                  STOP
               ELSE
                  READ (NIN,*) (A(NBASEK+(J-1)*BLKSTR(1,I)+K),J=1,
     +              BLKSTR(2,I))
               END IF
   20       CONTINUE
            NBASEK = NBASEK + BLKSTR(2,I)*BLKSTR(1,I)
            N = N + BLKSTR(1,I)
   40    CONTINUE
         IF (N.GT.NMAX) THEN
            WRITE (NOUT,*) ' N is too large'
            STOP
         END IF
         TOL = 0.0e0
         IFAIL = -1
*
         CALL F01LHF(N,NBLOKS,BLKSTR,A,LENA,PIVOT,TOL,INDEX,IFAIL)
*
         IF (IFAIL.EQ.0) THEN
            READ (NIN,*) IR
            IF (IR.LE.IRMAX) THEN
               READ (NIN,*) ((B(I,J),I=1,N),J=1,IR)
               IFAIL = -1
*
```

```
               CALL F04LHF('N',N,NBLOKS,BLKSTR,A,LENA,PIVOT,B,LDB,IR,
     +                     IFAIL)
*
               IF (IFAIL.EQ.0) THEN
                  WRITE (NOUT,*) 'Component Solution'
                  WRITE (NOUT,*)
                  DO 60 I = 1, N
                     WRITE (NOUT,99999) I, (B(I,J),J=1,IR)
   60             CONTINUE
               END IF
            ELSE
               WRITE (NOUT,*) ' Too many right hand sides specified'
            END IF
         END IF
      ELSE
         WRITE (NOUT,*) ' NBLOKS is invalid'
      END IF
      STOP
*
99999 FORMAT (1X,I5,6X,5F6.4)
      END
```

9.2 Program Data

```
F01LHF Example Program Data
  5        : Number of blocks
  2  4  3  : Number of rows, columns and column overlap, block 1
-1.00 -0.98 -0.79 -0.15
-1.00  0.25 -0.87  0.35                          : End block 1
  4  7  4  : Number of rows, columns and column overlap, block 2
 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          : End block 2
  5  8  2  : Number of rows, columns and column overlap, block 3
-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    : End block 3
  3  6  3  : Number of rows, columns and column overlap, block 4
 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                : End block 4
  4  5  0  : Number of rows, columns and column overlap, block 5
 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 block 5
  1        : Number of right hand sides
-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                              : End right hand side 1
```

9.3 Program Results

```
F01LHF Example Program Results

Component Solution

    1      1.0000
    2      1.0000
    3      1.0000
    4      1.0000
    5      1.0000
    6      1.0000
    7      1.0000
    8      1.0000
    9      1.0000
   10      1.0000
   11      1.0000
   12      1.0000
   13      1.0000
   14      1.0000
   15      1.0000
   16      1.0000
   17      1.0000
   18      1.0000
```