

# NAG Library Routine Document

## F01LHF

**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

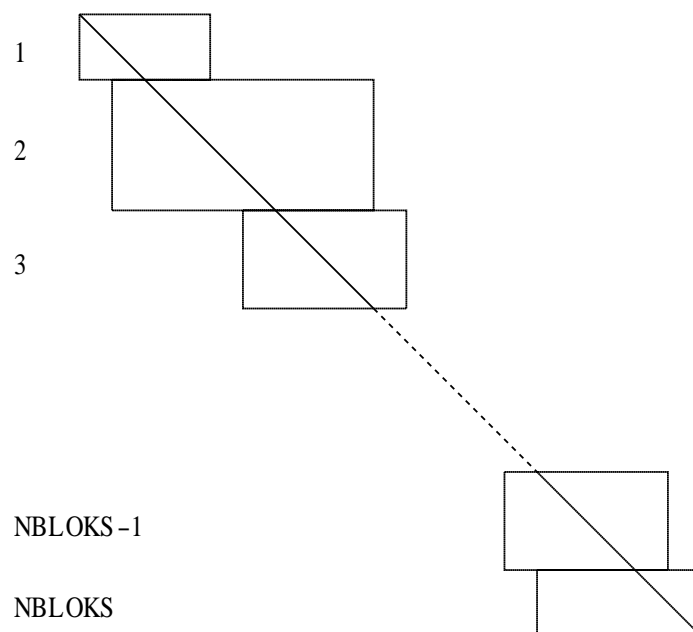
```

SUBROUTINE F01LHF (N, NBLOKS, BLKSTR, A, LENA, PIVOT, TOL, KPIVOT,      &
                  IFAIL)
INTEGER              N, NBLOKS, BLKSTR(3,NBLOKS), LENA, PIVOT(N),      &
                  KPIVOT, IFAIL
REAL (KIND=nag_wp) A(LENA), TOL

```

### 3 Description

F01LHF 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 routine may be followed by F04LHF, which is designed to solve sets of linear equations  $AX = B$  or  $A^T X = B$ .

### 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 Arguments

- 1: N – INTEGER Input

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

*Constraint:*  $N > 0$ .

- 2: NBLOKS – INTEGER Input

*On entry:*  $n$ , the total number of blocks of the matrix  $A$ .

*Constraint:*  $0 < \text{NBLOKS} \leq N$ .

- 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  $k$ th block,  $k = 1, 2, \dots, \text{NBLOKS}$ ;

BLKSTR(2, $k$ ) must contain the number of columns in the  $k$ th block,  $k = 1, 2, \dots, \text{NBLOKS}$ ;

BLKSTR(3, $k$ ) must contain the number of columns of overlap between the  $k$ th and ( $k+1$ )th blocks,  $k = 1, 2, \dots, \text{NBLOKS} - 1$ . BLKSTR(3,NBLOKS) need not be set.

The following conditions delimit the structure of  $A$ :

$$\text{BLKSTR}(1, k), \text{BLKSTR}(2, k) > 0, \quad k = 1, 2, \dots, \text{NBLOKS},$$

$$\text{BLKSTR}(3, k) \geq 0, \quad k = 1, 2, \dots, \text{NBLOKS} - 1,$$

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

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

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

$$\text{BLKSTR}(2, 1) \geq \text{BLKSTR}(1, 1),$$

$$\text{BLKSTR}(2, 1) + \sum_{k=2}^j [\text{BLKSTR}(2, k) - \text{BLKSTR}(3, k-1)] \geq \sum_{k=1}^j \text{BLKSTR}(1, k),$$

$$j = 2, 3, \dots, \text{NBLOKS} - 1,$$

$$\sum_{k=1}^j [\text{BLKSTR}(2, k) - \text{BLKSTR}(3, k)] \leq \sum_{k=1}^j \text{BLKSTR}(1, k), \quad j = 1, 2, \dots, \text{NBLOKS} - 1,$$

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

$$\sum_{k=1}^{\text{NBLOKS}} \text{BLKSTR}(1, k) = n,$$

$$\text{BLKSTR}(2, 1) + \sum_{k=2}^{\text{NBLOKS}} [\text{BLKSTR}(2, k) - \text{BLKSTR}(3, k-1)] = nk,$$

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

- 4: A(LENA) – REAL (KIND=nag\_wp) 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 argument BLKSTR.

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

$$A(p_k + (j - r)m_k + (i - s) + 1)$$

where

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

is the base address of the  $k$ th block, and

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

is the number of rows of the  $k$ th block.

See Section 9 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.

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

6: PIVOT(N) – INTEGER array *Output*

*On exit:* details of the interchanges.

7: TOL – REAL (KIND=nag\_wp) *Input/Output*

*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 9. If TOL is non-positive, then TOL is reset to  $10\epsilon$ , where  $\epsilon$  is the *machine precision*.

*On exit:* unchanged unless  $\text{TOL} \leq 0.0$  on entry, in which case it is set to  $10\epsilon$ .

8: KPIVOT – INTEGER *Output*

*On exit:* if 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.

9: IFAIL – INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

*On exit:* IFAIL = 0 unless the routine detects an error or a warning has been flagged (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 or warnings 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.

$IFAIL = -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

$IFAIL = -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

$IFAIL = -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

The accuracy of F01LHF depends on the conditioning of the matrix  $A$ .

## 8 Parallelism and Performance

F01LHF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 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  $\infty$ -norms of the rows and columns of  $A$  are all of approximately the same order of magnitude. (The  $\infty$ -norm is the maximum absolute value of any element in the row or column.)



```

      Go To 100
    Else
      Do k = 1, blkstr(1,i)
        Read (nin,*)(a(nbasek+(j-1)*blkstr(1,i)+k),j=1,blkstr(2,i))
      End Do
    End If
    nbasek = nbasek + blkstr(2,i)*blkstr(1,i)
    n = n + blkstr(1,i)
  End Do
  ldb = n
  Read (nin,*) ir
  Allocate (b(ldb,ir),pivot(n))
  tol = 0.0E0_nag_wp

!      ifail: behaviour on error exit
!      =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
  ifail = 0
  Call f01lhf(n,nbloks,blkstr,a,lens,pivot,tol,kpivot,ifail)

  Read (nin,*)(b(1:n,j),j=1,ir)

  ifail = 0
  Call f04lhf('N',n,nbloks,blkstr,a,lens,pivot,b,ldb,ir,ifail)

  Call x04caf('G','X',n,ir,b,ldb,'Component Solution',ifail)
100  Continue

      End Program f01lhfe

```

## 10.2 Program Data

F01LHF Example Program Data

```

      5      : nbloks
      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

```

## 10.3 Program Results

F01LHF Example Program Results

Component Solution

```

      1
      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

---