NAG Library Routine Document

F04LHF

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

F04LHF calculates the approximate solution of a set of real linear equations with multiple right-hand sides, $AX = B$ or $A^{\mathrm{T}}X = B$, where A is an almost block-diagonal matrix which has been factorized by F01LHF.

2 Specification

```
SUBROUTINE F04LHF (TRANS, N, NBLOKS, BLKSTR, A, LENA, PIVOT, B, LDB, IR,    &
                   IFAIL)
INTEGER            N, NBLOKS, BLKSTR(3,NBLOKS), LENA, PIVOT(N), LDB,        &
                   IR, IFAIL
REAL (KIND=nag_wp) A(LENA), B(LDB,IR)
CHARACTER(1)       TRANS
```

3 Description

F04LHF solves a set of real linear equations $AX = B$ or $A^{\mathrm{T}}X = B$, where A is almost block-diagonal. A must first be factorized by F01LHF. F04LHF then computes X by forward and backward substitution over the blocks.

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

1: TRANS – CHARACTER(1) *Input*

On entry: specifies the equations to be solved.

TRANS = 'N'
Solve $AX = B$.

TRANS = 'T'
Solve $A^{\mathrm{T}}X = B$.

Constraint: TRANS = 'N' or 'T'.

2: N – INTEGER *Input*

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

Constraint: N > 0.

3: NBLOKS – INTEGER *Input*

On entry: the total number of blocks of the matrix A, as supplied to F04LHF.

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

4: BLKSTR(3, NBLOKS) – INTEGER array *Input*

On entry: information which describes the block structure of A, as supplied to F04LHF.

5: A(LENA) – REAL (KIND=nag_wp) array *Input*

On entry: the elements in the factorization of A, as returned by F04LHF.

6: LENA – INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which F04LHF is called.

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

7: PIVOT(N) – INTEGER array *Input*

On entry: details of the interchanges in the factorization, as returned by F04LHF.

8: B(LDB, IR) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the n by r right-hand side matrix B.

On exit: B is overwritten by the n by r solution matrix X.

9: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F04LHF is called.

Constraint: LDB $\geq$ N.

10: IR – INTEGER *Input*

On entry: r, the number of right-hand sides.

Constraint: IR > 0.

11: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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 IR < 1,

or LDB $<$ N,
or N $<$ NBLOKS,
or LENA is too small,
or illegal values detected in BLKSTR,
or TRANS $\neq$ 'N' or 'T'.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The accuracy of the computed solution depends on the conditioning of the original matrix A.

8 Parallelism and Performance

F04LHF is not threaded by NAG in any implementation.

F04LHF 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

None.

10 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

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

10.1 Program Text

```
    Program f04lhfe

!     F04LHF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f01lhf, f04lhf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lena = 200, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: tol
      Integer                          :: i, ifail, index, ir, j, k, ldb, n,   &
                                          nbasek, nbloks
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:,:)
      Integer, Allocatable             :: blkstr(:,:), pivot(:)
!     .. Executable Statements ..
      Write (nout,*) 'F04LHF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nbloks
      Allocate (a(lena),blkstr(3,nbloks))
      nbasek = 0
      n = 0
      Do i = 1, nbloks
        Read (nin,*) blkstr(1:3,i)
        Do k = 1, blkstr(1,i)
          If (nbasek+blkstr(2,i)*blkstr(1,i)>lena) Then
            Write (nout,*) ' Array A is too small for this problem'
            Go To 100
          Else
            Read (nin,*)(a(nbasek+(j-1)*blkstr(1,i)+k),j=1,blkstr(2,i))
          End If
        End Do
        nbasek = nbasek + blkstr(2,i)*blkstr(1,i)
        n = n + blkstr(1,i)
      End Do
```

```
      Read (nin,*) ir
      ldb = n
      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,lena,pivot,tol,index,ifail)

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

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

      Write (nout,*) 'Component Solution'
      Do i = 1, n
        Write (nout,99999) i, b(i,1:ir)
      End Do
100   Continue

99999 Format (1X,I5,6X,5F7.4)
    End Program f04lhfe
```

10.2 Program Data

```
F04LHF 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
```

10.3 Program Results

```
 F04LHF 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
```