NAG Library Routine Document

F11MHF

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

F11MHF returns error bounds for the solution of a real sparse system of linear equations with multiple right-hand sides, $AX = B$ or $A^{\mathrm{T}}X = B$. It improves the solution by iterative refinement in standard precision, in order to reduce the backward error as much as possible.

2 Specification

```
SUBROUTINE F11MHF (TRANS, N, ICOLZP, IROWIX, A, IPRM, IL, LVAL, IU, UVAL,      &
                   NRHS, B, LDB, X, LDX, FERR, BERR, IFAIL)
INTEGER            N, ICOLZP(*), IROWIX(*), IPRM(7*N), IL(*), IU(*), NRHS,     &
                   LDB, LDX, IFAIL
REAL (KIND=nag_wp) A(*), LVAL(*), UVAL(*), B(LDB,*), X(LDX,*), FERR(NRHS),     &
                   BERR(NRHS)
CHARACTER(1)       TRANS
```

3 Description

F11MHF returns the backward errors and estimated bounds on the forward errors for the solution of a real system of linear equations with multiple right-hand sides $AX = B$ or $A^{\mathrm{T}}X = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of F11MHF in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that if x is the exact solution of a perturbed system:

$$(A + \delta A)x = b + \delta b$$
$$\text{then} \quad \left|\delta a_{ij}\right| \leq \beta\left|a_{ij}\right| \quad \text{and} \quad |\delta b_i| \leq \beta|b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i|x_i - \hat{x}_i|/\max_i|x_i|$$

where $\hat{x}$ is the true solution.

The routine uses the LU factorization $P_rAP_c = LU$ computed by F11MEF and the solution computed by F11MFF.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1: TRANS – CHARACTER(1) *Input*

On entry: specifies whether $AX = B$ or $A^{\mathrm{T}}X = B$ is solved.

TRANS = 'N'
$AX = B$ is solved.

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

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

2: N – INTEGER *Input*

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

Constraint: $\mathrm{N} \geq 0$.

3: ICOLZP($*$) – INTEGER array *Input*

Note: the dimension of the array ICOLZP must be at least $\mathrm{N} + 1$.

On entry: ICOLZP(i) contains the index in A of the start of a new column. See Section 2.1.3 in the F11 Chapter Introduction.

4: IROWIX($*$) – INTEGER array *Input*

Note: the dimension of the array IROWIX must be at least $\mathrm{ICOLZP}(\mathrm{N} + 1) - 1$, the number of nonzeros of the sparse matrix A.

On entry: the row index array of sparse matrix A.

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

Note: the dimension of the array A must be at least $\mathrm{ICOLZP}(\mathrm{N} + 1) - 1$, the number of nonzeros of the sparse matrix A.

On entry: the array of nonzero values in the sparse matrix A.

6: IPRM($7 \times \mathrm{N}$) – INTEGER array *Input*

On entry: the column permutation which defines P_c, the row permutation which defines P_r, plus associated data structures as computed by F11MEF.

7: IL($*$) – INTEGER array *Input*

Note: the dimension of the array IL must be at least as large as the dimension of the array of the same name in F11MEF.

On entry: records the sparsity pattern of matrix L as computed by F11MEF.

8: LVAL($*$) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array LVAL must be at least as large as the dimension of the array of the same name in F11MEF.

On entry: records the nonzero values of matrix L and some nonzero values of matrix U as computed by F11MEF.

9: IU($*$) – INTEGER array *Input*

Note: the dimension of the array IU must be at least as large as the dimension of the array of the same name in F11MEF.

On entry: records the sparsity pattern of matrix U as computed by F11MEF.

10: UVAL($*$) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array UVAL must be at least as large as the dimension of the array of the same name in F11MEF.

On entry: records some nonzero values of matrix U as computed by F11MEF.

11: NRHS – INTEGER *Input*

On entry: *nrhs*, the number of right-hand sides in B.

Constraint: NRHS ≥ 0.

12: B(LDB,$*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array B must be at least $\max(1, \text{NRHS})$.

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

13: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F11MHF is called.

Constraint: LDB $\geq \max(1, \text{N})$.

14: X(LDX,$*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X must be at least $\max(1, \text{NRHS})$.

On entry: the n by *nrhs* solution matrix X, as returned by F11MFF.

On exit: the n by *nrhs* improved solution matrix X.

15: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F11MHF is called.

Constraint: LDX $\geq \max(1, \text{N})$.

16: FERR(NRHS) – REAL (KIND=nag_wp) array *Output*

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots,$ *nrhs*.

17: BERR(NRHS) – REAL (KIND=nag_wp) array *Output*

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots,$ *nrhs*.

18: 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, TRANS $\neq$ 'N' or 'T',
or N < 0,
or NRHS < 0,
or LDB $< \max(1, \mathrm{N})$,
or LDX $< \max(1, \mathrm{N})$.

IFAIL $= 2$

Ill-defined row permutation in array IPRM. Internal checks have revealed that the IPRM array is corrupted.

IFAIL $= 3$

Ill-defined column permutations in array IPRM. Internal checks have revealed that the IPRM array is corrupted.

IFAIL $= 301$

Unable to allocate required internal workspace.

7 Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8 Further Comments

At most five steps of iterative refinement are performed, but usually only one or two steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^{\mathrm{T}}x = b$;

9 Example

This example solves the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 2.00 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 1.00 & -1.00 & 0 \\ 4.00 & 0 & 1.00 & 0 & 1.00 \\ 0 & 0 & 0 & 1.00 & 2.00 \\ 0 & -2.00 & 0 & 0 & 3.00 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1.56 & 3.12 \\ -0.25 & -0.50 \\ 3.60 & 7.20 \\ 1.33 & 2.66 \\ 0.52 & 1.04 \end{pmatrix}.$$

Here A is nonsymmetric and must first be factorized by F11MEF.

9.1 Program Text

```
    Program f11mhfe

!     F11MHF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
```

```
      Use nag_library, Only: f11mdf, f11mef, f11mff, f11mhf, nag_wp, x04caf,   &
                             x04cbf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.E0_nag_wp
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: flop, thresh
      Integer                          :: i, ifail, j, ldb, ldx, n, nnz, nnzl, &
                                          nnzu, nrhs, nzlmx, nzlumx, nzumx
      Character (1)                    :: spec, trans
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:,:), berr(:), ferr(:),      &
                                          lval(:), uval(:), x(:,:)
      Integer, Allocatable             :: icolzp(:), il(:), iprm(:),           &
                                          irowix(:), iu(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F11MHF Example Program Results'
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)

!     Read order of matrix and number of right hand sides

      Read (nin,*) n, nrhs
      ldb = n
      ldx = n

      Allocate (b(ldb,nrhs),berr(nrhs),ferr(nrhs),x(ldx,nrhs),icolzp(n+1), &
        iprm(7*n))

!     Read the matrix A

      Read (nin,*) icolzp(1:n+1)
      nnz = icolzp(n+1) - 1

      Allocate (a(nnz),lval(8*nnz),uval(8*nnz),il(7*n+8*nnz+4),irowix(nnz), &
        iu(2*n+8*nnz+1))

      Do i = 1, nnz
        Read (nin,*) a(i), irowix(i)
      End Do

!     Read the right hand sides

      Do j = 1, nrhs
        Read (nin,*) x(1:n,j)
        b(1:n,j) = x(1:n,j)
      End Do

!     Calculate COLAMD permutation

      spec = 'M'

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11mdf(spec,n,icolzp,irowix,iprm,ifail)

!     Factorise

      thresh = one
      ifail = 0
      nzlmx = 8*nnz
      nzlumx = 8*nnz
      nzumx = 8*nnz

      Call f11mef(n,irowix,a,iprm,thresh,nzlmx,nzlumx,nzumx,il,lval,iu,uval, &
        nnzl,nnzu,flop,ifail)
```

```
!     Compute solution in array X

      trans = 'N'

      ifail = 0
      Call f11mff(trans,n,iprm,il,lval,iu,uval,nrhs,x,ldx,ifail)

!     Improve solution, and compute backward errors and estimated
!     bounds on the forward errors

      Call f11mhf(trans,n,icolzp,irowix,a,iprm,il,lval,iu,uval,nrhs,b,ldb,x, &
        ldx,ferr,berr,ifail)

!     Print solution

      Write (nout,*)
      Flush (nout)

      Call x04caf('G',' ',n,nrhs,x,ldx,'Solutions',ifail)
      Call x04cbf('G','X',nrhs,1,ferr,nrhs,'1PE8.1','Estimated Forward Error', &
        'N',rlabs,'N',clabs,80,0,ifail)
      Call x04cbf('G','X',nrhs,1,berr,nrhs,'1PE8.1','Backward Error','N', &
        rlabs,'N',clabs,80,0,ifail)

    End Program f11mhfe
```

9.2 Program Data

```
F11MHF Example Program Data
  5 2  N, NRHS
 1
 3
 5
 7
 9
 12   ICOLZP(I) I=1,..,N+1
  2.   1
  4.   3
  1.   1
 -2.   5
  1.   2
  1.   3
 -1.   2
  1.   4
  1.   3
  2.   4
  3.   5    A(I), IROWIX(I) I=1,NNZ
 1.56 -.25 3.6 1.33 .52
 3.12 -.50 7.2 2.66 1.04 X(I,J) J=1,NRHS I=1,N
```

9.3 Program Results

```
 F11MHF Example Program Results

 Solutions
             1          2
 1      0.7000     1.4000
 2      0.1600     0.3200
 3      0.5200     1.0400
 4      0.7700     1.5400
 5      0.2800     0.5600
 Estimated Forward Error
   5.0E-15
   5.0E-15
 Backward Error
   3.6E-17
   3.6E-17
```