NAG Library Routine Document

F11MDF

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

F11MDF computes a column permutation suitable for LU factorization (by F11MEF) of a real sparse matrix in compressed column (Harwell–Boeing) format and applies it to the matrix. This routine must be called prior to F11MEF.

2 Specification

```
SUBROUTINE F11MDF (SPEC, N, ICOLZP, IROWIX, IPRM, IFAIL)
INTEGER       N, ICOLZP(*), IROWIX(*), IPRM(7*N), IFAIL
CHARACTER(1) SPEC
```

3 Description

Given a sparse matrix in compressed column (Harwell–Boeing) format A and a choice of column permutation schemes, the routine computes those data structures that will be needed by the LU factorization routine F11MEF and associated routines F11MMF, F11MFF and F11MHF. The column permutation choices are:

original order (that is, no permutation);

user-supplied permutation;

a permutation, computed by the routine, designed to minimize fill-in during the LU factorization.

The algorithm for this computed permutation is based on the approximate minimum degree column ordering algorithm COLAMD. The computed permutation is not sensitive to the magnitude of the nonzero values of A.

4 References

Amestoy P R, Davis T A and Duff I S (1996) An approximate minimum degree ordering algorithm *SIAM J. Matrix Anal. Appl.* **17** 886–905

Gilbert J R and Larimore S I (2004) A column approximate minimum degree ordering algorithm *ACM Trans. Math. Software* **30,3** 353–376

Gilbert J R, Larimore S I and Ng E G (2004) Algorithm 836: COLAMD, an approximate minimum degree ordering algorithm *ACM Trans. Math. Software* **30, 3** 377–380

5 Parameters

1: SPEC – CHARACTER(1) *Input*

On entry: indicates the permutation to be applied.

SPEC = 'N'
The identity permutation is used (i.e., the columns are not permuted).

SPEC = 'U'
The permutation in the IPRM array is used, as supplied by you.

SPEC = 'M'

The permutation computed by the COLAMD algorithm is used

Constraint: SPEC = 'N', 'U' or 'M'.

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: IROWIX(i) contains the row index in A for element $A(i)$. See Section 2.1.3 in the F11 Chapter Introduction.

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

On entry: the first N entries contain the column permutation supplied by you. This will be used if SPEC = 'U', and ignored otherwise. If used, it must consist of a permutation of all the integers in the range $[0, (\mathrm{N} - 1)]$, the leftmost column of the matrix A denoted by 0 and the rightmost by $\mathrm{N} - 1$. Labelling columns in this way, $\mathrm{IPRM}(i) = j$ means that column $i - 1$ of A is in position j in AP_c, where $P_rAP_c = LU$ expresses the factorization to be performed.

On exit: a new permutation is returned in the first N entries. The rest of the array contains data structures that will be used by other routines. The routine computes the column elimination tree for A and a post-order permutation on the tree. It then compounds the IPRM permutation given or computed by the COLAMD algorithm with the post-order permutation. This array is needed by the LU factorization routine F11MEF and associated routines F11MFF, F11MHF and F11MMF and should be passed to them unchanged.

6: 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, SPEC $\neq$ 'N', 'U' or 'M',
or N < 0.

IFAIL $= 2$

On entry, SPEC = 'U', but IPRM does not represent a valid permutation of the integers in $[0, (\mathrm{N} - 1)]$. An input value of IPRM is either out of range or repeated.

IFAIL $= 3$

Unspecified failure of the COLAMD algorithm. This should not happen and should be reported to NAG.

IFAIL $= 4$

On entry, the array ICOLZP failed to satisfy the following constraints:

$\mathrm{ICOLZP}(1) = 1$;

$\mathrm{ICOLZP}(i-1) \leq \mathrm{ICOLZP}(i)$, for $i = 2, 3, \ldots, \mathrm{N} + 1$;

$\mathrm{ICOLZP}(i) \leq \mathrm{N} \times \mathrm{N} + 1$, for $i = 1, 2, \ldots, \mathrm{N} + 1$.

IFAIL $= 5$

On entry, the array IROWIX failed to satisfy the following constraints:

$1 \leq \mathrm{IROWIX}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots, \mathrm{ICOLZP}(\mathrm{N} + 1) - 1$;

for each column $i = 1 \ldots \mathrm{N}$, the row indices $\mathrm{IROWIX}(j)$, where $j = \mathrm{ICOLZP}(i) \ldots \mathrm{ICOLZP}(i + 1) - 1$, do not repeat.

IFAIL $= 301$

Unable to allocate required internal workspace.

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

Not applicable. This computation does not use floating-point numbers.

8 Parallelism and Performance

F11MDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

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

We recommend calling this routine with SPEC = 'M' before calling F11MEF. The COLAMD algorithm computes a sparsity-preserving permutation P_c solely from the pattern of A such that the LU factorization $P_rAP_c = LU$ remains as sparse as possible, regardless of the subsequent choice of P_r. The algorithm takes advantage of the existence of super-columns (columns with the same sparsity pattern) to reduce running time.

10 Example

This example computes a sparsity preserving column permutation for the LU factorization of the matrix A, where

$$A = \begin{pmatrix} 2.00 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 1.00 & -1.00 & 0 \\ 4.00 & 0 & 1.00 & 0 & 1.00 \\ 0 & 0 & 0 & 1.00 & 2.00 \\ 0 & -2.00 & 0 & 0 & 3.00 \end{pmatrix}.$$

10.1 Program Text

```
    Program f11mdfe

!     F11MDF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f11mdf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: ifail, n, nnz
      Character (1)                    :: spec
!     .. Local Arrays ..
      Integer, Allocatable             :: icolzp(:), iprm(:), irowix(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11MDF Example Program Results'
!     Skip heading in data file
      Read (nin,*)

!     Read order of matrix

      Read (nin,*) n

      Allocate (icolzp(n+1),iprm(7*n))

!     Read the matrix

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

      Allocate (irowix(nnz))

      Read (nin,*) irowix(1:nnz)
```

```
!     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)

!     Output results

      Write (nout,*)
      Write (nout,*) 'COLAMD Permutation'
      Write (nout,'(10I6)') iprm(1:n)

!     Calculate user permutation

      spec = 'U'
      iprm(1) = 4
      iprm(2) = 3
      iprm(3) = 2
      iprm(4) = 1
      iprm(5) = 0

      ifail = 0
      Call f11mdf(spec,n,icolzp,irowix,iprm,ifail)

!     Output results

      Write (nout,*)
      Write (nout,*) 'User Permutation'
      Write (nout,'(10I6)') iprm(1:n)

!     Calculate natural permutation

      spec = 'N'

      ifail = 0
      Call f11mdf(spec,n,icolzp,irowix,iprm,ifail)

!     Output results

      Write (nout,*)
      Write (nout,*) 'Natural Permutation'
      Write (nout,'(10I6)') iprm(1:n)

    End Program f11mdfe
```

10.2 Program Data

```
F11MDF Example Program Data
  5   N
 1
 3
 5
 7
 9
 12   ICOLZP(I) I=1,..,N+1
 1
 3
 1
 5
 2
 3
 2
 4
 3
 4
 5    IROWIX(I) I=1,...,NNZ
```

10.3 Program Results

```
 F11MDF Example Program Results

 COLAMD Permutation
     1     0     4     3     2

 User Permutation
     4     3     2     1     0

 Natural Permutation
     0     1     2     3     4
```