NAG Toolbox for MATLAB

f11me

1 Purpose

f11me computes the LU factorization of a real sparse matrix in compressed column (Harwell–Boeing), column-permuted format.

2 Syntax

```
[iprm, nzlumx, il, lval, iu, uval, nnzl, nnzu, flop, ifail] = f11me(n,
irowix, a, iprm, thresh, nzlmx, nzlumx, nzumx)
```

3 Description

Given a real sparse matrix A, f11me computes an LU factorization of A with partial pivoting, $P_rAP_c = LU$, where P_r is a row permutation matrix (computed by f11me), P_c is a (supplied) column permutation matrix, L is unit lower triangular and U is upper triangular. The column permutation matrix, P_c, must be computed by a prior call to f11md. The matrix A must be presented in the column permuted, compressed column (Harwell–Boeing) format.

The LU factorization is output in the form of four one-dimensional arrays: integer arrays **il** and **iu** and real-valued arrays **lval** and **uval**. These describe the sparsity pattern and numerical values in the L and U matrices. The minimum required dimensions of these arrays cannot be given as a simple function of the size parameters (order and number of nonzero values) of the matrix A. This is due to unpredictable fill-in created by partial pivoting. f11me will, on return, indicate which dimensions of these arrays were not adequate for the computation or (in the case of one of them) give a firm bound. You should then allocate more storage and try again.

4 References

Demmel J W, Eisenstat S C, Gilbert J R, Li X S and Li J W H 1999 A Supernodal Approach to Sparse Partial Pivoting *SIAM J. Matrix Anal. Appl.* **20** 720–755

Demmel J W, Gilbert J R and Li X S 1999 An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination *SIAM J. Matrix Anal. Appl.* **20** 915–952

5 Parameters

5.1 Compulsory Input Parameters

1: **n – int32 scalar**

n, the order of the matrix A.

Constraint: **n** ≥ 0.

2: **irowix**$(*)$ **– int32 array**

Note: the dimension of the array **irowix** must be at least nnz, the number of nonzeros of the sparse matrix A.

The row index array of sparse matrix A.

3: **a**$(*)$ **– double array**

Note: the dimension of the array **a** must be at least nnz, the number of nonzeros of the sparse matrix A.

The array of nonzero values in the sparse matrix A.

4: **iprm**($\mathbf{7} \times \mathbf{n}$) – **int32 array**

Contains the column permutation which defines the permutation P_c and associated data structures as computed by function f11md.

5: **thresh – double scalar**

The diagonal pivoting threshold, t. At step j of the Gaussian elimination, if $|A_{jj}| \geq t\left(\max_{i \geq j}|A_{ij}|\right)$, use A_{jj} as a pivot, otherwise use $\max_{i \geq j}|A_{ij}|$. A value of $t = 1$ corresponds to partial pivoting, a value of $t = 0$ corresponds to always choosing the pivot on the diagonal (unless it is zero).

Constraint: $0 \leq$ **thresh** ≤ 1.

Suggested value: **thresh** $= 1.0$. Smaller values may result in a faster factorization, but the benefits are likely to be small in most cases. It might be possible to use **thresh** $= 0.0$ if you are confident about the stability of the factorization, for example, if A is diagonally dominant.

6: **nzlmx – int32 scalar**

Indicates the available size of array **il**. The dimension of **il** should be at least $7 \times$ **n** $+$ **nzlmx** $+ 4$. A good range for **nzlmx** that works for many problems is nnz to $8 \times nnz$, where nnz is the number of nonzeros in the sparse matrix A. If, on exit, **ifail** $= 2$, the given **nzlmx** was too small and you should attempt to provide more storage and call the function again.

Constraint: **nzlmx** ≥ 1.

7: **nzlumx – int32 scalar**

Indicates the available size of array **lval**. The dimension of **lval** should be at least **nzlumx**.

Constraint: **nzlumx** ≥ 1.

8: **nzumx – int32 scalar**

Indicates the available sizes of arrays **iu** and **uval**. The dimension of **iu** should be at least $2 \times$ **n** $+$ **nzumx** $+ 1$ and the dimension of **uval** should be at least **nzumx**. A good range for **nzumx** that works for many problems is nnz to $8 \times nnz$, where nnz is the number of nonzeros in the sparse matrix A. If, on exit, **ifail** $= 3$, the given **nzumx** was too small and you should attempt to provide more storage and call the function again.

Constraint: **nzumx** ≥ 1.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **iprm**($\mathbf{7} \times \mathbf{n}$) – **int32 array**

Part of the array is modified to record the row permutation P_r determined by pivoting.

2: **nzlumx – int32 scalar**

If **ifail** $= 4$, the given **nzlumx** was too small and is reset to a value that will be sufficient. You should then provide the indicated storage and call the function again.

3: **il($7 \times$ n + nzlmx + 4) – int32 array**

Encapsulates the sparsity pattern of matrix L.

4: **lval($*$) – double array**

Note: the dimension of the array **lval** must be at least **nzlumx**.

Records the nonzero values of matrix L and some of the nonzero values of matrix U.

5: **iu($2 \times$ n + nzumx + 1) – int32 array**

Encapsulates the sparsity pattern of matrix U.

6: **uval(nzumx) – double array**

Records some of the nonzero values of matrix U.

7: **nnzl – int32 scalar**

The number of nonzero values in the matrix L.

8: **nnzu – int32 scalar**

The number of nonzero values in the matrix U.

9: **flop – double scalar**

The number of floating-point operations performed.

10: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **n** < 0,
or **nzlmx** < 1,
or **nzlumx** < 1,
or **nzumx** < 1,
or **thresh** < 0.0,
or **thresh** > 1.0.

ifail = 2

nzlmx was not large enough. You should repeat the call with a larger value of **nzlmx**, providing more storage for the output array **il**.

ifail = 3

nzumx was not large enough. You should repeat the call with a larger value of **nzumx**, providing more storage for the output arrays **iu** and **uval**.

ifail = 4

nzlumx was not large enough. You should repeat the call with the value of **nzlumx** returned on exit, providing more storage for the output array **lval**.

ifail $= 5$

The matrix A is singular and no factorization will be attempted.

ifail $= 301$

Unable to allocate required internal workspace.

7 Accuracy

The computed factors L and U are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\epsilon|L||U|,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***, when partial pivoting is used. If no partial pivoting is used, the factorization accuracy can be considerably worse. A call to f11mm after f11me can help determine the quality of the factorization.

8 Further Comments

The total number of floating-point operations depends on the sparsity pattern of the matrix A.

A call to f11me may be followed by calls to the functions:

f11mf to solve $AX = B$ or $A^{\mathrm{T}}X = B$;

f11mg to estimate the condition number of A;

f11mm to estimate the reciprocal pivot growth of the LU factorization.

9 Example

```
n = int32(5);
irowix = [int32(1);
     int32(3);
     int32(1);
     int32(5);
     int32(2);
     int32(3);
     int32(2);
     int32(4);
     int32(3);
     int32(4);
     int32(5)];
a = [2;
     4;
     1;
     -2;
     1;
     1;
     -1;
     1;
     1;
     2;
     3];
iprm = [int32(1);
     int32(0);
     int32(4);
     int32(3);
     int32(2);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
```

```
      int32(2);
      int32(0);
      int32(8);
      int32(6);
      int32(4);
      int32(4);
      int32(2);
      int32(11);
      int32(8);
      int32(6);
      int32(1);
      int32(2);
      int32(3);
      int32(4);
      int32(5);
      int32(2);
      int32(2);
      int32(2);
      int32(2);
      int32(1);
      int32(1);
      int32(1);
      int32(1);
      int32(2);
      int32(0)];
thresh = 1;
nzlmx = int32(88);
nzlumx = int32(88);
nzumx = int32(88);
[iprmOut, nzlumxOut, il, lval, iu, uval, nnzl, nnzu, flop, ifail] = ...
    f11me(n, irowix, a, iprm, thresh, nzlmx, nzlumx, nzumx)
```

```
iprmOut =
            1
            0
            4
            3
            2
            4
            3
            1
            2
            0
            2
            0
            8
            6
            4
            4
            2
           11
            8
            6
            1
            2
            3
            4
            5
            2
            2
            2
            2
            1
            1
            1
            1
            2
            0
nzlumxOut =
           88
```

```
il =
     array elided
lval =
     array elided
iu =
     array elided
uval =
     array elided
nnzl =
           9
nnzu =
          10
flop =
    19
ifail =
           0
```