NAG Toolbox for MATLAB

f11mf

1 Purpose

f11mf solves a real sparse system of linear equations with multiple right-hand sides given an LU factorization of the sparse matrix computed by f11me.

2 Syntax

```
[b, ifail] = f11mf(trans, iprm, il, lval, iu, uval, b, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

f11mf solves a real system of linear equations with multiple right-hand sides $AX = B$ or $A^{\mathrm{T}}X = B$, according to the value of the parameter **trans**, where the matrix factorization $P_rAP_c = LU$ corresponds to an LU decomposition of a sparse matrix stored in compressed column (Harwell–Boeing) format, as computed by f11me.

In the above decomposition L is a lower triangular sparse matrix with unit diagonal elements and U is an upper triangular sparse matrix; P_r and P_c are permutation matrices.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **trans – string**

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

The column permutation which defines P_c, the row permutation which defines P_r, plus associated data structures as computed by f11me.

3: **il**$(*)$ **– int32 array**

Note: the dimension of the array **il** must be at least as large as the dimension of the array of the same name in f11me.

Records the sparsity pattern of matrix L as computed by f11me.

4: **lval**$(*)$ – **double array**

Note: the dimension of the array **lval** must be at least as large as the dimension of the array of the same name in f11me.

Records the nonzero values of matrix L and some nonzero values of matrix U as computed by f11me.

5: **iu**$(*)$ – **int32 array**

Note: the dimension of the array **iu** must be at least as large as the dimension of the array of the same name in f11me.

Records the sparsity pattern of matrix U as computed by f11me.

6: **uval**$(*)$ – **double array**

Note: the dimension of the array **uval** must be at least as large as the dimension of the array of the same name in f11me.

Records some nonzero values of matrix U as computed by f11me.

7: **b**(**ldb**,$*$) – **double array**

The first dimension of the array **b** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{nrhs\_p})$

The **n** by **nrhs_p** right-hand side matrix B.

5.2 Optional Input Parameters

1: **n** – **int32 scalar**

n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

2: **nrhs_p** – **int32 scalar**

Default: The second dimension of the array **b**.

$nrhs$, the number of right-hand sides in B.

Constraint: $\mathbf{nrhs\_p} \geq 0$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldb

5.4 Output Parameters

1: **b**(**ldb**,$*$) – **double array**

The first dimension of the array **b** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{nrhs\_p})$

The **n** by **nrhs_p** solution matrix X.

2: **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, **trans** $\neq$ 'N' or 'T',
or **n** < 0,
or **nrhs_p** < 0,
or **ldb** $< \max(1, \mathbf{n})$.

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

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A + E)x = b$, 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 $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\,\mathrm{cond}(A, x)\epsilon$$

where $\mathrm{cond}(A, x) = \left\| |A^{-1}||A||x| \right\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A, x)$ can be much smaller than $\mathrm{cond}(A)$, and $\mathrm{cond}\left(A^{\mathrm{T}}\right)$ can be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling f11mh, and an estimate for $\kappa_\infty(A)$ can be obtained by calling f11mg.

8 Further Comments

f11mf may be followed by a call to f11mh to refine the solution and return an error estimate.

9 Example

```
trans = 'N';
iprm = [int32(1);
     int32(0);
     int32(4);
     int32(3);
     int32(2);
     int32(4);
     int32(3);
     int32(1);
     int32(2);
     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)];
il = [int32(0);
     int32(1);
     int32(2);
     int32(3);
     int32(-1);
     int32(-1);
     int32(1);
     int32(2);
     int32(3);
     int32(5);
     int32(-1);
     int32(0);
     int32(1);
     int32(2);
     int32(3);
     int32(3);
     int32(3);
     int32(0);
     int32(2);
     int32(4);
     int32(6);
     int32(14);
     int32(8);
     int32(2);
     int32(4);
     int32(6);
     int32(8);
     int32(15);
     int32(0);
     int32(2);
     int32(4);
     int32(6);
     int32(8);
     int32(-1);
     int32(2);
     int32(4);
     int32(6);
     int32(8);
     int32(10);
     int32(0);
     int32(4);
     int32(1);
     int32(4);
     int32(2);
     int32(4);
```

```
      int32(3);
      int32(4);
      int32(3);
      int32(0);
      int32(3);
      int32(0);
      int32(1);
      int32(0);
      int32(0);
      int32(0);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
```

```
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1)];
lval = [-2;
      -0.5;
      4;
      0.5;
      2;
      0.5;
      -1;
      0.5;
      1;
      -1;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
```

```
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0];
iu = [int32(0);
      int32(0);
      int32(0);
      int32(2);
      int32(3);
      int32(-1);
      int32(0);
      int32(0);
      int32(2);
      int32(3);
      int32(4);
      int32(1);
      int32(0);
      int32(2);
      int32(1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
```

```
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1);
      int32(-1)];
uval = [1;
      3;
      1;
      1;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
```

```
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
      0;
```

```
     0;
     0;
     0;
     0;
     0;
     0;
     0];
b = [1.56, 3.12;
     -0.25, -0.5;
     3.6, 7.2;
     1.33, 2.66;
     0.52, 1.04];
[bOut, ifail] = f11mf(trans, iprm, il, lval, iu, uval, b)
```

```
bOut =
    0.7000    1.4000
    0.1600    0.3200
    0.5200    1.0400
    0.7700    1.5400
    0.2800    0.5600
ifail =
           0
```