NAG Toolbox

nag_sparse_direct_real_gen_solve (f11mf)

1 Purpose

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

2 Syntax

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

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

3 Description

nag_sparse_direct_real_gen_solve (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 argument **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 nag_sparse_direct_real_gen_lu (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** – CHARACTER(1)

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**($\mathbf{7} \times \mathbf{n}$) – INTEGER array

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

3: **il**(:) – INTEGER array

The dimension of the array **il** must be at least as large as the dimension of the array of the same name in nag_sparse_direct_real_gen_lu (f11me)

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

4: **lval**(:) – REAL (KIND=nag_wp) array

The dimension of the array **lval** must be at least as large as the dimension of the array of the same name in nag_sparse_direct_real_gen_lu (f11me)

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

5: **iu**(:) – INTEGER array

The dimension of the array **iu** must be at least as large as the dimension of the array of the same name in nag_sparse_direct_real_gen_lu (f11me)

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

6: **uval**(:) – REAL (KIND=nag_wp) array

The dimension of the array **uval** must be at least as large as the dimension of the array of the same name in nag_sparse_direct_real_gen_lu (f11me)

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

7: **b**(ldb, :) – REAL (KIND=nag_wp) array

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

The second dimension of the array **b** 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** – INTEGER

Default: the first dimension of the array **b**.

n, the order of the matrix A.

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

2: **nrhs_p** – INTEGER

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 Output Parameters

1: **b**(ldb, :) – REAL (KIND=nag_wp) array

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

The second dimension of the array **b** will be $\max(1, \mathbf{nrhs\_p})$.

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

2: **ifail** – INTEGER

$\mathbf{ifail} = 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: *ldb* $\geq \max(1, \mathbf{n})$.

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

Constraint: **nrhs_p** ≥ 0.

On entry, **trans** $= \langle value \rangle$.
Constraint: **trans** = 'N' or 'T'.

ifail $= 2$

Incorrect row permutations in array **iprm**.

ifail $= 3$

Incorrect column permutations in array **iprm**.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

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

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

8 Further Comments

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

9 Example

This example solves the system of equations $AX = B$, 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 nag_sparse_direct_real_gen_lu (f11me).

9.1 Program Text

```
function f11mf_example

fprintf('f11mf example results\n\n');

% A and b
n = nag_int(5);
nz = nag_int(11);
icolzp = [nag_int(1); 3; 5; 7; 9; 12];
irowix = [nag_int(1); 3; 1; 5; 2; 3; 2; 4; 3; 4; 5];
a = [2; 4; 1; -2; 1; 1; -1; 1; 1; 2; 3];
b = [  1.56,  3.12;
      -0.25, -0.50;
       3.60,  7.20;
       1.33,  2.66;
       0.52,  1.04];

% Calculate COLAMD permutation
spec = 'M';
iprm = zeros(1, 7*n, nag_int_name);

[iprm, ifail] = f11md( ...
                       spec, n, icolzp, irowix, iprm);

% Factorise
thresh = 1;
nzlmx = nag_int(8*nz);
nzlumx = nag_int(8*nz);
nzumx = nag_int(8*nz);

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

% Solve
trans = 'N';
[x, ifail] = f11mf( ...
                    trans, iprm, il, lval, iu, uval, b);
disp('Solutions');
disp(x);
```

9.2 Program Results

```
f11mf example results

Solutions
    0.7000    1.4000
    0.1600    0.3200
    0.5200    1.0400
    0.7700    1.5400
    0.2800    0.5600
```