NAG Toolbox

nag_sparse_direct_real_gen_refine (f11mh)

1 Purpose

nag_sparse_direct_real_gen_refine (f11mh) 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 Syntax

```
[x, ferr, berr, ifail] = nag_sparse_direct_real_gen_refine(trans, icolzp,
irowix, a, iprm, il, lval, iu, uval, b, x, 'n', n, 'nrhs_p', nrhs_p)
[x, ferr, berr, ifail] = f11mh(trans, icolzp, irowix, a, iprm, il, lval, iu,
uval, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_sparse_direct_real_gen_refine (f11mh) 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 function handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of nag_sparse_direct_real_gen_refine (f11mh) in terms of a single right-hand side b and solution x.

Given a computed solution x, the function 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:

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

Then the function 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 function uses the LU factorization $P_r A P_c = LU$ computed by nag_sparse_direct_real_gen_lu (f11me) and the solution computed by nag_sparse_direct_real_gen_solve (f11mf).

4 References

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

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: **icolzp**(:) – INTEGER array

The dimension of the array **icolzp** must be at least $\mathbf{n} + 1$

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

3: **irowix**(:) – INTEGER array

The dimension of the array **irowix** must be at least **icolzp**$(\mathbf{n} + 1) - 1$, the number of nonzeros of the sparse matrix A

The row index array of sparse matrix A.

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

The dimension of the array **a** must be at least **icolzp**$(\mathbf{n} + 1) - 1$, the number of nonzeros of the sparse matrix A

The array of nonzero values in the sparse matrix A.

5: **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).

6: **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).

7: **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).

8: **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).

9: **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).

10: **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 $nrhs$ right-hand side matrix B.

11: **x**$(ldx,:)$ – REAL (KIND=nag_wp) array

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

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

The n by $nrhs$ solution matrix X, as returned by nag_sparse_direct_real_gen_solve (f11mf).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **b**, **x**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix A.

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

2: **nrhs_p** – INTEGER

Default: the second dimension of the arrays **b**, **x**.

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

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

5.3 Output Parameters

1: **x**$(ldx,:)$ – REAL (KIND=nag_wp) array

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

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

The n by $nrhs$ improved solution matrix X.

2: **ferr**$(\mathbf{nrhs\_p})$ – REAL (KIND=nag_wp) array

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

3: **berr**$(\mathbf{nrhs\_p})$ – REAL (KIND=nag_wp) array

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

4: **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:

$\mathbf{ifail} = 1$

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

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

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

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

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

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 nag_sparse_direct_real_gen_lu (f11me).

9.1 Program Text

```
function f11mh_example

fprintf('f11mh example results\n\n');

% Solve AX=B for sparse A

% 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);

% Compute solution in x
x     = b;
trans = 'N';

[x, ifail] = f11mf( ...
                    trans, iprm, il, lval, iu, uval, x);

% Improve solution, and compute backward errors and estimated
% bounds on the forward errors
[x, ferr, berr, ifail] = ...
  f11mh( ...
         trans, icolzp, irowix, a, iprm, il, lval, iu, uval, b, x);

fprintf('Solutions:\n');
disp(x);
fprintf('Estmated Forward Error:\n');
fprintf('%8.1e\n', ferr);
fprintf('\nEstmated Backward Error:\n');
fprintf('%8.1e\n', berr);
```

9.2 Program Results

```
     f11mh example results

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

Estmated Forward Error:
 5.0e-15
 5.0e-15

Estmated Backward Error:
 3.6e-17
 3.6e-17
```