NAG Toolbox

nag_lapack_dgerfs (f07ah)

1 Purpose

nag_lapack_dgerfs (f07ah) returns error bounds for the solution of a real 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 order to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = nag_lapack_dgerfs(trans, a, af, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
[x, ferr, berr, info] = f07ah(trans, a, af, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_dgerfs (f07ah) 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_lapack_dgerfs (f07ah) 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 x is the exact solution of a perturbed system

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

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.

For details of the method, see the F07 Chapter Introduction.

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)

Indicates the form of the linear equations for which X is the computed solution.

trans = 'N'
The linear equations are of the form $AX = B$.

trans = 'T' or 'C'
The linear equations are of the form $A^{\mathrm{T}}X = B$.

Constraint: **trans** = 'N', 'T' or 'C'.

2: **a**$(lda, :)$ – REAL (KIND=nag_wp) array

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

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

The n by n original matrix A as supplied to nag_lapack_dgetrf (f07ad).

3: **af**$(ldaf, :)$ – REAL (KIND=nag_wp) array

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

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

The LU factorization of A, as returned by nag_lapack_dgetrf (f07ad).

4: **ipiv**$(:)$ – INTEGER array

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

The pivot indices, as returned by nag_lapack_dgetrf (f07ad).

5: **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 r right-hand side matrix B.

6: **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 r solution matrix X, as returned by nag_lapack_dgetrs (f07ae).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **a**, **af**, **b**, **x** and the second dimension of the arrays **a**, **af**, **ipiv**.

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**. (An error is raised if these dimensions are not equal.)

r, the number of right-hand sides.

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 improved solution matrix X.

2: **ferr**(**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 $j = 1, 2, \ldots, r$.

3: **berr**(**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 $j = 1, 2, \ldots, r$.

4: **info** – INTEGER

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

6 Error Indicators and Warnings

info < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

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

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. 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 $A^{\mathrm{T}}x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogue of this function is nag_lapack_zgerfs (f07av).

9 Example

This example solves the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 9.52 & 18.47 \\ 24.35 & 2.25 \\ 0.77 & -13.28 \\ -6.22 & -6.21 \end{pmatrix}.$$

Here A is nonsymmetric and must first be factorized by nag_lapack_dgetrf (f07ad).

9.1 Program Text

```
     function f07ah_example

fprintf('f07ah example results\n\n');

a = [ 1.80,  2.88,  2.05, -0.89;
      5.25, -2.95, -0.95, -3.80;
      1.58, -2.69, -2.90, -1.04;
     -1.11, -0.66, -0.59,  0.80];
b = [ 9.52, 18.47;
     24.35,  2.25;
      0.77,-13.28;
     -6.22, -6.21];

% Factorize a
[af, ipiv, info] = f07ad(a);

% Compute solution x
trans = 'N';
[x, ferr, berr, info] = f07ah( ...
                              trans, a, af, ipiv, b, b);

[ifail] = x04ca( ...
                'General', ' ', x, 'Solution(s)');

fprintf('\nBackward errors (machine-dependent)\n   ')
fprintf('%11.1e', berr);
fprintf('\nEstimated forward error bounds (machine-dependent)\n   ')
fprintf('%11.1e', ferr);
fprintf('\n');
```

9.2 Program Results

```
     f07ah example results

 Solution(s)
             1          2
 1      1.0000     3.0000
 2     -1.0000     2.0000
 3      3.0000     4.0000
 4     -5.0000     1.0000

Backward errors (machine-dependent)
       6.5e-17    8.9e-17
Estimated forward error bounds (machine-dependent)
       2.5e-14    3.5e-14
```