# NAG Toolbox

# nag_lapack_dporfs (f07fh)

## 1 Purpose

nag_lapack_dporfs (f07fh) returns error bounds for the solution of a real symmetric positive definite system of linear equations with multiple right-hand sides, $AX = B$. 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_dporfs(uplo, a, af, b, x, 'n', n, 'nrhs_p', nrhs_p)
[x, ferr, berr, info] = f07fh(uplo, a, af, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

## 3 Description

nag_lapack_dporfs (f07fh) returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric positive definite system of linear equations with multiple right-hand sides $AX = B$. 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_dporfs (f07fh) 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* $\beta$. 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

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

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.

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: **uplo** – CHARACTER(1)

Specifies whether the upper or lower triangular part of $A$ is stored and how $A$ is to be factorized.

**uplo** = 'U'

The upper triangular part of $A$ is stored and $A$ is factorized as $U^{\mathrm{T}}U$, where $U$ is upper triangular.

**uplo** = 'L'
The lower triangular part of $A$ is stored and $A$ is factorized as $LL^{\mathrm{T}}$, where $L$ is lower triangular.

*Constraint*: **uplo** = 'U' or 'L'.

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 symmetric positive definite matrix $A$ as supplied to nag_lapack_dpotrf (f07fd).

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 Cholesky factor of $A$, as returned by nag_lapack_dpotrf (f07fd).

4: **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 $B$.

5: **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_dpotrs (f07fe).

## 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**.

$n$, 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 $j$th solution vector, that is, the $j$th 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 $\beta$ for the $j$th solution vector, that is, the $j$th 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 $Ax = b$; 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_zporfs (f07fv).

## 9 Example

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

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here $A$ is symmetric positive definite and must first be factorized by nag_lapack_dpotrf (f07fd).

### 9.1 Program Text

```
function f07fh_example

fprintf('f07fh example results\n\n');

% Lower triangular part of symmetric matrix A
uplo = 'Lower';
a = [ 4.16,  0,    0,    0;
     -3.12,  5.03, 0,    0;
      0.56, -0.83, 0.76, 0;
     -0.10,  1.18, 0.34, 1.18];
```

```
[L, info] = f07fd( ...
                   uplo, a);

% Rhs
b = [  8.70, 8.30;
     -13.35, 2.13;
       1.89, 1.61;
      -4.14, 5.00];

% Solve
[x, info]  = f07fe( ...
                    uplo, L, b);

% Refine
[x, ferr, berr, info] = f07fh( ...
                               uplo, a, L, b, x);

[ifail] = x04ca( ...
                 'General', 'N', 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

```
     f07fh example results

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

Backward errors (machine-dependent)
       1.0e-16    5.0e-17
Estimated forward error bounds (machine-dependent)
       2.3e-14    2.3e-14
```