

NAG Library Routine Document

F07FHF (DPORFS)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details.

1 Purpose

F07FHF (DPORFS) 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 Specification

```
SUBROUTINE F07FHF (UPLO, N, NRHS, A, LDA, AF, LDAF, B, LDB, X, LDX, FERR,      &
                  BERR, WORK, IWORK, INFO)
INTEGER          N, NRHS, LDA, LDAF, LDB, LDX, IWORK(N), INFO
REAL (KIND=nag_wp) A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), FERR(NRHS),  &
                  BERR(NRHS), WORK(3*N)
CHARACTER(1)    UPLO
```

The routine may be called by its LAPACK name *dporf*s.

3 Description

F07FHF (DPORFS) 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 routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of F07FHF (DPORFS) in terms of a single right-hand side b and solution x .

Given a computed solution x , the routine 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

$$| \delta a_{ij} | \leq \beta | a_{ij} | \quad \text{and} \quad | \delta b_i | \leq \beta | b_i |.$$

Then the routine 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

- 1: UPLO – CHARACTER(1) *Input*
On entry: 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 $U^T U$, where U is upper triangular.
 UPLO = 'L'
 The lower triangular part of A is stored and A is factorized as LL^T , where L is lower triangular.
Constraint: UPLO = 'U' or 'L'.
- 2: N – INTEGER *Input*
On entry: n , the order of the matrix A .
Constraint: $N \geq 0$.
- 3: NRHS – INTEGER *Input*
On entry: r , the number of right-hand sides.
Constraint: NRHS ≥ 0 .
- 4: A(LDA,*) – REAL (KIND=nag_wp) array *Input*
Note: the second dimension of the array A must be at least $\max(1, N)$.
On entry: the n by n original symmetric positive definite matrix A as supplied to F07FDF (DPOTRF).
- 5: LDA – INTEGER *Input*
On entry: the first dimension of the array A as declared in the (sub)program from which F07FHF (DPORFS) is called.
Constraint: LDA $\geq \max(1, N)$.
- 6: AF(LDAF,*) – REAL (KIND=nag_wp) array *Input*
Note: the second dimension of the array AF must be at least $\max(1, N)$.
On entry: the Cholesky factor of A , as returned by F07FDF (DPOTRF).
- 7: LDAF – INTEGER *Input*
On entry: the first dimension of the array AF as declared in the (sub)program from which F07FHF (DPORFS) is called.
Constraint: LDAF $\geq \max(1, N)$.
- 8: B(LDB,*) – REAL (KIND=nag_wp) array *Input*
Note: the second dimension of the array B must be at least $\max(1, NRHS)$.
On entry: the n by r right-hand side matrix B .
- 9: LDB – INTEGER *Input*
On entry: the first dimension of the array B as declared in the (sub)program from which F07FHF (DPORFS) is called.
Constraint: LDB $\geq \max(1, N)$.

- 10: X(LDX,*) – REAL (KIND=nag_wp) array Input/Output
Note: the second dimension of the array X must be at least $\max(1, \text{NRHS})$.
On entry: the n by r solution matrix X , as returned by F07FEF (DPOTRS).
On exit: the improved solution matrix X .
- 11: LDX – INTEGER Input
On entry: the first dimension of the array X as declared in the (sub)program from which F07FHF (DPORFS) is called.
Constraint: $\text{LDX} \geq \max(1, N)$.
- 12: FERR(NRHS) – REAL (KIND=nag_wp) array Output
On exit: FERR(j) contains an estimated error bound for the j th solution vector, that is, the j th column of X , for $j = 1, 2, \dots, r$.
- 13: BERR(NRHS) – REAL (KIND=nag_wp) array Output
On exit: BERR(j) contains the component-wise backward error bound β for the j th solution vector, that is, the j th column of X , for $j = 1, 2, \dots, r$.
- 14: WORK($3 \times N$) – REAL (KIND=nag_wp) array Workspace
- 15: IWORK(N) – INTEGER array Workspace
- 16: INFO – INTEGER Output
On exit: INFO = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

INFO < 0

If INFO = $-i$, the i th parameter 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 routine is F07FVF (ZPORFS).

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 F07FDF (DPOTRF).

9.1 Program Text

```

Program f07fhfe

!      F07FHF Example Program Text

!      Mark 24 Release. NAG Copyright 2012.

!      .. Use Statements ..
Use nag_library, Only: dporfs, dpotrf, dpotrs, f06qff, nag_wp, x04caf
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
Integer                     :: i, ifail, info, lda, ldaf, ldb, ldx, &
                             n, nrhs
Character (1)              :: uplo
!      .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: a(:,,:), af(:,,:), b(:,,:), berr(:), &
                             ferr(:), work(:), x(:,,:)
Integer, Allocatable       :: iwork(:)
!      .. Executable Statements ..
Write (nout,*) 'F07FHF Example Program Results'
!      Skip heading in data file
Read (nin,*)
Read (nin,*) n, nrhs
lda = n
ldaf = n
ldb = n
ldx = n
Allocate (a(lda,n),af(ldaf,n),b(ldb,nrhs),berr(nrhs),ferr(nrhs), &
         work(3*n),x(ldx,n),iwork(n))

!      Read A and B from data file, and copy A to AF and B to X

Read (nin,*) uplo
If (uplo=='U') Then
  Read (nin,*)(a(i,i:n),i=1,n)
Else If (uplo=='L') Then
  Read (nin,*)(a(i,1:i),i=1,n)
End If
Read (nin,*)(b(i,1:nrhs),i=1,n)

Call f06qff(uplo,n,n,a,lda,af,ldaf)

x(1:n,1:nrhs) = b(1:n,1:nrhs)

!      Factorize A in the array AF
!      The NAG name equivalent of dpotrf is f07fdf
Call dpotrf(uplo,n,af,ldaf,info)

Write (nout,*)
Flush (nout)
If (info==0) Then

!      Compute solution in the array X
!      The NAG name equivalent of dpotrs is f07fef
Call dpotrs(uplo,n,nrhs,af,ldaf,x,ldx,info)

```

```

!      Improve solution, and compute backward errors and
!      estimated bounds on the forward errors

!      The NAG name equivalent of dporfs is f07fhf
!      Call dporfs(uplo,n,nrhs,a,lda,af,ldaf,b,ldb,x,ldx,ferr,berr,work, &
!      iwork,info)

!      Print solution

!      ifail: behaviour on error exit
!      =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
!      ifail = 0
!      Call x04caf('General',' ',n,nrhs,x,ldx,'Solution(s)',ifail)

!      Write (nout,*)
!      Write (nout,*) 'Backward errors (machine-dependent)'
!      Write (nout,99999) berr(1:nrhs)
!      Write (nout,*) 'Estimated forward error bounds (machine-dependent)'
!      Write (nout,99999) ferr(1:nrhs)
!      Else
!      Write (nout,*) 'A is not positive definite'
!      End If

99999 Format ((3X,1P,7E11.1))
End Program f07fhfe

```

9.2 Program Data

F07FHF Example Program Data

```

4 2 :Values of N and NRHS
'L' :Value of UPLO
4.16
-3.12 5.03
0.56 -0.83 0.76
-0.10 1.18 0.34 1.18 :End of matrix A
8.70 8.30
-13.35 2.13
1.89 1.61
-4.14 5.00 :End of matrix B

```

9.3 Program Results

F07FHF Example Program 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)

7.6E-17	5.0E-17
---------	---------

Estimated forward error bounds (machine-dependent)

2.4E-14	2.3E-14
---------	---------