# NAG Toolbox for MATLAB

# f07qv

## 1 Purpose

f07qv returns error bounds for the solution of a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

## 2 Syntax

```
[x, ferr, berr, info] = f07qv(uplo, ap, afp, ipiv, b, x, 'n', n,
'nrhs_p', nrhs_p)
```

## 3 Description

f07qv returns the backward errors and estimated bounds on the forward errors for the solution of a complex symmetric system of linear equations with multiple right-hand sides $AX = B$, using packed storage. The function handles each right-hand side vector (stored as a column of the matrix $B$) independently, so we describe the function of f07qv 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

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

Then the function estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i\left|x_i - \hat{x}_i\right| / \max_i\left|x_i\right|$$

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 – string**

Indicates 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 $PUDU^{\mathrm{T}}P^{\mathrm{T}}$, where $U$ is upper triangular.

**uplo** = 'L'

The lower triangular part of $A$ is stored and $A$ is factorized as $PLDL^{\mathrm{T}}P^{\mathrm{T}}$, where $L$ is lower triangular.

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

2: **ap**($*$) – **complex array**

**Note**: the dimension of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

The $n$ by $n$ original symmetric matrix $A$ as supplied to f07qr.

3: **afp**($*$) – **complex array**

**Note**: the dimension of the array **afp** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

The factorization of $A$ stored in packed form, as returned by f07qr.

4: **ipiv**($*$) – **int32 array**

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

Details of the interchanges and the block structure of $D$, as returned by f07qr.

5: **b**(**ldb**,$*$) – **complex array**

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

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

The $n$ by $r$ right-hand side matrix $B$.

6: **x**(**ldx**,$*$) – **complex array**

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

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

The $n$ by $r$ solution matrix $X$, as returned by f07qs.

## 5.2 Optional Input Parameters

1: **n – int32 scalar**

*Default*: The first dimension of the array **ap** and the second dimension of the array **ap**. (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 – int32 scalar**

*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 Input Parameters Omitted from the MATLAB Interface

ldb, ldx, work, rwork

## 5.4 Output Parameters

1: **x**(**ldx**,$*$) – **complex array**

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

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

The improved solution matrix $X$.

2: **ferr**($*$) – **double array**

**Note**: the dimension of the array **ferr** must be at least $\max(1, \textbf{nrhs\_p})$.

**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**($*$) – **double array**

**Note**: the dimension of the array **berr** must be at least $\max(1, \textbf{nrhs\_p})$.

**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 – int32 scalar**

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

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**info** $= -i$

If **info** $= -i$, parameter $i$ had an illegal value on entry. The parameters are numbered as follows:

1: **uplo**, 2: **n**, 3: **nrhs_p**, 4: **ap**, 5: **afp**, 6: **ipiv**, 7: **b**, 8: **ldb**, 9: **x**, 10: **ldx**, 11: **ferr**, 12: **berr**, 13: **work**, 14: **rwork**, 15: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

## 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 $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real 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 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this function is f07ph.

## 9 Example

```
uplo = 'L';
ap = [complex(-0.39, -0.71);
     complex(5.14, -0.64);
     complex(-7.86, -2.96);
     complex(3.8, +0.92);
     complex(8.859999999999999, +1.81);
     complex(-3.52, +0.58);
     complex(5.32, -1.59);
     complex(-2.83, -0.03);
     complex(-1.54, -2.86);
```

```
     complex(-0.5600000000000001, +0.12)];
afp = [complex(-0.39, -0.71);
     complex(-7.86, -2.96);
     complex(0.5278724801640799, -0.3714660014825906);
     complex(0.442558238872675, +0.1936483698297402);
     complex(-2.83, -0.03);
     complex(-0.6078391056683192, +0.281079647893122);
     complex(-0.4822822975185383, +0.01498936219105284);
     complex(4.407906236731014, +5.399120676796941);
     complex(-0.1070821880092683, -0.3156780862488454);
     complex(-2.095414887840057, -2.201139281440786)];
ipiv = [int32(-3);
     int32(-3);
     int32(3);
     int32(4)];
b = [complex(-55.64, +41.22), complex(-19.09, -35.97);
     complex(-48.18, +66), complex(-12.08, -27.02);
     complex(-0.49, -1.47), complex(6.95, +20.49);
     complex(-6.43, +19.24), complex(-4.59, -35.53)];
x    =    [complex(0.9999999999999996,    -1.000000000000002),    complex(-
1.999999999999999, -1.000000000000001);
                      complex(-2.000000000000001,    +5.000000000000002),
complex(0.9999999999999993, -3);
                       complex(3.000000000000002,    -1.999999999999997),
complex(2.999999999999999, +2.000000000000001);
            complex(-3.999999999999999,   +3.000000000000002),   complex(-
0.9999999999999991, +1.000000000000001)];
[xOut, ferr, berr, info] = f07qv(uplo, ap, afp, ipiv, b, x)
```

```
xOut =
   1.0000 - 1.0000i  -2.0000 - 1.0000i
  -2.0000 + 5.0000i   1.0000 - 3.0000i
   3.0000 - 2.0000i   3.0000 + 2.0000i
  -4.0000 + 3.0000i  -1.0000 + 1.0000i
ferr =
   1.0e-13 *
    0.1235
    0.1261
berr =
   1.0e-15 *
    0.1055
    0.0932
info =
           0
```