

NAG Library Routine Document

F07QPF (ZSPSVX)

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

F07QPF (ZSPSVX) uses the diagonal pivoting factorization

$$A = UDU^T \quad \text{or} \quad A = LDL^T$$

to compute the solution to a complex system of linear equations

$$AX = B,$$

where A is an n by n symmetric matrix stored in packed format and X and B are n by r matrices. Error bounds on the solution and a condition estimate are also provided.

2 Specification

```
SUBROUTINE F07QPF (FACT, UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX,      &
                  RCOND, FERR, BERR, WORK, RWORK, INFO)
```

```
INTEGER          N, NRHS, IPIV(N), LDB, LDX, INFO
REAL (KIND=nag_wp) RCOND, FERR(NRHS), BERR(NRHS), RWORK(N)
COMPLEX (KIND=nag_wp) AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(2*N)
CHARACTER(1)     FACT, UPLO
```

The routine may be called by its LAPACK name *zpsvx*.

3 Description

F07QPF (ZSPSVX) performs the following steps:

1. If FACT = 'N', the diagonal pivoting method is used to factor A as $A = UDU^T$ if UPLO = 'U' or $A = LDL^T$ if UPLO = 'L', where U (or L) is a product of permutation and unit upper (lower) triangular matrices and D is symmetric and block diagonal with 1 by 1 and 2 by 2 diagonal blocks.
2. If some $d_{ii} = 0$, so that D is exactly singular, then the routine returns with INFO = i . Otherwise, the factored form of A is used to estimate the condition number of the matrix A . If the reciprocal of the condition number is less than *machine precision*, INFO = $N + 1$ is returned as a warning, but the routine still goes on to solve for X and compute error bounds as described below.
3. The system of equations is solved for X using the factored form of A .
4. Iterative refinement is applied to improve the computed solution matrix and to calculate error bounds and backward error estimates for it.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia <http://www.netlib.org/lapack/lug>

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

Higham N J (2002) *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

5 Parameters

- 1: FACT – CHARACTER(1) *Input*
On entry: specifies whether or not the factorized form of the matrix A has been supplied.
 FACT = 'F'
 AFP and IPIV contain the factorized form of the matrix A . AFP and IPIV will not be modified.
 FACT = 'N'
 The matrix A will be copied to AFP and factorized.
Constraint: FACT = 'F' or 'N'.
- 2: UPLO – CHARACTER(1) *Input*
On entry: if UPLO = 'U', the upper triangle of A is stored.
 If UPLO = 'L', the lower triangle of A is stored.
Constraint: UPLO = 'U' or 'L'.
- 3: N – INTEGER *Input*
On entry: n , the number of linear equations, i.e., the order of the matrix A .
Constraint: $N \geq 0$.
- 4: NRHS – INTEGER *Input*
On entry: r , the number of right-hand sides, i.e., the number of columns of the matrix B .
Constraint: NRHS ≥ 0 .
- 5: AP(*) – COMPLEX (KIND=nag_wp) array *Input*
Note: the dimension of the array AP must be at least $\max(1, N \times (N + 1)/2)$.
On entry: the n by n symmetric matrix A , packed by columns.
 More precisely,
 if UPLO = 'U', the upper triangle of A must be stored with element A_{ij} in
 AP($i + j(j - 1)/2$) for $i \leq j$;
 if UPLO = 'L', the lower triangle of A must be stored with element A_{ij} in
 AP($i + (2n - j)(j - 1)/2$) for $i \geq j$.
- 6: AFP(*) – COMPLEX (KIND=nag_wp) array *Input/Output*
Note: the dimension of the array AFP must be at least $\max(1, N \times (N + 1)/2)$.
On entry: if FACT = 'F', AFP contains the block diagonal matrix D and the multipliers used to obtain the factor U or L from the factorization $A = UDU^T$ or $A = LDL^T$ as computed by F07QRF (ZSPTRF), stored as a packed triangular matrix in the same storage format as A .
On exit: if FACT = 'N', AFP contains the block diagonal matrix D and the multipliers used to obtain the factor U or L from the factorization $A = UDU^T$ or $A = LDL^T$ as computed by F07QRF (ZSPTRF), stored as a packed triangular matrix in the same storage format as A .

- 7: IPIV(N) – INTEGER array *Input/Output*
On entry: if FACT = 'F', IPIV contains details of the interchanges and the block structure of D , as determined by F07QRF (ZSPTRF).
 if $IPIV(i) = k > 0$, d_{ii} is a 1 by 1 pivot block and the i th row and column of A were interchanged with the k th row and column;
 if UPLO = 'U' and $IPIV(i - 1) = IPIV(i) = -l < 0$, $\begin{pmatrix} d_{i-1,i-1} & \bar{d}_{i,i-1} \\ \bar{d}_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i - 1)$ th row and column of A were interchanged with the l th row and column;
 if UPLO = 'L' and $IPIV(i) = IPIV(i + 1) = -m < 0$, $\begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i + 1)$ th row and column of A were interchanged with the m th row and column.
On exit: if FACT = 'N', IPIV contains details of the interchanges and the block structure of D , as determined by F07QRF (ZSPTRF), as described above.
- 8: B(LDB,*) – COMPLEX (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 F07QPF (ZSPSVX) is called.
Constraint: $LDB \geq \max(1, N)$.
- 10: X(LDX,*) – COMPLEX (KIND=nag_wp) array *Output*
Note: the second dimension of the array X must be at least $\max(1, NRHS)$.
On exit: if INFO = 0 or $N + 1$, the n by r solution matrix X .
- 11: LDX – INTEGER *Input*
On entry: the first dimension of the array X as declared in the (sub)program from which F07QPF (ZSPSVX) is called.
Constraint: $LDX \geq \max(1, N)$.
- 12: RCOND – REAL (KIND=nag_wp) *Output*
On exit: the estimate of the reciprocal condition number of the matrix A . If $RCOND = 0.0$, the matrix may be exactly singular. This condition is indicated by $INFO > 0$ and $INFO \leq N$. Otherwise, if RCOND is less than the *machine precision*, the matrix is singular to working precision. This condition is indicated by $INFO = N + 1$.
- 13: FERR(NRHS) – REAL (KIND=nag_wp) array *Output*
On exit: if INFO = 0 or $N + 1$, an estimate of the forward error bound for each computed solution vector, such that $\|\hat{x}_j - x_j\|_\infty / \|x_j\|_\infty \leq FERR(j)$ where \hat{x}_j is the j th column of the computed solution returned in the array X and x_j is the corresponding column of the exact solution X . The estimate is as reliable as the estimate for RCOND, and is almost always a slight overestimate of the true error.

- 14: BERR(NRHS) – REAL (KIND=nag_wp) array Output
On exit: if INFO = 0 or N + 1, an estimate of the component-wise relative backward error of each computed solution vector \hat{x}_j (i.e., the smallest relative change in any element of A or B that makes \hat{x}_j an exact solution).
- 15: WORK(2 × N) – COMPLEX (KIND=nag_wp) array Workspace
- 16: RWORK(N) – REAL (KIND=nag_wp) array Workspace
- 17: 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 argument had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0 and INFO ≤ N

If INFO ≤ N, $d(i, i)$ is exactly zero. The factorization has been completed, but the factor D is exactly singular, so the solution and error bounds could not be computed. RCOND = 0.0 is returned.

INFO = N + 1

D is nonsingular, but RCOND is less than *machine precision*, meaning that the matrix is singular to working precision. Nevertheless, the solution and error bounds are computed because there are a number of situations where the computed solution can be more accurate than the value of RCOND would suggest.

7 Accuracy

For each right-hand side vector b , the computed solution \hat{x} is the exact solution of a perturbed system of equations $(A + E)\hat{x} = b$, where

$$\|E\|_1 = O(\epsilon)\|A\|_1,$$

where ϵ is the *machine precision*. See Chapter 11 of Higham (2002) for further details.

If \hat{x} is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|\hat{x}\|_\infty} \leq w_c \text{cond}(A, \hat{x}, b)$$

where $\text{cond}(A, \hat{x}, b) = \frac{\| |A| |\hat{x}| + |b| \|_\infty}{\|\hat{x}\|_\infty} \leq \text{cond}(A) = \| |A^{-1}| |A| \|_\infty \leq \kappa_\infty(A)$. If \hat{x} is the j th column of X , then w_c is returned in BERR(j) and a bound on $\|x - \hat{x}\|_\infty / \|\hat{x}\|_\infty$ is returned in FERR(j). See Section 4.4 of Anderson *et al.* (1999) for further details.

8 Further Comments

The factorization of A requires approximately $\frac{4}{3}n^3$ floating point operations.

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ floating point operations. Each step of iterative refinement involves an additional $24n^2$ 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 equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $8n^2$ operations.

The real analogue of this routine is F07PBF (DSPSVX). The complex Hermitian analogue of this routine is F07PPF (ZHPSVX).

9 Example

This example solves the equations

$$AX = B,$$

where A is the complex symmetric matrix

$$A = \begin{pmatrix} -0.56 + 0.12i & -1.54 - 2.86i & 5.32 - 1.59i & 3.80 + 0.92i \\ -1.54 - 2.86i & -2.83 - 0.03i & -3.52 + 0.58i & -7.86 - 2.96i \\ 5.32 - 1.59i & -3.52 + 0.58i & 8.86 + 1.81i & 5.14 - 0.64i \\ 3.80 + 0.92i & -7.86 - 2.96i & 5.14 - 0.64i & -0.39 - 0.71i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -6.43 + 19.24i & -4.59 - 35.53i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -55.64 + 41.22i & -19.09 - 35.97i \end{pmatrix}.$$

Error estimates for the solutions, and an estimate of the reciprocal of the condition number of the matrix A are also output.

9.1 Program Text

```

Program f07qpfe

!      F07QPF Example Program Text

!      Mark 24 Release. NAG Copyright 2012.

!      .. Use Statements ..
Use nag_library, Only: nag_wp, x04dbf, zspsvx
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6
Character (1), Parameter   :: uplo = 'U'
!      .. Local Scalars ..
Real (Kind=nag_wp)         :: rcond
Integer                     :: i, ifail, info, j, ldb, ldx, n, nrhs
!      .. Local Arrays ..
Complex (Kind=nag_wp), Allocatable :: afp(:,), ap(:,), b(:,,:), work(:,) &
                                     x(:,,:)
Real (Kind=nag_wp), Allocatable :: berr(:,), ferr(:,), rwork(:,)
Integer, Allocatable           :: ipiv(:,)
Character (1)                  :: clabs(1), rlabs(1)
!      .. Executable Statements ..
Write (nout,*) 'F07QPF Example Program Results'
Write (nout,*)
Flush (nout)
!      Skip heading in data file
Read (nin,*)
Read (nin,*) n, nrhs
ldb = n
ldx = n
Allocate (afp((n*(n+1))/2), ap((n*(n+1))/2), b(ldb,nrhs), work(2*n), x(ldx, &
    nrhs), berr(nrhs), ferr(nrhs), rwork(n), ipiv(n))

!      Read the upper or lower triangular part of the matrix A from
!      data file

```

```

      If (uplo=='U') Then
        Read (nin,*)((ap(i+(j*(j-1))/2),j=i,n),i=1,n)
      Else If (uplo=='L') Then
        Read (nin,*)((ap(i+((2*n-j)*(j-1))/2),j=1,i),i=1,n)
      End If

!      Read B from data file

      Read (nin,*)(b(i,1:nrhs),i=1,n)

!      Solve the equations AX = B for X
!      The NAG name equivalent of zspsvx is f06qpf
      Call zspsvx('Not factored',uplo,n,nrhs,ap,afp,ipiv,b,ldb,x,ldx,rcond, &
        ferr,berr,work,rwork,info)

      If ((info==0) .Or. (info==n+1)) Then

!      Print solution, error bounds and condition number

!      ifail: behaviour on error exit
!      =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call x04dbf('General',' ',n,nrhs,x,ldx,'Bracketed','F7.4', &
        'Solution(s)','Integer',rlabs,'Integer',clabs,80,0,ifail)

      Write (nout,*)
      Write (nout,*) 'Backward errors (machine-dependent)'
      Write (nout,99999) berr(1:nrhs)
      Write (nout,*)
      Write (nout,*) 'Estimated forward error bounds (machine-dependent)'
      Write (nout,99999) ferr(1:nrhs)
      Write (nout,*)
      Write (nout,*) 'Estimate of reciprocal condition number'
      Write (nout,99999) rcond
      Write (nout,*)

      If (info==n+1) Then
        Write (nout,*)
        Write (nout,*) 'The matrix A is singular to working precision'
      End If
      Else
        Write (nout,99998) 'The diagonal block ', info, ' of D is zero'
      End If

99999 Format ((3X,1P,7E11.1))
99998 Format (1X,A,I3,A)
      End Program f07qpfe

```

9.2 Program Data

F07QPF Example Program Data

```

      4          2                                     :N and NRHS
      (-0.56, 0.12) (-1.54, -2.86) ( 5.32, -1.59) ( 3.80, 0.92)
              (-2.83,-0.03) (-3.52, 0.58) (-7.86, -2.96)
              ( 8.86, 1.81) ( 5.14, -0.64)
              ( -0.39,-0.71) :End matrix A

      (-6.43, 19.24) (-4.59,-35.53)
      (-0.49, -1.47) ( 6.95, 20.49)
      (-48.18, 66.00) (-12.08,-27.02)
      (-55.64, 41.22) (-19.09,-35.97)                                     :End matrix B

```

9.3 Program Results

F07QPF Example Program Results

```

Solution(s)
              1          2
1  (-4.0000, 3.0000) (-1.0000, 1.0000)
2  ( 3.0000,-2.0000) ( 3.0000, 2.0000)

```

3 (-2.0000, 5.0000) (1.0000,-3.0000)
4 (1.0000,-1.0000) (-2.0000,-1.0000)

Backward errors (machine-dependent)
8.1E-17 3.0E-17

Estimated forward error bounds (machine-dependent)
1.2E-14 1.2E-14

Estimate of reciprocal condition number
4.9E-02
