NAG Toolbox

nag_lapack_zcposv (f07fq)

1 Purpose

nag_lapack_zcposv (f07fq) uses the Cholesky factorization

$$A = U^{\mathrm{H}}U \quad \text{or} \quad A = LL^{\mathrm{H}}$$

to compute the solution to a complex system of linear equations

$$AX = B,$$

where A is an n by n Hermitian positive definite matrix and X and B are n by r matrices.

2 Syntax

```
[a, x, iter, info] = nag_lapack_zcposv(uplo, a, b, 'n', n, 'nrhs_p', nrhs_p)
[a, x, iter, info] = f07fq(uplo, a, b, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zcposv (f07fq) first attempts to factorize the matrix in reduced precision and use this factorization within an iterative refinement procedure to produce a solution with full precision normwise backward error quality (see below). If the approach fails the method switches to a full precision factorization and solve.

The iterative refinement can be more efficient than the corresponding direct full precision algorithm. Since the strategy implemented by nag_lapack_zcposv (f07fq) must perform iterative refinement on each right-hand side, any efficiency gains will reduce as the number of right-hand sides increases. Conversely, as the matrix size increases the cost of these iterative refinements become less significant relative to the cost of factorization. Thus, any efficiency gains will be greatest for a very small number of right-hand sides and for large matrix sizes. The cut-off values for the number of right-hand sides and matrix size, for which the iterative refinement strategy performs better, depends on the relative performance of the reduced and full precision factorization and back-substitution. nag_lapack_zcposv (f07fq) always attempts the iterative refinement strategy first; you are advised to compare the performance of nag_lapack_zcposv (f07fq) with that of its full precision counterpart nag_lapack_zposv (f07fn) to determine whether this strategy is worthwhile for your particular problem dimensions.

The iterative refinement process is stopped if **iter** > 30 where **iter** is the number of iterations carried out thus far. The process is also stopped if for all right-hand sides we have

$$\|resid\| < \sqrt{\mathbf{n}}\|x\|\|A\|\epsilon,$$

where $\|resid\|$ is the ∞-norm of the residual, $\|x\|$ is the ∞-norm of the solution, $\|A\|$ is the ∞-norm of the matrix A and ϵ is the ***machine precision*** returned by nag_machine_precision (x02aj).

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

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

Specifies whether the upper or lower triangular part of A is stored.

uplo = 'U'
The upper triangular part of A is stored.

uplo = 'L'
The lower triangular part of A is stored.

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

2: **a**$(lda,:)$ – COMPLEX (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 Hermitian positive definite matrix A.

If **uplo** = 'U', the upper triangular part of a must be stored and the elements of the array below the diagonal are not referenced.

If **uplo** = 'L', the lower triangular part of a must be stored and the elements of the array above the diagonal are not referenced.

3: **b**$(ldb,:)$ – COMPLEX (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 right-hand side matrix B.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the array **n**.

n, the number of linear equations, i.e., the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

2: **nrhs_p** – INTEGER

Default: the second dimension of the array **b**.

r, the number of right-hand sides, i.e., the number of columns of the matrix B.

Constraint: $\mathbf{nrhs\_p} \geq 0$.

5.3 Output Parameters

1: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

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

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

If iterative refinement has been successfully used (**info** $= 0$ and **iter** ≥ 0, see description below), then **a** is unchanged. If full precision factorization has been used (**info** $= 0$ and **iter** < 0, see description below), then the array A contains the factor U or L from the Cholesky factorization $A = U^{\mathrm{H}}U$ or $A = LL^{\mathrm{H}}$.

2: **x**$(ldx,:)$ – COMPLEX (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 $\max(1,\mathbf{nrhs\_p})$.

If **info** $= 0$, the n by r solution matrix X.

3: **iter** – INTEGER

Information on the progress of the interative refinement process.

iter < 0
Iterative refinement has failed for one of the reasons given below, full precision factorization has been performed instead.

-1 The function fell back to full precision for implementation- or machine-specific reasons.

-2 Narrowing the precision induced an overflow, the function fell back to full precision.

-3 An intermediate reduced precision factorization failed.

-31 The maximum permitted number of iterations was exceeded.

iter > 0
Iterative refinement has been sucessfully used. **iter** returns the number of iterations.

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.

info > 0 and **info** $\leq$ **n**

The leading minor of order ⟨*value*⟩ of A is not positive definite, so the factorization could not be completed, and the solution has not been computed.

7 Accuracy

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

if **uplo** = 'U', $|E| \leq c(n)\epsilon|U^{\mathrm{H}}||U|$;

if **uplo** = 'L', $|E| \leq c(n)\epsilon|L||L^{\mathrm{H}}|$,

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***. See Section 10.1 of Higham (2002) for further details.

An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x}-x\|_1}{\|x\|_1} \leq \kappa(A)\frac{\|E\|_1}{\|A\|_1}$$

where $\kappa(A) = \left\|A^{-1}\right\|_1 \|A\|_1$, the condition number of A with respect to the solution of the linear equations. See Section 4.4 of Anderson *et al.* (1999) for further details.

8 Further Comments

The real analogue of this function is nag_lapack_dsposv (f07fc).

9 Example

This example solves the equations

$$AX = B,$$

where A is the Hermitian positive definite matrix

$$A = \begin{pmatrix} 3.23 & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.93 - 6.14i \\ 6.17 + 9.42i \\ -7.17 - 21.83i \\ 1.99 - 14.38i \end{pmatrix}.$$

9.1 Program Text

```
function f07fq_example

fprintf('f07fq example results\n\n');

% Hermitian matrix A
a = [3.23 + 0i,     1.51 - 1.92i,  1.90 + 0.84i,  0.42 + 2.50i;
     1.51 + 1.92i,  3.58 + 0i,    -0.23 + 1.11i, -1.18 + 1.37i;
     1.90 - 0.84i, -0.23 - 1.11i,  4.09 + 0i,     2.33 - 0.14i;
     0.42 - 2.50i, -1.18 - 1.37i,  2.33 + 0.14i,  4.29 + 0i];

% Rhs
b = [3.93 -  6.14i;
     6.17 +  9.42i;
    -7.17 - 21.83i;
     1.99 - 14.38i];

% Solve Ax = b for x
[af, x, iter, info] = f07fq( ...
                           'Upper', a, b);

fprintf('Solution:\n');
disp(x);
```

9.2 Program Results

```
f07fq example results

Solution:
   1.0000 - 1.0000i
  -0.0000 + 3.0000i
  -4.0000 - 5.0000i
   2.0000 + 1.0000i
```