

NAG Toolbox

nag_lapack_zptrfs (f07jv)

1 Purpose

nag_lapack_zptrfs (f07jv) computes error bounds and refines the solution to a complex system of linear equations $AX = B$, where A is an n by n Hermitian positive definite tridiagonal matrix and X and B are n by r matrices, using the modified Cholesky factorization returned by nag_lapack_zpttrf (f07jr) and an initial solution returned by nag_lapack_zptrs (f07js). Iterative refinement is used to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = nag_lapack_zptrfs(uplo, d, e, df, ef, b, x, 'n', n,
'nrhs_p', nrhs_p)
[x, ferr, berr, info] = f07jv(uplo, d, e, df, ef, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zptrfs (f07jv) should normally be preceded by calls to nag_lapack_zpttrf (f07jr) and nag_lapack_zptrs (f07js). nag_lapack_zpttrf (f07jr) computes a modified Cholesky factorization of the matrix A as

$$A = LDL^H,$$

where L is a unit lower bidiagonal matrix and D is a diagonal matrix, with positive diagonal elements. nag_lapack_zptrs (f07js) then utilizes the factorization to compute a solution, \hat{X} , to the required equations. Letting \hat{x} denote a column of \hat{X} , nag_lapack_zptrfs (f07jv) computes a *component-wise backward error*, β , the smallest relative perturbation in each element of A and b such that \hat{x} is the exact solution of a perturbed system

$$(A + E)\hat{x} = b + f, \quad \text{with } |e_{ij}| \leq \beta|a_{ij}|, \quad \text{and } |f_j| \leq \beta|b_j|.$$

The function also estimates a bound for the *component-wise forward error* in the computed solution defined by $\max |x_i - \hat{x}_i| / \max |\hat{x}_i|$, where x is the corresponding column of the exact solution, X .

Note that the modified Cholesky factorization of A can also be expressed as

$$A = U^H DU,$$

where U is unit upper bidiagonal.

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>

5 Parameters

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

Specifies the form of the factorization as follows:

uplo = 'U'
 $A = U^H DU.$

uplo = 'L'
 $A = LDL^H$.

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

- 2: **d**(:) – REAL (KIND=nag_wp) array
 The dimension of the array **d** must be at least $\max(1, \mathbf{n})$
 Must contain the n diagonal elements of the matrix of A .
- 3: **e**(:) – COMPLEX (KIND=nag_wp) array
 The dimension of the array **e** must be at least $\max(1, \mathbf{n} - 1)$
 If **uplo** = 'U', **e** must contain the $(n - 1)$ superdiagonal elements of the matrix A .
 If **uplo** = 'L', **e** must contain the $(n - 1)$ subdiagonal elements of the matrix A .
- 4: **df**(:) – REAL (KIND=nag_wp) array
 The dimension of the array **df** must be at least $\max(1, \mathbf{n})$
 Must contain the n diagonal elements of the diagonal matrix D from the LDL^T factorization of A .
- 5: **ef**(:) – COMPLEX (KIND=nag_wp) array
 The dimension of the array **ef** must be at least $\max(1, \mathbf{n} - 1)$
 If **uplo** = 'U', **ef** must contain the $(n - 1)$ superdiagonal elements of the unit upper bidiagonal matrix U from the $U^H DU$ factorization of A .
 If **uplo** = 'L', **ef** must contain the $(n - 1)$ subdiagonal elements of the unit lower bidiagonal matrix L from the LDL^H factorization of A .
- 6: **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 n by r matrix of right-hand sides B .
- 7: **x**(ldx,:) – COMPLEX (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 initial solution matrix X .

5.2 Optional Input Parameters

- 1: **n** – INTEGER
Default: the first dimension of the arrays **b**, **x** and the dimension of the arrays **d**, **df**,
 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**.
 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: **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 $\max(1, \mathbf{nrhs_p})$.

The n by r refined solution matrix X .

2: **ferr**(*nrhs_p*) – REAL (KIND=*nag_wp*) array

Estimate of the forward error bound for each computed solution vector, such that $\|\hat{x}_j - x_j\|_\infty / \|\hat{x}_j\|_\infty \leq \mathbf{ferr}(j)$, 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 almost always a slight overestimate of the true error.

3: **berr**(*nrhs_p*) – REAL (KIND=*nag_wp*) array

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).

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 computed solution for a single right-hand side, \hat{x} , satisfies an equation of the form

$$(A + E)\hat{x} = b,$$

where

$$\|E\|_\infty = O(\epsilon)\|A\|_\infty$$

and ϵ is the *machine precision*. An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x} - x\|_\infty}{\|x\|_\infty} \leq \kappa(A) \frac{\|E\|_\infty}{\|A\|_\infty},$$

where $\kappa(A) = \|A^{-1}\|_\infty \|A\|_\infty$, 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.

Function `nag_lapack_zptcon` (f07ju) can be used to compute the condition number of A .

8 Further Comments

The total number of floating-point operations required to solve the equations $AX = B$ is proportional to nr . At most five steps of iterative refinement are performed, but usually only one or two steps are required.

The real analogue of this function is `nag_lapack_dptrfs` (f07jh).

9 Example

This example solves the equations

$$AX = B,$$

where A is the Hermitian positive definite tridiagonal matrix

$$A = \begin{pmatrix} 16.0 & 16.0 - 16.0i & 0 & 0 \\ 16.0 + 16.0i & 41.0 & 18.0 + 9.0i & 0 \\ 0 & 18.0 - 9.0i & 46.0 & 1.0 + 4.0i \\ 0 & 0 & 1.0 - 4.0i & 21.0 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 64.0 + 16.0i & -16.0 - 32.0i \\ 93.0 + 62.0i & 61.0 - 66.0i \\ 78.0 - 80.0i & 71.0 - 74.0i \\ 14.0 - 27.0i & 35.0 + 15.0i \end{pmatrix}.$$

Estimates for the backward errors and forward errors are also output.

9.1 Program Text

```
function f07jv_example

fprintf('f07jv example results\n\n');

% Hermitian tridiagonal A stored as two diagonals
d = [ 16      41      46      21];
e = [ 16 + 16i    18 - 9i    1 - 4i    ];

% Factorize
[df, ef, info] = f07jr( ...
    d, e);

% RHS
b = [ 64 + 16i,  -16 - 32i;
     93 + 62i,   61 - 66i;
     78 - 80i,   71 - 74i;
     14 - 27i,   35 + 15i];

% Solve
uplo = 'L';
[x, info] = f07js( ...
    uplo, df, ef, b);

% Refine
[x, ferr, berr, info] = f07jv( ...
    uplo, d, e, df, ef, b, x);

disp('Solution(s)');
disp(x);
fprintf('Forward error bounds = %10.1e  %10.1e\n',ferr);
fprintf('Backward error bounds = %10.1e  %10.1e\n',berr);
```

9.2 Program Results

f07jv example results

Solution(s)

2.0000 + 1.0000i	-3.0000 - 2.0000i
1.0000 + 1.0000i	1.0000 + 1.0000i
1.0000 - 2.0000i	1.0000 - 2.0000i
1.0000 - 1.0000i	2.0000 + 1.0000i

Forward error bounds =	9.0e-12	6.1e-12
Backward error bounds =	0.0e+00	0.0e+00
