NAG Toolbox

nag_lapack_dptrfs (f07jh)

1 Purpose

nag_lapack_dptrfs (f07jh) computes error bounds and refines the solution to a real system of linear equations $AX = B$, where A is an n by n symmetric positive definite tridiagonal matrix and X and B are n by r matrices, using the modified Cholesky factorization returned by nag_lapack_dpttrf (f07jd) and an initial solution returned by nag_lapack_dpttrs (f07je). Iterative refinement is used to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = nag_lapack_dptrfs(d, e, df, ef, b, x, 'n', n, 'nrhs_p', nrhs_p)
[x, ferr, berr, info] = f07jh(d, e, df, ef, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_dptrfs (f07jh) should normally be preceded by calls to nag_lapack_dpttrf (f07jd) and nag_lapack_dpttrs (f07je). nag_lapack_dpttrf (f07jd) computes a modified Cholesky factorization of the matrix A as

$$A = LDL^{\mathrm{T}},$$

where L is a unit lower bidiagonal matrix and D is a diagonal matrix, with positive diagonal elements. nag_lapack_dpttrs (f07je) 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_dptrfs (f07jh) 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

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

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

$$A = U^{\mathrm{T}} DU,$$

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: **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.

2: **e**(:) – REAL (KIND=nag_wp) array

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

Must contain the $(n - 1)$ subdiagonal elements of the matrix A.

3: **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.

4: **ef**(:) – REAL (KIND=nag_wp) array

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

Must contain the $(n - 1)$ subdiagonal elements of the unit bidiagonal matrix L from the LDL^{T} factorization of A.

5: **b**(ldb, :) – REAL (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.

6: **x**(ldx, :) – REAL (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**, **ef**.

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, :) – REAL (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 $\left\|\hat{x}_j - x_j\right\|_\infty / \left\|\hat{x}_j\right\|_\infty \leq \mathbf{ferr}(j)$, where $\hat{x}_j$ is the jth 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 $\kappa(A) = \left\|A^{-1}\right\|_\infty \|A\|_\infty$, 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_dptcon (f07jg) 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 complex analogue of this function is nag_lapack_zptrfs (f07jv).

9 Example

This example solves the equations

$$AX = B,$$

where A is the symmetric positive definite tridiagonal matrix

$$A = \begin{pmatrix} 4.0 & -2.0 & 0 & 0 & 0 \\ -2.0 & 10.0 & -6.0 & 0 & 0 \\ 0 & -6.0 & 29.0 & 15.0 & 0 \\ 0 & 0 & 15.0 & 25.0 & 8.0 \\ 0 & 0 & 0 & 8.0 & 5.0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 6.0 & 10.0 \\ 9.0 & 4.0 \\ 2.0 & 9.0 \\ 14.0 & 65.0 \\ 7.0 & 23.0 \end{pmatrix}.$$

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

9.1 Program Text

```
     function f07jh_example

fprintf('f07jh example results\n\n');

% Symmetric tridiagonal A stored as two diagonals
d = [ 4     10     29     25     5];
e = [-2     -6     15     8       ];

% RHS
b = [ 6, 10;
      9,  4;
      2,  9;
     14, 65;
      7, 23];

% Factorize
[df, ef, info] = f07jd( ...
                        d, e);

%Solve
[x, info] = f07je( ...
                   df, ef, b);

% Refine
ef = [ef 0];
[x, ferr, berr, info] = f07jh( ...
                               d, e, df, ef, b, x);

disp('Solution');
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

```
     f07jh example results

Solution
    2.5000    2.0000
    2.0000   -1.0000
    1.0000   -3.0000
   -1.0000    6.0000
    3.0000   -5.0000

Forward  error bounds =    2.4e-14     4.7e-14
Backward error bounds =    0.0e+00     7.4e-17
```