NAG Toolbox

nag_lapack_zhptrs (f07ps)

1 Purpose

nag_lapack_zhptrs (f07ps) solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides,

$$AX = B,$$

where A has been factorized by nag_lapack_zhptrf (f07pr), using packed storage.

2 Syntax

```
[b, info] = nag_lapack_zhptrs(uplo, ap, ipiv, b, 'n', n, 'nrhs_p', nrhs_p)
[b, info] = f07ps(uplo, ap, ipiv, b, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zhptrs (f07ps) is used to solve a complex Hermitian indefinite system of linear equations $AX = B$, the function must be preceded by a call to nag_lapack_zhptrf (f07pr) which computes the Bunch–Kaufman factorization of A, using packed storage.

If **uplo** = 'U', $A = PUDU^{\mathrm{H}}P^{\mathrm{T}}$, where P is a permutation matrix, U is an upper triangular matrix and D is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^{\mathrm{H}}P^{\mathrm{T}}X = Y$.

If **uplo** = 'L', $A = PLDL^{\mathrm{H}}P^{\mathrm{T}}$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^{\mathrm{H}}P^{\mathrm{T}}X = Y$.

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** – CHARACTER(1)

Specifies how A has been factorized.

uplo = 'U'
$A = PUDU^{\mathrm{H}}P^{\mathrm{T}}$, where U is upper triangular.

uplo = 'L'
$A = PLDL^{\mathrm{H}}P^{\mathrm{T}}$, where L is lower triangular.

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

2: **ap**(:) – COMPLEX (KIND=nag_wp) array

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

The factorization of A stored in packed form, as returned by nag_lapack_zhptrf (f07pr).

3: **ipiv**(:) – INTEGER array

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 nag_lapack_zhptrf (f07pr).

4: **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 right-hand side matrix B.

5.2 Optional Input Parameters

1: **n** – INTEGER

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** – INTEGER

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

r, the number of right-hand sides.

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

5.3 Output Parameters

1: **b**(ldb, :) – COMPLEX (KIND=nag_wp) array

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

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

The n by r solution matrix X.

2: **info** – INTEGER

$\mathbf{info} = 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

$\mathbf{info} < 0$

If $\mathbf{info} = -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

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', $|E| \leq c(n)\epsilon P|U||D||U^{\mathrm{H}}|P^{\mathrm{T}}$;

if **uplo** = 'L', $|E| \leq c(n)\epsilon P|L||D||L^{\mathrm{H}}|P^{\mathrm{T}}$,

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

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

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\,\mathrm{cond}(A,x)\epsilon$$

where $\mathrm{cond}(A,x) = \left\| |A^{-1}||A||x| \right\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$.

Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling nag_lapack_zhprfs (f07pv), and an estimate for $\kappa_\infty(A)$ $(=\kappa_1(A))$ can be obtained by calling nag_lapack_zhpcon (f07pu).

8 Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This function may be followed by a call to nag_lapack_zhprfs (f07pv) to refine the solution and return an error estimate.

The real analogue of this function is nag_lapack_dsptrs (f07pe).

9 Example

This example solves the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.36+0.00i & 1.58+0.90i & 2.21-0.21i & 3.91+1.50i \\ 1.58-0.90i & -8.87+0.00i & -1.84-0.03i & -1.78+1.18i \\ 2.21+0.21i & -1.84+0.03i & -4.63+0.00i & 0.11+0.11i \\ 3.91-1.50i & -1.78-1.18i & 0.11-0.11i & -1.84+0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 7.79+5.48i & -35.39+18.01i \\ -0.77-16.05i & 4.23-70.02i \\ -9.58+3.88i & -24.79-8.40i \\ 2.98-10.18i & 28.68-39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by nag_lapack_zhptrf (f07pr).

9.1 Program Text

```
     function f07ps_example

fprintf('f07ps example results\n\n');

% Hermitian indefinite matrix A (Lower triangular part stored in packed form)
uplo = 'L';
n = nag_int(4);
ap = [-1.36 + 0i;  1.58 - 0.9i;   2.21 + 0.21i;  3.91 - 1.5i;
                  -8.87 + 0i;    -1.84 + 0.03i; -1.78 - 1.18i;
                                 -4.63 + 0i;     0.11 - 0.11i;
                                                -1.84 + 0i];

% Factorize
[apf, ipiv, info] = f07pr( ...
                           uplo, n, ap);

% RHS
b = [ 7.79 +  5.48i, -35.39 + 18.01i;
     -0.77 - 16.05i,   4.23 - 70.02i;
     -9.58 +  3.88i, -24.79 -  8.40i;
      2.98 - 10.18i,  28.68 - 39.89i];

% Solve
```

```
[x, info] = f07ps( ...
                   uplo, apf, ipiv, b);

disp('Solution(s)');
disp(x);
```

9.2 Program Results

```
     f07ps example results

Solution(s)
   1.0000 - 1.0000i   3.0000 - 4.0000i
  -1.0000 + 2.0000i  -1.0000 + 5.0000i
   3.0000 - 2.0000i   7.0000 - 2.0000i
   2.0000 + 1.0000i  -8.0000 + 6.0000i
```