

# NAG C Library Function Document

## nag\_zhptrs (f07psc)

### 1 Purpose

nag\_zhptrs (f07psc) solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides,  $AX = B$ , where  $A$  has been factorized by nag\_zhptrf (f07prc), using packed storage.

### 2 Specification

```
void nag_zhptrs (Nag_OrderType order, Nag_UploType uplo, Integer n, Integer nrhs,
               const Complex ap[], const Integer ipiv[], Complex b[], Integer pdb,
               NagError *fail)
```

### 3 Description

To solve a complex Hermitian indefinite system of linear equations  $AX = B$ , this function must be preceded by a call to nag\_zhptrf (f07prc) which computes the Bunch–Kaufman factorization of  $A$  using packed storage.

If **uplo** = **Nag\_Upper**,  $A = PUDU^H P^T$ , 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  $U^H P^T X = Y$ .

If **uplo** = **Nag\_Lower**,  $A = PLDL^H P^T$ , where  $L$  is a lower triangular matrix; the solution  $X$  is computed by solving  $PLDY = B$  and then  $L^H P^T X = Y$ .

### 4 References

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

### 5 Parameters

1: **order** – Nag\_OrderType *Input*

*On entry:* the **order** parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = **Nag\_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

*Constraint:* **order** = **Nag\_RowMajor** or **Nag\_ColMajor**.

2: **uplo** – Nag\_UploType *Input*

*On entry:* indicates how  $A$  has been factorized as follows:

if **uplo** = **Nag\_Upper**,  $A = PUDU^H P^T$ , where  $U$  is upper triangular;

if **uplo** = **Nag\_Lower**,  $A = PLDL^H P^T$ , where  $L$  is lower triangular.

*Constraint:* **uplo** = **Nag\_Upper** or **Nag\_Lower**.

3: **n** – Integer *Input*

*On entry:*  $n$ , the order of the matrix  $A$ .

*Constraint:*  $n \geq 0$ .

- 4: **nrhs** – Integer *Input*  
*On entry:*  $r$ , the number of right-hand sides.  
*Constraint:*  $\mathbf{nrhs} \geq 0$ .
- 5: **ap**[ $dim$ ] – const Complex *Input*  
**Note:** the dimension,  $dim$ , of the array **ap** must be at least  $\max(1, \mathbf{n} \times (\mathbf{n} + 1)/2)$ .  
*On entry:* details of the factorization of  $A$  stored in packed form, as returned by nag\_zhptrf (f07prc).
- 6: **ipiv**[ $dim$ ] – const Integer *Input*  
**Note:** the dimension,  $dim$ , of the array **ipiv** must be at least  $\max(1, \mathbf{n})$ .  
*On entry:* details of the interchanges and the block structure of  $D$ , as returned by nag\_zhptrf (f07prc).
- 7: **b**[ $dim$ ] – Complex *Input/Output*  
**Note:** the dimension,  $dim$ , of the array **b** must be at least  $\max(1, \mathbf{pdb} \times \mathbf{nrhs})$  when **order** = **Nag\_ColMajor** and at least  $\max(1, \mathbf{pdb} \times \mathbf{n})$  when **order** = **Nag\_RowMajor**.  
If **order** = **Nag\_ColMajor**, the  $(i, j)$ th element of the matrix  $B$  is stored in  $\mathbf{b}[(j - 1) \times \mathbf{pdb} + i - 1]$  and if **order** = **Nag\_RowMajor**, the  $(i, j)$ th element of the matrix  $B$  is stored in  $\mathbf{b}[(i - 1) \times \mathbf{pdb} + j - 1]$ .  
*On entry:* the  $n$  by  $r$  right-hand side matrix  $B$ .  
*On exit:* the  $n$  by  $r$  solution matrix  $X$ .
- 8: **pdb** – Integer *Input*  
*On entry:* the stride separating matrix row or column elements (depending on the value of **order**) in the array **b**.  
*Constraints:*  
if **order** = **Nag\_ColMajor**,  $\mathbf{pdb} \geq \max(1, \mathbf{n})$ ;  
if **order** = **Nag\_RowMajor**,  $\mathbf{pdb} \geq \max(1, \mathbf{nrhs})$ .
- 9: **fail** – NagError \* *Output*  
The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry, **n** =  $\langle value \rangle$ .

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

On entry, **nrhs** =  $\langle value \rangle$ .

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

On entry, **pdb** =  $\langle value \rangle$ .

Constraint:  $\mathbf{pdb} > 0$ .

### NE\_INT\_2

On entry, **pdb** =  $\langle value \rangle$ , **n** =  $\langle value \rangle$ .

Constraint:  $\mathbf{pdb} \geq \max(1, \mathbf{n})$ .

On entry, **pdb** =  $\langle value \rangle$ , **nrhs** =  $\langle value \rangle$ .

Constraint:  $\mathbf{pdb} \geq \max(1, \mathbf{nrhs})$ .

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

On entry, parameter  $\langle value \rangle$  had an illegal value.

**NE\_INTERNAL\_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

**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** = **Nag\_Upper**,  $|E| \leq c(n)\epsilon P|U||D||U^H|P^T$ ;

if **uplo** = **Nag\_Lower**,  $|E| \leq c(n)\epsilon P|L||D||L^H|P^T$ ,

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

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

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

where  $\text{cond}(A, x) = \| |A^{-1}| |A| |x| \|_\infty / \|x\|_\infty \leq \text{cond}(A) = \| |A^{-1}| |A| \|_\infty \leq \kappa_\infty(A)$ . Note that  $\text{cond}(A, x)$  can be much smaller than  $\text{cond}(A)$ .

Forward and backward error bounds can be computed by calling `nag_zhprfs` (f07pvc), and an estimate for  $\kappa_\infty(A)$  ( $= \kappa_1(A)$ ) can be obtained by calling `nag_zhpcon` (f07puc).

**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_zhprfs` (f07pvc) to refine the solution and return an error estimate.

The real analogue of this function is `nag_dsptsr` (f07pec).

**9 Example**

To solve 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_zhptrf` (f07prc).

## 9.1 Program Text

```

/* nag_zhptrs (f07psc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer ap_len, i, j, n, nrhs, pdb;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    Complex *ap=0, *b=0;

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
#define B(I,J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
#else
#define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
#define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
#define B(I,J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07psc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^\\n] ");
    Vscanf("%ld%ld%*[^\\n] ", &n, &nrhs);
    ap_len = n * (n + 1)/2;
#ifdef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(ap = NAG_ALLOC(n * (n + 1)/2, Complex)) ||
        !(b = NAG_ALLOC(n * nrhs, Complex)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A and B from data file */
    Vscanf(" ' %1s '%*[^\\n] ", uplo);
    if (*(unsigned char *)uplo == 'L')
        uplo_enum = Nag_Lower;
    else if (*(unsigned char *)uplo == 'U')
        uplo_enum = Nag_Upper;
    else
    {

```

```

    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}
if (uplo_enum == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf(" ( %lf , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
    }
    Vscanf("%*[\n] ");
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        Vscanf(" ( %lf , %lf )", &B(i,j).re, &B(i,j).im);
}
Vscanf("%*[\n] ");

/* Factorize A */
f07prc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07prc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
f07psc(order, uplo_enum, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07psc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb,
        Nag_BracketForm, "%7.4f", "Solution(s)", Nag_IntegerLabels,
        0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04dbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ipiv) NAG_FREE(ipiv);
if (ap) NAG_FREE(ap);
if (b) NAG_FREE(b);
return exit_status;
}

```

## 9.2 Program Data

f07psc Example Program Data

```

4 2                                     :Values of N and NRHS
'L'                                     :Value of UPLO
(-1.36, 0.00)
( 1.58,-0.90) (-8.87, 0.00)
( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)

```

```
( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00) :End of matrix A
( 7.79,  5.48) (-35.39, 18.01)
(-0.77,-16.05) (  4.23,-70.02)
(-9.58,  3.88) (-24.79, -8.40)
( 2.98,-10.18) ( 28.68,-39.89)                               :End of matrix B
```

### 9.3 Program Results

f07psc Example Program Results

Solution(s)

	1	2
1	( 1.0000,-1.0000)	( 3.0000,-4.0000)
2	(-1.0000, 2.0000)	(-1.0000, 5.0000)
3	( 3.0000,-2.0000)	( 7.0000,-2.0000)
4	( 2.0000, 1.0000)	(-8.0000, 6.0000)

---