

# NAG Library Function Document

## nag\_zsptrs (f07qsc)

### 1 Purpose

nag\_zsptrs (f07qsc) solves a complex symmetric system of linear equations with multiple right-hand sides,

$$AX = B,$$

where  $A$  has been factorized by nag\_zsptrf (f07qrc), using packed storage.

### 2 Specification

```
#include <nag.h>
#include <nagf07.h>

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

### 3 Description

nag\_zsptrs (f07qsc) is used to solve a complex symmetric system of linear equations  $AX = B$ , the function must be preceded by a call to nag\_zsptrf (f07qrc) which computes the Bunch–Kaufman factorization of  $A$ , using packed storage.

If **uplo** = Nag\_Upper,  $A = PUDU^T P^T$ , where  $P$  is a permutation matrix,  $U$  is an upper triangular matrix and  $D$  is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution  $X$  is computed by solving  $PUDY = B$  and then  $U^T P^T X = Y$ .

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

### 4 References

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

### 5 Arguments

1: **order** – Nag\_OrderType *Input*

*On entry:* the **order** argument 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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

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

2: **uplo** – Nag\_UploType *Input*

*On entry:* specifies how  $A$  has been factorized.

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

**uplo** = Nag\_Lower  
 $A = PLDL^T P^T$ , 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:* **nrhs**  $\geq 0$ .
- 5: **ap**[*dim*] – const Complex *Input*  
**Note:** the dimension, *dim*, of the array **ap** must be at least  $\max(1, n \times (n + 1)/2)$ .  
*On entry:* the factorization of  $A$  stored in packed form, as returned by nag\_zsptf (f07qrc).
- 6: **ipiv**[*dim*] – const Integer *Input*  
**Note:** the dimension, *dim*, of the array **ipiv** must be at least  $\max(1, n)$ .  
*On entry:* details of the interchanges and the block structure of  $D$ , as returned by nag\_zsptf (f07qrc).
- 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;  
 $\max(1, n \times \mathbf{pdb})$  when **order** = Nag\_RowMajor.  
The  $(i, j)$ th element of the matrix  $B$  is stored in  
 $\mathbf{b}[(j - 1) \times \mathbf{pdb} + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{b}[(i - 1) \times \mathbf{pdb} + j - 1]$  when **order** = Nag\_RowMajor.  
*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 row or column elements (depending on the value of **order**) in the array **b**.  
*Constraints:*  
if **order** = Nag\_ColMajor, **pdb**  $\geq \max(1, n)$ ;  
if **order** = Nag\_RowMajor, **pdb**  $\geq \max(1, \mathbf{nrhs})$ .
- 9: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE\_BAD\_PARAM**

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

**NE\_INT**

On entry,  $\mathbf{n} = \langle value \rangle$ .

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

On entry,  $\mathbf{nrhs} = \langle value \rangle$ .

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

On entry,  $\mathbf{pdb} = \langle value \rangle$ .

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

**NE\_INT\_2**

On entry,  $\mathbf{pdb} = \langle value \rangle$  and  $\mathbf{n} = \langle value \rangle$ .

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

On entry,  $\mathbf{pdb} = \langle value \rangle$  and  $\mathbf{nrhs} = \langle value \rangle$ .

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

**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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 in the Essential Introduction for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 3.6.5 in the Essential Introduction for further information.

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

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

$c(n)$  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  $\text{cond}(A, x) = \frac{\|A^{-1}\|_{\infty}\|A\|_{\infty}\|x\|_{\infty}}{\|x\|_{\infty}} \leq \text{cond}(A) = \frac{\|A^{-1}\|_{\infty}\|A\|_{\infty}}{1} \leq \kappa_{\infty}(A)$ .

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_zsprfs` (f07qvc), and an estimate for  $\kappa_{\infty}(A)$  ( $= \kappa_1(A)$ ) can be obtained by calling `nag_zspcon` (f07quc).

**8 Parallelism and Performance**

`nag_zsprts` (f07qsc) is not threaded by NAG in any implementation.

`nag_zsprts` (f07qsc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

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

This function may be followed by a call to nag\_zsprfs (f07qvc) to refine the solution and return an error estimate.

The real analogue of this function is nag\_dsprts (f07pec).

## 10 Example

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

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here  $A$  is symmetric, stored in packed form, and must first be factorized by nag\_zsprtf (f07qrc).

### 10.1 Program Text

```

/* nag_zsprts (f07qsc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 * Mark 7b revised, 2004.
 */

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

int main(void)
{
    /* Scalars */
    Integer      i, j, n, nrhs, pdb;
    Integer      exit_status = 0;
    NagError     fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    Integer      *ipiv = 0;
    char         nag_enum_arg[40];
    Complex      *ap = 0, *b = 0;

#ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker
       to load floating point support */
    float        force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

#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);

    printf("nag_zsptrs (f07qsc) Example Program Results\n\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#ifdef _WIN32
    scanf_s("%"NAG_IFMT%"NAG_IFMT"%*[\n] ", &n, &nrhs);
#else
    scanf("%"NAG_IFMT%"NAG_IFMT"%*[\n] ", &n, &nrhs);
#endif
#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)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A and B from data file */
#ifdef _WIN32
    scanf_s(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n] ", nag_enum_arg);
#endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

    if (uplo == Nag_Upper)
    {
        for (i = 1; i <= n; ++i)
            for (j = i; j <= n; ++j)
#ifdef _WIN32
                scanf_s(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#else
                scanf(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
    }
    else
    {
        for (i = 1; i <= n; ++i)

```

```

        for (j = 1; j <= i; ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &A_LOWER(i, j).re, &A_LOWER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &A_LOWER(i, j).re, &A_LOWER(i, j).im);
#endif
#ifdef _WIN32
            scanf_s("%*[\n] ");
#else
            scanf("%*[\n] ");
#endif
        }
        for (i = 1; i <= n; ++i)
            for (j = 1; j <= nrhs; ++j)
#ifdef _WIN32
                scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#else
                scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#endif
#ifdef _WIN32
            scanf_s("%*[\n] ");
#else
            scanf("%*[\n] ");
#endif

/* Factorize A */
/* nag_zsptrf (f07qrc).
 * Bunch-Kaufman factorization of complex symmetric matrix,
 * packed storage
 */
nag_zsptrf(order, uplo, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zsptrf (f07qrc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
/* nag_zsptrs (f07qsc).
 * Solution of complex symmetric system of linear equations,
 * multiple right-hand sides, matrix already factorized by
 * nag_zsptrf (f07qrc), packed storage
 */
nag_zsptrs(order, uplo, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zsptrs (f07qsc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
fflush(stdout);
nag_gen_complx_mat_print_comp(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)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}
END:

```

```

NAG_FREE(ipiv);
NAG_FREE(ap);
NAG_FREE(b);
return exit_status;
}

```

## 10.2 Program Data

```

nag_zspttrs (f07qsc) Example Program Data
  4  2                                     :Values of n and nrhs
  Nag_Lower                               :Value of uplo
(-0.39,-0.71)
( 5.14,-0.64) ( 8.86, 1.81)
(-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12) :End of matrix A
(-55.64, 41.22) (-19.09,-35.97)
(-48.18, 66.00) (-12.08,-27.02)
( -0.49, -1.47) ( 6.95, 20.49)
( -6.43, 19.24) ( -4.59,-35.53)           :End of matrix B

```

## 10.3 Program Results

```

nag_zspttrs (f07qsc) Example Program Results

Solution(s)
           1           2
1 ( 1.0000,-1.0000) (-2.0000,-1.0000)
2 (-2.0000, 5.0000) ( 1.0000,-3.0000)
3 ( 3.0000,-2.0000) ( 3.0000, 2.0000)
4 (-4.0000, 3.0000) (-1.0000, 1.0000)

```

---