

NAG C Library Function Document

nag_zsytrs (f07nsc)

1 Purpose

nag_zsytrs (f07nsc) solves a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by nag_zsytrf (f07nrc).

2 Specification

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

3 Description

To solve a complex symmetric system of linear equations $AX = B$, this function must be preceded by a call to nag_zsytrf (f07nrc) which computes the Bunch–Kaufman factorization of A .

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 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**, then $A = PUDU^T P^T$, where U is upper triangular;
 if **uplo** = **Nag_Lower**, then $A = PLDL^T P^T$, 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** ≥ 0 .
- 5: **a**[*dim*] – const Complex *Input*
Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$.
On entry: details of the factorization of A , as returned by nag_zsytrf (f07nrc).
- 6: **pda** – Integer *Input*
On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix in the array **a**.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.
- 7: **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_zsytrf (f07nrc).
- 8: **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 **b**[($j - 1$) $\times$ **pdb** + $i - 1$] and if **order** = **Nag_RowMajor**, the (i, j) th element of the matrix B is stored in **b**[($i - 1$) $\times$ **pdb** + $j - 1$].
On entry: the n by r right-hand side matrix B .
On exit: the n by r solution matrix X .
- 9: **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**, **pdb** $\geq \max(1, \mathbf{n})$;
if **order** = **Nag_RowMajor**, **pdb** $\geq \max(1, \mathbf{nrhs})$.
- 10: **fail** – NagError * *Output*
The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle \text{value} \rangle$.
Constraint: **n** ≥ 0 .
On entry, **nrhs** = $\langle \text{value} \rangle$.
Constraint: **nrhs** ≥ 0 .
On entry, **pda** = $\langle \text{value} \rangle$.
Constraint: **pda** > 0 .
On entry, **pdb** = $\langle \text{value} \rangle$.
Constraint: **pdb** > 0 .

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.
 Constraint: **pda** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = $\langle value \rangle$, **n** = $\langle value \rangle$.
 Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = $\langle value \rangle$, **nrhs** = $\langle value \rangle$.
 Constraint: **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**, $|E| \leq c(n)\epsilon P|U||D||U^T|P^T$;

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 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

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

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_zsyrrfs` (f07nvc), and an estimate for $\kappa_\infty(A)$ ($= \kappa_1(A)$) can be obtained by calling `nag_zsycon` (f07nuc).

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_zsyrrfs` (f07nvc) to refine the solution and return an error estimate.

The real analogue of this function is `nag_dsytrs` (f07mec).

9 Example

To solve 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 and must first be factorized by nag_zsytrf (f07nrc).

9.1 Program Text

```

/* nag_zsytrs (f07nsc) 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 i, j, n, nrhs, pda, pdb;
    Integer exit_status=0;
    Nag_UploType uplo_enum;

    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    Complex *a=0, *b=0;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
#define B(I,J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
#define B(I,J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07nsc Example Program Results\n\n");

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

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

```

```

    }

    /* Read A and B from data file */

    Vscanf(" ' %ls '%*[\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(i,j).re, &A(i,j).im);
        }
        Vscanf("%*[\n] ");
    }
    else
    {
        for (i = 1; i <= n; ++i)
        {
            for (j = 1; j <= i; ++j)
                Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(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 */
    f07nrc(order, uplo_enum, n, a, pda, ipiv, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07nrc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Compute solution */
    f07nsc(order, uplo_enum, n, nrhs, a, pda, ipiv, b, pdb,
        &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07nsc.\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 (a) NAG_FREE(a);
    if (b) NAG_FREE(b);
    return exit_status;
}

```

9.2 Program Data

f07nsc Example Program Data

```

    4  2                                     :Values of N and NRHS
    'L'                                     :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

```

9.3 Program Results

f07nsc 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)

```
