

NAG C Library Function Document

nag_zpptrs (f07gsc)

1 Purpose

nag_zpptrs (f07gsc) solves a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by nag_zpptrf (f07grc), using packed storage.

2 Specification

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

3 Description

To solve a complex Hermitian positive-definite system of linear equations $AX = B$, this function must be preceded by a call to nag_zpptrf (f07grc) which computes the Cholesky factorization of A using packed storage. The solution X is computed by forward and backward substitution.

If **uplo** = **Nag_Upper**, $A = U^H U$, where U is upper triangular; the solution X is computed by solving $U^H Y = B$ and then $UX = Y$.

If **uplo** = **Nag_Lower**, $A = LL^H$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^H X = Y$.

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 whether A has been factorized as $U^H U$ or LL^H as follows:
 if **uplo** = **Nag_Upper**, $A = U^H U$, where U is upper triangular;
 if **uplo** = **Nag_Lower**, $A = LL^H$, 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: the Cholesky factor of A stored in packed form, as returned by nag_zpptrf (f07grc).
- 6: **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 .
- 7: **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**, $\mathbf{pdb} \geq \max(1, \mathbf{nrhs})$.
- 8: **fail** – NagError * *Output*
The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

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

On entry, $\mathbf{pdb} = \langle value \rangle$, $\mathbf{nrhs} = \langle value \rangle$.
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**, $|E| \leq c(n)\epsilon|U^H||U|$;

if **uplo** = **Nag_Lower**, $|E| \leq c(n)\epsilon|L||L^H|$,

$c(n)$ 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_zpprfs` (f07gvc), and an estimate for $\kappa_\infty(A)$ ($= \kappa_1(A)$) can be obtained by calling `nag_zppcon` (f07guc).

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_zpprfs` (f07gvc) to refine the solution and return an error estimate.

The real analogue of this function is `nag_dpptrs` (f07gec).

9 Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.93 & - & 6.14i & 1.48 & + & 6.58i \\ 6.17 & + & 9.42i & 4.65 & - & 4.75i \\ -7.17 & - & 21.83i & -4.91 & + & 2.29i \\ 1.99 & - & 14.38i & 7.64 & - & 10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by `nag_zpptrf` (f07grc).

9.1 Program Text

```
/* nag_zpptrs (f07gsc) 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 */
    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("f07gsc 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 ( !(ap = NAG_ALLOC(ap_len, 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_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 */
f07grc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07grc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
f07gsc(order, uplo_enum, n, nrhs, ap, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07gsc.\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 (ap) NAG_FREE(ap);
if (b) NAG_FREE(b);
return exit_status;
}

```

9.2 Program Data

f07gsc Example Program Data

```

4 2                                     :Values of N and NRHS
'L'                                     :Value of UPLO
(3.23, 0.00)
(1.51, 1.92) ( 3.58, 0.00)
(1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
(0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix A
( 3.93, -6.14) ( 1.48, 6.58)
( 6.17, 9.42) ( 4.65, -4.75)
(-7.17,-21.83) (-4.91, 2.29)
( 1.99,-14.38) ( 7.64,-10.79)           :End of matrix B

```

9.3 Program Results

f07gsc Example Program Results

```

Solution(s)
1          1          2
1 ( 1.0000,-1.0000) (-1.0000, 2.0000)

```

```
2  (-0.0000, 3.0000)  ( 3.0000,-4.0000)
3  (-4.0000,-5.0000) (-2.0000, 3.0000)
4  ( 2.0000, 1.0000)  ( 4.0000,-5.0000)
```
