NAG C Library Function Document

nag_zpotrs (f07fsc)

1 Purpose

nag_zpotrs (f07fsc) solves a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, \( AX = B \), where \( A \) has been factorized by nag_zpotrf (f07frc).

2 Specification

```c
void nag_zpotrs (Nag_OrderType order, Nag_UploType uplo, Integer n, Integer nrhs,
               const Complex a[], Integer pda, 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_zpotrf (f07frc) which computes the Cholesky factorization of \( A \). The solution \( X \) is computed by forward and backward substitution.

If \( \text{uplo} = \text{Nag_Upper} \), \( A = U^H U \), where \( U \) is upper triangular; the solution \( X \) is computed by solving \( U^H Y = B \) and then \( UX = Y \).

If \( \text{uplo} = \text{Nag_Lower} \), \( A = LL^H \), where \( L \) is lower triangular; the solution \( X \) is computed by solving \( LY = B \) and then \( L^H X = Y \).

4 References


5 Parameters

1: \( \text{order} \) – Nag_OrderType

On entry: the \text{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 \text{order} = \text{Nag_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{uplo} \) – Nag_UploType

On entry: indicates whether \( A \) has been factorized as \( U^H U \) or \( LL^H \) as follows:

- if \( \text{uplo} = \text{Nag_Upper} \), then \( A = U^H U \), where \( U \) is upper triangular;
- if \( \text{uplo} = \text{Nag_Lower} \), then \( A = LL^H \), where \( L \) is lower triangular.

Constraint: \( \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

3: \( n \) – Integer

On entry: \( n \), the order of the matrix \( A \).

Constraint: \( n \geq 0 \).
4: \( \text{nrhs} \) – Integer  
\( \text{Input} \)  
On entry: \( r \), the number of right-hand sides.  
Constraint: \( \text{nrhs} \geq 0. \)

5: \( a[i] \) – const Complex  
\( \text{Input} \)  
Note: the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \text{max}(1, \text{pda} \times \text{n}) \).  
On entry: the Cholesky factor of \( A \), as returned by nag_zpotrf (f07frc).

6: \( \text{pda} \) – Integer  
\( \text{Input} \)  
On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) of the matrix in the array \( a \).  
Constraint: \( \text{pda} \geq \text{max}(1, n) \).

7: \( b[i] \) – Complex  
\( \text{Input/Output} \)  
Note: the dimension, \( \text{dim} \), of the array \( b \) must be at least \( \text{max}(1, \text{pdb} \times \text{nrhs}) \) when \( \text{order} = \text{Nag\_ColMajor} \) and at least \( \text{max}(1, \text{pdb} \times \text{n}) \) when \( \text{order} = \text{Nag\_RowMajor} \).  
If \( \text{order} = \text{Nag\_ColMajor} \), the \((i, j)\)th element of the matrix \( B \) is stored in \( b[(j-1) \times \text{pdb} + i - 1] \) and if \( \text{order} = \text{Nag\_RowMajor} \), the \((i, j)\)th element of the matrix \( B \) is stored in \( b[(i-1) \times \text{pdb} + j - 1] \).  
On entry: the \( n \) by \( r \) right-hand side matrix \( B \).  
On exit: the \( n \) by \( r \) solution matrix \( X \).

8: \( \text{pdb} \) – Integer  
\( \text{Input} \)  
On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( b \).  
Constraints:  
- if \( \text{order} = \text{Nag\_ColMajor} \), \( \text{pdb} \geq \text{max}(1, n) \);  
- if \( \text{order} = \text{Nag\_RowMajor} \), \( \text{pdb} \geq \text{max}(1, \text{nrhs}) \).

9: \( \text{fail} \) – NagError *  
\( \text{Output} \)  
The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT  
On entry, \( n = \langle \text{value} \rangle \).  
Constraint: \( n \geq 0. \)  
On entry, \( \text{nrhs} = \langle \text{value} \rangle \).  
Constraint: \( \text{nrhs} \geq 0. \)  
On entry, \( \text{pda} = \langle \text{value} \rangle \).  
Constraint: \( \text{pda} > 0. \)  
On entry, \( \text{pdb} = \langle \text{value} \rangle \).  
Constraint: \( \text{pdb} > 0. \)

NE_INT_2  
On entry, \( \text{pda} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).  
Constraint: \( \text{pda} \geq \text{max}(1, n) \).  
On entry, \( \text{pdb} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).  
Constraint: \( \text{pdb} \geq \text{max}(1, n) \).
On entry, \( \text{pdb} = \text{(value)} \), \( \text{nrhs} = \text{(value)} \).
Constraint: \( \text{pdb} \geq \max(1, \text{nrhs}) \).

**NE_ALLOC_FAIL**
Memory allocation failed.

**NE_BAD_PARAM**
On entry, parameter \( \text{(value)} \) 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 \( \text{uplo} = \text{Nag_Upper} \), \( |E| \leq c(n)\epsilon|U^H||U| \);
- if \( \text{uplo} = \text{Nag_Lower} \), \( |E| \leq c(n)\epsilon|L||L^H| \),

\( c(n) \) 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 \( \text{cond}(A,x) = \|A^{-1}\| |A| |x| \|/\|\|x\|| \leq \text{cond}(A) = \|A^{-1}\| |A| \leq \kappa_\infty(A) \). 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_zporfs (f07fvc), and an estimate for \( \kappa_\infty(A) (= \kappa_1(A)) \) can be obtained by calling nag_zpocon (f07fuc).

### 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_zporfs (f07fvc) to refine the solution and return an error estimate.

The real analogue of this function is nag_dpotrs (f07fec).

### 9 Example
To solve the system of equations \( AX = B \), where

\[
A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 1.41i & 4.29 + 0.00i \end{pmatrix}
\]

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 and must first be factorized by nag_zpotrf (f07frc).
9.1 Program Text

/* nag_zpotrs (f07fsc) 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 */
    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("f07fsc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^
] ");
    Vscanf("%ld%ld%*[^
] ", &n, &nrhs);

    pda = n;
    pdb = n;
    #else
    pda = n;
    pdb = nrhs;
    #endif

    /* Allocate memory */
    if (!(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(" ' %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");

    END:
    return exit_status;
}
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 */
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07frc.\n\n", fail.message);
  exit_status = 1;
goto END;
}

/* Compute solution */
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07fsc.\n\n", fail.message);
  exit_status = 1;
goto END;
}

/* Print solution */
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04dbc.\n\n", fail.message);
  exit_status = 1;
goto END;
}

END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
return exit_status;

9.2 Program Data

f07fsc Example Program Data

4 2  
'L'  :Values of N and NRHS  
(3.23, 0.00)  
(1.51, 1.92) ( 3.58, 0.00)  

[NP3645/7]  

f07fsc.5
9.3 Program Results

f07fsc Example Program Results

Solution(s)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 1.0000, -1.0000)</td>
<td>(-1.0000, 2.0000)</td>
</tr>
<tr>
<td>2</td>
<td>(-0.0000, 3.0000)</td>
<td>( 3.0000, -4.0000)</td>
</tr>
<tr>
<td>3</td>
<td>(-4.0000, -5.0000)</td>
<td>(-2.0000, 3.0000)</td>
</tr>
<tr>
<td>4</td>
<td>( 2.0000, 1.0000)</td>
<td>( 4.0000, -5.0000)</td>
</tr>
</tbody>
</table>