NAG C Library Function Document

nag_zhetrs (f07msc)

1 Purpose

nag_zhetrs (f07msc) solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by nag_zhetrf (f07mrc).

2 Specification

```
void nag_zhetrs (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 Hermitian indefinite system of linear equations $AX = B$, this function must be preceded by a call to nag_zhetrf (f07mrc) which computes the Bunch–Kaufman factorization of A.

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

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 $\max(1, \textbf{pda} \times \textbf{n})$.

On entry: details of the factorization of A, as returned by nag_zhetrf (f07mrc).

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** $\geq \max(1, \textbf{n})$.

7: **ipiv**[dim] – const Integer *Input*

Note: the dimension, dim, of the array **ipiv** must be at least $\max(1, \textbf{n})$.

On entry: details of the interchanges and the block structure of D, as returned by nag_zhetrf (f07mrc).

8: **b**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **b** must be at least $\max(1, \textbf{pdb} \times \textbf{nrhs})$ when **order** = **Nag_ColMajor** and at least $\max(1, \textbf{pdb} \times \textbf{n})$ when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix B is stored in $\textbf{b}[(j-1) \times \textbf{pdb} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix B is stored in $\textbf{b}[(i-1) \times \textbf{pdb} + j - 1]$.

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** $\geq \max(1, \textbf{n})$;
if **order** = **Nag_RowMajor**, **pdb** $\geq \max(1, \textbf{nrhs})$.

10: **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: **n** ≥ 0.

On entry, **nrhs** = $\langle$*value*$\rangle$.
Constraint: **nrhs** ≥ 0.

On entry, **pda** = $\langle$*value*$\rangle$.
Constraint: **pda** > 0.

On entry, **pdb** = $\langle$*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**, $|E| \leq c(n)\epsilon P|U|\,|D|\,|U^H|P^T$;

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 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

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

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

Forward and backward error bounds can be computed by calling nag_zherfs (f07mvc), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling nag_zhecon (f07muc).

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_zherfs (f07mvc) 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

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}$$

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 and must first be factorized by nag_zhetrf (f07mrc).

9.1 Program Text

```
/* nag_zhetrs (f07msc) 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("f07msc 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(" ' %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(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 */
  f07mrc(order, uplo_enum, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07mrc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Compute solution */
  f07msc(order, uplo_enum, n, nrhs, a, pda, ipiv, b, pdb,
         &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07msc.\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

```
f07msc 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

```
f07msc 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)
```