NAG 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

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

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

3 Description

nag_zpptrs (f07gsc) is used to solve a complex Hermitian positive definite system of linear equations \( AX = B \), the 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 \( \text{uplo} = \text{Nag Upper} \), \( A = U^H U \), 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 Arguments

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

\( \text{On entry:} \) the \( \text{order} \) 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 \( \text{order} = \text{Nag RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\( \text{Constraint:} \) \( \text{order} = \text{Nag RowMajor} \) or \( \text{Nag ColMajor} \).

2: \( \text{uplo} \) – Nag_UploType

\( \text{Input} \)

\( \text{On entry:} \) specifies how \( A \) has been factorized.

\( \text{uplo} = \text{Nag Upper} \)

\( A = U^H U \), where \( U \) is upper triangular.
uplo = Nag_Lower
    A = LL^H, where L is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.

3:  n – Integer
    On entry: n, the order of the matrix A.
    Constraint: n ≥ 0.

4:  nrhs – Integer
    On entry: r, the number of right-hand sides.
    Constraint: nrhs ≥ 0.

5:  ap[dim] – const Complex
    Note: the dimension, dim, of the array ap must be at least max(1, n × (n + 1)/2).
    On entry: the Cholesky factor of A stored in packed form, as returned by nag_zpptrf (f07grc).

6:  b[dim] – Complex
    Note: the dimension, dim, of the array b must be at least max(1, pdb × nrhs) when order = Nag_ColMajor;
    max(1, n × pdb) when order = Nag_RowMajor.
    The (i,j)th element of the matrix B is stored in
    b[(j - 1) × pdb + i - 1] when order = Nag_ColMajor;
    b[(i - 1) × 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.

7:  pdb – Integer
    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 ≥ max(1, n);
    if order = Nag_RowMajor, pdb ≥ max(1, nrhs).

8:  fail – NagError *
    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 <value> had an illegal value.
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, \( \text{pdb} = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} > 0 \).

On entry, \( \text{pdb} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} \leq \max(1, n) \).

On entry, \( \text{pdb} = \langle \text{value} \rangle \) and \( \text{nrhs} = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} \leq \max(1, \text{nrhs}) \).

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.

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.

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 \( \text{uplo} = \text{NagUpper} \), \( |E| \leq c(n)\epsilon|U|\|L|\|U|\|U\|\);  
- if \( \text{uplo} = \text{NagLower} \), \( |E| \leq c(n)\epsilon|L|\|L|\|L\|\).

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

Forward and backward error bounds can be computed by calling \( \text{nag_zzprfs} \) (f07gvc), and an estimate for \( \kappa_\infty(A) \) \( (= \kappa_1(A)) \) can be obtained by calling \( \text{nag_zppcon} \) (f07guc).

\( \text{nag_zpptrs} \) (f07gsc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\( \text{nag_zpptrs} \) (f07gsc) 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_zpprfs (f07gvc) to refine the solution and return an error estimate.

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

10 Example

This example solves 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).

10.1 Program Text

/* nag_zpptrs (f07gsc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 7, 2001. */

#include <stdio.h>
#include <nag.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;
    Nag_OrderType order;
    /* Arrays */
    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]
    #endif

    ...
```c
#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_zpptrs (f07gsc) Example Program Results\n\n");

/* Skip heading in data file */
#if defined _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#if defined _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &nrhs);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &nrhs);
#endif
    ap_len = n * (n + 1)/2;
    if (NAG_COLUMN_MAJOR
        pdb = n;
    )
    else
        pdb = nrhs;
#endif

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

/* Read A and B from data file */
#if defined _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)
            {
                if defined _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);
            }
            if defined _WIN32
                scanf_s("%*[\n] ");
            else
                scanf("%*[\n] ");
        }
    }
```

Mark 25

f07gsc.5
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_zpptrf (f07gsc). */
    /* Cholesky factorization of complex Hermitian */
    /* positive-definite matrix, packed storage */
    /* */
    nag_zpptrf(order, uplo, n, ap, &fail);
    if (fail.code != NE_NOERROR)
      {
        printf("Error from nag_zpptrf (f07gsc).
%s
", fail.message);
        exit_status = 1;
        goto END;
      }
    /* Compute solution */
    /* nag_zpgrpc (f07gsc). */
    /* Solution of complex Hermitian positive-definite system of */
    /* linear equations, multiple right-hand sides, matrix */
    /* already factorized by nag_zpptrf (f07gsc), packed storage */
    /* */
    nag_zpptrs(order, uplo, n, nrhs, ap, b, pdb, &fail);
    if (fail.code != NE_NOERROR)
      {
        printf("Error from nag_zpptrs (f07gsc).
%s
", 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, NagBracketForm, "%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).
%s
", fail.message);
        exit_status = 1;
        goto END;
      }
  }
fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ap);
NAG_FREE(b);
return exit_status;
}

10.2 Program Data

nag_zpptrs (f07gsc) Example Program Data

<table>
<thead>
<tr>
<th>Values of n and nrhs</th>
<th>:Value of uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Nag_Lower</td>
</tr>
<tr>
<td>(3.23, 0.00)</td>
<td></td>
</tr>
<tr>
<td>(1.51, 1.92)</td>
<td>(3.58, 0.00)</td>
</tr>
<tr>
<td>(1.90, -0.84)</td>
<td>(-0.23, -1.11)</td>
</tr>
<tr>
<td>(0.42, -2.50)</td>
<td>(4.09, 0.00)</td>
</tr>
<tr>
<td>(-1.18, -1.37)</td>
<td>(2.33, 0.14)</td>
</tr>
<tr>
<td>(3.93, -6.14)</td>
<td>(4.29, 0.00)</td>
</tr>
<tr>
<td>(6.17, 9.42)</td>
<td>(1.48, 6.58)</td>
</tr>
<tr>
<td>(-7.17, -21.83)</td>
<td>(-4.91, 2.29)</td>
</tr>
<tr>
<td>(1.99, -14.38)</td>
<td>(7.64, -10.79)</td>
</tr>
</tbody>
</table>

:End of matrix A

:End of matrix B

10.3 Program Results

nag_zpptrs (f07gsc) Example Program Results

Solution(s)

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