NAG Library Function Document

nag_zpttrs (f07jsc)

1 Purpose

nag_zpttrs (f07jsc) computes the solution to a complex system of linear equations $AX = B$, where $A$ is an $n$ by $n$ Hermitian positive definite tridiagonal matrix and $X$ and $B$ are $n$ by $r$ matrices, using the $LDL^H$ factorization returned by nag_zpttrf (f07jrc).

2 Specification

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

void nag_zpttrs (Nag_OrderType order, Nag_UploType uplo, Integer n,
                Integer nrhs, const double d[], const Complex e[], Complex b[],
                Integer pdb, NagError *fail)
```

3 Description

nag_zpttrs (f07jsc) should be preceded by a call to nag_zpttrf (f07jrc), which computes a modified Cholesky factorization of the matrix $A$ as

$$A = LDL^H,$$

where $L$ is a unit lower bidiagonal matrix and $D$ is a diagonal matrix, with positive diagonal elements. nag_zpttrs (f07jsc) then utilizes the factorization to solve the required equations. Note that the factorization may also be regarded as having the form $U^H DU$, where $U$ is a unit upper bidiagonal matrix.

4 References


5 Arguments

1:  
   `order` – Nag_OrderType
   
   *Input*
   
   On entry: the `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 `order = Nag_RowMajor`. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   
   Constraint: `order = Nag_RowMajor` or `Nag_ColMajor`.

2:  
   `uplo` – Nag_UploType
   
   *Input*
   
   On entry: specifies the form of the factorization as follows:
   
   `uplo = Nag_Upper`
   
   $A = U^H DU$.
   
   `uplo = Nag_Lower`
   
   $A = LDL^H$.
   
   Constraint: `uplo = Nag_Upper` or `Nag_Lower`. 
3: \( n \) – Integer

\textit{Input}

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

\textit{Constraint:} \( n \geq 0 \).

4: \( \text{nrhs} \) – Integer

\textit{Input}

\textit{On entry:} \( r_h \), the number of right-hand sides, i.e., the number of columns of the matrix \( B \).

\textit{Constraint:} \( \text{nrhs} \geq 0 \).

5: \( d[\text{dim}] \) – const double

\textit{Input}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( d \) must be at least \( \max(1, n) \).

\textit{On entry:} must contain the \( n \) diagonal elements of the diagonal matrix \( D \) from the \( LDL^H \) or \( U^H DU \) factorization of \( A \).

6: \( e[\text{dim}] \) – const Complex

\textit{Input}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( e \) must be at least \( \max(1, n-1) \).

\textit{On entry:} if \( \text{uplo} = \text{Nag} \_ \text{Upper} \), \( e \) must contain the \( (n-1) \) superdiagonal elements of the unit upper bidiagonal matrix \( U \) from the \( U^H DU \) factorization of \( A \).

If \( \text{uplo} = \text{Nag} \_ \text{Lower} \), \( e \) must contain the \( (n-1) \) subdiagonal elements of the unit lower bidiagonal matrix \( L \) from the \( LDL^H \) factorization of \( A \).

7: \( b[\text{dim}] \) – Complex

\textit{Input/Output}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( b \) must be at least

\( \max(1, \text{pdb} \times \text{nrhs}) \) when \( \text{order} = \text{Nag} \_ \text{ColMajor} \);

\( \max(1, n \times \text{pdb}) \) when \( \text{order} = \text{Nag} \_ \text{RowMajor} \).

The \( (i,j) \)th element of the matrix \( B \) is stored in

\( b[(j-1) \times \text{pdb} + i - 1] \) when \( \text{order} = \text{Nag} \_ \text{ColMajor} \);

\( b[(i-1) \times \text{pdb} + j - 1] \) when \( \text{order} = \text{Nag} \_ \text{RowMajor} \).

\textit{On entry:} the \( n \) by \( r_h \) matrix of right-hand sides \( B \).

\textit{On exit:} the \( n \) by \( r_h \) solution matrix \( X \).

8: \( \text{pdb} \) – Integer

\textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( b \).

\textit{Constraints:}

\( \text{if} \ \text{order} = \text{Nag} \_ \text{ColMajor}, \ \text{pdb} \geq \max(1, n) \);

\( \text{if} \ \text{order} = \text{Nag} \_ \text{RowMajor}, \ \text{pdb} \geq \max(1, \text{nrhs}) \).

9: \( \text{fail} \) – NagError*

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \ Error Indicators and Warnings

\textbf{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 \langle value \rangle had an illegal value.

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

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

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 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.

NE_NO_LICENCE
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.

7 Accuracy
The computed solution for a single right-hand side, \( \hat{x} \), satisfies an equation of the form
\[
(A + E)\hat{x} = b,
\]
where
\[
\|E\|_1 = O(\epsilon)\|A\|_1
\]
and \( \epsilon \) is the machine precision. An approximate error bound for the computed solution is given by
\[
\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\|E\|_1\|A\|_1,
\]
where \( \kappa(A) = \|A^{-1}\|_1\|A\|_1 \), the condition number of \( A \) with respect to the solution of the linear
equations. See Section 4.4 of Anderson et al. (1999) for further details.
Following the use of this function nag_zptcon (f07juc) can be used to estimate the condition number of
\( A \) and nag_zptrfs (f07jvc) can be used to obtain approximate error bounds.

8 Parallelism and Performance
nag_zpttrs (f07jsc) is not threaded by NAG in any implementation.
nag_zpttrs (f07jsc) 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 floating-point operations required to solve the equations $AX = B$ is proportional to $nr$.

The real analogue of this function is nag_dpttrs (f07jec).

10 Example

This example solves the equations

$$AX = B,$$

where $A$ is the Hermitian positive definite tridiagonal matrix

$$A = \begin{pmatrix}
16.0 & 16.0 - 16.0i & 0 & 0 \\
16.0 + 16.0i & 41.0 & 18.0 + 9.0i & 0.0 \\
0 & 18.0 - 9.0i & 46.0 & 1.0 + 4.0i \\
0 & 0 & 1.0 - 4.0i & 21.0
\end{pmatrix}$$

and

$$B = \begin{pmatrix}
64.0 + 16.0i & -16.0 - 32.0i \\
93.0 + 62.0i & 61.0 - 66.0i \\
78.0 - 80.0i & 71.0 - 74.0i \\
14.0 - 27.0i & 35.0 + 15.0i
\end{pmatrix}.$$
INIT_FAIL(fail);

printf("nag_zpttrs (f07jsc) Example Program Results\n\n");
/* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[\n]");
#else
  scanf("%*[\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n, &nrhs);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n, &nrhs);
#endif
if (n < 0 || nrhs < 0)
{
  printf("Invalid n or nrhs\n");
  exit_status = 1;
  goto END;
}
/* Allocate memory */
if (!(b = NAG_ALLOC(n * nrhs, Complex)) || /* Allocate memory */
  !(e = NAG_ALLOC(n-1, Complex)) || /* Allocate memory */
  !(d = NAG_ALLOC(n, double)))
{
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}
#endif NAG_COLUMN_MAJOR
pdb = n;
#else
pdb = nrhs;
#endif
/* Read the upper bidiagonal part of the tridiagonal matrix A from */
/* data file */
#ifdef _WIN32
  for (i = 0; i < n - 1; ++i) scanf_s(" ( %lf , %lf )", &e[i].re, &e[i].im);
#else
  for (i = 0; i < n - 1; ++i) scanf(" ( %lf , %lf )", &e[i].re, &e[i].im);
#endif
#ifdef _WIN32
  scanf_s("%*[\n]");
#else
  scanf("%*[\n]");
#endif
#ifdef _WIN32
  for (i = 0; i < n; ++i) scanf("%lf", &d[i]);
#else
  for (i = 0; i < n; ++i) scanf("%lf", &d[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[\n]");
#else
  scanf("%*[\n]");
#endif
/* Read the right hand matrix B */
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

/* Read the right hand matrix B */
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
scanf("%*[\n"]);
#endif

/* Factorize the tridiagonal matrix A using nag_zpttrf (f07jrc). */
nag_zpttrf(n, d, e, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zpttrf (f07jrc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Solve the equations AX = B using nag_zpttrs (f07jsc). */
nag_zpttrs(order, Nag_Upper, n, nrhs, d, e, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zpttrs (f07jsc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print the solution using nag_gen_complx_mat_print_comp (x04dbc). */
fflush(stdout);
nag_gen_complx_mat_print_comp(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)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(b);
NAG_FREE(e);
NAG_FREE(d);
return exit_status;
#endif

10.2 Program Data

nag_zpttrs (f07jsc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>nrhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>16.0, -16.0</td>
<td>18.0, 9.0</td>
</tr>
<tr>
<td>16.0</td>
<td>41.0</td>
</tr>
<tr>
<td>64.0, 16.0</td>
<td>-16.0, -32.0</td>
</tr>
<tr>
<td>93.0, 62.0</td>
<td>61.0, -66.0</td>
</tr>
<tr>
<td>78.0, -80.0</td>
<td>71.0, -74.0</td>
</tr>
<tr>
<td>14.0, -27.0</td>
<td>35.0, 15.0</td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_zpttrs (f07jsc) Example Program Results

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