NAG Library Function Document

nag_zptsv (f07jnc)

1 Purpose
nag_zptsv (f07jnc) 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.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_zptsv (Nag_OrderType order, Integer n, Integer nrhs, double d[],
    Complex e[], Complex b[], Integer pdb, NagError *fail)
```

3 Description

nag_zptsv (f07jnc) factors \( A \) as \( A = LDL^H \). The factored form of \( A \) is then used to solve the system of equations.

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: \( \text{n} \) – Integer

\( \text{Input} \)

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

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

3: \( \text{nrhs} \) – Integer

\( \text{Input} \)

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

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

4: \( \text{d}[/\text{dim}] \) – double

\( \text{Input/Output} \)

\( \text{Note:} \) the dimension, \( \text{dim} \), of the array \( \text{d} \) must be at least \( \max(1, \text{n}) \).
On entry: the \( n \) diagonal elements of the tridiagonal matrix \( A \).

On exit: the \( n \) diagonal elements of the diagonal matrix \( D \) from the factorization \( A = LDL^H \).

5: \( \mathbf{e}[\text{dim}] \) – Complex

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

On entry: the \( (n - 1) \) subdiagonal elements of the tridiagonal matrix \( A \).

On exit: the \( (n - 1) \) subdiagonal elements of the unit bidiagonal factor \( L \) from the \( LDL^H \) factorization of \( A \). (\( e \) can also be regarded as the superdiagonal of the unit bidiagonal factor \( U \) from the \( U^H D U \) factorization of \( A \).)

6: \( \mathbf{b}[\text{dim}] \) – Complex

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

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

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

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

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

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

On entry: the \( n \) by \( r \) right-hand side matrix \( B \).

On exit: if \( \text{fail.code} = \text{NE_NOERROR} \), the \( n \) by \( r \) solution matrix \( X \).

7: \( \text{pdb} \) – Integer

**Input**

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

**Constraints:**

- if \( \text{order} = \text{Nag ColMajor} \), \( \text{pdb} \geq \max(1, n) \);
- if \( \text{order} = \text{Nag RowMajor} \), \( \text{pdb} \geq \max(1, \text{nrhs}) \).

8: \( \text{fail} \) – NagError *

**Input/Output**

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 \( \langle \text{value} \rangle \) had an illegal value.

**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, \( \text{pdb} = \langle \text{value} \rangle \).

Constraint: \( \text{pdb} > 0 \).
NE_INT_2

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

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

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_MAT_NOT_POS_DEF

The leading minor of order \( \langle \text{value} \rangle \) is not positive definite, and the solution has not been computed. The factorization has not been completed unless \( \text{n} = \langle \text{value} \rangle \).

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

\[
\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\frac{\|E\|_1}{\|A\|_1},
\]

where \( \kappa(A) = \|A^{-1}\|_1 \|A\|_1 \) is 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.

\text{f07jnc} \text{ is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, f04cgc solves } Ax = b \text{ and returns a forward error bound and condition estimate. f04cgc calls f07jnc to solve the equations.}

8 Parallelism and Performance

\text{f07jnc} \text{ is not threaded by NAG in any implementation.}

\text{f07jnc} \text{ 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 number of floating-point operations required for the factorization of $A$ is proportional to $n$, and the number of floating-point operations required for the solution of the equations is proportional to $nr$, where $r$ is the number of right-hand sides.

The real analogue of this function is nag_dptsv (f07jac).

10 Example

This example solves the equations

$$Ax = b,$$

where $A$ is the Hermitian positive definite tridiagonal matrix

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

and

$$b = \begin{pmatrix} 64.0 + 16.0i \\ 93.0 + 62.0i \\ 78.0 - 80.0i \\ 14.0 - 27.0i \end{pmatrix}.$$ 

Details of the $LDL^H$ factorization of $A$ are also output.

10.1 Program Text

/* nag_zptsv (f07jnc) Example Program. */
" * Copyright 2014 Numerical Algorithms Group. */
" * Mark 23, 2011. */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>

int main(void)
{
    /* Scalars */
    Integer exit_status = 0, i, j, n, nrhs, pdb;
    
    /* Arrays */
    Complex *b = 0, *e = 0;
    double *d = 0;
    
    /* Nag Types */
    NagError fail;
    Nag_OrderType order;
    
    #ifdef NAG_COLUMN_MAJOR
    #define B(I, J) b[(J-1)*pdb +I-1]
    #else
    #define B(I, J) b[(I-1)*pdb +J-1]
    #endif

    INIT_FAIL(fail);
    printf("nag_zptsv (f07jnc) Example Program Results
n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    #endif
}
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))
 || !(e = NAG_ALLOC(n-1, Complex))
 || !(d = NAG_ALLOC(n, double)))
{
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}
#ifdef NAG_COLUMN_MAJOR
  pdb = n;
#else
  pdb = nrhs;
#endif
/* Read the lower bidiagonal part of the tridiagonal matrix A and */
/* the right hand side b from data file */
#ifdef _WIN32
  for (i = 0; i < n; ++i) scanf_s("%lf", &d[i]);
#else
  for (i = 0; i < n; ++i) scanf("%lf", &d[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[\n]");
#else
  scanf("%*[\n]");
#endif
#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
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
/* Solve the equations Ax = b for x using nag_zptsv (f07jnc). */
  nag_zptsv(order, n, nrhs, d, e, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
printf("Error from nag_zptsv (f07jnc).\n", fail.message);
exit_status = 1;
goto END;
}

/* Print solution */
printf("Solution\n");
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        printf("(%8.4f, %8.4f)%s", B(i, j).re, B(i, j).im, j%4 == 0?"\n":" ");
    printf("\n");
}

/* Print details of factorization */
printf("\nDiagonal elements of the diagonal matrix D\n");
for (i = 0; i < n; ++i) printf("%7.4f%s", d[i], i%8 == 7?"\n":" ");

printf("\nSub-diagonal elements of the Cholesky factor L\n");
for (i = 0; i < n-1; ++i)
    printf("(%8.4f, %8.4f)%s", e[i].re, e[i].im, i%8 == 7?"\n":" ");

END:
NAG_FREE(b);
NAG_FREE(e);
NAG_FREE(d);
return exit_status;
}

10.2 Program Data

nag_zptsv (f07jnc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>nrhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>diagonal d</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.0 41.0 46.0 21.0</td>
</tr>
<tr>
<td>(16.0, 16.0) (18.0, -9.0) (1.0, -4.0)</td>
</tr>
<tr>
<td>(64.0, 16.0) (93.0, 62.0) (78.0, -80.0) (14.0, -27.0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sub-diagonal e</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0000, 1.0000)</td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_zptsv (f07jnc) Example Program Results

Solution

| 2.0000 1.0000 |
| 1.0000 1.0000 |
| 1.0000 -2.0000 |
| 1.0000 -1.0000 |

Diagonal elements of the diagonal matrix D

| 16.0000 9.0000 1.0000 4.0000 |

Sub-diagonal elements of the Cholesky factor L

| 1.0000 1.0000 2.0000 -1.0000 1.0000 -4.0000 |