

# NAG Library Function Document

## nag\_dptsv (f07jac)

### 1 Purpose

nag\_dptsv (f07jac) computes the solution to a real system of linear equations

$$AX = B,$$

where  $A$  is an  $n$  by  $n$  symmetric positive definite tridiagonal matrix, and  $X$  and  $B$  are  $n$  by  $r$  matrices.

### 2 Specification

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

### 3 Description

nag\_dptsv (f07jac) factors  $A$  as  $A = LDL^T$ . The factored form of  $A$  is then used to solve the system of equations.

### 4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia <http://www.netlib.org/lapack/lug>

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.  
*Constraint:* **order** = Nag\_RowMajor or Nag\_ColMajor.
- 2: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:* **n**  $\geq 0$ .
- 3: **nrhs** – Integer *Input*  
*On entry:*  $r$ , the number of right-hand sides, i.e., the number of columns of the matrix  $B$ .  
*Constraint:* **nrhs**  $\geq 0$ .
- 4: **d**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **d** must be at least  $\max(1, \mathbf{n})$ .

*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  $A = LDL^T$ .

5: **e**[*dim*] – double *Input/Output*

**Note:** the dimension, *dim*, of the array **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  $LDL^T$  factorization of  $A$ . (**e** can also be regarded as the superdiagonal of the unit bidiagonal factor  $U$  from the  $U^T D U$  factorization of  $A$ .)

6: **b**[*dim*] – double *Input/Output*

**Note:** the dimension, *dim*, of the array **b** must be at least

$\max(1, \mathbf{pdb} \times \mathbf{nrhs})$  when **order** = Nag\_ColMajor;

$\max(1, n \times \mathbf{pdb})$  when **order** = Nag\_RowMajor.

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

**b**[( $j - 1$ )  $\times$  **pdb** +  $i - 1$ ] when **order** = Nag\_ColMajor;

**b**[( $i - 1$ )  $\times$  **pdb** +  $j - 1$ ] when **order** = Nag\_RowMajor.

*On entry:* the  $n$  by  $r$  right-hand side matrix  $B$ .

*On exit:* if **fail.code** = NE\_NOERROR, the  $n$  by  $r$  solution matrix  $X$ .

7: **pdb** – Integer *Input*

*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**  $\geq \max(1, n)$ ;

if **order** = Nag\_RowMajor, **pdb**  $\geq \max(1, \mathbf{nrhs})$ .

8: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation 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**  $\geq \max(1, \mathbf{n})$ .

On entry, **pdb** =  $\langle value \rangle$  and **nrhs** =  $\langle value \rangle$ .

Constraint: **pdb**  $\geq \max(1, \mathbf{nrhs})$ .

**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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_MAT\_NOT\_POS\_DEF**

The leading minor of order  $\langle value \rangle$  is not positive definite, and the solution has not been computed. The factorization has not been completed unless **n** =  $\langle value \rangle$ .

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation 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  $\kappa(A) = \|A^{-1}\|_1 \|A\|_1$ , 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.

nag\_dptsvx (f07jbc) is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, nag\_real\_sym\_posdef\_tridiag\_lin\_solve (f04bgc) solves  $Ax = b$  and returns a forward error bound and condition estimate. nag\_real\_sym\_posdef\_tridiag\_lin\_solve (f04bgc) calls nag\_dptsv (f07jac) to solve the equations.

**8 Parallelism and Performance**

nag\_dptsv (f07jac) 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 complex analogue of this function is nag\_zptsv (f07jnc).

## 10 Example

This example solves the equations

$$Ax = b,$$

where  $A$  is the symmetric positive definite tridiagonal matrix

$$A = \begin{pmatrix} 4.0 & -2.0 & 0 & 0 & 0 \\ -2.0 & 10.0 & -6.0 & 0 & 0 \\ 0 & -6.0 & 29.0 & 15.0 & 0 \\ 0 & 0 & 15.0 & 25.0 & 8.0 \\ 0 & 0 & 0 & 8.0 & 5.0 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 6.0 \\ 9.0 \\ 2.0 \\ 14.0 \\ 7.0 \end{pmatrix}.$$

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

### 10.1 Program Text

```
/* nag_dptsv (f07jac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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;
    Nag_OrderType order;

    /* Arrays */
    double *b = 0, *d = 0, *e = 0;

    /* Nag Types */
    NagError fail;

#ifdef NAG_COLUMN_MAJOR
#define B(I, J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
#else
#define B(I, J) b[(I-1)*pdb + J - 1]
    order = Nag_RowMajor;
#endif
    INIT_FAIL(fail);

    printf("nag_dptsv (f07jac) 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
}
```

```

    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, double)) ||
        !(d = NAG_ALLOC(n, double)) || !(e = NAG_ALLOC(n - 1, 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", &e[i]);
#else
    for (i = 0; i < n - 1; ++i)
        scanf("%lf", &e[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    for (i = 1; i <= n; ++i)
#ifdef _WIN32
        for (j = 1; j <= nrhs; ++j)
            scanf_s("%lf", &B(i, j));
#else
        for (j = 1; j <= nrhs; ++j)
            scanf("%lf", &B(i, j));
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    /* Solve the equations Ax = b for x using nag_dptsv (f07jac). */
    nag_dptsv(order, n, nrhs, d, e, b, pdb, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dptsv (f07jac).\n%s\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("%11.4f%s", B(i, j), j % 7 == 0 ? "\n" : " ");
    printf("\n");
}

/* Print details of factorization */
printf("\nDiagonal elements of the diagonal matrix D\n");
for (i = 0; i < n; ++i)
    printf("%11.4f%s", d[i], i % 7 == 6 ? "\n" : " ");

printf("\n\nSubdiagonal elements of the Cholesky factor L\n");
for (i = 0; i < n - 1; ++i)
    printf("%11.4f%s", e[i], i % 7 == 6 ? "\n" : " ");
printf("\n");
END:
    NAG_FREE(b);
    NAG_FREE(d);
    NAG_FREE(e);

    return exit_status;
}

```

## 10.2 Program Data

```

nag_dptsv (f07jac) Example Program Data
   5      1      : n nrhs
   4.0   10.0   29.0   25.0   5.0 : diagonal d
  -2.0   -6.0   15.0    8.0      : sub-diagonal e
   6.0    9.0    2.0   14.0   7.0 : vector b

```

## 10.3 Program Results

```
nag_dptsv (f07jac) Example Program Results
```

```

Solution
   2.5000
   2.0000
   1.0000
  -1.0000
   3.0000

Diagonal elements of the diagonal matrix D
   4.0000    9.0000   25.0000   16.0000    1.0000

Subdiagonal elements of the Cholesky factor L
  -0.5000   -0.6667    0.6000    0.5000

```

---