

NAG C Library Function Document

nag_real_sym_posdef_tridiag_lin_solve (f04bgc)

1 Purpose

nag_real_sym_posdef_tridiag_lin_solve (f04bgc) 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. An estimate of the condition number of A and an error bound for the computed solution are also returned.

2 Specification

```
#include <nag.h>
#include <nagf04.h>

void nag_real_sym_posdef_tridiag_lin_solve (Nag_OrderType order, Integer n,
      Integer nrhs, double d[], double e[], double b[], Integer pdb, double *rcond,
      double *errbnd, NagError *fail)
```

3 Description

A is factorized as $A = LDL^T$, where L is a unit lower bidiagonal matrix and D is diagonal, and 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 URL: <http://www.netlib.org/lapack/lug>

Higham N J (2002) *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this argument.
Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.
- 2: **n** – Integer *Input*
On entry: the number of linear equations n , i.e., the order of the matrix A .
Constraint: $n \geq 0$.
- 3: **nrhs** – Integer *Input*
On entry: the number of right-hand sides r , i.e., the number of columns of the matrix B .
Constraint: **nrhs** ≥ 0 .
- 4: **d**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **d** must be at least $\max(1, n)$.
On entry: must contain the n diagonal elements of the tridiagonal matrix A .

On exit: if **fail.code** = **NE_NOERROR** or **NE_RCOND**, **d** is overwritten by the n diagonal elements of the diagonal matrix D from the LDL^T factorization of A .

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

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

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

On exit: if **fail.code** = **NE_NOERROR** or **NE_RCOND**, **e** is overwritten by the $(n - 1)$ subdiagonal elements of the unit lower bidiagonal matrix L from the LDL^T factorization of A . (**e** can also be regarded as the superdiagonal of the unit upper 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

$\max(1, \text{pdb} \times \text{nrhs})$ when **order** = **Nag_ColMajor**;
 $\max(1, \text{pdb} \times n)$ when **order** = **Nag_RowMajor**.

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

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

On entry: the n by r matrix of right-hand sides B .

On exit: if **fail.code** = **NE_NOERROR** or **NE_RCOND**, the n by r solution matrix X .

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

8: **rcond** – double * *Output*

On exit: if **fail.code** = **NE_NOERROR** or **NE_RCOND**, an estimate of the reciprocal of the condition number of the matrix A , computed as $\text{rcond} = 1 / (\|A\|_1 \|A^{-1}\|_1)$.

9: **errbnd** – double * *Output*

On exit: if **fail.code** = **NE_NOERROR** or **NE_RCOND**, an estimate of the forward error bound for a computed solution $\hat{x}$, such that $\|\hat{x} - x\|_1 / \|x\|_1 \leq \text{errbnd}$, where $\hat{x}$ is a column of the computed solution returned in the array **b** and x is the corresponding column of the exact solution X . If **rcond** is less than *machine precision*, then **errbnd** is returned as unity.

10: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle \text{value} \rangle$ had an illegal value.

NE_INT

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

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

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

NE_INT_2

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_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.

NE_POS_DEF

The principal minor of order $\langle value \rangle$ of the matrix A is not positive-definite. The factorization has not been completed and the solution could not be computed.

NE_RCOND

A solution has been computed, but **rcond** is less than *machine precision* so that the matrix A is numerically singular.

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 ϵ 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. `nag_real_sym_posdef_tridiag_lin_solve (f04bgc)` uses the approximation $\|E\|_1 = \epsilon \|A\|_1$ to estimate **errbnd**. See Section 4.4 of Anderson *et al.* (1999) for further details.

8 Further Comments

The total number of floating-point operations required to solve the equations $AX = B$ is proportional to nr . The condition number estimation requires $O(n)$ floating-point operations.

See Section 15.3 of Higham (2002) for further details on computing the condition number of tridiagonal matrices.

The complex analogue of `nag_real_sym_posdef_tridiag_lin_solve (f04bgc)` is `nag_herm_posdef_tridiag_lin_solve (f04cgc)`.

9 Example

To solve the equations

$$AX = B,$$

where A is the symmetric positive-definite tridiagonal matrix

$$A = \begin{pmatrix} 4.0 & -2.0 & 0 & 0 & 0 \\ -2.0 & 10.0 & -6.0 & 0 & 0 \\ 0 & -6.0 & 29.0 & 15.0 & 0 \\ 0 & 0 & 15.0 & 25.0 & 8.0 \\ 0 & 0 & 0 & 8.0 & 5.0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 6.0 & 10.0 \\ 9.0 & 4.0 \\ 2.0 & 9.0 \\ 14.0 & 65.0 \\ 7.0 & 23.0 \end{pmatrix}.$$

An estimate of the condition number of A and an approximate error bound for the computed solutions are also printed.

9.1 Program Text

```
/* nag_real_sym_posdef_tridiag_lin_solve (f04bgc) Example Program.
 *
 * Copyright 2004 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf04.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double errbnd, rcond;
    Integer exit_status, i, j, n, nrhs, pdb;

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

    /* Nag Types */
    NagError fail;
    Nag_OrderType order;

#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
#define b_ref(a_1,a_2) b[(a_2)*8 + a_1 - 9]

    exit_status = 0;
    INIT_FAIL(fail);

    Vprintf("nag_real_sym_posdef_tridiag_lin_solve (f04bgc) Example Program"
           " Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^\\n] ");
    Vscanf("%ld%ld%*[^\\n] ", &n, &nrhs);
    if (n>0 && nrhs>0)
    {
        /* Allocate memory */
        if ( !(b = NAG_ALLOC(n*nrhs, double)) ||
            !(d = NAG_ALLOC(n, double)) ||
```

```

        !(e = NAG_ALLOC(n-1, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
#ifdef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif
}
else
{
    Vprintf("%s\n", "n and/or nrhs too small");
    exit_status = 1;
    return exit_status;
}

/* Read A from data file */
for (i = 1; i<=n; ++i)
{
    Vscanf("%lf", &d[i-1]);
}
Vscanf("%*[^\\n] ");

for (i = 1; i<=n-1; ++i)
{
    Vscanf("%lf", &e[i-1]);
}
Vscanf("%*[^\\n] ");

/* Read B from data file */
for (i=1; i<= n; ++i)
{
    for (j=1; j<=nrhs; ++j)
    {
        Vscanf("%lf", &B(i,j));
    }
}
Vscanf("%*[^\\n] ");

/* Solve the equations AX = B for X */
/* nag_real_sym_posdef_tridiag_lin_solve (f04bgc).
 * Computes the solution and error-bound to a real symmetric
 * positive-definite tridiagonal system of linear equations
 */
nag_real_sym_posdef_tridiag_lin_solve(order, n, nrhs, d, e, b, pdb, &rcond,
                                     &errbnd, &fail);
if (fail.code == NE_NOERROR)
{
    /* Print solution, estimate of condition number and approximate */
    /* error bound */

    /* nag_gen_real_mat_print (x04cac).
     * Print real general matrix (easy-to-use)
     */
    nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                          nrhs, b, pdb, "Solution", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }
    Vprintf("\n%s\n%6s%9.1e\n", "Estimate of condition number", "",
            1.0/rcond);
    Vprintf("\n\n");
    Vprintf("%s\n%6s", "Estimate of error bound for computed solutions", "");
}

```

```

    Vprintf("%9.1e\n\n", errbnd);
}
else if (fail.code == NE_RCOND)
{
    /* Matrix A is numerically singular. Print estimate of */
    /* reciprocal of condition number and solution */
    Vprintf("\n");
    Vprintf("%s\n%6s%9.1e\n", "Estimate of reciprocal of condition number",
            "", rcond);
    Vprintf("\n\n");

    /* nag_gen_real_mat_print (x04cac), see above. */
    nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                          nrhs, b, pdb, "Solution", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }
}
else if (fail.code == NE_POS_DEF)
{
    Vprintf("%s%3ld%s\n\n", "The leading minor of order ",
            fail.errnum, " is not positive definite");
}
END:
if (b) NAG_FREE(b);
if (d) NAG_FREE(d);
if (e) NAG_FREE(e);
return exit_status;
}

```

9.2 Program Data

nag_real_sym_posdef_tridiag_lin_solve (f04bgc) Example Program Data

```

5      2                      :Values of N and NRHS

4.0  10.0  29.0  25.0   5.0 :End of diagonal D
-2.0  -6.0  15.0   8.0      :End of sub-diagonal E

6.0  10.0
9.0   4.0
2.0   9.0
14.0  65.0
7.0  23.0                      :End of matrix B

```

9.3 Program Results

nag_real_sym_posdef_tridiag_lin_solve (f04bgc) Example Program Results

Solution

	1	2
1	2.5000	2.0000
2	2.0000	-1.0000
3	1.0000	-3.0000
4	-1.0000	6.0000
5	3.0000	-5.0000

Estimate of condition number
1.1e+02

Estimate of error bound for computed solutions
1.2e-14