NAG C Library Function Document

nag_herm_posdef_tridiag_lin_solve (f04cgc)

1 Purpose

nag_herm_posdef_tridiag_lin_solve (f04cgc) 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. 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_herm_posdef_tridiag_lin_solve (Nag_OrderType order, Integer n,
     Integer nrhs, double d[], Complex e[], Complex b[], Integer pdb,
     double *rcond, double *errbnd, NagError *fail)
```

3 Description

A is factorized as $A = LDL^{\mathrm{H}}$, where L is a unit lower bidiagonal matrix and D is a real diagonal matrix, 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: $\mathbf{n} \geq 0$.

3: **nrhs** – Integer *Input*

On entry: the number of right-hand sides r, i.e., the number of columns of the matrix B.

Constraint: $\mathbf{nrhs} \geq 0$.

4: **d**[dim] – double *Input/Output*

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

On entry: 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^{H} factorization of A.

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

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

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^{H} factorization of A. (**e** can also be regarded as the conjugate of the superdiagonal of the unit upper bidiagonal factor U from the $U^{\mathrm{H}}DU$ factorization of A.)

6: **b**[*dim*] – Complex *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**;
$\max(1, \mathbf{pdb} \times \mathbf{n})$ when **order** = **Nag_RowMajor**.

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

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

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

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 $\mathbf{rcond} = 1/\left(\|A\|_1, \left\|A^{-1}\right\|_1\right)$.

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 $\|\hat{x} - x\|_1 / \|x\|_1 \leq$ **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

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

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

NE_INT_2

On entry,**pdb** $= \langle value \rangle$, $\mathbf{n} = \langle value \rangle$.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

$$\|E\|_1 = \mathrm{O}(\epsilon)\|A\|_1$$

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 $\kappa(A) = \left\|A^{-1}\right\|_1 \|A\|_1$, the condition number of A with respect to the solution of the linear equations. nag_herm_posdef_tridiag_lin_solve (f04cgc) 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 $\mathrm{O}(n)$ floating-point operations.

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

The real analogue of nag_herm_posdef_tridiag_lin_solve (f04cgc) is nag_real_sym_posdef_tridiag_lin_solve (f04bgc).

9 Example

To solve 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

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

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_herm_posdef_tridiag_lin_solve (f04cgc) 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 */
  char *clabs=0, *rlabs=0;
  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]
  order = Nag_ColMajor;
#else
#define B(I,J) b[(I-1)*pdb + J - 1]
  order = Nag_RowMajor;
#endif

  exit_status = 0;
  INIT_FAIL(fail);

  Vprintf("nag_herm_posdef_tridiag_lin_solve (f04cgc) 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 ( !(clabs = NAG_ALLOC(2, char)) ||
           !(rlabs = NAG_ALLOC(2, char)) ||
           !(b = NAG_ALLOC(n*nrhs, Complex)) ||
           !(e = NAG_ALLOC(n-1, Complex)) ||
           !(d__ = NAG_ALLOC(n, 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 , %lf )", &e[i - 1].re, &e[i - 1].im);
    }
  Vscanf("%*[^\n] ");

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

  /* Solve the equations AX = B for X */
  /* nag_herm_posdef_tridiag_lin_solve (f04cgc).
   * Computes the solution and error-bound to a complex
   * Hermitian positive-definite tridiagonal system of linear
   * equations
   */
  nag_herm_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_complx_mat_print_comp (x04dbc).
       * Print complex general matrix (comprehensive)
       */
      nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                                    n, nrhs, b, pdb, Nag_BracketForm, 0,
                                    "Solution", Nag_IntegerLabels, 0,
                                    Nag_IntegerLabels, 0, 80, 0, 0, &fail);
      if (fail.code != NE_NOERROR)
```

```
        {
          Vprintf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }

      Vprintf("\n");
      Vprintf("%s\n%8s%9.1e\n", "Estimate of condition number", "", 1.0/rcond);
      Vprintf("\n\n");
      Vprintf("%s\n%8s%9.1e\n\n",
              "Estimate of error bound for computed solutions", "", errbnd);
    }
  if (fail.code == NE_RCOND)
    {
      /* Matrix A is numerically singular.  Print estimate of */
      /* reciprocal of condition number and solution */

      Vprintf("\n");
      Vprintf("%s\n%8s%9.1e\n\n\n",
              "Estimate of reciprocal of condition number", "", rcond);
      /* nag_gen_complx_mat_print_comp (x04dbc), see above. */
      nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                                    n, nrhs, b, pdb, Nag_BracketForm, 0,
                                    "Solution", Nag_IntegerLabels, 0,
                                    Nag_IntegerLabels, 0, 80, 0, 0, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
    }
  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 (clabs) NAG_FREE(clabs);
  if (rlabs) NAG_FREE(rlabs);
  if (b) NAG_FREE(b);
  if (e) NAG_FREE(e);
  if (d__) NAG_FREE(d__);

  return exit_status;
}
#undef B
```

9.2 Program Data

```
nag_herm_posdef_tridiag_lin_solve (f04cgc) Example Program Data

    4             2                                :Values of N and NRHS

   16.0          41.0          46.0          21.0 :End of diagonal D
 ( 16.0, 16.0) ( 18.0, -9.0) (  1.0, -4.0)        :End of sub-diagonal E

 ( 64.0, 16.0) (-16.0,-32.0)
 ( 93.0, 62.0) ( 61.0,-66.0)
 ( 78.0,-80.0) ( 71.0,-74.0)
 ( 14.0,-27.0) ( 35.0, 15.0)                      :End of matrix B
```

9.3 Program Results

```
nag_herm_posdef_tridiag_lin_solve (f04cgc) Example Program Results

 Solution

                              1                          2
 1  (     2.0000,     1.0000)  (    -3.0000,    -2.0000)
 2  (     1.0000,     1.0000)  (     1.0000,     1.0000)
 3  (     1.0000,    -2.0000)  (     1.0000,    -2.0000)
 4  (     1.0000,    -1.0000)  (     2.0000,     1.0000)

Estimate of condition number
          9.2e+03


Estimate of error bound for computed solutions
          1.0e-12
```