

nag_hermitian_lin_eqn_mult_rhs (f04awc)

1. Purpose

nag_hermitian_lin_eqn_mult_rhs (f04awc) calculates the approximate solution of a set of complex Hermitian positive-definite linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by **nag_complex_cholesky (f01bnc)**.

2. Specification

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

void nag_hermitian_lin_eqn_mult_rhs(Integer n, Integer nrhs, Complex a[],
    Integer tda, double p[], Complex b[], Integer tdb, Complex x[],
    Integer tdx, NagError *fail)
```

3. Description

To solve a set of complex linear equations, $AX = B$, where A is positive-definite Hermitian, this routine must be preceded by a call to **nag_complex_cholesky (f01bnc)** which computes a Cholesky factorization $A = U^H U$, where U is an upper triangular matrix with real diagonal elements. The columns x of the solution X are found in two steps $U^H y = b$ and $Ux = y$, where b is a column of the right-hand side matrix B .

4. Parameters

n

Input: n , the order of the matrix A .
Constraint: $n \geq 1$.

nrhs

Input: r , the number of right-hand sides.
Constraint: **nrhs** ≥ 1 .

a[n][tda]

Input: the off-diagonal elements of the upper triangular matrix U as returned by **nag_complex_cholesky (f01bnc)**. The lower triangle of the array is not used.

tda

Input: the second dimension of the array **a** as declared in the function from which **nag_hermitian_lin_eqn_mult_rhs** is called.
Constraint: **tda** $\geq n$.

p[n]

Input: the reciprocals of the diagonal elements of U , as returned by **nag_complex_cholesky (f01bnc)**.

b[n][tdb]

Input: the n by r right-hand side matrix B . See also Section 6.

tdb

Input: the second dimension of the array **b** as declared in the function from which **nag_hermitian_lin_eqn_mult_rhs** is called.
Constraint: **tdb** $\geq$ **nrhs**.

x[n][tdx]

Output: the n by r solution matrix X . See also Section 6.

tdx

Input: the second dimension of the array **x** as declared in the function from which **nag_hermitian_lin_eqn_mult_rhs** is called.
Constraint: **tdx** $\geq$ **nrhs**.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_INT_ARG_LT

On entry, **nrhs** must not be less than 1: **nrhs** = $\langle value \rangle$.

On entry, **n** must not be less than 1: **n** = $\langle value \rangle$.

NE_2_INT_ARG_LT

On entry, **tda** = $\langle value \rangle$ while **n** = $\langle value \rangle$. These parameters must satisfy **tda** $\geq$ **n**.

On entry, **tdb** = $\langle value \rangle$ while **nrhs** = $\langle value \rangle$. These parameters must satisfy **tdb** $\geq$ **nrhs**.

On entry, **tdx** = $\langle value \rangle$ while **nrhs** = $\langle value \rangle$. These parameters must satisfy **tdx** $\geq$ **nrhs**.

6. Further Comments

The time taken by the function is approximately proportional to n^2r .

The function may be called with the same actual array supplied for parameters **b** and **x**, in which case the solution vectors will overwrite the right-hand sides.

6.1. Accuracy

The solutions should be the best possible for the precision of computation having regard to the condition of A . The computed solution x , corresponding to the right-hand side b , satisfies the bound

$$\frac{\|x - A^{-1}b\|}{\|A^{-1}b\|} \leq c\epsilon k.$$

Here c is a modest function of n , ϵ is the **machine precision**, and k is the condition number defined by

$$k = \|A\| \|A^{-1}\|.$$

6.2. References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation (Vol II, Linear Algebra)* Springer-Verlag pp 9–30.

7. See Also

nag_complex_cholesky (f01bnc)

8. Example

To solve the set of linear equations $AX = B$ where A is the positive-definite Hermitian matrix:

$$\begin{pmatrix} 15 & 1-2i & 2 & -4+3i \\ 1+2i & 20 & -2+i & 3-3i \\ 2 & -2-i & 18 & -1+2i \\ -4-3i & 3+3i & -1-2i & 26 \end{pmatrix}$$

and B is the single column vector:

$$\begin{pmatrix} 25+34i \\ 21+19i \\ 12-21i \\ 21-27i \end{pmatrix}.$$

8.1. Program Text

```

/* nag_hermitian_lin_eqn_mult_rhs(f04awc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagf04.h>

#define NMAX 8
#define TDA NMAX
#define TDB 2
#define TDx 2
#define COMPLEX(A) A.re, A.im

main()
{
    Integer i, j, n;
    Complex a[NMAX][TDA], b[NMAX][TDB], x[NMAX][TDx];
    double p[NMAX];
    Integer nrhs = 1;
    static NagError fail;

    Vprintf("f04awc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\\n]");
    Vscanf("%ld", &n);
    if (n<0 || n>NMAX)
    {
        Vfprintf(stderr, "\\nn is out of range: n = %ld\\n", n);
        exit(EXIT_FAILURE);
    }
    for (i=0; i<n; ++i)
        for (j=0; j<=i; ++j)
            Vscanf(" ( %lf , %lf ) ", COMPLEX(&a[i][j]));
    for (i=0; i<n; ++i)
        for (j=0; j<nrhs; ++j)
            Vscanf(" ( %lf , %lf ) ", COMPLEX(&b[i][j]));
    /* Factorize A */
    fail.print = TRUE;
    f01bnc(n, (Complex *)a, (Integer)TDA, p, &fail);
    if (fail.code != NE_NOERROR)
        exit(EXIT_FAILURE);
    /* Solve A */
    f04awc(n, nrhs, (Complex *)a, (Integer)TDA, p, (Complex *)b,
           (Integer)TDB, (Complex *)x, (Integer)TDx, &fail);
    if (fail.code != NE_NOERROR)
        exit(EXIT_FAILURE);
    Vprintf("\\nSolution\\n\\n");
    for (i=0; i<n; ++i)
    {
        for (j=0; j<nrhs; ++j)
            Vprintf(" (%7.4f,%7.4f)", COMPLEX(x[i][j]));
        Vprintf("\\n");
    }
    exit(EXIT_SUCCESS);
}

```

8.2. Program Data

f04awc Example Program Data

```
4
(15.0, 0.0)
( 1.0, 2.0) (20.0, 0.0)
( 2.0, 0.0) (-2.0, -1.0) (18.0, 0.0)
(-4.0, -3.0) ( 3.0, 3.0) (-1.0, -2.0) (26.0, 0.0)
(25.0, 34.0) (21.0, 19.0) (12.0, -21.0) (21.0, -27.0)
```

8.3. Program Results

f04awc Example Program Results

Solution

```
( 1.4917, 2.1788)
( 1.1629, 0.7698)
( 0.5506, -1.3977)
( 0.8691, -0.7655)
```
