NAG Library Function Document

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[], const 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 function 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 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

5 Arguments

1: **n** – Integer *Input*

On entry: n, the order of the matrix A.

Constraint: **n** ≥ 1.

2: **nrhs** – Integer *Input*

On entry: r, the number of right-hand sides.

Constraint: **nrhs** ≥ 1.

3: **a**[**n** $\times$ **tda**] – Complex *Input*

Note: the (i, j)th element of the matrix A is stored in **a**$[(i-1) \times$ **tda** $+ j - 1]$.

On entry: 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.

4: **tda** – Integer *Input*

On entry: the stride separating matrix column elements in the array **a**.

Constraint: **tda** $\geq$ **n**.

5: **p**[**n**] – double *Input*

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

6: **b**[**n** × **tdb**] – const Complex *Input*

Note: the (i, j)th element of the matrix B is stored in **b**$[(i - 1) \times$ **tdb** $+ j - 1]$.

On entry: the n by r right-hand side matrix B. See also Section 9.

7: **tdb** – Integer *Input*

On entry: the stride separating matrix column elements in the array **b**.

Constraint: **tdb** ≥ **nrhs**.

8: **x**[**n** × **tdx**] – Complex *Output*

Note: the (i, j)th element of the matrix X is stored in **x**$[(i - 1) \times$ **tdx** $+ j - 1]$.

On exit: the n by r solution matrix X. See also Section 9.

9: **tdx** – Integer *Input*

On entry: the stride separating matrix column elements in the array **x**.

Constraint: **tdx** ≥ **nrhs**.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **tda** ≥ **n**.

On entry, **tdb** = ⟨*value*⟩ while **nrhs** = ⟨*value*⟩. These arguments must satisfy **tdb** ≥ **nrhs**.

On entry, **tdx** = ⟨*value*⟩ while **nrhs** = ⟨*value*⟩. These arguments must satisfy **tdx** ≥ **nrhs**.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

On entry, **nrhs** = ⟨*value*⟩.
Constraint: **nrhs** ≥ 1.

7 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

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

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_hermitian_lin_eqn_mult_rhs (f04awc) is approximately proportional to n^2r.

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

10 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:

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

10.1 Program Text

```
/* nag_hermitian_lin_eqn_mult_rhs (f04awc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

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


#define COMPLEX(A) A.re, A.im
#define A(I, J)    a[(I) *tda + J]
#define B(I, J)    b[(I) *tdb + J]
#define X(I, J)    x[(I) *tdx + J]
int main(void)
{
  Complex  *a = 0, *b = 0, *x = 0;
  Integer  exit_status = 0, i, j, n, nrhs, tda, tdb, tdx;
  NagError fail;
  double   *p = 0;

  INIT_FAIL(fail);

  printf(
          "nag_hermitian_lin_eqn_mult_rhs (f04awc) Example Program Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  nrhs = 1;
  if (n >= 1)
    {
      if (!(p = NAG_ALLOC(n, double)) ||
          !(a = NAG_ALLOC(n*n, Complex)) ||
```

```
        !(b = NAG_ALLOC(n*nrhs, Complex)) ||
        !(x = NAG_ALLOC(n*nrhs, Complex)))
      {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
      }
    tda = n;
    tdb = nrhs;
    tdx = nrhs;
  }
else
  {
    printf("Invalid n.\n");
    exit_status = 1;
    return exit_status;
  }
for (i = 0; i < n; ++i)
  for (j = 0; j <= i; ++j)
    scanf(" ( %lf,  %lf ) ", COMPLEX(&A(i, j)));
for (i = 0; i < n; ++i)
  for (j = 0; j < nrhs; ++j)
    scanf(" ( %lf,  %lf ) ", COMPLEX(&B(i, j)));
/* Factorize A */
/* nag_complex_cholesky (f01bnc).
 * UU^H factorization of complex Hermitian positive-definite
 * matrix
 */
nag_complex_cholesky(n, a, tda, p, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_complex_cholesky (f01bnc).\n%s\n",
            fail.message);
    exit_status = 1;
    goto END;
  }
/* Solve A */
/* nag_hermitian_lin_eqn_mult_rhs (f04awc).
 * Approximate solution of complex Hermitian
 * positive-definite simultaneous linear equations
 * (coefficient matrix already factorized by
 * nag_complex_cholesky (f01bnc))
 */
nag_hermitian_lin_eqn_mult_rhs(n, nrhs, a, tda, p, b, tdb, x, tdx, &fail);
if (fail.code != NE_NOERROR)
  {
    printf(
            "Error from nag_hermitian_lin_eqn_mult_rhs (f04awc).\n%s\n",
            fail.message);
    exit_status = 1;
    goto END;
  }
printf("\nSolution\n\n");
for (i = 0; i < n; ++i)
  {
    for (j = 0; j < nrhs; ++j)
      printf("  (%7.4f,%7.4f)", COMPLEX(X(i, j)));
    printf("\n");
  }
END:
NAG_FREE(p);
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(x);
return exit_status;
}
```

10.2 Program Data

```
nag_hermitian_lin_eqn_mult_rhs (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)
```

10.3 Program Results

```
nag_hermitian_lin_eqn_mult_rhs (f04awc) Example Program Results

Solution

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