NAG Library Function Document

nag_complex_lin_eqn_mult_rhs (f04adc)

1 Purpose

nag_complex_lin_eqn_mult_rhs (f04adc) calculates the solution of a set of complex linear equations with multiple right-hand sides.

2 Specification

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

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

3 Description

Given a set of complex linear equations $AX = B$, the function first computes an LU factorization of A with partial pivoting, $PA = LU$, where P is a permutation matrix, L is lower triangular and U is unit upper triangular. The columns x of the solution X are found by forward and backward substitution in $Ly = Pb$ and $Ux = y$, where b is a column of the right-hand side matrix B.

4 References

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

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/Output*

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

On entry: the n by n matrix A.

On exit: A is overwritten by the lower triangular matrix L and the off-diagonal elements of the upper triangular matrix U. The unit diagonal elements of U are not stored.

4: **tda** – Integer *Input*

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

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

5: **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.

6: **tdb** – Integer *Input*

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

Constraint: **tdb** $\geq$ **nrhs**.

7: **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 6.

8: **tdx** – Integer *Input*

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

Constraint: **tdx** $\geq$ **nrhs**.

9: **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** $= \langle value \rangle$ while **n** $= \langle value \rangle$. The arguments must satisfy **tda** $\geq$ **n**.

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

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

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

On entry, **nrhs** $= \langle value \rangle$.
Constraint: **nrhs** ≥ 1.

NE_SINGULAR

The matrix A is singular, possibly due to rounding errors.

7 Accuracy

The accuracy of the computed solution depends on the conditioning of the original matrix. For a detailed error analysis see page 106 of Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_complex_lin_eqn_mult_rhs (f04adc) is approximately proportional to n^3.

The function may be called with the same 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

$$A = \begin{pmatrix} 1 & 1+2i & 2+10i \\ 1+i & 3i & -5+14i \\ 1+i & 5i & -8+20i \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_complex_lin_eqn_mult_rhs (f04adc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1A revised, (Oct 1990).
 * Mark 8 revised, 2004.
 */

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

#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 = 1, tda, tdb, tdx;
  NagError fail;

  INIT_FAIL(fail);

  printf(
          "nag_complex_lin_eqn_mult_rhs (f04adc) Example Program Results\n");
  scanf("%*[^\n]");  /* Skip heading in data file */
  scanf("%ld", &n);
  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, Complex)) ||
          !(b = NAG_ALLOC(n*1, Complex)) ||
          !(x = NAG_ALLOC(n*1, Complex)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdb = 1;
      tdx = 1;
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 0; i < n; i++)
```

```
    for (j = 0; j < n; j++)
      scanf(" ( %lf,  %lf ) ", &A(i, j).re, &A(i, j).im);
  for (i = 0; i < n; i++)
    for (j = 0; j < tdx; j++)
      scanf(" ( %lf,  %lf ) ", &B(i, j).re, &B(i, j).im);
  /* nag_complex_lin_eqn_mult_rhs (f04adc).
   * Approximate solution of complex simultaneous linear
   * equations with multiple right-hand sides
   */
  nag_complex_lin_eqn_mult_rhs(n, nrhs, a, tda, b, tdb, x, tdx, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_complex_lin_eqn_mult_rhs (f04adc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
  else
    {
      printf("Solution\n");
      for (i = 0; i < n; i++)
        printf("(%7.4f, %7.4f)\n", X(i, 0).re, X(i, 0).im);
    }
 END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_complex_lin_eqn_mult_rhs (f04adc) Example Program Data
  3
( 1.0, 0.0 ) ( 1.0, 2.0 ) ( 2.0,10.0 )
( 1.0, 1.0 ) ( 0.0, 3.0 ) (-5.0,14.0 )
( 1.0, 1.0 ) ( 0.0, 5.0 ) (-8.0,20.0 )
( 1.0, 0.0 ) ( 0.0, 0.0 ) ( 0.0, 0.0 )
```

10.3 Program Results

```
nag_complex_lin_eqn_mult_rhs (f04adc) Example Program Results
Solution
(10.0000,  1.0000)
( 9.0000, -3.0000)
(-2.0000,  2.0000)
```