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, 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. Parameters

n
Input: n, the order of the matrix A.
Constraint: **n** ≥ 1.

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

a[n][tda]
Input: the n by n matrix A.
Output: 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.

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

b[n][tdb]
Input: the n by r right-hand side matrix B.

tdb
Input: the last dimension of the array **b** as declared in the function from which nag_complex_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 last dimension of the array **x** as declared in the function from which nag_complex_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_SINGULAR
The matrix A is singular, possibly due to rounding errors.

NE_INT_ARG_LT
On entry, **n** must not be less than 1: **n** = ⟨*value*⟩.
On entry, **nrhs** must not be less than 1: **nrhs** = ⟨*value*⟩.

NE_2_INT_ARG_LT
On entry, **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩. The parameters must satisfy **tda** ≥ **n**.
On entry, **tdb** = ⟨*value*⟩ while **nrhs** = ⟨*value*⟩. These parameters must satisfy **tdb** ≥ **nrhs**.
On entry, **tdx** = ⟨*value*⟩ while **nphs** = ⟨*value*⟩. These parameters must satisfy **tdx** ≥ **nrhs**.

NE_ALLOC_FAIL
Memory allocation failed.

6. Further Comments

The time taken by the function is approximately proportional to n^3.

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

6.1. Accuracy

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

6.2. References

Golub G H and Van Loan C F (1989) *Matrix Computations* (2nd Edn) Johns Hopkins University Press, Baltimore.
Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation (Vol II, Linear Algebra)* Springer-Verlag pp 93–110.

7. See Also

nag_complex_lu (f03ahc)
nag_complex_lu_solve_mult_rhs (f04akc)

8. Example

To solve the set of linear equations $AX = B$ where

$$A = \begin{pmatrix} 1 & 1+2i & 2+10i \\ 1+i & 3i & -5+14i \\ 1+i & 5i & -8+20i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

8.1. Program Text

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

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

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

main()
{
#define NMAX 5
#define TDA NMAX
#define TDB 1
#define TDX 1

  Complex a[NMAX][TDA], b[NMAX][TDB], x[NMAX][TDX];
  Integer i, j, n, nrhs = 1;
  static NagError fail;

  fail.print = TRUE;
  Vprintf("f04adc Example Program Results\n");
  Vscanf("%*[^\n]");  /* Skip heading in data file */
  Vscanf("%ld", &n);
  if (n > 0 && n <= NMAX)
    {
      for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
          Vscanf(" ( %lf , %lf ) ", &a[i][j].re, &a[i][j].im);
      for (i = 0; i < n; i++)
        for (j = 0; j < TDX; j++)
          Vscanf(" ( %lf , %lf ) ", &b[i][j].re, &b[i][j].im);
      f04adc(n, nrhs, (Complex *)a, (Integer)TDA, (Complex *)b, (Integer)TDB,
             (Complex *)x, (Integer)TDX, &fail);
      if (fail.code!=NE_NOERROR)
        exit(EXIT_FAILURE);
      else
        {
          Vprintf("Solution\n");
          for (i=0; i<n; i++)
            Vprintf("(%7.4f, %7.4f)\n", x[i][0].re, x[i][0].im);
        }
    }
  else
    {
      Vfprintf(stderr, "Error: n is out of range: n = %ld\n", n);
      exit(EXIT_FAILURE);
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
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 )
```

8.3. Program Results

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