NAG Library Function Document

nag_real_lu_solve_mult_rhs (f04ajc)

1 Purpose

nag_real_lu_solve_mult_rhs (f04ajc) calculates the approximate solution of a set of real linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by nag_real_lu (f03afc).

2 Specification

```
#include <nag.h>
#include <nagf04.h>
void nag_real_lu_solve_mult_rhs (Integer n, Integer nrhs, const double a[],
     Integer tda, const Integer pivot[], double b[], Integer tdb,
     NagError *fail)
```

3 Description

To solve a set of real linear equations $AX = B$, this function must be preceded by a call to nag_real_lu (f03afc) which 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 sides.

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**] – const double *Input*

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

On entry: details of the LU factorization, as returned by nag_real_lu (f03afc).

4: **tda** – Integer *Input*

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

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

5: **pivot**[**n**] – const Integer *Input*

On entry: details of the row interchanges as returned by nag_real_lu (f03afc).

6: **b**[**n** × **tdb**] – double *Input/Output*

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.

On exit: B is overwritten by the solution matrix X.

7: **tdb** – Integer *Input*

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

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

8: **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** $= \langle value \rangle$. These arguments must satisfy **tda** $\geq$ **n**.

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

NE_INT_ARG_LT

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

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

7 Accuracy

The accuracy of the computed solutions 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_real_lu_solve_mult_rhs (f04ajc) is approximately proportional to $n^2 r$.

10 Example

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

$$A = \begin{pmatrix} 33 & 16 & 72 \\ -24 & -10 & -57 \\ -8 & -4 & -17 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -359 \\ 281 \\ 85 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_real_lu_solve_mult_rhs (f04ajc) 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 <nagf03.h>
#include <nagf04.h>

#define A(I, J) a[(I) *tda + J]
#define B(I, J) b[(I) *tdb + J]
int main(void)
{
  Integer  dete, exit_status = 0, i, j, n, nrhs, *pivot = 0, tda, tdb;
  NagError fail;
  double   *a = 0, *b = 0, detf;

  INIT_FAIL(fail);

  printf(
          "nag_real_lu_solve_mult_rhs (f04ajc) Example Program Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  nrhs = 1;
  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, double)) ||
          !(b = NAG_ALLOC(n*nrhs, double)) ||
          !(pivot = NAG_ALLOC(n, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdb = nrhs;
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      goto END;
    }
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%lf", &A(i, j));
  /* Crout decomposition  */
  /* nag_real_lu (f03afc).
   * LU factorization and determinant of real matrix
   */
  nag_real_lu(n, a, tda, pivot, &detf, &dete, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_lu (f03afc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (i = 0; i < n; i++)
    for (j = 0; j < nrhs; j++)
      scanf("%lf", &B(i, j));
  /*
   * Approximate solution of linear equations
   */
  /* nag_real_lu_solve_mult_rhs (f04ajc).
   * Approximate solution of real simultaneous linear
   * equations (coefficient matrix already factorized by
   * nag_real_lu (f03afc))
   */
  nag_real_lu_solve_mult_rhs(n, nrhs, a, tda, pivot, b, nrhs, &fail);
  if (fail.code != NE_NOERROR)
    {
```

```
      printf("Error from nag_real_lu_solve_mult_rhs (f04ajc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
  printf("Solution\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < nrhs; j++)
      printf("%9.4f\n", B(i, j));
 END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(pivot);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_lu_solve_mult_rhs (f04ajc) Example Program Data
  3
   33   16   72
  -24  -10  -57
   -8   -4  -17
 -359  281   85
```

10.3 Program Results

```
nag_real_lu_solve_mult_rhs (f04ajc) Example Program Results
Solution
   1.0000
  -2.0000
  -5.0000
```