NAG Library Function Document

nag_real_lin_eqn (f04arc)

1 Purpose

nag_real_lin_eqn (f04arc) calculates the approximate solution of a set of real linear equations with a single right-hand side, using an LU factorization with partial pivoting.

2 Specification

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

```
void nag_real_lin_eqn (Integer n, double a[], Integer tda, const double b[],
     double x[], NagError *fail)
```

3 Description

Given a set of 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 approximate solution x is found by forward and backward substitution in $Ly = Pb$ and $Ux = y$, where b is the right-hand side.

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: $\mathbf{n} \geq 1$.

2: **a**[**n** × **tda**] – double *Input/Output*

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

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.

3: **tda** – Integer *Input*

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

Constraint: $\mathbf{tda} \geq \mathbf{n}$.

4: **b**[**n**] – const double *Input*

On entry: the right-hand side vector b.

5: **x**[**n**] – double *Output*

On exit: the solution vector x.

6: **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**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 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 107 of Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_lin_eqn (f04arc) is approximately proportional to n^3.

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_lin_eqn (f04arc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 * 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]
int main(void)
{
```

```
  Integer  exit_status = 0, i, j, n, tda;
  NagError fail;
  double   *a = 0, *b = 0, *x = 0;

  INIT_FAIL(fail);

  printf("nag_real_lin_eqn (f04arc) Example Program Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, double)) ||
          !(b = NAG_ALLOC(n, double)) ||
          !(x = NAG_ALLOC(n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
    }
  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", &A(i, j));
  for (i = 0; i < n; i++)
    scanf("%lf", &b[i]);
  /* nag_real_lin_eqn (f04arc).
   * Approximate solution of real simultaneous linear
   * equations, one right-hand side
   */
  nag_real_lin_eqn(n, a, tda, b, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_real_lin_eqn (f04arc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Solution\n");
  for (i = 0; i < n; i++)
    printf("%9.4f\n", x[i]);
 END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_lin_eqn (f04arc) Example Program Data
  3
   33   16   72
  -24  -10  -57
   -8   -4  -17
 -359  281   85
```

10.3 Program Results

```
nag_real_lin_eqn (f04arc) Example Program Results
Solution
   1.0000
  -2.0000
  -5.0000
```