NAG Library Function Document

nag_real_lu (f03afc)

1 Purpose

nag_real_lu (f03afc) computes an LU factorization of a real matrix, with partial pivoting, and evaluates the determinant.

2 Specification

```
#include <nag.h>
#include <nagf03.h>
void nag_real_lu (Integer n, double a[], Integer tda, Integer pivot[],
     double *detf, Integer *dete, NagError *fail)
```

3 Description

nag_real_lu (f03afc) computes an LU factorization of a real matrix A with partial pivoting: $PA = LU$, where P is a permutation matrix, L is lower triangular and U is unit upper triangular. The determinant of A is the product of the diagonal elements of L with the correct sign determined by the row interchanges.

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: **a**[**n** $\times$ **tda**] – double *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.

3: **tda** – Integer *Input*

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

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

4: **pivot**[**n**] – Integer *Output*

On exit: **pivot**[$i-1$] gives the row index of the ith pivot.

5: **detf** – double * *Output*

6: **dete** – Integer * *Output*

On exit: the determinant of A is given by **detf** $\times 2.0^{\textbf{dete}}$. It is given in this form to avoid overflow or underflow.

7: **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*⟩. The arguments must satisfy **tda** ≥ **n**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

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

NE_SINGULAR

The matrix A is singular, possibly due to rounding errors. The factorization could not be completed. **detf** and **dete** are set to zero.

7 Accuracy

The accuracy of the determinant depends on the conditioning of the original matrix. For a detailed error analysis, see Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_lu (f03afc) is approximately proportional to n^3.

10 Example

To compute the LU factorization with partial pivoting, and calculate the determinant, of the real matrix

$$\begin{pmatrix} 33 & 16 & 72 \\ -24 & -10 & -57 \\ -8 & -4 & -17 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_real_lu (f03afc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf03.h>

#define A(I, J) a[(I) *tda + J]
```

```
int main(void)
{
  Integer  dete, exit_status = 0, i, j, n, *pivot = 0, tda;
  NagError fail;
  double   *a = 0, detf, two = 2.0;

  INIT_FAIL(fail);

  printf("nag_real_lu (f03afc) Example Program Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%" NAG_IFMT "", &n);

  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, double)) ||
          !(pivot = NAG_ALLOC(n, Integer)))
        {
          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));
  /* 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;
    }
  else
    {
      printf("Array A after factorization\n");
      for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
          printf("%9.4f%s", A(i, j), (j%8 == 7 || j == n-1)?"\n":" ");
      printf("\nArray P\n");
      for (i = 0; i < n; i++)
        printf("%3" NAG_IFMT "%s", pivot[i], (i%8 == 7 || i == n-1)?"\n":" ");
      printf("\ndetf = %9.4f  dete = %2" NAG_IFMT "\n", detf, dete);
      detf = detf * pow(two, (double) dete);
      printf("\nValue of determinant = %9.4f\n", detf);
    }
 END:
  NAG_FREE(a);
  NAG_FREE(pivot);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_lu (f03afc) Example Program Data
  3
   33   16   72
  -24  -10  -57
   -8   -4  -17
```

10.3 Program Results

```
nag_real_lu (f03afc) Example Program Results
Array A after factorization
  -8.0000    0.5000    2.1250
 -24.0000    2.0000   -3.0000
  33.0000   -0.5000    0.3750

Array P
  3   2   3

detf =    0.3750  dete =  4

Value of determinant =    6.0000
```