

NAG Library Function Document

nag_complex_lu (f03ahc)

1 Purpose

nag_complex_lu (f03ahc) computes an LU factorization of a complex matrix, with partial pivoting, and evaluates the determinant.

2 Specification

```
#include <nag.h>
#include <nagf03.h>
```

```
void nag_complex_lu (Integer n, Complex a[], Integer tda, Integer pivot[],
    Complex *det, Integer *dete, NagError *fail)
```

3 Description

nag_complex_lu (f03ahc) computes an LU factorization of a complex 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 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: $n \geq 1$.
 - 2: **a[n × tda]** – Complex *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 row elements in \mathbf{a} .
Constraint: $\mathbf{tda} \geq n$
 - 4: **pivot[n]** – Integer *Output*
On exit: **pivot**[$i - 1$] gives the row index of the i th pivot.
 - 5: **det** – Complex * *Output*
 - 6: **dete** – Integer * *Output*
- On exit:* the determinant of A is given by $(\mathbf{det.re} + i\mathbf{det.im}) \times 2.0^{\mathbf{dete}}$. It is given in this form to avoid overflow and 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** = $\langle value \rangle$ while **n** = $\langle value \rangle$. The 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** \geq 1.

NE_SINGULAR

The matrix A is singular, possibly due to rounding errors. The factorization could not be completed. **det.re**, **det.im** 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 Further Comments

The time taken by `nag_complex_lu` (f03ahc) is approximately proportional to n^3 .

9 Example

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

$$\begin{pmatrix} 2 & 1+2i & 2+10i \\ 1+i & 1+3i & -5+14i \\ 1+i & 5i & -7+20i \end{pmatrix}.$$

9.1 Program Text

```
/* nag_complex_lu (f03ahc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1A revised, (Oct 1990).
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <nagx04.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf03.h>

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

int main(int argc, char *argv[])
{
    FILE      *fpin, *fpout;
```

```

Complex  *a = 0, det;
Integer  dete, exit_status = 0, i, j, n, *pivot = 0, tda;
NagError fail;

INIT_FAIL(fail);

/* Check for command-line IO options */
fpin = nag_example_file_io(argc, argv, "-data", NULL);
fpout = nag_example_file_io(argc, argv, "-results", NULL);

fprintf(fpout, "nag_complex_lu (f03ahc) Example Program Results\n");
fscanf(fpin, "%*[\n]"); /* Skip heading in data file */

fscanf(fpin, "%ld", &n);
if (n >= 1)
{
    if (!(pivot = NAG_ALLOC(n, Integer)) ||
        !(a = NAG_ALLOC(n*n, Complex)))
    {
        fprintf(fpout, "Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tda = n;
}
else
{
    fprintf(fpout, "Invalid n.\n");
    exit_status = 1;
    return exit_status;
}
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        fscanf(fpin, "( %lf, %lf ) ", &A(i, j).re, &A(i, j).im);
/* nag_complex_lu (f03ahc).
 * LU factorization and determinant of complex matrix
 */
nag_complex_lu(n, a, tda, pivot, &det, &dete, &fail);
if (fail.code != NE_NOERROR)
{
    fprintf(fpout, "Error from nag_complex_lu (f03ahc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
else
{
    fprintf(fpout, "Array a after factorization\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            fprintf(fpout, "(%7.3f, %7.3f) ", A(i, j).re, A(i, j).im);
        fprintf(fpout, "\n");
    }
    fprintf(fpout, "\nArray pivot\n");
    for (i = 0; i < n; i++)
        fprintf(fpout, "%5ld", pivot[i]);
    fprintf(fpout, "\n\ndet.re = %7.4f, det.im = %7.4f, dete = %2ld.\n",
        det.re, det.im, dete);
    det.re = ldexp(det.re, (int) dete);
    det.im = ldexp(det.im, (int) dete);
    fprintf(fpout, "\nValue of determinant = (%7.4f, %7.4f)\n", det.re,
        det.im);
}
END:
if (fpin != stdin) fclose(fpin);

```

```

    if (fpout != stdout) fclose(fpout);
    if (pivot) NAG_FREE(pivot);
    if (a) NAG_FREE(a);
    return exit_status;
}

```

9.2 Program Data

```

nag_complex_lu (f03ahc) Example Program Data
  3
(2.0, 0.0) (1.0, 2.0) (2.0,10.0)
(1.0, 1.0) (1.0, 3.0) (-5.0,14.0)
(1.0, 1.0) (0.0, 5.0) (-7.0,20.0)

```

9.3 Program Results

```

nag_complex_lu (f03ahc) Example Program Results
Array a after factorization
( 2.000,  0.000) ( 0.500,  1.000) ( 1.000,  5.000)
( 1.000,  1.000) ( 0.500,  3.500) ( 3.800,  1.400)
( 1.000,  1.000) ( 1.500,  1.500) (-4.600,  0.200)

Array pivot
  1   3   3

det.re =  0.0234, det.im =  0.1250, dete =  8.

Value of determinant = ( 6.0000, 32.0000)

```
