NAG Library Function Document

nag_det_complex_gen (f03bnc)

1 Purpose

nag_det_complex_gen (f03bnc) computes the determinant of a complex n by n matrix A. nag_zgetrf (f07arc) must be called first to supply the matrix A in factorized form.

2 Specification

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

```
void nag_det_complex_gen (Nag_OrderType order, Integer n, const Complex a[],
     Integer pda, const Integer ipiv[], Complex *d, Integer id[],
     NagError *fail)
```

3 Description

nag_det_complex_gen (f03bnc) computes the determinant of a complex n by n matrix A that has been factorized by a call to nag_zgetrf (f07arc). The determinant of A is the product of the diagonal elements of U 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: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **n** – Integer *Input*

On entry: n, the order of the matrix A.

Constraint: **n** > 0.

3: **a**$[dim]$ – const Complex *Input*

Note: the dimension, *dim*, of the array **a** must be at least **pda** $\times$ **n**.

The (i, j)th element of the factorized form of the matrix A is stored in

a$[(j-1) \times$ **pda** $+ i - 1]$ when **order** = Nag_ColMajor;
a$[(i-1) \times$ **pda** $+ j - 1]$ when **order** = Nag_RowMajor.

On entry: the n by n matrix A in factorized form as returned by nag_zgetrf (f07arc).

4: **pda** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **a**.

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

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

On entry: the row interchanges used to factorize matrix A as returned by nag_zgetrf (f07arc).

6: **d** – Complex * *Output*

On exit: the mantissa of the real and imaginary parts of the determinant.

7: **id**[**2**] – Integer *Output*

On exit: the exponents for the real and imaginary parts of the determinant. The determinant, $d = (d_r, d_i)$, is returned as $d_r = D_r \times 2^j$ and $d_i = D_i \times 2^k$, where **d** $= (D_r, D_i)$ and j and k are stored in the first and second elements respectively of the array **id** on successful exit.

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle$*value*$\rangle$ had an illegal value.

NE_INT

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

NE_INT_2

On entry, **pda** $= \langle$*value*$\rangle$ and **n** $= \langle$*value*$\rangle$.
Constraint: **pda** $\geq$ **n**.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

NE_SINGULAR

The matrix A is approximately singular.

7 Accuracy

The accuracy of the determinant 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_det_complex_gen (f03bnc) is approximately proportional to n.

10 Example

This example calculates the determinant of the complex matrix

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

10.1 Program Text

```
/* nag_det_complex_gen (f03bnc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf03.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer        exit_status = 0;
  Integer        i, j, n, pda;
  Complex        d;
  /* Arrays */
  Integer        *ipiv = 0;
  Integer        id[2];
  Complex        *a = 0;
  /* NAG types */
  NagError       fail;
  Nag_OrderType  order;
  Nag_MatrixType matrix = Nag_GeneralMatrix;
  Nag_DiagType   diag = Nag_NonUnitDiag;

  printf("nag_det_complex_gen (f03bnc) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld%*[^\n]", &n);
  pda = n;

  if (!(a = NAG_ALLOC((n)*(n), Complex)) ||
      !(ipiv = NAG_ALLOC((n), Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Define matrix element A_ij in terms of elements of array a[k] */
#ifdef NAG_COLUMN_MAJOR
  order = Nag_ColMajor;
#define A(I, J) a[(J-1)*pda+(I-1)]
#else
  order = Nag_RowMajor;
#define A(J, I) a[(J-1)*pda+(I-1)]
#endif
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      scanf(" ( %lf , %lf ) ", &A(i,j).re, &A(i,j).im);
  scanf("%*[^\n] ");

  INIT_FAIL(fail);
  /* Factorize A using nag_zgetrf (f07arc)
   * LU factorization of complex m by n matrix
   */
  nag_zgetrf(order, n, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_gen_complx_mat_print (x04dac)
   * Print complex general matrix (easy-to-use)
   */
  fflush(stdout);
  nag_gen_complx_mat_print(order, matrix, diag, n, n, a, pda,
                           "Array A after factorization", NULL,
                           &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("%s\n", fail.message);
      exit_status = 2;
      goto END;
    }

  printf("Pivots:\n  ");
  for (j = 0; j < n; j++) printf("%12" NAG_IFMT, ipiv[j]);
  printf("\n");

  /* nag_det_complex_gen (f03bnc) - Determinant of complex matrix */
  nag_det_complex_gen(order, n, a, pda, ipiv, &d, id, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("%s\n", fail.message);
      exit_status = 3;
      goto END;
    }

  printf("d = (%9.5f, %9.5f)    id = (%2ld, %2ld)\n",
         d.re, d.im, id[0], id[1]);
  printf("Value of determinant = (%12.5e, %12.5e)\n",
         pow(2.0, id[0])*(d.re), pow(2.0, id[1])*(d.im));

 END:
  NAG_FREE(a);
  NAG_FREE(ipiv);

  return exit_status;
}
```

10.2 Program Data

```
nag_det_complex_gen (f03bnc) Example Program Data

   3                                    : n

 ( 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)  : A
```

10.3 Program Results

```
nag_det_complex_gen (f03bnc) Example Program Results
 Array A after factorization
            1           2           3
 1      1.0000      0.0000     -5.0000
        1.0000      3.0000     14.0000

 2      1.0000      0.0000     -3.0000
        0.0000      2.0000      6.0000

 3      0.5000      0.2500     -0.2500
       -0.5000      0.2500     -0.2500

Pivots:
            2           3           3
d = (  0.06250,   0.00000)    id = ( 4,  0)
Value of determinant = ( 1.00000e+00,  0.00000e+00)
```