

NAG Library Function Document

nag_det_real_gen (f03bac)

1 Purpose

nag_det_real_gen (f03bac) computes the determinant of a real n by n matrix A . nag_dgetrf (f07adc) must be called first to supply the matrix A in factorized form.

2 Specification

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

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

3 Description

nag_det_real_gen (f03bac) computes the determinant of a real n by n matrix A that has been factorized by a call to nag_dgetrf (f07adc). 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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 double *Input*

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

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

$$\begin{aligned} &\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1] \text{ when } \mathbf{order} = \text{Nag_ColMajor}; \\ &\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1] \text{ when } \mathbf{order} = \text{Nag_RowMajor}. \end{aligned}$$

On entry: the n by n matrix A in factorized form as returned by nag_dgetrf (f07adc).

- 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_dgetrf (f07adc).
- 6: **d** – double * *Output*
7: **id** – Integer * *Output*
On exit: the determinant of *A* is given by **d** \times 2.0^{id}. It is given in this form to avoid overflow or underflow.
- 8: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument *value* had an illegal value.

NE_INT

On entry, **n** = *value*.

Constraint: **n** \geq 1.

NE_INT_2

On entry, **pda** = *value* and **n** = *value*.

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.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

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

nag_det_real_gen (f03bac) is not threaded in any implementation.

9 Further Comments

The time taken by nag_det_real_gen (f03bac) is approximately proportional to n .

10 Example

This example computes the LU factorization with partial pivoting, and calculates the determinant, of the real matrix

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

10.1 Program Text

```

/* nag_det_real_gen (f03bac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf03.h>
#include <nagf07.h>
#include <nagx04.h>

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

    printf("nag_det_real_gen (f03bac) Example Program Results\n");
    fflush(stdout);

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#ifdef _WIN32

```

```

    scanf_s("%" NAG_IFMT "%*[\n]", &n);
#else
    scanf("%" NAG_IFMT "%*[\n]", &n);
#endif
    pda = n;
    if (!(a = NAG_ALLOC(n * n, double)) || !(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++)
#ifdef _WIN32
            scanf_s("%lf", &A(i, j));
#else
            scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    INIT_FAIL(fail);
    /* nag_dgetrf (f07adc) - LU factorization of real m by n matrix */
    nag_dgetrf(order, n, n, a, pda, ipiv, &fail);
    if (fail.code != NE_NOERROR) {
        printf("%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* nag_gen_real_mat_print (x04cac).
     * Print real general matrix (easy-to-use)
     */
    printf("\n");
    fflush(stdout);
    nag_gen_real_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("\nPivots:\n  ");
    for (j = 0; j < n; j++)
        printf("%11" NAG_IFMT " ", ipiv[j]);
    printf("\n");

    /* nag_det_real_gen (f03bac).
     * LU factorization and determinant of real matrix
     */
    nag_det_real_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 = %12.5f  id = %12" NAG_IFMT "\n", d, id);

```

```

    printf("Value of determinant = %13.5e\n", d * pow((double) 2.0, id));
END:
    NAG_FREE(a);
    NAG_FREE(ipiv);

    return exit_status;
}

```

10.2 Program Data

```

nag_det_real_gen (f03bac) Example Program Data
  3
  33   16   72
-24  -10  -57
  -8   -4  -17      : A

```

10.3 Program Results

```

nag_det_real_gen (f03bac) Example Program Results

```

```

Array A after factorization
      1      2      3
1    33.0000  16.0000  72.0000
2    -0.7273   1.6364 -4.6364
3    -0.2424  -0.0741   0.1111

```

```

Pivots:
      1      2      3
d =    0.37500  id =    4
Value of determinant =  6.00000e+00

```
