NAG Library Function Document

nag_det_real_gen (f03bac)

1 Purpose

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

2 Specification

```c
#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 \( n \times n \) 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


5 Arguments

1: \( \text{order} \) – Nag_OrderType

   On entry: the \text{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 \text{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: \text{order} = Nag_RowMajor or Nag_ColMajor.

2: \( n \) – Integer

   On entry: \( n \), the order of the matrix \( A \).

   Constraint: \( n > 0 \).

3: \( \text{a}[\text{dim}] \) – const double

   Note: the dimension, \( \text{dim} \), of the array \( \text{a} \) must be at least \( \text{pda} \times n \).

   The \((i,j)\)th element of the factorized form of the matrix \( A \) is stored in
   \[ \text{a}[(j-1) \times \text{pda} + i - 1] \] when \text{order} = Nag_ColMajor;
   \[ \text{a}[(i-1) \times \text{pda} + j - 1] \] when \text{order} = Nag_RowMajor.

   On entry: the \( n \) by \( n \) matrix \( A \) in factorized form as returned by nag_dgetrf (f07adc).
4: \textbf{pda} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{a}.

\textit{Constraint: pda} \geq n.

5: \textbf{ipiv}[n] – const Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the row interchanges used to factorize matrix \textbf{A} as returned by \texttt{nag_dgetrf} (f07adc).

6: \textbf{d} – double * \hspace{1cm} \textit{Output}

7: \textbf{id} – Integer * \hspace{1cm} \textit{Output}

\textit{On exit:} the determinant of \textbf{A} is given by \textbf{d} \times 2.0^{\textbf{id}}. It is given in this form to avoid overflow or underflow.

8: \textbf{fail} – \texttt{NagError} * \hspace{1cm} \textit{Input/Output}

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

6 \ Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \langle\textit{value}\rangle had an illegal value.

\textbf{NE_INT}

On entry, \textbf{n} = \langle\textit{value}\rangle.

\textit{Constraint:} \textbf{n} \geq 1.

\textbf{NE_INT_2}

On entry, \textbf{pda} = \langle\textit{value}\rangle and \textbf{n} = \langle\textit{value}\rangle.

\textit{Constraint:} \textbf{pda} \geq \textbf{n}.

\textbf{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 3.6.6 in the Essential Introduction for further information.

\textbf{NE_NO_LICENCE}

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

See Section 3.6.5 in the Essential Introduction for further information.

\textbf{NE_SINGULAR}

The matrix \textbf{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_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

\[
\begin{bmatrix}
33 & 16 & 72 \\
-24 & -10 & -57 \\
-8 & -4 & -17
\end{bmatrix}
\]

10.1 Program Text

/* nag_det_real_gen (f03bac) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 23, 2011.
 */
#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("%"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");
    }
    if (a && ipiv)
    {
        // Perform LU factorization
        // Calculate determinant
    }
    free(a);
    free(ipiv);
    return exit_status;
}
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]);
print("\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

n 3
-24 -10 -57
-8 -4 -17

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