NAG C Library Function Document

nag_dgetri (f07ajc)

1 Purpose

nag_dgetri (f07ajc) computes the inverse of a real matrix $A$, where $A$ has been factorized by nag_dgetrf (f07adc).

2 Specification

```c
void nag_dgetri (Nag_OrderType order, Integer n, double a[], Integer pda,
               const Integer ipiv[], NagError *fail)
```

3 Description

To compute the inverse of a real matrix $A$, the function must be preceded by a call to nag_dgetrf (f07adc), which computes the $LU$ factorization of $A$ as $A = PLU$. The inverse of $A$ is computed by forming $U^{-1}$ and then solving the equation $XPL = U^{-1}$ for $X$.

4 References


5 Parameters

1:  
order – Nag_OrderType

On entry: the order parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: order = Nag_RowMajor or Nag_ColMajor.

2:  
n – Integer

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

Constraint: $n \geq 0$.

3:  
a[dim] – double

Note: the dimension, $dim$, of the array $a$ must be at least max(1,$pda \times n$).

If order = Nag_ColMajor, the $(i,j)$th element of the matrix $A$ is stored in $a[(j-1) \times pda + i - 1]$ and if order = Nag_RowMajor, the $(i,j)$th element of the matrix $A$ is stored in $a[(i-1) \times pda + j - 1]$. 

On entry: the LU factorization of $A$, as returned by nag_dgetrf (f07adc).

On exit: the factorization is overwritten by the $n$ by $n$ matrix $A^{-1}$.

4:  
pda – Integer

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

Constraint: $pda \geq \max(1,n)$.
5: \( \text{ipiv}[\text{dim}] \) – const Integer
   \( Input \)
   \( \text{Note: the dimension, } \text{dim}, \text{ of the array } \text{ipiv} \text{ must be at least } \max(1, n). \)
   \( \text{On entry: the pivot indices, as returned by nag_dgetrf (f07adc).} \)

6: \( \text{fail} \) – NagError *
   \( Output \)
   \( \text{The NAG error parameter (see the Essential Introduction).} \)

6  \textbf{Error Indicators and Warnings}

\textbf{NE_INT}
   \( \text{On entry, } n = \langle \text{value} \rangle. \)
   \( \text{Constraint: } n \geq 0. \)
   \( \text{On entry, } pda = \langle \text{value} \rangle. \)
   \( \text{Constraint: } pda > 0. \)

\textbf{NE_INT_2}
   \( \text{On entry, } pda = \langle \text{value} \rangle, \ n = \langle \text{value} \rangle. \)
   \( \text{Constraint: } pda \geq \max(1, n). \)

\textbf{NE_SINGULAR}
   \( \text{Element } \langle \text{value} \rangle \text{ of the diagonal is zero. } U \text{ is singular, and the inverse of } A \text{ cannot be computed.} \)

\textbf{NE_ALLOC_FAIL}
   \( \text{Memory allocation failed.} \)

\textbf{NE_BAD_PARAM}
   \( \text{On entry, parameter } \langle \text{value} \rangle \text{ had an illegal value.} \)

\textbf{NE_INTERNAL_ERROR}
   \( \text{An internal error has occurred in this function. Check the function call and any array sizes. If the} \)
   \( \text{call is correct then please consult NAG for assistance.} \)

7  \textbf{Accuracy}

The computed inverse \( X \) satisfies a bound of the form:
\[ |XA - I| \leq c(n)\epsilon |X||P||L||U|, \]
where \( c(n) \) is a modest linear function of \( n \), and \( \epsilon \) is the \textit{machine precision}.

Note that a similar bound for \( |AX - I| \) cannot be guaranteed, although it is almost always satisfied. See

8  \textbf{Further Comments}

The total number of floating-point operations is approximately \( \frac{4}{3}n^3 \).

The complex analogue of this function is nag_zgetri (f07awc).
9 Example

To compute the inverse of the matrix $A$, where

$$
A = \begin{pmatrix}
  1.80 & 2.88 & 2.05 & -0.89 \\
  5.25 & -2.95 & -0.95 & -3.80 \\
  1.58 & -2.69 & -2.90 & -1.04 \\
 -1.11 & -0.66 & -0.59 & 0.80
\end{pmatrix}
$$

Here $A$ is nonsymmetric and must first be factorized by nag_dgetrf (f07adc).

9.1 Program Text

```c
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
   _sta__int i, ipiv_len, j, n, pda;
   _sta__int exit_status=0;
    NagError fail;
    Nag_OrderType order;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda+I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda+J-1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07ajc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n ] ");
    Vscanf("%d%*[\n ] ", &n);

    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;
    #endif

    ipiv_len = n;

    /* Allocate memory */
    if ( !(a = NAG_ALLOC(n * n, double)) ||
        !(ipiv = NAG_ALLOC(n, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A from data file */
```

for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &A(i,j));
} Vscanf("%*[\n] ");

/* Factorize A */
f07adc(order, n, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07adc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute inverse of A */
f07ajc(order, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07ajc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print inverse */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, a, pda, "Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
if (ipiv) NAG_FREE(ipiv);
return exit_status;

9.2 Program Data

f07ajc Example Program Data

<table>
<thead>
<tr>
<th>Value of N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.80 2.88 2.05 -0.89</td>
</tr>
<tr>
<td>5.25 -2.95 -0.95 -3.80</td>
</tr>
<tr>
<td>1.58 -2.69 -2.90 -1.04</td>
</tr>
<tr>
<td>-1.11 -0.66 -0.59 0.80</td>
</tr>
</tbody>
</table>

:End of matrix A

9.3 Program Results

f07ajc Example Program Results

Inverse

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7720</td>
<td>0.5757</td>
<td>0.0843</td>
<td>4.8155</td>
</tr>
<tr>
<td>-0.1175</td>
<td>-0.4456</td>
<td>0.4114</td>
<td>-1.7126</td>
</tr>
<tr>
<td>0.1799</td>
<td>0.4527</td>
<td>-0.6676</td>
<td>1.4824</td>
</tr>
<tr>
<td>2.4944</td>
<td>0.7650</td>
<td>-0.0360</td>
<td>7.6119</td>
</tr>
</tbody>
</table>