NAG C Library Function Document

nag_zgetri (f07awc)

1 Purpose

nag_zgetri (f07awc) computes the inverse of a complex matrix \( A \), where \( A \) has been factorized by nag_zgetrf (f07arc).

2 Specification

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

3 Description

To compute the inverse of a complex matrix \( A \), the function must be preceded by a call to nag_zgetrf (f07arc), 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  
    Input  
    On entry: the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.  
    Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2:  n – Integer  
    Input  
    On entry: \( n \), the order of the matrix \( A \).  
    Constraint: \( n \geq 0 \).

3:  a[dim] – Complex  
    Input/Output  
    Note: the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \max(1, \text{pda} \times n) \).  
    If \( \text{order} = \text{Nag_ColMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times \text{pda} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(i-1) \times \text{pda} + j - 1] \).  
    On entry: the \( LU \) factorization of \( A \), as returned by nag_zgetrf (f07arc).  
    On exit: the factorization is overwritten by the \( n \) by \( n \) matrix \( A^{-1} \).

4:  pda – Integer  
    Input  
    On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( a \).  
    Constraint: \( \text{pda} \geq \max(1, n) \).
5: \texttt{ipiv[dim]} – const Integer \hfill \textit{Input}

\textbf{Note:} the dimension, \textit{dim}, of the array \texttt{ipiv} must be at least \text{max}(1, n).

\textit{On entry:} the pivot indices, as returned by \texttt{nag_zgetrf (f07arc)}.

6: \texttt{fail} – \texttt{NagError * } \hfill \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 \quad \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

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

\textit{Constraint:} \texttt{n} \geq 0.

\textit{On entry,} \texttt{pda} = \langle\text{value}\rangle.

\textit{Constraint:} \texttt{pda} > 0.

\textbf{NE_INT_2}

\textit{On entry,} \texttt{pda} = \langle\text{value}\rangle, \texttt{n} = \langle\text{value}\rangle.

\textit{Constraint:} \texttt{pda} \geq \text{max}(1, n).

\textbf{NE_SINGULAR}

Element \langle\text{value}\rangle of the diagonal of factor \texttt{U} is zero. \texttt{U} is singular, and the inverse of \texttt{A} cannot be computed.

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

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

\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 consult NAG for assistance.

7 \quad \textbf{Accuracy}

The computed inverse \texttt{X} satisfies a bound of the form:

\[ |\texttt{X} \times \texttt{A} - \texttt{I}| \leq c(n)\epsilon |\texttt{X}| |\texttt{P}| |\texttt{L}| |\texttt{U}|, \]

where \(c(n)\) is a modest linear function of \(n\), and \(\epsilon\) is the \textit{machine precision}.

Note that a similar bound for \(|\texttt{A} \times \texttt{X} - \texttt{I}|\) cannot be guaranteed, although it is almost always satisfied. See Du Croz and Higham (1992).

8 \quad \textbf{Further Comments}

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

The real analogue of this function is \texttt{nag_dgetri (f07ajc)}. 

\texttt{f07awc.2} 

\textit{[NP3645/7]}
9 Example

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

$$A = \begin{pmatrix}
-1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\
-0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\
-3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\
2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \\
\end{pmatrix}.$$ 

Here $A$ is nonsymmetric and must first be factorized by nag_zgetrf (f07arc).

9.1 Program Text

/* nag_zgetri (f07awc) Example Program. 
 * Copyright 2001 Numerical Algorithms Group. 
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>
int main(void)
{
    /* Scalars */
    Integer i, ipiv_len, j, n, pda;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Complex *a=0;
    Integer *ipiv=0;
    #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("f07awc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^n]\n ");
    Vscanf("%ld%[^n] ", &n);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;
    #endif
    ipiv_len = n;
    /* Allocate memory */
    if ( !(a = NAG_ALLOC(n * n, Complex)) ||
        !(ipiv = NAG_ALLOC(ipiv_len, 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 , %lf )", &A(i,j).re, &A(i,j).im);    Vscanf("%*[\n] ");/* Factorize A */    f07arc(order, n, n, a, pda, ipiv, &fail);    if (fail.code != NE_NOERROR)    {        Vprintf("Error from f07arc.
%s
", fail.message);        exit_status = 1;        goto END;    }
/* Compute inverse of A */    f07awc(order, n, a, pda, ipiv, &fail);    if (fail.code != NE_NOERROR)    {        Vprintf("Error from f07awc.
%s
", fail.message);        exit_status = 1;        goto END;    }
/* Print inverse */    x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, a, pda, NagBracketForm, "%7.4f", "Inverse", Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);    if (fail.code != NE_NOERROR)    {        Vprintf("Error from x04dbc.
%s
", 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

f07awc Example Program Data
4 :Value of N
(-1.34, 2.55) (0.28, 3.17) (-6.39, -2.20) (0.72, -0.92)
(-0.17, -1.41) (3.31, -0.15) (-0.15, 1.34) (1.29, 1.38)
(-3.29, -2.39) (-1.91, 4.42) (-0.14, -1.35) (-1.96, 0.67) :End of matrix A

9.3 Program Results

f07awc Example Program Results

Inverse
1   2   3   4
1 (0.0757, -0.4324) (1.6512, -3.1342) (1.2663, 0.0418) (3.8181, 1.1195)
2 (-0.1942, 0.0798) (-1.1900, -0.1426) (-0.2401, -0.5889) (-0.0101, -1.4969)
3 (-0.0957, -0.0491) (0.7371, -0.4290) (0.3224, 0.0776) (0.6887, 0.7891)
4 (0.3702, -0.5040) (3.7253, -3.1813) (1.7014, 0.7267) (3.9367, 3.3255)