NAG C Library Function Document

nag_zhetri (f07mwc)

1 Purpose

nag_zhetri (f07mwc) computes the inverse of a complex Hermitian indefinite matrix \( A \), where \( A \) has been factorized by nag_zhetrf (f07mrc).

2 Specification

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

3 Description

To compute the inverse of a complex Hermitian indefinite matrix \( A \), this function must be preceded by a call to nag_zhetrf (f07mrc), which computes the Bunch–Kaufman factorization of \( A \).

If \( \text{uplo} = \text{Nag_Upper} \), \( A = PUDU^H P^T \) and \( A^{-1} \) is computed by solving \( U^H P^T XPU = D^{-1} \) for \( X \).

If \( \text{uplo} = \text{Nag_Lower} \), \( A = PLDL^H P^T \) and \( A^{-1} \) is computed by solving \( L^H P^T XPL = D^{-1} \) for \( X \).

4 References


5 Parameters

1: \( \text{order} \) – 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: \( \text{uplo} \) – Nag_UploType

*Input*

On entry: indicates how \( A \) has been factorized as follows:

- if \( \text{uplo} = \text{Nag_Upper} \), \( A = PUDU^H P^T \), where \( U \) is upper triangular;
- if \( \text{uplo} = \text{Nag_Lower} \), \( A = PLDL^H P^T \), where \( L \) is lower triangular.

Constraint: \( \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

3: \( \text{n} \) – Integer

*Input*

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

Constraint: \( n \geq 0 \).

4: \( \text{a}[\text{dim}] \) – Complex

*Input/Output*

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

On entry: details of the factorization of \( A \), as returned by nag_zhetrf (f07mrc).
On exit: the factorization is overwritten by the \( n \) by \( n \) Hermitian matrix \( A^{-1} \). If \( \text{uplo} = \text{Nag}_\text{Upper} \), the upper triangle of \( A^{-1} \) is stored in the upper triangular part of the array; if \( \text{uplo} = \text{Nag}_\text{Lower} \), the lower triangle of \( A^{-1} \) is stored in the lower triangular part of the array.

5: \( \text{pda} \) – Integer

\text{Input}

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

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

6: \( \text{ipiv}[\text{dim}] \) – const Integer

\text{Input}

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

On entry: details of the interchanges and the block structure of \( D \), as returned by \text{nag_zhetrf} (f07mrc).

7: \( \text{fail} \) – NagError *

\text{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\text{NE_INT}

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).

Constraint: \( \text{pda} > 0 \).

\text{NE_INT_2}

On entry, \( \text{pda} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).

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

\text{NE_SINGULAR}

The block diagonal matrix \( D \) is exactly singular.

\text{NE_ALLOC_FAIL}

Memory allocation failed.

\text{NE_BAD_PARAM}

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

\text{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 Accuracy

The computed inverse \( X \) satisfies a bound of the form

\[ |DU^H P^T X PU - I| \leq c(n)\epsilon(|D| |U^H P^T X P|U | + |D| |D^{-1}|); \]

\[ |DL^H P^T X PL - I| \leq c(n)\epsilon(|D| |L^H P^T X P|L | + |D| |D^{-1}|), \]

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

\text{f07mwc.2} [NP3645/7]
8 Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.
The real analogue of this function is nag_dsytri (f07mjc).

9 Example

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

$$A = \begin{pmatrix}
-1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\
1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\
2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\
3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i
\end{pmatrix}.$$ 

Here $A$ is Hermitian indefinite and must first be factorized by nag_zhetrf (f07mrc).

9.1 Program Text

/* nag_zhetri (f07mwc) 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, j, n, pda;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_MatrixType matrix;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    Complex *a=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

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

    INIT_FAIL(fail);
    Vprintf("f07mwc Example Program Results\n\n");

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

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

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||

! (a = NAG_ALLOC(n * n, Complex))

    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

/* Read A from data file */
Vscanf(" ' %ls ' *\n", uplo);
if (*(unsigned char *) uplo == 'L')
    {
        uplo_enum = Nag_Lower;
        matrix = Nag_LowerMatrix;
    }
else if (*(unsigned char *) uplo == 'U')
    {
        uplo_enum = Nag_Upper;
        matrix = Nag_UpperMatrix;
    }
else
    {
        Vprintf("Unrecognised character for Nag_UploType type\n");
        exit_status = -1;
        goto END;
    }

if (uplo_enum == Nag_Upper)
    {
        for (i = 1; i <= n; ++i)
            {
                for (j = i; j <= n; ++j)
                    Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
            }
    }
else
    {
        for (i = 1; i <= n; ++i)
            {
                for (j = 1; j <= i; ++j)
                    Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
            }
    }

/* Factorize A */
f07mrc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07mrc.\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Compute inverse of A */
f07mwc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07mwc.\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Print inverse */
x04dbc(order, matrix, Nag_NonUnitDiag, n, n, a, pda, Nag_BracketForm,
    "%7.4f", "Inverse", Nag_IntegerLabels, 0,
    Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04dbc.\n", fail.message);
        exit_status = 1;
        goto END;
    }

f07mwc.4

NAG C Library Manual
9.2 Program Data

f07mwc Example Program Data

4 :Value of N
'L' :Value of UPLO

(-1.36, 0.00)
(1.58, -0.90) (-8.87, 0.00)
(2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
(3.91, -1.50) (-1.78, -1.18) (0.11, -0.11) (-1.84, 0.00) :End of matrix A

9.3 Program Results

f07mwc Example Program Results

Inverse

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.0826, 0.0000)</td>
<td>(0.0000</td>
<td>(0.0000</td>
<td>(0.0000</td>
</tr>
<tr>
<td>2</td>
<td>(-0.0335, 0.0440)</td>
<td>(-0.1408, 0.0000)</td>
<td>(-0.2007, -0.0000)</td>
<td>(-0.2007, -0.0000)</td>
</tr>
<tr>
<td>3</td>
<td>(0.0603, -0.0105)</td>
<td>(0.0422, -0.0222)</td>
<td>(0.0982, -0.0635)</td>
<td>(0.0982, -0.0635)</td>
</tr>
<tr>
<td>4</td>
<td>(0.2391, -0.0926)</td>
<td>(0.0304, 0.0203)</td>
<td>(0.0973, 0.0000)</td>
<td>(0.0973, 0.0000)</td>
</tr>
</tbody>
</table>