NAG 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
#include <nag.h>
#include <nagf07.h>
void nag_zhetri (Nag_OrderType order, Nag_UploType uplo, Integer n,
     Complex a[], Integer pda, const Integer ipiv[], NagError *fail)
```

3 Description

nag_zhetri (f07mwc) is used to compute the inverse of a complex Hermitian indefinite matrix $A$, the function must be preceded by a call to nag_zhetrf (f07mrc), which computes the Bunch–Kaufman factorization of $A$.

If $\text{uplo} = \text{Nag_Upper}$, $A = PUDU^HP^T$ and $A^{-1}$ is computed by solving $U^HP^TXU = D^{-1}$ for $X$.

If $\text{uplo} = \text{Nag_Lower}$, $A = PLDL^HP^T$ and $A^{-1}$ is computed by solving $L^HP^TXPL = D^{-1}$ for $X$.

4 References


5 Arguments

1:  
   \textbf{order} – Nag_OrderType
   
   \textit{Input}
   
   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:  
   \textbf{uplo} – Nag_UploType
   
   \textit{Input}
   
   On entry: specifies how $A$ has been factorized.

   $\text{uplo} = \text{Nag_Upper}$
   
   $A = PUDU^HP^T$, where $U$ is upper triangular.

   $\text{uplo} = \text{Nag_Lower}$
   
   $A = PLDL^HP^T$, where $L$ is lower triangular.

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

3:  
   \textbf{n} – Integer
   
   \textit{Input}
   
   On entry: $n$, the order of the matrix $A$.

   Constraint: $n \geq 0$. 

Mark 25
4:  \(a[\text{dim}]\) – Complex
    \(\text{Input/Output}\)

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

    \textit{On entry}: details of the factorization of \(A\), as returned by \text{nag_zhetrf} (f07mrc).

    \textit{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
    \(\text{Input}\)

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

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

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

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

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

7:  \(\text{fail}\) – \text{NagError} *
    \(\text{Input/Output}\)

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

6 \quad \textbf{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 \(\langle\text{value}\rangle\) had an illegal value.

\textbf{NE_INT}

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

    Constraint: \(n \geq 0\).

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

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

\textbf{NE_INT_2}

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

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

\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.
Element (value) of the diagonal is exactly zero. D is singular and the inverse of A cannot be computed.

7 Accuracy
The computed inverse X satisfies a bound of the form

$$|D^{H}P^{T}XPU - I| \leq c(n)\epsilon\left(||D||^{H}||P^{T}|X|P|U| + ||D||D^{-1})\right);$$

if uplo = Nag_Upper,

$$|D^{H}P^{T}XPL - I| \leq c(n)\epsilon\left(||D||^{H}||P^{T}|X|P|L| + ||D||D^{-1})\right);$$

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

8 Parallelism and Performance
nag_zhetri (f07mwc) is not threaded by NAG in any implementation.

nag_zhetri (f07mwc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments
The total number of real floating-point operations is approximately $8n^{3}$.

The real analogue of this function is nag_dsytri (f07mjc).

10 Example
This example computes 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).

10.1 Program Text
/* nag_zhetri (f07mwc) Example Program.  
 * Copyright 2014 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;
Nag_MatrixType matrix;
Nag_OrderType order;
/* Arrays */
Integer *ipiv = 0;
char nag_enum_arg[40];
Complex *a = 0;
#ifdef NAG_LOAD_FP
/* The following line is needed to force the Microsoft linker
to load floating point support */
float force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

#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);
printf("nag_zhetri (f07mwc) Example Program Results

");
/* 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
#ifdef NAG_COLUMN_MAJOR
pda = n;
#else
pda = n;
#endif
/* Allocate memory */
if (!(ipiv = NAG_ALLOC(n, Integer)) || 
!(a = NAG_ALLOC(n * n, Complex)))
{
printf("Allocation failure\n"");
exit_status = -1;
goto END;
}
/* Read A from data file */
#ifdef _WIN32
scanf_s("%39s%*[\n ]", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s%*[\n ]", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    matrix = Nag_UpperMatrix;
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
#ifdef _WIN32
for (i = 1; i <= n; ++i)
for (j = i; j <= n; ++j)
```c
#else
scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
#else
scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
#else
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif
else
{
    matrix = Nag_LowerMatrix;
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
        {
            scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
    }
    scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif

/* Factorize A */
/* nag_zhetrf (f07mrc). *
  * Bunch-Kaufman factorization of complex Hermitian *
  * indefinite matrix *
*/
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhetrf (f07mrc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute inverse of A */
/* nag_zhetri (f07mwc). *
  * Inverse of complex Hermitian indefinite matrix, matrix *
  * already factorized by nag_zhetrf (f07mrc) *
*/
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhetri (f07mwc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print inverse */
/* nag_gen_complx_mat_print_comp (x04dbc). *
  * Print complex general matrix (comprehensive) *
*/
fflush(stdout);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
```

Mark 25

f07mwc.5
10.2 Program Data

nag_zhetri (f07mwc) Example Program Data
4 :Value of n
   Nag_Lower :Value of uplo
   (-1.36, 0.00) (-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

10.3 Program Results

nag_zhetri (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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(-0.0335, 0.0440)</td>
<td>(-0.1408, 0.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( 0.0603,-0.0105)</td>
<td>( 0.0422,-0.0222)</td>
<td>(-0.2007,-0.0000)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( 0.2391,-0.0926)</td>
<td>( 0.0304, 0.0203)</td>
<td>( 0.0982,-0.0635)</td>
<td>( 0.0073, 0.0000)</td>
</tr>
</tbody>
</table>