NAG C Library Function Document

nag_zpotri (f07fwc)

1 Purpose

nag_zpotri (f07fwc) computes the inverse of a complex Hermitian positive-definite matrix $A$, where $A$ has been factorized by nag_zpotrf (f07frc).

2 Specification

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

3 Description

To compute the inverse of a complex Hermitian positive-definite matrix $A$, the function must be preceded by a call to nag_zpotrf (f07frc), which computes the Cholesky factorization of $A$.

If $\text{uplo} = \text{Nag_Upper}$, $A = U^H U$ and $A^{-1}$ is computed by first inverting $U$ and then forming $(U^{-1})(U^{-1})^H$.

If $\text{uplo} = \text{Nag_Lower}$, $A = LL^H$ and $A^{-1}$ is computed by first inverting $L$ and then forming $(L^{-1})^H(L^{-1})$.

4 References


5 Parameters

1: order – Nag_OrderType

Input

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: uplo – Nag_UploType

Input

On entry: indicates whether $A$ has been factorized as $U^H U$ or $LL^H$ as follows:

- if $\text{uplo} = \text{Nag_Upper}$, $A = U^H U$, where $U$ is upper triangular;
- if $\text{uplo} = \text{Nag_Lower}$, $A = LL^H$, where $L$ is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer

Input

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

Constraint: $n \geq 0$.

4: a[dim] – Complex

Input/Output

Note: the dimension, dim, of the array a must be at least max(1,pda × n).
On entry: the upper triangular matrix $U$ if $\text{uplo} = \text{Nag_Upper}$ or the lower triangular matrix $L$ if $\text{uplo} = \text{Nag_Lower}$, as returned by nag_zpotrf (f07frc).

On exit: $U$ is overwritten by the upper triangle of $A^{-1}$ if $\text{uplo} = \text{Nag_Upper}$; $L$ is overwritten by the lower triangle of $A^{-1}$ if $\text{uplo} = \text{Nag_Lower}$.

5:  

$pda$ – 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 $a$.

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

6:  

$\text{fail}$ – NagError $\ast$

$\text{Output}$

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

Constraint: $n \geq 0$.

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

Constraint: $pda > 0$.

NE_INT_2

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

Constraint: $pda \geq \max(1, n)$.

NE_SINGULAR

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

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

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

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

$$
\|XA - I\|_2 \leq c(n)\kappa_2(A) \quad \text{and} \quad \|AX - I\|_2 \leq c(n)\kappa_2(A),
$$

where $c(n)$ is a modest function of $n$, $\epsilon$ is the machine precision and $\kappa_2(A)$ is the condition number of $A$ defined by

$$
\kappa_2(A) = \|A\|_2\|A^{-1}\|_2.
$$

8 Further Comments

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

f07fwc.2 [NP3645/7]
The real analogue of this function is nag_dpotri (f07fjc).

9 Example

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

$$A = \begin{pmatrix}
3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\
1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\
1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\
0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i
\end{pmatrix}.$$  

Here $A$ is Hermitian positive-definite and must first be factorized by nag_zpotrf (f07frc).

9.1 Program Text

/* nag_zpotri (f07fwc) Example Program.*/
* Copyright 2001 Numerical Algorithms Group.
* Mark 7, 2001.*

#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 */
    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

    INIT_FAIL(fail);
    Vprintf("f07fwc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    Vscanf("%ld%*[\n] ", &n);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;
    #endif

    /* Allocate memory */
    if ( !(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);
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
    Vscanf("%*[\n] ");
}
/* Factorize A */
f07frc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07frc.\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute inverse of A */
f07fwc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fwc.\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print inverse */
x04dbc(order, matrix, 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.\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
return exit_status;
9.2 Program Data

f07fwc Example Program Data

<table>
<thead>
<tr>
<th>Value of N</th>
<th>Value of UPLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>'L'</td>
<td></td>
</tr>
<tr>
<td>(3.23, 0.00)</td>
<td></td>
</tr>
<tr>
<td>(1.51, 1.92)</td>
<td>(3.58, 0.00)</td>
</tr>
<tr>
<td>(1.90, -0.84)</td>
<td>(-0.23, -1.11)</td>
</tr>
<tr>
<td>(0.42, -2.50)</td>
<td>(-1.18, -1.37)</td>
</tr>
</tbody>
</table>

:End of matrix A

9.3 Program Results

f07fwc 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>(5.4691, 0.0000)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(-1.2624, -1.5491)</td>
<td>(1.1024, 0.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(-2.9746, -0.9616)</td>
<td>(0.8989, -0.5672)</td>
<td>(2.1589, 0.0000)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(1.1962, 2.9772)</td>
<td>(-0.9826, -0.2566)</td>
<td>(-1.3756, -1.4550)</td>
<td>(2.2934, 0.0000)</td>
</tr>
</tbody>
</table>