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

```
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 **uplo** = **Nag_Upper**, $A = PUDU^H P^T$ and A^{-1} is computed by solving $U^H P^T XPU = D^{-1}$ for X.

If **uplo** = **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

Du Croz J J and Higham N J (1992) Stability of methods for matrix inversion *IMA J. Numer. Anal.* **12** 1–19

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 how A has been factorized as follows:

if **uplo** = **Nag_Upper**, $A = PUDU^H P^T$, where U is upper triangular;

if **uplo** = **Nag_Lower**, $A = PLDL^H P^T$, 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** ≥ 0.

4: **a**[dim] – Complex *Input/Output*

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

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 **uplo** = **Nag_Upper**, the upper triangle of A^{-1} is stored in the upper triangular part of the array; if **uplo** = **Nag_Lower**, the lower triangle of A^{-1} is stored in the lower triangular part of the array.

5: **pda** – Integer *Input*

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

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

6: **ipiv**[dim] – const Integer *Input*

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

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

7: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: $\mathbf{n} \geq 0$.

On entry, **pda** = ⟨*value*⟩.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.

NE_SINGULAR

The block diagonal matrix D is exactly singular.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ 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 a bound of the form

if **uplo** = **Nag_Upper**, $|DU^H P^T XPU - I| \leq c(n)\epsilon(|D|\,|U^H|P^T|X|P|U| + |D|\,|D^{-1}|)$;

if **uplo** = **Nag_Lower**, $|DL^H P^T XPL - I| \leq c(n)\epsilon(|D|\,|L^H|P^T|X|P|L| + |D|\,|D^{-1}|)$,

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

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.
 *
 * 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 */
  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

  INIT_FAIL(fail);
  Vprintf("f07mwc 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 ( !(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(" ' %1s '%*[^\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 */
  f07mrc(order, uplo_enum, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07mrc.\n%s\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%s\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%s\n",fail.message);
      exit_status = 1;
      goto END;
    }
```

```
 END:
  if (ipiv) NAG_FREE(ipiv);
  if (a) NAG_FREE(a);
  return exit_status;
}
```

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
                   1                   2                   3                   4
 1  ( 0.0826, 0.0000)
 2  (-0.0335, 0.0440)  (-0.1408, 0.0000)
 3  ( 0.0603,-0.0105)  ( 0.0422,-0.0222)  (-0.2007,-0.0000)
 4  ( 0.2391,-0.0926)  ( 0.0304, 0.0203)  ( 0.0982,-0.0635)  ( 0.0073, 0.0000)
```