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

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

If **uplo** $=$ Nag_Lower, $A = PLDL^{\mathrm{H}}P^{\mathrm{T}}$ and A^{-1} is computed by solving $L^{\mathrm{H}}P^{\mathrm{T}}XPL = D^{-1}$ 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 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **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 **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: **order** $=$ Nag_RowMajor or Nag_ColMajor.

2: **uplo** – Nag_UploType *Input*

On entry: specifies how A has been factorized.

uplo $=$ Nag_Upper
$A = PUDU^{\mathrm{H}}P^{\mathrm{T}}$, where U is upper triangular.

uplo $=$ Nag_Lower
$A = PLDL^{\mathrm{H}}P^{\mathrm{T}}$, 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 $\max(1, \mathbf{pda} \times \mathbf{n})$.

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 * *Input/Output*

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

On entry, **pda** $= \langle value \rangle$.
Constraint: **pda** > 0.

NE_INT_2

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

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.

NE_SINGULAR

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

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

$c(n)$ is a modest linear function of n, and ϵ 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 Users' Note for your implementation for any additional implementation-specific information.

9 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).

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 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;
  Nag_MatrixType matrix;
  Nag_OrderType  order;
  /* Arrays */
  Integer        *ipiv = 0;
  char           nag_enum_arg[40];
  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);

  printf("nag_zhetri (f07mwc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%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)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }


  /* Read A from data file */
  scanf(" %39s%*[^\n] ", nag_enum_arg);
  /* 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)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }
  else
    {
      matrix = Nag_LowerMatrix;
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }

  /* Factorize A */
  /* nag_zhetrf (f07mrc).
   * Bunch-Kaufman factorization of complex Hermitian
   * indefinite matrix
   */
  nag_zhetrf(order, uplo, n, a, pda, ipiv, &fail);
  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)
   */
  nag_zhetri(order, uplo, n, a, pda, ipiv, &fail);
  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);
  nag_gen_complx_mat_print_comp(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)
    {
      printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(ipiv);
  NAG_FREE(a);
  return exit_status;
}
```

10.2 Program Data

```
nag_zhetri (f07mwc) Example Program Data
  4                                                        :Value of n
  Nag_Lower                                                :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
```

10.3 Program Results

```
nag_zhetri (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)
```