NAG C Library Function Document

nag_zsptri (f07qwc)

1 Purpose
nag_zsptri (f07qwc) computes the inverse of a complex symmetric matrix A, where A has been factorized by nag_zsptrfs (f07qrc), using packed storage.

2 Specification

void nag_zsptri (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex ap[], const Integer ipiv[], NagError *fail)

3 Description
To compute the inverse of a complex symmetric matrix A, this function must be preceded by a call to nag_zsptrfs (f07qrc), which computes the Bunch–Kaufman factorization of A using packed storage.

If uplo = Nag_Upper, A = PUDU^T P^T and A^{-1} is computed by solving U^T P^T XPU = D^{-1}.

If uplo = Nag_Lower, A = PLDL^T P^T and A^{-1} is computed by solving L^T P^T XPL = D^{-1}.

4 References

5 Parameters
1: order – Nag_OrderType
   
   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
   
   On entry: indicates how A has been factorized as follows:
   
   if uplo = Nag_Upper, A = PUDU^T P^T, where U is upper triangular;
   
   if uplo = Nag_Lower, A = PLDL^T P^T, where L is lower triangular.

   Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer
   
   On entry: n, the order of the matrix A.

   Constraint: n ≥ 0.

4: ap[dimensions] – Complex
   
   Note: the dimension, dim, of the array ap must be at least max(1,n × (n + 1)/2).

   On entry: details of the factorization of A stored in packed form, as returned by nag_zsptrfs (f07qre).

   On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1} stored in packed form.
5: \( \text{ipiv}[\text{dim}] \) – \text{const Integer} \hspace{1cm} \text{Input}

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

\textbf{On entry:} details of the interchanges and the block structure of \( D \), as returned by \text{nag_zsptrf} (f07qrc).

6: \( \text{fail} \) – \text{NagError *} \hspace{1cm} \text{Output}

The NAG error parameter (see the Essential Introduction).

6 \ Error Indicators and Warnings

\textbf{NE_INT}

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

Constraint: \( n \geq 0 \).

\textbf{NE_SINGULAR}

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

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

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

\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 consult NAG for assistance.

7 \ Accuracy

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

\[
|D^{UH} P^T X P L - I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U| + |D||D^{-1}|),
\]

\[
|D^{LH} P^T X P L - I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L| + |D||D^{-1}|),
\]

where \( c(n) \) is a modest linear function of \( n \), and \( \varepsilon \) is the \emph{machine precision}.

8 \ Further Comments

The total number of real floating-point operations is approximately \( \frac{2}{3}n^3 \).

The real analogue of this function is \text{nag_dsptri} (f07pjc).

9 \ Example

To compute the inverse of the matrix \( A \), where

\[
A = \begin{pmatrix}
-0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\
5.14 - 0.64i & 8.86 + 1.30i & -3.52 + 0.58i & 5.32 - 1.59i \\
-7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\
3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i
\end{pmatrix}.
\]

Here \( A \) is symmetric, stored in packed form, and must first be factorized by \text{nag_zsptrf} (f07qrc).
9.1 Program Text

/* nag_zsptri (f07qwc) 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 ap_len, i, j, n;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;

    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    Complex *ap=0;

    #ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
    #else
    #define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
    #define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07qwc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n ] ");
    Vscanf("%ld%*[\n ] ", &n);
    ap_len = n * (n + 1)/2;

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(ap = NAG_ALLOC(ap_len, 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;
    else if (*(unsigned char *)uplo == 'U')
        uplo_enum = Nag_Upper;
    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)
        {....

[NP3645/7] f07qwc.3

f07 – Linear Equations (LAPACK)
for (j = i; j <= n; ++j)
    Vscanf(" ( %lf , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
} 
Vscanf("%*[\n ] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
    } 
Vscanf("%*[\n ] ");
}

/* Factorize A */
f07qrc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07qrc.
%s
", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute inverse of A */
f07qwc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07qwc.
%s
", fail.message);
    exit_status = 1;
    goto END;
}
/* Print inverse */
x04ddc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
    Nag_BracketForm, "%7.4f", "Inverse", Nag_IntegerLabels,
    0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04ddc.
%s
", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ipiv) NAG_FREE(ipiv);
if (ap) NAG_FREE(ap);
return exit_status;
}

9.2 Program Data
f07qwc Example Program Data

4 :Value of N
'L' :Value of UPLO
(-0.39,-0.71)
( 5.14,-0.64) ( 8.86, 1.81)
(-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12) :End of matrix A

9.3 Program Results
f07qwc Example Program Results
Inverse

\begin{tabular}{cccc}
1 & 2 & 3 & 4 \\
-0.1562 & -0.1014 & 0.0946 & -0.1475 \\
0.0400 & 0.1527 & -0.0326 & -0.1370 \\
0.0550 & 0.0845 & -0.1320 & -0.0102 \\
0.2162 & -0.0742 & -0.0995 & 0.2383 \\
\end{tabular}