NAG C Library Function Document

**nag_zpptri (f07gwc)**

1 **Purpose**

*nag_zpptri (f07gwc)* computes the inverse of a complex Hermitian positive-definite matrix \( A \), where \( A \) has been factorized by *nag_zpptrf (f07grc)*, using packed storage.

2 **Specification**

```c
void nag_zpptri (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex ap[], NagError *fail)
```

3 **Description**

To compute the inverse of a complex Hermitian positive-definite matrix \( A \), this function must be preceded by a call to *nag_zpptrf (f07grc)*, which computes the Cholesky factorization of \( A \) using packed storage.

If \( \text{uplo} = \text{Nag\_Upper} \), \( A = UU^H \) 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: \text{order} – Nag\_OrderType

*Input*

*On entry*: the \text{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 \text{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*: \text{order} = Nag\_RowMajor or Nag\_ColMajor.

2: \text{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*: \text{uplo} = Nag\_Upper or Nag\_Lower.

3: \text{n} – Integer

*Input*

*On entry*: \( n \), the order of the matrix \( A \).

*Constraint*: \( n \geq 0 \).

4: \text{ap}[\text{dim}] – Complex

*Input/Output*

*Note*: the dimension, \( \text{dim} \), of the array \text{ap} must be at least \( \max(1, n \times (n + 1)/2) \).
On entry: the upper triangular matrix $U$ stored in packed form if uplo = Nag_Upper or the lower triangular matrix $L$ stored in packed form if uplo = Nag_Lower, as returned by nag_zpptrf (f07grc).

On exit: $U$ is overwritten by the upper triangle of $A^{-1}$ if uplo = Nag_Upper; $L$ is overwritten by the lower triangle of $A^{-1}$ if uplo = Nag_Lower, using the same storage scheme as on entry.

5: fail – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, n = ⟨value⟩.

Constraint: n ≥ 0.

NE_SINGULAR

Element ⟨value⟩ 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 ⟨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

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

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 $\frac{4}{3}n^3$.

The real analogue of this function is nag_dpstr (f07gjc).

9 Example

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

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

### NAG C Library Manual

f07gwc.2 [NP3645/7]
Here $A$ is Hermitian positive-definite, stored in packed form, and must first be factorized by nag_zpptrf (f07grc).

### 9.1 Program Text

```c
#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 */
    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("f07gwc 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 ( !(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)
    {
        /* Linear Equations (LAPACK)
        f07 – Linear Equations (LAPACK)
        f07gwc
        */
    }
}
```
for (i = 1; i <= n; ++i)
{
    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 */
f07grc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07grc.
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute inverse of A */
f07gwc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07gwc.
%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 (ap) NAG_FREE(ap);
return exit_status;
}

9.2 Program Data

f07gwc Example Program Data
4
'L'
(3.23, 0.00)
(1.51, 1.92) ( 3.58, 0.00)
(1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
(0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix A

9.3 Program Results

f07gwc Example Program Results

Inverse 1 2 3 4
1 ( 5.4691, 0.0000)
2 (-1.2624,-1.5491) ( 1.1024, 0.0000)
3 (-2.9746,-0.9616) ( 0.8989,-0.5672) ( 2.1589, 0.0000)
4 ( 1.1962, 2.9772) (-0.9826,-0.2566) (-1.3756,-1.4550) ( 2.2934, 0.0000)