NAG C Library Function Document

nag_zppcon (f07guc)

1 Purpose

nag_zppcon (f07guc) estimates the condition number 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_zppcon (Nag_OrderType order, Nag_UploType uplo, Integer n,
                  const Complex *ap[], double anorm, double *rcond, NagError *fail)
```

3 Description

nag_zppcon (f07guc) estimates the condition number (in the 1-norm) of a complex Hermitian positive-definite matrix \( A \):

\[
\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.
\]

Since \( A \) is Hermitian, \( \kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty. \)

Because \( \kappa_1(A) \) is infinite if \( A \) is singular, the function actually returns an estimate of the reciprocal of \( \kappa_1(A) \).

The function should be preceded by a call to nag_zhp_norm (f16udc) to compute \( \|A\|_1 \) and a call to nag_zpptrf (f07grc) to compute the Cholesky factorization of \( A \). The function then uses Higham’s implementation of Hager’s method (see Higham (1988)) to estimate \( \|A^{-1}\|_1 \).

4 References

Higham N J (1988) FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation ACM Trans. Math. Software 14 381–396

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 \( \text{order} = \text{Nag_RowMajor} \). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
  - Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

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: \( \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

3: \( n \) – Integer

- **Input**
  - On entry: \( n \), the order of the matrix \( A \).
  - Constraint: \( n \geq 0 \).
4: \( \text{ap}[\text{dim}] \) – const Complex

\textbf{Input}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \text{ap} must be at least \( \max(1, n \times (n + 1)/2) \).

\textit{On entry:} the Cholesky factor of \( A \) stored in packed form, as returned by \text{nag_zpptrf (f07grc)}.

5: \( \text{anorm} \) – double

\textbf{Input}

\textit{On entry:} the 1-norm of the original matrix \( A \), which may be computed by calling \text{nag_zhp_norm (f16udc)}. \text{anorm} must be computed either before calling \text{nag_zpptrf (f07grc)} or else from a copy of the original matrix \( A \).

\textit{Constraint:} \( \text{anorm} \geq 0.0 \).

6: \( \text{rcond} \) – double *

\textbf{Output}

\textit{On exit:} an estimate of the reciprocal of the condition number of \( A \). \text{rcond} is set to zero if exact singularity is detected or the estimate underflows. If \text{rcond} is less than \textit{machine precision}, \( A \) is singular to working precision.

7: \( \text{fail} \) – NagError *

The NAG error parameter (see the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

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

\textit{Constraint:} \( n \geq 0 \).

\textbf{NE_REAL}

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

\textit{Constraint:} \( \text{anorm} \geq 0.0 \).

\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 \textbf{Accuracy}

The computed estimate \( \text{rcond} \) is never less than the true value \( \rho \), and in practice is nearly always less than \( 10\rho \), although examples can be constructed where \( \text{rcond} \) is much larger.

8 \textbf{Further Comments}

A call to \text{nag_zppcon (f07guc)} involves solving a number of systems of linear equations of the form \( Ax = b \); the number is usually 5 and never more than 11. Each solution involves approximately \( 8n^2 \) real floating-point operations but takes considerably longer than a call to \text{nag_zpptrs (f07gsc)} with 1 right-hand side, because extra care is taken to avoid overflow when \( A \) is approximately singular.

The real analogue of this function is \text{nag_dppcon (f07gge)}.
9 Example

To estimate the condition number in the 1-norm (or infinity-norm) 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, stored in packed form, and must first be factorized by nag_zpptrf (f07grc). The true condition number in the 1-norm is 201.92.

9.1 Program Text

```c
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf07.h>
#include <nagf16.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double anorm, rcond;
    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("f07guc 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(" %ls \%*[\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)
    {
        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] ");
}
/* Compute norm of A */
fl6udc(order, Nag_OneNorm, uplo_enum, n, ap, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from fl6udc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Factorize A */
f07grc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07grc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Estimate condition number */
f07guc(order, uplo_enum, n, ap, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07guc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
if (rcond >= X02AJC)
    Vprintf("Estimate of condition number =%10.2e\n\n", 1.0/rcond);
else
{

}
END:
if (ap) NAG_FREE(ap);
return exit_status;
}
9.2 Program Data

f07guc Example Program Data

:Value of N
4

:Value of UPLO
‘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

f07guc Example Program Results

Estimate of condition number = 1.51e+02