NAG C Library Function Document

nag_ztpcon (f07uuc)

1 Purpose

nag_ztpcon (f07uuc) estimates the condition number of a complex triangular matrix, using packed storage.

2 Specification

void nag_ztpcon (Nag_OrderType order, Nag_NormType norm, Nag_UploType uplo,
   Nag_DiagType diag, Integer n, const Complex *ap[], double *rcond,
   NagError *fail)

3 Description

nag_ztpcon (f07uuc) estimates the condition number of a complex triangular matrix $A$, in either the $1$-norm or the infinity-norm, using packed storage:

$$
\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.
$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if $A$ is singular, the function actually returns an estimate of the reciprocal of the condition number.

The function computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham’s implementation of Hager’s method (Higham (1988)) to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

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: 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: norm – Nag_NormType

   On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

   if norm = Nag_OneNorm, then $\kappa_1(A)$ is estimated;
   if norm = Nag_InfNorm, then $\kappa_\infty(A)$ is estimated.

   Constraint: norm = Nag_OneNorm or Nag_InfNorm.

3: uplo – Nag_UploType

   On entry: indicates whether $A$ is upper or lower triangular as follows:
if uplo = Nag_Upper, \( A \) is upper triangular;
if uplo = Nag_Lower, \( A \) is lower triangular.

**Constraint:** \( \text{uplo} = \text{Nag\_Upper} \) or \( \text{Nag\_Lower} \).

4: \( \text{diag} \) – Nag_DiagType

*Input*

\( \text{On entry:} \) indicates whether \( A \) is a non-unit or unit triangular matrix as follows:

- if \( \text{diag} = \text{Nag\_NonUnitDiag} \), \( A \) is a non-unit triangular matrix;
- if \( \text{diag} = \text{Nag\_UnitDiag} \), \( A \) is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

**Constraint:** \( \text{diag} = \text{Nag\_NonUnitDiag} \) or \( \text{Nag\_UnitDiag} \).

5: \( n \) – Integer

*Input*

\( \text{On entry:} \) \( n \), the order of the matrix \( A \).

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

6: \( \text{ap}[\text{dim}] \) – const Complex

*Input*

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

\( \text{On entry:} \) the \( n \) by \( n \) triangular matrix \( A \), packed by rows or columns. The storage of elements \( a_{ij} \) depends on the \( \text{order} \) and \( \text{uplo} \) parameters as follows:

- if \( \text{order} = \text{Nag\_ColMajor} \) and \( \text{uplo} = \text{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[(j-1) \times j/2 + i - 1] \), for \( i \leq j \);
- if \( \text{order} = \text{Nag\_ColMajor} \) and \( \text{uplo} = \text{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(2n-j) \times (j-1)/2 + i - 1] \), for \( i \geq j \);
- if \( \text{order} = \text{Nag\_RowMajor} \) and \( \text{uplo} = \text{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[(2n-i) \times (i-1)/2 + j - 1] \), for \( i \leq j \);
- if \( \text{order} = \text{Nag\_RowMajor} \) and \( \text{uplo} = \text{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(i-1) \times i/2 + j - 1] \), for \( i \geq j \).

7: \( \text{rcond} \) – double *

*Output*

\( \text{On exit:} \) 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 *machine precision*, \( A \) is singular to working precision.

8: \( \text{fail} \) – NagError *

*Output*

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE\_INT**

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

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

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) 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 estimate 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 rcond is much larger.

8 Further Comments
A call to nag_ztpcon (f07uuc) involves solving a number of systems of linear equations of the form
$Ax = b$ or $A^Hx = b$; the number is usually 5 and never more than 11. Each solution involves
approximately $4n^2$ real floating-point operations but takes considerably longer than a call to nag_ztptrs
(f07usc) with one right-hand side, because extra care is taken to avoid overflow when $A$ is approximately
singular.

The real analogue of this function is nag_dtpcon (f07ugc).

9 Example
To estimate the condition number in the 1-norm of the matrix

$$
A = \begin{pmatrix}
4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\
2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\
-1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i
\end{pmatrix},
$$

using packed storage. The true condition number in the 1-norm is 70.27.

9.1 Program Text
/* nag_ztpcon (f07uuc) Example Program.
 * Copyright 2001 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double rcond;
    Integer ap_len, i, j, n;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    Complex *ap=&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]
    #else
    #endif

    #ifndef A_UPPER
    define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
    #endif
    #ifndef A_LOWER
    define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    #endif

    order = Nag_ColMajor;
#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("f07uuc Example Program Results\n");

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

/* Allocate memory */
if ( !(ap = NAG_ALLOC(n * n, 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 ]");
}

/* Estimate condition number */
f07uuc(order, Nag_OneNorm, uplo_enum, Nag_NonUnitDiag, n, ap, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07uuc.\n%\ns\n", fail.message);
    exit_status = 1;
    goto END;
}
Vprintf("\n");
if (rcond >= X02AJC)
    Vprintf("Estimate of condition number =%10.2e\n", 1.0/rcond);
else
    Vprintf("A is singular to working precision\n");
END:
if (ap) NAG_FREE(ap);
return exit_status;
}


9.2 Program Data

f07uuc Example Program Data

4 :Value of N

'L' :Value of UPLO

( 4.78, 4.56)
( 2.00, -0.30) (-4.11, 1.25)
( 2.89, -1.34) ( 2.36, -4.25) ( 4.15, 0.80)
(-1.89, 1.15) ( 0.04, -3.69) (-0.02, 0.46) ( 0.33, -0.26) :End of matrix A

9.3 Program Results

f07uuc Example Program Results

Estimate of condition number = 3.74e+01