NAG C Library Function Document

nag_ztrcon (f07tuc)

1 Purpose
nag_ztrcon (f07tuc) estimates the condition number of a complex triangular matrix.

2 Specification

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

3 Description
nag_ztrcon (f07tuc) estimates the condition number of a complex triangular matrix \( A \), in either the 1-norm or the infinity-norm:

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

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 (see 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: \texttt{order} – Nag_OrderType \hspace{1cm} \textit{Input}

\textit{On entry}: the \texttt{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 \texttt{order = Nag_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\textit{Constraint}: \texttt{order = Nag_RowMajor} or \texttt{Nag_ColMajor}.

2: \texttt{norm} – Nag_NormType \hspace{1cm} \textit{Input}

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

if \texttt{norm = Nag_OneNorm}, \( \kappa_1(A) \) is estimated;

if \texttt{norm = Nag_Infnorm}, \( \kappa_\infty(A) \) is estimated.

\textit{Constraint}: \texttt{norm = Nag_OneNorm} or \texttt{Nag_Infnorm}.

3: \texttt{uplo} – Nag_UploType \hspace{1cm} \textit{Input}

\textit{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: uplo = Nag_Upper or Nag_Lower.

4: diag – Nag_DiagType
   On entry: indicates whether A is a non-unit or unit triangular matrix as follows:
   if diag = Nag_NonUnitDiag, A is a non-unit triangular matrix;
   if diag = Nag_UnitDiag, A is a unit triangular matrix; the diagonal elements are not
   referenced and are assumed to be 1.

Constraint: diag = Nag_NonUnitDiag or Nag_UnitDiag.

5: n – Integer
   On entry: n, the order of the matrix A.
   Constraint: n ≥ 0.

6: a[dim] – const Complex
   Note: the dimension, dim, of the array a must be at least max(1, pda × n).
   On entry: the n by n triangular matrix A. If uplo = Nag_Upper, A is upper triangular and the
   elements of the array below the diagonal are not referenced; if uplo = Nag_Lower, A is lower
   triangular and the elements of the array above the diagonal are not referenced. If diag = Nag_UnitDiag,
   the diagonal elements of A are not referenced, but are assumed to be 1.

7: pda – Integer
   On entry: the stride separating row or column elements (depending on the value of order) of the
   matrix A in the array a.
   Constraint: pda ≥ max(1, n).

8: rcond – double *
   On exit: an estimate of the reciprocal of the condition number of A. rcond is set to zero if exact
   singularity is detected or the estimate underflows. If rcond is less than machine precision, A is
   singular to working precision.

9: fail – NagError *
   The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT
   On entry, n = ⟨value⟩.
   Constraint: n ≥ 0.
   On entry, pda = ⟨value⟩.
   Constraint: pda > 0.

NE_INT_2
   On entry, pda = ⟨value⟩, n = ⟨value⟩.
   Constraint: pda ≥ max(1, n).

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 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_ztrcon (f07tuc) involves solving a number of systems of linear equations of the form \( Ax = b \) or \( A^H x = b \); 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_ztrtrs (f07tsc) 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 nag_dtrcon (f07tgc).

9 Example
To estimate the condition number in the 1-norm of the matrix \( A \), where

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

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

9.1 Program Text
/* nag_ztrcon (f07tuc) Example Program. */
* * Copyright 2001 Numerical Algorithms Group. *
* * Mark 7, 2001. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx02.h>

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

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda + I - 1]
    #endif

order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
order = Nag_RowMajor;
#endif
INIT_FAIL(fail);
Vprintf("f07tuc Example Program Results\n");

// Skip heading in data file */
Vscanf("%[^\n] ");
Vscanf("%ld%[^\n] ", &n);
#else if NAG_COLUMN_MAJOR
pda = n;
#endif
pda = n;

/* Allocate memory */
if ( !(a = 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(i,j).re, &A(i,j).im);
  }
  Vscanf("%[^\n] ");
}
else
{
  for (i = 1; i <= n; ++i)
  {
    for (j = 1; j <= i; ++j)
      Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
  }
  Vscanf("%[^\n] ");
}

/* Estimate condition number */
f07tuc(order, Nag_OneNorm, uplo_enum, Nag_NonUnitDiag, n, a, pda, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07tuc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
Vprintf("\n");
if (rcond >= X02AJC)
{
  Vprintf("Estimate of condition number =%10.2e\n\n", 1.0/rcond);
}
else
    Vprintf("A is singular to working precision\n");
END:
if (a) NAG_FREE(a);
return exit_status;
}

9.2 Program Data

f07tuc Example Program Data

4
'L'
( 4.78, 4.56)  :Value of N
( 2.00,-0.30)  :Value of UPLO
( 2.89,-1.34)  (-4.11, 1.25)
(-1.89, 1.15)  ( 2.36,-4.25)  ( 4.15, 0.80)
( 0.04,-3.69)  (-0.02, 0.46)  ( 0.33,-0.26)  :End of matrix A

9.3 Program Results

f07tuc Example Program Results

Estimate of condition number = 3.74e+01