

NAG C Library Function Document

nag_dtrcon (f07tgc)

1 Purpose

nag_dtrcon (f07tgc) estimates the condition number of a real triangular matrix.

2 Specification

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

3 Description

nag_dtrcon (f07tgc) estimates the condition number of a real 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: **order** – Nag_OrderType *Input*

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 *Input*

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

if **norm = Nag_OneNorm**, $\kappa_1(A)$ is estimated;

if **norm = Nag_InfNorm**, $\kappa_\infty(A)$ is estimated.

Constraint: **norm = Nag_OneNorm** or **Nag_InfNorm**.

3: **uplo** – Nag_UploType *Input*

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 *Input*

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 *Input*

On entry: n , the order of the matrix A .

Constraint: $n \geq 0$.

6: **a**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$.

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 *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix A in the array **a**.

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

8: **rcond** – double * *Output*

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

9: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle value \rangle$.

Constraint: $\mathbf{n} \geq 0$.

On entry, **pda** = $\langle value \rangle$.

Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.

Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ 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 ρ , and in practice is nearly always less than 10ρ , although examples can be constructed where **rcond** is much larger.

8 Further Comments

A call to nag_dtrcon (f07tgc) involves solving a number of systems of linear equations of the form $Ax = b$ or $A^T x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately n^2 floating-point operations but takes considerably longer than a call to nag_dtrtrs (f07tec) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this function is nag_ztrcon (f07tuc).

9 Example

To estimate the condition number in the 1-norm of the matrix A , where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix}.$$

The true condition number in the 1-norm is 116.41.

9.1 Program Text

```

/* nag_dtrcon (f07tgc) 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];
    double *a=0;

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

```

```

    order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07tgc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    Vscanf("%ld%*[\n] ", &n);
#ifdef NAG_COLUMN_MAJOR
    pda = n;
#else
    pda = n;
#endif

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

    /* Estimate condition number */
    f07tgc(order, Nag_OneNorm, uplo_enum, Nag_NonUnitDiag, n,
          a, pda, &rcond, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07tgc.\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

```
f07tgc Example Program Data
  4                               :Value of N
  'L'                             :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12   :End of matrix A
```

9.3 Program Results

```
f07tgc Example Program Results
```

```
Estimate of condition number = 1.16e+02
```
