1 Purpose

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

2 Specification

```c
#include <nag.h>
#include <nagf07.h>

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 \( \infty \)-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 Arguments

1: **order** – Nag_OrderType  
   *Input*
   
   *On entry:* the \( \text{order} \) argument 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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   
   *Constraint:* \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: **norm** – Nag_NormType  
   *Input*
   
   *On entry:* indicates whether \( \kappa_1(A) \) or \( \kappa_\infty(A) \) is estimated.
   
   **norm** = Nag_OneNorm  
   \( \kappa_1(A) \) is estimated.
   
   **norm** = Nag_InfNorm  
   \( \kappa_\infty(A) \) is estimated.
   
   *Constraint:* \( \text{norm} = \text{Nag_OneNorm} \) or \( \text{Nag_InfNorm} \).
3: \hfill \textit{uplo} – Nag_UploType \hfill \textit{Input}

\textit{On entry:} specifies whether \( A \) is upper or lower triangular.

\textit{uplo} = Nag_Upper
\hfill \( A \) is upper triangular.

\textit{uplo} = Nag_Lower
\hfill \( A \) is lower triangular.

\textit{Constraint:} \textit{uplo} = Nag_Upper or Nag_Lower.

4: \hfill \textit{diag} – Nag_DiagType \hfill \textit{Input}

\textit{On entry:} indicates whether \( A \) is a nonunit or unit triangular matrix.

\textit{diag} = Nag_NonUnitDiag
\hfill \( A \) is a nonunit triangular matrix.

\textit{diag} = Nag_UnitDiag
\hfill \( A \) is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

\textit{Constraint:} \textit{diag} = Nag_NonUnitDiag or Nag_UnitDiag.

5: \hfill \textit{n} – Integer \hfill \textit{Input}

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

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

6: \hfill \textit{a[dim]} – const double \hfill \textit{Input}

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

\textit{On entry:} the \( n \) by \( n \) triangular matrix \( A \).

If \textit{order} = Nag_ColMajor, \( A_{ij} \) is stored in \( a[(j-1) \times \textit{pda} + i - 1] \).

If \textit{order} = Nag_RowMajor, \( A_{ij} \) is stored in \( a[(i-1) \times \textit{pda} + j - 1] \).

If \textit{uplo} = Nag_Upper, the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced.

If \textit{uplo} = Nag_Lower, the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

If \textit{diag} = Nag_UnitDiag, the diagonal elements of \( A \) are assumed to be 1, and are not referenced.

7: \hfill \textit{pda} – Integer \hfill \textit{Input}

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

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

8: \hfill \textit{rcond} – double * \hfill \textit{Output}

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

9: \hfill \textit{fail} – NagError * \hfill \textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).
6 Error Indicators and Warnings

**NE_ALLOC_FAIL**
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**
On entry, argument \(\text{value}\) had an illegal value.

**NE_INT**
On entry, \(n = \text{value}\).
Constraint: \(n \geq 0\).
On entry, \(pda = \text{value}\).
Constraint: \(pda > 0\).

**NE_INT_2**
On entry, \(pda = \text{value}\) and \(n = \text{value}\).
Constraint: \(pda \geq \max(1,n)\).

**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 contact NAG for assistance.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

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

8 Parallelism and Performance
\n\text{arg_dtrcon} (f07tgc) is not threaded by NAG in any implementation.
\n\text{arg_dtrcon} (f07tgc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.
Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments
A call to \text{arg_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 \text{arg_dtrtrs} (f07tec) with one right-hand side, because extra care is taken to avoid overflow when \(A\) is approximately singular.

The complex analogue of this function is \text{arg_ztrcon} (f07tuc).
This example estimates 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.

### 10.1 Program Text

```c
#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;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    char nag_enum_arg[40];
    double *a = 0;

    #ifdef NAG_COLUMN_MAJOR
        pda = n;
    #else
        pda = n;
    #endif

    INIT_FAIL(fail);
    printf("nag_dtrcon (f07tgc) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT"%*[\n] ", &n);
    #else
        scanf("%"NAG_IFMT"%*[\n] ", &n);
    #endif
    #ifdef NAG_COLUMN_MAJOR
        order = Nag_ColMajor;
    #else
        order = Nag_RowMajor;
    #endif

    if (!(a = NAG_ALLOC(n * n, double)))
    {
        NAG_FALSE_RETURN;
        return -1;
    }

    */* Allocate memory */
    if (!((a = NAGALLOC(n * n, double))))
    {
        NAG_FALSE_RETURN;
        return -1;
    }

    */* Solve the system */
    NAG_TRUE_RETURN;
    return 0;
}
```
{ printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}

/* Read A from data file */

#ifdef _WIN32
  scanf_s(" %39s%[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf(" %39s%[\n] ", nag_enum_arg);
#endif

/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

if (uplo == Nag_Upper)
  {
    for (i = 1; i <= n; ++i)
    {
      for (j = i; j <= n; ++j)
      #ifdef _WIN32
        scanf_s("%lf", &A(i, j));
      #else
        scanf("%lf", &A(i, j));
      #endif
    }
    #ifdef _WIN32
      scanf_s("%*[\n] ");
    #else
      scanf("%*[\n] ");
    #endif
  }
else
  {
    for (i = 1; i <= n; ++i)
    {
      for (j = 1; j <= i; ++j)
      #ifdef _WIN32
        scanf_s("%lf", &A(i, j));
      #else
        scanf("%lf", &A(i, j));
      #endif
    }
    #ifdef _WIN32
      scanf_s("%*[\n] ");
    #else
      scanf("%*[\n] ");
    #endif
  }

/* Estimate condition number */
/* nag_dtrcon (f07tgc).
 * Estimate condition number of real triangular matrix
 */
  nag_dtrcon(order, Nag_OneNorm, uplo, Nag_UpperDiag, n,
             a, pda, &rcond, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dtrcon (f07tgc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

printf("\n");
/* nag_machine_precision (x02ajc).
 * The machine precision
if (rcond >= nag_machine_precision)
    {
        printf("Estimate of condition number =%11.2e\n\n", 1.0 / rcond);
    }
else
    printf("A is singular to working precision\n");
END:
NAG_FREE(a);
return exit_status;
}

10.2 Program Data

nag_dtrcon (f07tgc) Example Program Data

<table>
<thead>
<tr>
<th>Value of n</th>
<th>Value of uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Nag_Lower</td>
</tr>
<tr>
<td>4.30</td>
<td></td>
</tr>
<tr>
<td>-3.96</td>
<td>-4.87</td>
</tr>
<tr>
<td>0.40</td>
<td>0.31</td>
</tr>
<tr>
<td>-0.27</td>
<td>0.07</td>
</tr>
<tr>
<td>-8.02</td>
<td>-5.95</td>
</tr>
<tr>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of matrix A</td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_dtrcon (f07tgc) Example Program Results

Estimate of condition number = 1.16e+02