NAG C Library Function Document
nag_dtpcon (f07ugc)

1 Purpose
nag_dtpcon (f07ugc) estimates the condition number of a real triangular matrix, using packed storage.

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

3 Description
nag_dtpcon (f07ugc) estimates the condition number of a real triangular matrix $A$, in either the 1-norm or the infinity-norm, using packed storage:

$$
k_1(A) = \|A\|_1 \|A^{-1}\|_1 \quad \text{or} \quad k_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.
$$

Note that $k_\infty(A) = k_1(A^T)$.

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 $k_1(A)$ or $k_\infty(A)$ is estimated as follows:

if norm = Nag_OneNorm, $k_1(A)$ is estimated;

if norm = Nag_InfNorm, $k_\infty(A)$ 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: 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: ap[dim] – const double

Note: the dimension, dim, of the array ap must be at least max(1, n × (n + 1)/2).

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

if order = Nag_ColMajor and uplo = Nag_Upper,

$a_{ij}$ is stored in $ap[(j - 1) \times j/2 + i - 1]$, for $i \leq j$;

if order = Nag_ColMajor and uplo = Nag_Lower,

$a_{ij}$ is stored in $ap[(2n - j) \times (j - 1)/2 + i - 1]$, for $i \geq j$;

if order = Nag_RowMajor and uplo = Nag_Upper,

$a_{ij}$ is stored in $ap[(2n - i) \times (i - 1)/2 + j - 1]$, for $i \leq j$;

if order = Nag_RowMajor and uplo = Nag_Lower,

$a_{ij}$ is stored in $ap[(i - 1) \times i/2 + j - 1]$, for $i \geq j$.

7: 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.

8: fail – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, n = (value).

Constraint: n ≥ 0.

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 \texttt{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 \texttt{rcond} is much larger.

8 Further Comments
A call to \texttt{nag_dtpcon (f07ugc)} 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 \texttt{nag_dtptrs (f07uec)} 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 \texttt{nag_ztpcon (f07uuc)}.

9 Example
To estimate the condition number in the 1-norm of the matrix \(A\), where
\[
A = \begin{pmatrix}
0.40 & 0.31 & -8.02 & 0.00 \\
-3.96 & -4.87 & 0.00 & 0.00 \\
-0.27 & 0.07 & -5.95 & 0.12 \\
4.30 & 0.00 & 0.00 & 0.00 \\
\end{pmatrix},
\]
using packed storage. The true condition number in the 1-norm is 116.41.

9.1 Program Text
/* nag_dtpcon (f07ugc) 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 ap_len, i, j, n;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;

    NagOrderType order;
    /* Arrays */
    char uplo[2];
    double *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("f07ugc Example Program Results\n");
/* Skip heading in data file */
Vscanf("%*[\n ] ");
Vscanf("%ld%*[\n ] ", &n);

/* Allocate memory */
ap_len = n*(n+1)/2;
if ( !(ap = NAG_ALLOC(ap_len, 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_UPPER(i,j));
    }
    Vscanf("%*[\n ] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A_LOWER(i,j));
    }
    Vscanf("%*[\n ] ");
}

/* Estimate condition number */
f07ugc(order, Nag_OneNorm, uplo_enum, Nag_NonUnitDiag, n,
ap, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07ugc.\n");
    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

f07ugc Example Program Data

Value of N
L
Value of UPLO
4

Value of A

-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

f07ugc Example Program Results

Estimate of condition number = 1.16e+02