NAG C Library Function Document

nag_dtbcon (f07vgc)

1 Purpose

nag_dtbcon (f07vgc) estimates the condition number of a real triangular band matrix.

2 Specification

```
void nag_dtbcon (Nag_OrderType order, Nag_NormType norm, Nag_UploType uplo,
     Nag_DiagType diag, Integer n, Integer kd, const double ab[], Integer pdab,
     double *rcond, NagError *fail)
```

3 Description

nag_dtbcon (f07vgc) estimates the condition number of a real triangular band 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 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** ≥ 0.

6: **kd** – Integer *Input*

On entry: k, the number of super-diagonals of the matrix A if **uplo** = **Nag_Upper** or the number of sub-diagonals if **uplo** = **Nag_Lower**.

Constraint: **kd** ≥ 0.

7: **ab**[dim] – const double *Input*

Note: the dimension, dim, of the array **ab** must be at least max(1, **pdab** × **n**).

On entry: the n by n triangular matrix A. This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. 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 **ab**$[k + i - j + (j - 1) \times$ **pdab**], for $i = 1, \ldots, n$ and $j = i, \ldots, \min(n, i + k)$;

if **order** = **Nag_ColMajor** and **uplo** = **Nag_Lower**,
a_{ij} is stored in **ab**$[i - j + (j - 1) \times$ **pdab**], for $i = 1, \ldots, n$ and $j = \max(1, i - k), \ldots, i$;

if **order** = **Nag_RowMajor** and **uplo** = **Nag_Upper**,
a_{ij} is stored in **ab**$[j - i + (i - 1) \times$ **pdab**], for $i = 1, \ldots, n$ and $j = i, \ldots, \min(n, i + k)$;

if **order** = **Nag_RowMajor** and **uplo** = **Nag_Lower**,
a_{ij} is stored in **ab**$[k + j - i + (i - 1) \times$ **pdab**], for $i = 1, \ldots, n$ and $j = \max(1, i - k), \ldots, i$.

8: **pdab** – Integer *Input*

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

Constraint: **pdab** $\geq$ **kd** $+ 1$.

9: **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 the estimate underflows. If **rcond** is less than ***machine precision***, A is singular to working precision.

10: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **kd** = ⟨*value*⟩.
Constraint: **kd** ≥ 0.

On entry, **pdab** = ⟨*value*⟩.
Constraint: **pdab** > 0.

NE_INT_2

On entry, **pdab** = ⟨*value*⟩, **kd** = ⟨*value*⟩.
Constraint: **pdab** $\geq$ **kd** $+ 1$.

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 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_dtbcon (f07vgc) 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 $2nk$ floating-point operations (assuming $n \gg k$) but takes considerably longer than a call to nag_dtbtrs (f07vec) 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 nag_ztbcon (f07vuc).

9 Example

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

$$A = \begin{pmatrix} -4.16 & 0.00 & 0.00 & 0.00 \\ -2.25 & 4.78 & 0.00 & 0.00 \\ 0.00 & 5.86 & 6.32 & 0.00 \\ 0.00 & 0.00 & -4.82 & 0.16 \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with 1 sub-diagonal. The true condition number in the 1-norm is 69.62.

9.1 Program Text

```
/* nag_dtbcon (f07vgc) 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 */
  Integer  i, j, k, kd, n, pdab;
  Integer  exit_status=0;
  double   rcond;
  NagError fail;
  Nag_UploType uplo_enum;
  Nag_OrderType order;

  /* Arrays */
  char    uplo[2];
  double  *ab=0;

#ifdef NAG_COLUMN_MAJOR
#define AB_UPPER(I,J) ab[(J-1)*pdab + k + I - J - 1]
#define AB_LOWER(I,J) ab[(J-1)*pdab + I - J]
  order = Nag_ColMajor;
#else
#define AB_UPPER(I,J) ab[(I-1)*pdab + J - I]
#define AB_LOWER(I,J) ab[(I-1)*pdab + k + J - I - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  Vprintf("f07vgc Example Program Results\n\n");

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

  /* Allocate memory */
  if ( !(ab = NAG_ALLOC((kd+1) * 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;
    }
  k = kd + 1;
  if (uplo_enum == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
```

```
        {
          for (j = i; j <= MIN(i+kd,n); ++j)
            Vscanf("%lf", &AB_UPPER(i,j));
        }
      Vscanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = MAX(1,i-kd); j <= i; ++j)
            Vscanf("%lf", &AB_LOWER(i,j));
        }
      Vscanf("%*[^\n] ");
    }
  /* Estimate condition number */
  f07vgc(order, Nag_OneNorm, uplo_enum, Nag_NonUnitDiag, n,
         kd, ab, pdab, &rcond, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07vgc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  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 (ab) NAG_FREE(ab);
  return exit_status;
}
```

9.2 Program Data

```
f07vgc Example Program Data
  4  1                        :Values of N and KD
  'L'                         :Value of UPLO
 -4.16
 -2.25   4.78
         5.86   6.32
               -4.82   0.16   :End of matrix A
```

9.3 Program Results

```
f07vgc Example Program Results

Estimate of condition number =  6.96e+01
```