NAG Library Function Document

nag_dtbcon (f07vgc)

1 Purpose

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

2 Specification

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

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 $\infty$-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 **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 **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   **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.

   **norm** = Nag_OneNorm

   $\kappa_1(A)$ is estimated.

   **norm** = Nag_InfNorm

   $\kappa_{\infty}(A)$ is estimated.

   **Constraint:** norm = Nag_OneNorm or Nag_InfNorm.
3:  **uplo** – Nag_UploType

*Input*

*On entry:* specifies whether $A$ is upper or lower triangular.

- **uplo** = Nag_Upper
  - $A$ is upper triangular.
- **uplo** = Nag_Lower
  - $A$ is lower triangular.

*Constraint:* $\text{uplo} = \text{Nag\_Upper}$ or $\text{Nag\_Lower}$.

4:  **diag** – Nag_DiagType

*Input*

*On entry:* indicates whether $A$ is a nonunit or unit triangular matrix.

- **diag** = Nag_NonUnitDiag
  - $A$ is a nonunit triangular matrix.
- **diag** = Nag_UnitDiag
  - $A$ is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

*Constraint:* $\text{diag} = \text{Nag\_NonUnitDiag}$ or $\text{Nag\_UnitDiag}$.

5:  **n** – Integer

*Input*

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

*Constraint:* $n \geq 0$.

6:  **kd** – Integer

*Input*

*On entry:* $kd$, the number of superdiagonals of the matrix $A$ if uplo = Nag_Upper, or the number of subdiagonals if uplo = Nag_Lower.

*Constraint:* $kd \geq 0$.

7:  **ab[dim]** – const double

*Input*

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

*On entry:* the $n$ by $n$ triangular band matrix $A$.

This is stored as a notional two-dimensional array with row elements or column elements stored continguously. The storage of elements of $A_{ij}$, depends on the **order** and **uplo** arguments as follows:

- **order** = Nag_ColMajor and **uplo** = Nag_Upper,
  - $A_{ij}$ is stored in $\text{ab}[kd + i - j + (j - 1) \times pdab]$, for $j = 1, \ldots, n$ and $i = \max(1, j - k_d), \ldots, j$;
- **order** = Nag_ColMajor and **uplo** = Nag_Lower,
  - $A_{ij}$ is stored in $\text{ab}[i - j + (j - 1) \times pdab]$, for $j = 1, \ldots, n$ and $i = j, \ldots, \min(n, j + k_d)$;
- **order** = Nag_RowMajor and **uplo** = Nag_Upper,
  - $A_{ij}$ is stored in $\text{ab}[j - i + (i - 1) \times pdab]$, for $i = 1, \ldots, n$ and $j = i, \ldots, \min(n, i + k_d)$;
- **order** = Nag_RowMajor and **uplo** = Nag_Lower,
  - $A_{ij}$ is stored in $\text{ab}[k_d + j - i + (i - 1) \times pdab]$, for $i = 1, \ldots, n$ and $j = \max(1, i - k_d), \ldots, i$.

If **diag** = Nag_UnitDiag, the diagonal elements of $AB$ are assumed to be 1, and are not referenced.
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 ≥ kd + 1.

DOUBLE PRECISION 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.

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 value had an illegal value.

NE_INT
On entry, kd = value. Constraint: kd ≥ 0.

On entry, n = value. Constraint: n ≥ 0.

On entry, pdab = value. Constraint: pdab > 0.

NE_INT_2
On entry, pdab = value and kd = value. Constraint: pdab ≥ kd + 1.

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 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 Parallelism and Performance

nag_dtbcon (f07vgc) is not threaded by NAG in any implementation.

nag_dtbcon (f07vgc) 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 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).

10 Example

This example estimates 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 one subdiagonal. The true condition number in the 1-norm is 69.62.

10.1 Program Text

/* nag_dtbcon (f07vgc) Example Program. *
 * Copyright 2014 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;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    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;
    #endif

    // Code to estimate the condition number
    // and print the result
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);

printf("nag_dtbcon (f07vgc) Example Program Results\n\n");

/* Skip heading in data file */
#define _WIN32
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif
#undef _WIN32
scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n ] ", &n, &kd);
#else
scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n ] ", &n, &kd);
#endif

pdab = kd + 1;

/* Allocate memory */
if (!(ab = NAG_ALLOC((kd+1) * n, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
#define _WIN32
scanf_s(" %39s%*[\n ] ", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf(" %39s%*[\n ] ", nag_enum_arg);
#endif

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 <= MIN(i+kd, n); ++j)
            #ifdef _WIN32
                scanf_s("%lf", &AB_UPPER(i, j));
            #else
                scanf("%lf", &AB_UPPER(i, j));
            #endif
    }
    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
} else {
    for (i = 1; i <= n; ++i)
        for (j = MAX(1, i-kd); j <= i; ++j)
            #ifdef _WIN32
                scanf_s("%lf", &AB_LOWER(i, j));
            #else
                scanf("%lf", &AB_LOWER(i, j));
            #endif
}
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

} /* Estimate condition number */

/* Estimate condition number of real band triangular matrix */

nag_dtbcon(order, Nag_OneNorm, uplo, Nag_NonUnitDiag, n, 
  kd, ab, pdab, &rcond, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dtbcon (f07vgc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* 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(ab);
return exit_status;

10.2 Program Data

nag_dtbcon (f07vgc) Example Program Data

4 1 :Values of n and kd
Nag_Lower :Value of uplo
-4.16
-2.25 4.78
  5.86 6.32
  -4.82 0.16 :End of matrix A

10.3 Program Results

nag_dtbcon (f07vgc) Example Program Results

Estimate of condition number = 6.96e+01