NAG C Library Function Document

nag_dgbcon (f07bgc)

1 Purpose

nag_dgbcon (f07bgc) estimates the condition number of a real band matrix $A$, where $A$ has been factorized by nag_dgbtrf (f07bdc).

2 Specification

```c
void nag_dgbcon (Nag_OrderType order, Nag_NormType norm, Integer n, Integer kl,
                 Integer ku, const double ab[], Integer pdab, const Integer ipiv[],
                 double anorm, double *rcond, NagError *fail)
```

3 Description

nag_dgbcon (f07bgc) estimates the condition number of a real 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 reciprocal of the condition number.

The function should be preceded by a call to nag_dgb_norm (f16rbc) to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to nag_dgbtrf (f07bdc) to compute the $LU$ factorization of $A$. The function then 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: n – Integer
   Input
   On entry: $n$, the order of the matrix $A$.
   Constraint: $n \geq 0$. 
4:  kl – Integer
    Input
    On entry: $k_l$, the number of sub-diagonals within the band of $A$.
    Constraint: $kl \geq 0$.

5:  ku – Integer
    Input
    On entry: $k_u$, the number of super-diagonals within the band of $A$.
    Constraint: $ku \geq 0$.

6:  ab[\textit{dim}] – const double
    Input
    Note: the dimension, \textit{dim}, of the array \textit{ab} must be at least max$(1, pdab \times n)$.
    On entry: the \textit{LU} factorization of $A$, as returned by \texttt{nag_dgbtrf (f07bdc)}.

7:  pdab – Integer
    Input
    On entry: the stride separating row or column elements (depending on the value of \textit{order}) of the matrix in the array \textit{ab}.
    Constraint: $pdab \geq 2 \times kl + ku + 1$.

8:  ipiv[\textit{dim}] – const Integer
    Input
    Note: the dimension, \textit{dim}, of the array \textit{ipiv} must be at least max$(1, n)$.
    On entry: the pivot indices, as returned by \texttt{nag_dgbtrf (f07bdc)}.

9:  anorm – double
    Input
    On entry: if \textit{norm} = \texttt{Nag_OneNorm}, the 1-norm of the original matrix $A$; if \textit{norm} = \texttt{Nag_InfNorm}, the infinity-norm of the original matrix $A$. \textit{anorm} may be computed by calling \texttt{nag_dgb_norm (f16rbc)} with the same value for the parameter \textit{norm}. \textit{anorm} must be computed either before calling \texttt{nag_dgbtrf (f07bdc)} or else from a copy of the original matrix $A$.
    Constraint: $\textit{anorm} \geq 0.0$.

10: rcond – double *
    Output
    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 the estimate underflows. If \textit{rcond} is less than \textit{machine precision}, $A$ is singular to working precision.

11: fail – NagError *
    Output
    The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT
    On entry, $n = \langle\textit{value}\rangle$.
    Constraint: $n \geq 0$.
    On entry, $kl = \langle\textit{value}\rangle$.
    Constraint: $kl \geq 0$.
    On entry, $ku = \langle\textit{value}\rangle$.
    Constraint: $ku \geq 0$.
    On entry, $pdab = \langle\textit{value}\rangle$.
    Constraint: $pdab > 0$.  

On entry, \( pdab = \text{(value)} \), \( kl = \text{(value)} \), \( ku = \text{(value)} \).
Constraint: \( pdab \geq 2 \times kl + ku + 1 \).

On entry, \( anorm = \text{(value)} \).
Constraint: \( anorm \geq 0.0 \).

Memory allocation failed.

On entry, parameter \( \text{(value)} \) had an illegal value.

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.

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

A call to \( \text{nag_dgbcon (f07bgc)} \) 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 \( 2n(2k_l + k_u) \) floating-point operations (assuming \( n \gg k_l \) and \( n \gg k_u \)) but takes considerably longer than a call to \( \text{nag_dgbtrs (f07bec)} \) 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 \( \text{nag_zgbcon (f07buc)} \).

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

\[
A = \begin{pmatrix}
-0.23 & 2.54 & -3.66 & 0.00 \\
-6.98 & 2.46 & -2.73 & -2.13 \\
0.00 & 2.56 & 2.46 & 4.07 \\
0.00 & 0.00 & -4.78 & -3.82
\end{pmatrix}
\]

Here \( A \) is nonsymmetric and is treated as a band matrix, which must first be factorized by \( \text{nag_dgbtrf (f07bdc)} \). The true condition number in the 1-norm is 56.40.

/* \( \text{nag_dgbcon (f07bgc)} \) 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, ipiv_len, j, kl, ku, n, pdab;
    Integer exit_status=0;
    double anorm, rcond, sum;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    double *ab=0;
    Integer *ipiv=0;

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

    INIT_FAULT(fail);
    Vprintf("f07bgc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n"]);
    Vscanf("%ld%ld%ld%*[\n"] , &n, &kl, &ku);
    ipiv_len = n;
    pdab = 2*kl + ku + 1;

    /* Allocate memory */
    if ( !(ab = NAG_ALLOC((2*kl+ku+1) * n, double)) ||
        !(ipiv = NAG_ALLOC(ipiv_len, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read AB from data file */
    for (i = 1; i <= n; ++i)
    {
        for (j = MAX(i-kl,1); j <= MIN(i+ku,n); ++j)
            Vscanf("%lf", &AB(i,j));
    }
    Vscanf("%*[\n"]);

    /* Compute norm of A */
    anorm = 0.0;
    for (j = 1; j <= n; ++j)
    {
        sum = 0.0;
        for (i = MAX(j-ku,1); i <= MIN(j+kl,n); ++i)
            sum = sum + ABS(AB(i,j));
        anorm = MAX(anorm,sum);
    }

    /* Factorize A */
    f07bdc(order, n, n, kl, ku, ab, pdab, ipiv, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07bdc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Estimate condition number */
    f07bgc(order, Nag_OneNorm, n, kl, ku, ab, pdab, ipiv, anorm, &rcond, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07bgc.\n%s\n", fail.message);
        exit_status = 1;
    }
}

f07bgc
NAG C Library Manual
goto END;
}

/* Print condition number */
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 (ab) NAG_FREE(ab);
if (ipiv) NAG_FREE(ipiv);
return exit_status;
}

9.2 Program Data

f07bgc Example Program Data

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-0.23</td>
<td>2.54</td>
<td>-3.66</td>
</tr>
<tr>
<td>-6.98</td>
<td>2.46</td>
<td>-2.73</td>
</tr>
<tr>
<td>2.56</td>
<td>2.46</td>
<td>4.07</td>
</tr>
<tr>
<td></td>
<td>-4.78</td>
<td>-3.82</td>
</tr>
</tbody>
</table>

Values of N, KL and KU

End of matrix A

9.3 Program Results

f07bgc Example Program Results

Estimate of condition number = 5.64e+01