NAG C Library Function Document

nag_zgbcon (f07buc)

1 Purpose

nag_zgbcon (f07buc) estimates the condition number of a complex band matrix $A$, where $A$ has been factorized by nag_zgbtrf (f07brc).

2 Specification

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

3 Description

nag_zgbcon (f07buc) estimates the condition number of a complex 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 $\kappa_\infty(A) = \kappa_1(A^H)$.

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_zgb_norm (f16ubc) to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to nag_zgbtrf (f07brc) 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[\( dim \)]** – const Complex  
   *Input*
   
   Note: the dimension, \( dim \), of the array \( ab \) must be at least \( \max(1, pdab \times n) \).
   
   On entry: the \( LU \) factorization of \( A \), as returned by \( \text{nag_zgbtrf (f07brc)} \).

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

8:  **ipiv[\( dim \)]** – const Integer  
   *Input*
   
   Note: the dimension, \( dim \), of the array \( ipiv \) must be at least \( \max(1, n) \).
   
   On entry: the pivot indices, as returned by \( \text{nag_zgbtrf (f07brc)} \).

9:  **anorm** – double  
   *Input*
   
   On entry: if \( \text{norm} = \text{Nag_OneNorm} \), the 1-norm of the original matrix \( A \); if \( \text{norm} = \text{Nag_InfNorm} \), the infinity-norm of the original matrix \( A \). \( \text{anorm} \) may be computed by calling \( \text{nag_zgb_norm (f16ubc)} \) with the same value for the parameter \( \text{norm} \). \( \text{anorm} \) must be computed either before calling \( \text{nag_zgbtrf (f07brc)} \) or else from a copy of the original matrix \( A \).
   
   Constraint: \( anorm \geq 0.0 \).

10:  **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 \( \text{machine precision} \), \( A \) 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, \( n = \langle value \rangle \).
   
   Constraint: \( n \geq 0 \).
   
   On entry, \( kl = \langle value \rangle \).
   
   Constraint: \( kl \geq 0 \).
   
   On entry, \( ku = \langle value \rangle \).
   
   Constraint: \( ku \geq 0 \).
   
   On entry, \( pdab = \langle value \rangle \).
   
   Constraint: \( pdab > 0 \).
On entry, \( \text{pdab} = (\text{value}), \text{kl} = (\text{value}), \text{ku} = (\text{value}) \).  
Constraint: \( \text{pdab} \geq 2 \times \text{kl} + \text{ku} + 1 \).

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

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 nag_zgbcon (f07buc) involves solving a number of systems of linear equations of the form \( Ax = b \) or \( A^H x = b \); the number is usually 5 and never more than 11. Each solution involves approximately \( 8n(2k_l + k_u) \) real floating-point operations (assuming \( n \gg k_l \) and \( n \gg k_u \)) but takes considerably longer than a call to nag_zgbtrs (f07bsc) with 1 right-hand side, because extra care is taken to avoid overflow when \( A \) is approximately singular.

The real analogue of this function is nag_dgbcon (f07bgc).

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

\[
A = \begin{pmatrix}
-1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\
0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\
0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\
0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i
\end{pmatrix}
\]

9.1 Program Text

/* nag_zgbcon (f07buc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf07.h>
#include <nagx02.h>

int main(void)

int

f07buc
{ /* Scalars */
    Integer i, ipiv_len, j, kl, ku, n, pdab;
    Integer exit_status=0;
    double anorm, rcond, sum;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    Complex *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_FAIL(fail);
    Vprintf("f07buc 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, Complex)) ||
        !(ipiv = NAG_ALLOC(ipiv_len, Integer)) )
        {
            Vprintf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }

    /* Read A from data file */
    for (i = 1; i <= n; ++i)
        {
            for (j = MAX(i-kl,1); j <= MIN(i+ku,n); ++j)
                Vscanf(" ( %lf , %lf )", &AB(i,j).re, &AB(i,j).im);
        }
    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 + a02dbc(AB(i,j));
            anorm = MAX(anorm,sum);
        }

    /* Factorize A */
    f07brc(order, n, n, kl, ku, ab, pdab, ipiv, &fail);
    if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from f07brc.\n\n", fail.message);
            exit_status = 1;
            goto END;
        }

    /* Estimate condition number */
    f07buc(order, Nag_OneNorm, n, kl, ku, ab, pdab, ipiv, anorm, &rcond, &fail);
    if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from f07buc.\n\n", fail.message);
            exit_status = 1;
            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
f07buc Example Program Data
4 1 2 :Values of N, KL and KU
(-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
(-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
( 4.48,-1.09) (-0.46,-1.72) :End of matrix A

9.3 Program Results
f07buc Example Program Results
Estimate of condition number = 1.04e+02