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

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

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 \( \infty \)-norm:

\[
\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.
\]

Note that \( \kappa_\infty(A) = \kappa_1(A^H) \).

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 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.
\[ \|A\|_\infty \] is estimated.

**Constraint:** \( \text{norm} = \text{Nag\_OneNorm} \) or \( \text{Nag\_InfNorm} \).

3: \( n \) – Integer

**Input**

*On entry:* \( n \), the order of the matrix \( A \).

**Constraint:** \( n \geq 0 \).

4: \( k_l \) – Integer

**Input**

*On entry:* \( k_l \), the number of subdiagonals within the band of the matrix \( A \).

**Constraint:** \( k_l \geq 0 \).

5: \( k_u \) – Integer

**Input**

*On entry:* \( k_u \), the number of superdiagonals within the band of the matrix \( A \).

**Constraint:** \( k_u \geq 0 \).

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:** \( pdab \geq 2 \times k_l + k_u + 1 \).

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: \( \text{anorm} \) – 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 \( \infty \)-norm of the original matrix \( A \).

\( \text{anorm} \) may be computed by calling \( \text{nag\_zgb\_norm} \) (f16ubc) with the same value for the argument \( \text{norm} \).

\( \text{anorm} \) must be computed either before calling \( \text{nag\_zgbtrf} \) (f07brc) or else from a copy of the original matrix \( A \) (see Section 10).

**Constraint:** \( \text{anorm} \geq 0.0 \).

10: \( \text{rcond} \) – double *

**Output**

*On exit:* an estimate of the reciprocal of the condition number of \( A \). \( \text{rcond} \) is set to zero if exact singularity is detected or the estimate underflows. If \( \text{rcond} \) is less than \( \text{machine precision} \), \( A \) is singular to working precision.

11: \( \text{fail} \) – NagError *

**Input/Output**

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, kl = ⟨value⟩.
Constraint: kl ≥ 0.
On entry, ku = ⟨value⟩.
Constraint: ku ≥ 0.
On entry, n = ⟨value⟩.
Constraint: n ≥ 0.
On entry, pdab = ⟨value⟩.
Constraint: pdab > 0.

NE_INT_3
On entry, pdab = ⟨value⟩, kl = ⟨value⟩ and ku = ⟨value⟩.
Constraint: pdab ≥ 2 × kl + ku + 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.

NE_REAL
On entry, anorm = ⟨value⟩.
Constraint: anorm ≥ 0.0.

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_zgbcon (f07buc) is not threaded by NAG in any implementation.
nag_zgbcon (f07buc) 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_zgbcon (f07buc) involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Hx = b$; 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_zgbrs (f07bsc) with one 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).

10 Example

This example estimates the condition number in the 1-norm of the matrix $A$, where

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

10.1 Program Text

/* nag_zgbcon (f07buc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.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 */
    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);
    printf("nag_zgbcon (f07buc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n ] ");
    #else
    scanf("%*[\n ] ");
    #endif
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n ] ", &n, &kl, &ku);
    */
```c
#include <stdio.h>
#include <nag.h>

#define _WIN32

int main(int argc, char *argv[]) {
    int n, kl, ku;
    double *ab, *ipiv;
    int pdab = 2*kl + ku + 1;
    int ipiv_len = n;

    /* Allocate memory */
    if (!ab = NAG_ALLOC((2*kl+ku+1) * n, Complex) || 
        !ipiv = NAG_ALLOC(ipiv_len, Integer)) {
        printf("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)
            scanf(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
    }

    /* 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 + nag_complex_abs(AB(i, j));
        anorm = MAX(anorm, sum);
    }

    /* Factorize A */
    /* nag_zgbtrf (f07brc).
     * LU factorization of complex m by n band matrix
     */
    nag_zgbtrf(order, n, n, kl, ku, ab, pdab, ipiv, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zgbtrf (f07brc).\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Estimate condition number */
    /* nag_zgbcon (f07buc).
     * Estimate condition number of complex band matrix, matrix
     * already factorized by nag_zgbtrf (f07brc)
     */
    nag_zgbcon(order, Nag_OneNorm, n, kl, ku, ab, pdab, ipiv, 
                anorm, &rcond, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zgbcon (f07buc).\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print condition number */
    /* nag_machine_precision (x02ajc).
     * The machine precision
     */
    if (rcond > nag_machine_precision)
        printf("Warning: Condition number too large\n");

    return exit_status;
}
```

This program demonstrates how to perform LU factorization and estimate the condition number of a complex band matrix using the LAPACK routines nag_zgbtrf and nag_zgbcon, respectively. It also prints a warning if the condition number is too large.
printf("Estimate of condition number = %11.2e\n", 1.0/rcond);
else
  printf("A is singular to working precision\n");
END;
NAG_FREE(ab);
NAG_FREE(ipiv);
return exit_status;
}

10.2 Program Data

nag_zgbcon (f07buc) Example Program Data

4  1  2
(-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

10.3 Program Results

nag_zgbcon (f07buc) Example Program Results

Estimate of condition number = 1.04e+02