

# NAG 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

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

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  $\infty$ -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 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 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: **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 subdiagonals within the band of the matrix  $A$ .  
*Constraint:*  $kl \geq 0$ .
- 5: **ku** – Integer *Input*  
*On entry:*  $k_u$ , the number of superdiagonals within the band of the matrix  $A$ .  
*Constraint:*  $ku \geq 0$ .
- 6: **ab**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **ab** must be at least  $\max(1, \mathbf{pdab} \times \mathbf{n})$ .  
*On entry:* the  $LU$  factorization of  $A$ , as returned by nag\_dgbtrf (f07bdc).
- 7: **pdab** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) of the matrix in the array **ab**.  
*Constraint:*  $\mathbf{pdab} \geq 2 \times \mathbf{kl} + \mathbf{ku} + 1$ .
- 8: **ipiv**[*dim*] – const Integer *Input*  
**Note:** the dimension, *dim*, of the array **ipiv** must be at least  $\max(1, \mathbf{n})$ .  
*On entry:* the pivot indices, as returned by nag\_dgbtrf (f07bdc).
- 9: **anorm** – double *Input*  
*On entry:* if **norm** = Nag\_OneNorm, the 1-norm of the **original** matrix  $A$ .  
If **norm** = Nag\_InfNorm, the  $\infty$ -norm of the **original** matrix  $A$ .  
**anorm** may be computed by calling nag\_dgb\_norm (f16rbc) with the same value for the argument **norm**.  
**anorm** must be computed either **before** calling nag\_dgbtrf (f07bdc) or else from a **copy** of the original matrix  $A$  (see Section 10).  
*Constraint:*  $\mathbf{anorm} \geq 0.0$ .
- 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 *machine precision*,  $A$  is singular to working precision.
- 11: **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.

**NE\_BAD\_PARAM**

On entry, argument  $\langle value \rangle$  had an illegal value.

**NE\_INT**

On entry, **kl** =  $\langle value \rangle$ .

Constraint: **kl**  $\geq 0$ .

On entry, **ku** =  $\langle value \rangle$ .

Constraint: **ku**  $\geq 0$ .

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 0$ .

On entry, **pdab** =  $\langle value \rangle$ .

Constraint: **pdab**  $> 0$ .

**NE\_INT\_3**

On entry, **pdab** =  $\langle value \rangle$ , **kl** =  $\langle value \rangle$  and **ku** =  $\langle value \rangle$ .

Constraint: **pdab**  $\geq 2 \times \mathbf{kl} + \mathbf{ku} + 1$ .

**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.

**NE\_REAL**

On entry, **anorm** =  $\langle value \rangle$ .

Constraint: **anorm**  $\geq 0.0$ .

**7 Accuracy**

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.

**8 Parallelism and Performance**

nag\_dgbcon (f07bgc) is not threaded by NAG in any implementation.

nag\_dgbcon (f07bgc) 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 Users' Note for your implementation for any additional implementation-specific information.

**9 Further Comments**

A call to 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 nag\_dgbtrs (f07bec) 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\_zgbcon (f07buc).

## 10 Example

This example estimates 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 nag\_dgbtrf (f07bdc). The true condition number in the 1-norm is 56.40.

### 10.1 Program 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_FAIL(fail);

    printf("nag_dgbcon (f07bgc) Example Program Results\n\n");

    /* Skip heading in data file */
    scanf("%*[\n] ");
    scanf("%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)))
    {
        printf("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)
            scanf("%lf", &AB(i, j));
    }
scanf("%*[\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 */
/* nag_dgbtrf (f07bdc).
 * LU factorization of real m by n band matrix
 */
nag_dgbtrf(order, n, n, kl, ku, ab, pdab, ipiv, &fail);
if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dgbtrf (f07bdc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
/* Estimate condition number */
/* nag_dgbcon (f07bgc).
 * Estimate condition number of real band matrix, matrix
 * already factorized by nag_dgbtrf (f07bdc)
 */
nag_dgbcon(order, Nag_OneNorm, n, kl, ku, ab, pdab, ipiv,
           anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dgbcon (f07bgc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
/* Print condition number */
/* nag_machine_precision (x02ajc).
 * The machine precision
 */
if (rcond > nag_machine_precision)
    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_dgbcon (f07bgc) Example Program Data
  4  1  2           :Values of N, KL and KU
-0.23  2.54 -3.66
-6.98  2.46 -2.73 -2.13
        2.56  2.46  4.07
        -4.78 -3.82   :End of matrix A

```

## 10.3 Program Results

```

nag_dgbcon (f07bgc) Example Program Results
Estimate of condition number = 5.64e+01

```

---