

# NAG C Library Function Document

## nag\_dgecon (f07agc)

### 1 Purpose

nag\_dgecon (f07agc) estimates the condition number of a real matrix  $A$ , where  $A$  has been factorized by nag\_dgetrf (f07adc).

### 2 Specification

```
void nag_dgecon (Nag_OrderType order, Nag_NormType norm, Integer n,
                 const double a[], Integer pda, double anorm, double *rcond, NagError *fail)
```

### 3 Description

nag\_dgecon (f07agc) estimates the condition number of a real 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\_dge\_norm (f16rac) to compute  $\|A\|_1$  or  $\|A\|_\infty$ , and a call to nag\_dgetrf (f07adc) 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: **a**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **a** must be at least  $\max(1, \mathbf{pda} \times \mathbf{n})$ .  
 If **order** = **Nag\_ColMajor**, the (*i*, *j*)th element of the matrix *A* is stored in **a**[(*j* – 1) × **pda** + *i* – 1] and  
 if **order** = **Nag\_RowMajor**, the (*i*, *j*)th element of the matrix *A* is stored in **a**[(*i* – 1) × **pda** + *j* – 1].  
*On entry:* the *LU* factorization of *A*, as returned by nag\_dgetrf (f07adc).
- 5: **pda** – Integer *Input*  
*On entry:* the stride separating matrix row or column elements (depending on the value of **order**) in the array **a**.  
*Constraint:* **pda** ≥ max(1, **n**).
- 6: **anorm** – double *Input*  
*On entry:* if **norm** = **Nag\_OneNorm**, the 1-norm of the **original** matrix *A*; if **norm** = **Nag\_InfNorm**, the infinity-norm of the **original** matrix *A*. **anorm** may be computed by calling nag\_dge\_norm (f16rac) with the same value for the parameter **norm**. **anorm** must be computed either **before** calling nag\_dgetrf (f07adc) or else from a **copy** of the original matrix *A*.  
*Constraint:* **anorm** ≥ 0.0.
- 7: **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.
- 8: **fail** – NagError \* *Output*  
 The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry, **n** = *<value>*.

Constraint: **n** ≥ 0.

On entry, **pda** = *<value>*.

Constraint: **pda** > 0.

### NE\_INT\_2

On entry, **pda** = *<value>*, **n** = *<value>*.

Constraint: **pda** ≥ max(1, **n**).

### NE\_REAL

On entry, **anorm** = *<value>*.

Constraint: **anorm** ≥ 0.0.

### NE\_ALLOC\_FAIL

Memory allocation failed.

### NE\_BAD\_PARAM

On entry, parameter *<value>* had an illegal value.

**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 consult NAG for assistance.

**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 Further Comments**

A call to nag\_dgecon (f07agc) 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  $2n^2$  floating-point operations but takes considerably longer than a call to nag\_dgetrs (f07aec) 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 nag\_zgecon (f07auc).

**9 Example**

To estimate the condition number in the 1-norm of the matrix  $A$ , where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix}.$$

Here  $A$  is nonsymmetric and must first be factorized by nag\_dgetrf (f07adc). The true condition number in the 1-norm is 152.16.

**9.1 Program Text**

```
/* nag_dgecon (f07agc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf16.h>
#include <nagx02.h>
#include <math.h>

int main(void)
{
    /* Scalars */
    double    anorm, rcond;
    Integer    exit_status=0;
    Integer    i, ipiv_len, j, m, n, pda;
    NagError   fail;
    Nag_OrderType order;

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

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
    order = Nag_ColMajor;
#endif
```

```

#else
#define A(I,J) a[(I-1)*pda + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07agc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\\n] ");
    Vscanf("%ld%*[^\\n] ", &n);
    pda = n;
    m = n;
    ipiv_len = n;

    /* Allocate memory */
    if ( !(a = NAG_ALLOC(n * n, double)) ||
        !(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 = 1; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[^\\n] ");

    /* Compute norm of A */
    f16rac(order, Nag_OneNorm, n, n, a, pda, &anorm, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f16rac.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Factorize A */
    f07adc(order, n, n, a, pda, ipiv, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07adc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    Vprintf("\n");
    /* Estimate condition number */
    f07agc(order, Nag_OneNorm, n, a, pda, anorm, &rcond, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07agc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    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 (a) NAG_FREE(a);
    if (ipiv) NAG_FREE(ipiv);
    return exit_status;
}

```

## 9.2 Program Data

```
f07agc Example Program Data
4                               :Value of N
1.80    2.88    2.05   -0.89
5.25   -2.95   -0.95   -3.80
1.58   -2.69   -2.90   -1.04
-1.11   -0.66   -0.59    0.80   :End of matrix A
```

## 9.3 Program Results

```
f07agc Example Program Results
```

```
Estimate of condition number = 1.52e+02
```

---