NAG C Library Function Document

nag_zgecon (f07auc)

1 Purpose

nag_zgecon (f07auc) estimates the condition number of a complex matrix A, where A has been factorized by nag_zgetrf (f07arc).

2 Specification

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

3 Description

nag_zgecon (f07auc) estimates the condition number of a complex 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_zge_norm (f16uac) to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to nag_zgetrf (f07arc) 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** ≥ 0.

4: **a**[dim] – const Complex *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 $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix A is stored in $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$.

On entry: the LU factorization of A, as returned by nag_zgetrf (f07arc).

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: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

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_zge_norm (f16uac) with the same value for the parameter **norm**. **anorm** must be computed either **before** calling nag_zgetrf (f07arc) or else from a **copy** of the original matrix A.

Constraint: $\mathbf{anorm} \geq 0.0$.

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: $\mathbf{n} \geq 0$.

On entry, **pda** = ⟨*value*⟩.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

NE_REAL

On entry, **anorm** = ⟨*value*⟩.
Constraint: $\mathbf{anorm} \geq 0.0$.

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 ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where **rcond** is much larger.

8 Further Comments

A call to nag_zgecon (f07auc) 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 $8n^2$ real floating-point operations but takes considerably longer than a call to nag_zgetrs (f07asc) 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_dgecon (f07agc).

9 Example

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

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix}.$$

Here A is nonsymmetric and must first be factorized by nag_zgetrf (f07arc). The true condition number in the 1-norm is 231.86.

9.1 Program Text

```
/* nag_zgecon (f07auc) 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 <nagf16.h>
#include <nagx02.h>
#include <math.h>

int main(void)
{

  /* Scalars */
  double   anorm, rcond;
  Integer  exit_status=0;
  Integer  i, ipiv_len,j, n, pda;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  Complex  *a=0;
  Integer  *ipiv=0;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
```

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

  INIT_FAIL(fail);
  Vprintf("f07auc Example Program Results\n\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);

#ifdef NAG_COLUMN_MAJOR
  pda = n;
#else
  pda = n;
#endif
  ipiv_len = n;

  /* Allocate memory */
  if ( !(a = NAG_ALLOC(n * 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 = 1; j <= n; ++j)
        Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
  Vscanf("%*[^\n] ");
  /* Compute norm of A */
  f16uac(order, Nag_OneNorm, n, n, a, pda, &anorm, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f16uac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Factorize A */
  f07arc(order, n, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07arc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Estimate condition number */
  f07auc(order, Nag_OneNorm, n, a, pda, anorm, &rcond, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07auc.\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

```
f07auc Example Program Data
  4                                                          :Value of N
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67)  :End of matrix A
```

9.3 Program Results

```
f07auc Example Program Results

Estimate of condition number =   1.50e+02
```