NAG 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

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

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 $\infty$-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_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 Arguments

1: order – Nag_OrderType

   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

   On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated.

   norm = Nag_OneNorm

   $\kappa_1(A)$ is estimated.
\textbf{f07auc} \hfill NAG Library Manual

\begin{verbatim}

\textbf{norm} = Nag_InfNorm
\kappa_{\infty}(A) \text{ is estimated.}
\textbf{Constraint: norm} = Nag_OneNorm or Nag_InfNorm.

3: \textbf{n} – Integer \hfill \textit{Input}
\textit{On entry:} \textit{n}, the order of the matrix \textit{A}.
\textbf{Constraint: n} \geq 0.

4: \textbf{a[dim]} – const Complex \hfill \textit{Input}
\textbf{Note:} the dimension, \textit{dim}, of the array \textit{a} must be at least \textit{max}(1, pda \times n).
The \((i,j)\text{th} \text{ element} \text{ of} \text{ the matrix} \text{ } \textit{A} \text{ is stored in}
\text{ a}[(j-1) \times pda + i - 1] \text{ when order} = \text{Nag\_ColMajor;}
\text{ a}[(i-1) \times pda + j - 1] \text{ when order} = \text{Nag\_RowMajor.}
\textit{On entry:} \textit{the LU factorization of A}, \textit{as returned by nag_zgetrf (f07arc)}.

5: \textbf{pda} – Integer \hfill \textit{Input}
\textit{On entry:} \textit{the stride separating row or column elements (depending on the value of order) in the array a.}
\textbf{Constraint: pda} \geq \textit{max}(1, n).

6: \textbf{anorm} – double \hfill \textit{Input}
\textit{On entry:} if \textbf{norm} = \text{Nag\_OneNorm}, \textit{the} 1\text{-norm} \text{ of} \text{ the original matrix} \text{ } \textit{A}.
If \textbf{norm} = \text{Nag\_InfNorm}, \textit{the} \infty\text{-norm} \text{ of} \text{ the original matrix} \text{ } \textit{A}.
\textbf{anorm} \text{ may be computed by calling nag_zge_norm (f16uac) with the same value for the argument norm.}
\textbf{anorm} \text{ must be computed either before calling nag_zgetrf (f07arc) or else from a copy of the original matrix} \textit{A} (see Section 10).
\textbf{Constraint: anorm} \geq 0.0.

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

8: \textbf{fail} – NagError * \hfill \textit{Input/Output}
\textit{The NAG error argument} (see Section 3.6 in the Essential Introduction).
\end{verbatim}

\section{6 Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

On entry, argument \textit{\langle value\rangle} had an illegal value.
On entry, \( n = \langle value \rangle \).
Constraint: \( n \geq 0 \).
On entry, \( pda = \langle value \rangle \).
Constraint: \( pda > 0 \).

On entry, \( pda = \langle value \rangle \) and \( n = \langle value \rangle \).
Constraint: \( pda \geq \max(1, n) \).

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.

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.

On entry, \( anorm = \langle value \rangle \).
Constraint: \( anorm \geq 0 \).

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.

nag_zgecon (f07auc) is not threaded by NAG in any implementation.

nag_zgecon (f07auc) 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.

A call to nag_zgecon (f07auc) 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 \( 8n^2 \) real floating-point operations but takes considerably longer than a call to nag_zgetrs (f07asc) 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_dgecon (f07adc).
10 Example

This example estimates 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.

10.1 Program Text

/* nag_zgecon (f07auc) Example Program. *
 * Copyright 2014 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]
#else
    #define A(I, J) a[(I-1)*pda + J-1]
#endif

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

    INIT_FAIL(fail);
    printf("nag_zgecon (f07auc) Example Program Results\n\n");

    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
    #ifdef _WIN32
        scanf("%"NAG_IFMT"%*[\n ]", &n);
    #else
        scanf("%"NAG_IFMT"%*[\n ]", &n);
    #endif

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

    /* Skip heading in data file */
}
pda = n;
#endif
ipiv_len = n;

/* Allocate memory */
if (!((a = NAG_ALLOC(n * 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 = 1; j <= n; ++j)
    {
        #ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        #else
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        #endif
    }
    #ifdef _WIN32
        scanf_s("%*[^\n] ");
    #else
        scanf("%*[^\n] ");
    #endif
}

/* Compute norm of A */
/* nag_zge_norm (f16uac).
 * 1-norm, infinity-norm, Frobenius norm, largest absolute
 * element, complex general matrix
 */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zge_norm (f16uac).", fail.message);
    exit_status = 1;
    goto END;
}

/* Factorize A */
/* nag_zgetrf (f07arc).
 * LU factorization of complex m by n matrix
 */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgetrf (f07arc).", fail.message);
    exit_status = 1;
    goto END;
}

/* Estimate condition number */
/* nag_zgecon (f07auc).
 * Estimate condition number of complex matrix, matrix
 * already factorized by nag_zgetrf (f07arc)
 */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgecon (f07auc).", fail.message);
    exit_status = 1;
    goto END;
}

/* 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(a);
NAG_FREE(ipiv);
return exit_status;
}

10.2 Program Data

nag_zgecon (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

10.3 Program Results

nag_zgecon (f07auc) Example Program Results
Estimate of condition number = 1.50e+02