NAG C Library Function Document

nag_zsycon (f07nuc)

1 Purpose

nag_zsycon (f07nuc) estimates the condition number of a complex symmetric matrix $A$, where $A$ has been factorized by nag_zsytrf (f07nrc).

2 Specification

```c
void nag_zsycon (Nag_OrderType order, Nag_UploType uplo, Integer n,
                const Complex a[], Integer pda, const Integer ipiv[],
                double anorm, double *rcond, NagError *fail)
```

3 Description

nag_zsycon (f07nuc) estimates the condition number (in the 1-norm) of a complex symmetric matrix $A$:

$$
\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.
$$

Since $A$ is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if $A$ is singular, the function actually returns an estimate of the reciprocal of $\kappa_1(A)$.

The function should be preceded by a call to nag_zsy_norm (f16ufc) to compute $\|A\|_1$ and a call to nag_zsytrf (f07nrc) to compute the Bunch–Kaufman factorization of $A$. The function then uses Higham’s implementation of Hager’s method (see Higham (1988)) to estimate $\|A^{-1}\|_1$.

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:  
   uplo – Nag_UploType  
   
   **Input**  
   
   On entry: indicates how $A$ has been factorized as follows:  
   
   if uplo = Nag_Upper, $A = PUDU^T P^T$, where $U$ is upper triangular;  
   
   if uplo = Nag_Lower, $A = PLDL^T P^T$, where $L$ is lower triangular.  
   
   Constraint: uplo = Nag_Upper or Nag_Lower.

3:  
   n – Integer  
   
   **Input**  
   
   On entry: $n$, the order of the matrix $A$.  
   
   Constraint: $n \geq 0$. 


4: \[a]\text{[}\text{dim}\text{]} – \text{const Complex} \hspace{1cm} \text{Input}

\textbf{Note:} the dimension, \textit{dim}, of the array \textit{a} must be at least \text{max}(1, \textit{pda} \times \textit{n})..

\textit{On entry:} details of the factorization of \textit{A}, as returned by \texttt{nag_zsymrf (f07nrc)}.

5: \textit{pda} – Integer \hspace{1cm} \text{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) of the matrix in the array \textit{a}.

\textbf{Constraint:} \textit{pda} \geq \text{max}(1, \textit{n}).

6: \textit{ipiv}\text{[}\text{dim}\text{]} – \text{const Integer} \hspace{1cm} \text{Input}

\textbf{Note:} the dimension, \textit{dim}, of the array \textit{ipiv} must be at least \text{max}(1, \textit{n})..

\textit{On entry:} details of the interchanges and the block structure of \textit{D}, as returned by \texttt{nag_zsymrf (f07nrc)}.

7: \textit{anorm} – double \hspace{1cm} \text{Input}

\textit{On entry:} the 1-norm of the \textbf{original} matrix \textit{A}, which may be computed by calling \texttt{nag_zsym_norm (f16ufc)}. \textit{anorm} must be computed either \textbf{before} calling \texttt{nag_zsymrf (f07nrc)} or else from a copy of the original matrix \textit{A}.

\textbf{Constraint:} \textit{anorm} \geq 0.0.

8: \textit{rcond} – double * \hspace{1cm} \text{Output}

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

9: \textit{fail} – \texttt{NagError} * \hspace{1cm} \text{Output}

The NAG error parameter (see the Essential Introduction).

6 \hspace{0.5cm} \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

\textit{On entry,} \textit{n} = \textless\textit{value}\textgreater.

\textbf{Constraint:} \textit{n} \geq 0.

\textit{On entry,} \textit{pda} = \textless\textit{value}\textgreater.

\textbf{Constraint:} \textit{pda} > 0.

\textbf{NE_INT_2}

\textit{On entry,} \textit{pda} = \textless\textit{value}\textgreater, \textit{n} = \textless\textit{value}\textgreater.

\textbf{Constraint:} \textit{pda} \geq \text{max}(1, \textit{n}).

\textbf{NE_REAL}

\textit{On entry,} \textit{anorm} = \textless\textit{value}\textgreater.

\textbf{Constraint:} \textit{anorm} \geq 0.0.

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

\textit{On entry,} parameter \textless\textit{value}\textgreater 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 $r$, and in practice is nearly always less than
$10^6$, although examples can be constructed where rcond is much larger.

8 Further Comments
A call to nag_zsycon (f07nuc) involves solving a number of systems of linear equations of the form
$Ax = b$; 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_zsytrs (f07nsc) 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_dsycon (f07mgc).

9 Example
To estimate the condition number in the 1-norm (or infinity-norm) of the matrix $A$, where
\[
A = \begin{pmatrix}
-0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\
5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\
-7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\
3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \\
\end{pmatrix}
\]
Here $A$ is symmetric and must first be factorized by nag_zsytrf (f07nrc). The true condition number in the
1-norm is 32.92.

9.1 Program Text
/* nag_zsycon (f07nuc) 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>

int main(void)
{
    /* Scalars */
    double anorm, rcond;
    Integer i, j, n, pda;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    Complex *a=0;

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

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

INIT_FAIL(fail);
Vprintf("f07nuc 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

/* Allocate memory */
if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
     !(a = NAG_ALLOC(n * n, Complex)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
Vscanf(" ' %1s ' %*[\n ]", uplo);
if (*(unsigned char *)uplo == 'L')
    uplo_enum = Nag_Lower;
else if (*(unsigned char *)uplo == 'U')
    uplo_enum = Nag_Upper;
else
{
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}

if (uplo_enum == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
    Vscanf("%*[\n ]");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
    Vscanf("%*[\n ]");
}

/* Compute norm of A */
f16ufc(order, Nag_OneNorm, uplo_enum, n, a, pda, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f16ufc.\n%\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Factorize A */
f07nrc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
Vprintf("Error from f07nrc.\n\n", fail.message);
exit_status = 1;
goto END;
}
/* Estimate condition number */
f07nuc(order, uplo_enum, n, a, pda, ipiv, anorm, &rcond,
   &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07nuc.\n\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 (ipiv) NAG_FREE(ipiv);
if (a) NAG_FREE(a);
return exit_status;
}

9.2 Program Data

f07nuc Example Program Data

4
'L'

(-0.39,-0.71)
( 5.14,-0.64) ( 8.86, 1.81)
(-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12) :End of matrix A

9.3 Program Results

f07nuc Example Program Results

Estimate of condition number = 2.06e+01