NAG C Library Function Document

nag_zhpcon (f07puc)

1 Purpose

nag_zhpcon (f07puc) estimates the condition number of a complex Hermitian indefinite matrix $A$, where $A$ has been factorized by nag_zhptrf (f07prc), using packed storage.

2 Specification

void nag_zhpcon (Nag_OrderType order, Nag_UploType uplo, Integer n, const Complex ap[], const Integer ipiv[], double anorm, double *rcond, NagError *fail)

3 Description

nag_zhpcon (f07puc) estimates the condition number (in the 1-norm) of a complex Hermitian indefinite matrix $A$:

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

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

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_zhp_norm (f16udc) to compute $\|A\|_1$ and a call to nag_zhptrf (f07prc) 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, $A = PUDU^H PT$, where $U$ is upper triangular;

   if uplo = Nag_Lower, $A = PLDL^H PT$, where $L$ is lower triangular.

   Constraint: uplo = Nag_Upper or Nag_Lower.
3: \quad n – Integer

\textit{Input}

\textit{On entry:} \( n \), the order of the matrix \( A \).

\textit{Constraint:} \( n \geq 0 \).

4: \quad ap[dim] – const Complex

\textit{Input}

\textit{Note:} the dimension, \( dim \), of the array \( ap \) must be at least \( 1 + \lceil n/2 \rceil \).

\textit{On entry:} details of the factorization of \( A \) stored in packed form, as returned by \textit{nag_zhptrf (f07prc)}.

5: \quad ipiv[dim] – const Integer

\textit{Input}

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

\textit{On entry:} details of the interchanges and the block structure of \( D \), as returned by \textit{nag_zhptrf (f07prc)}.

6: \quad anorm – double

\textit{Input}

\textit{On entry:} the 1-norm of the \textbf{original} matrix \( A \), which may be computed by calling \textit{nag_zhp_norm (f16udc)}. \( anorm \) must be computed \textbf{either before} calling \textit{nag_zhptrf (f07prc)} or else from a copy of the original matrix \( A \).

\textit{Constraint:} \( anorm \geq 0.0 \).

7: \quad rcond – double *

\textit{Output}

\textit{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 \textit{machine precision}, \( A \) is singular to working precision.

8: \quad fail – NagError *

\textit{Output}

The NAG error parameter (see the Essential Introduction).

\section{Error Indicators and Warnings}

\textbf{NE_INT}

On entry, \( n = \langle \text{value} \rangle \).

\textit{Constraint:} \( n \geq 0 \).

\textbf{NE_REAL}

On entry, \( anorm = \langle \text{value} \rangle \).

\textit{Constraint:} \( anorm \geq 0.0 \).

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

\textbf{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_zhpccon (f07puc) 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_zhptrs (f07psc) 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_dspcon (f07pgc).

9 Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix $A$, where

$$A = \begin{pmatrix}
-1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\
1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\
2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\
3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i 
\end{pmatrix}. $$

Here $A$ is Hermitian indefinite, stored in packed form, and must first be factorized by nag_zhptrf (f07prc). The true condition number in the 1-norm is 9.10.

9.1 Program Text

/* nag_zhpccon (f07puc) Example Program.
 * Copyright 2001 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf07.h>
#include <nagf16.h>
#include <nagx02.h>

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

    #ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
    #else
    #define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
    #define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;

    /*...*/

    return exit_status;
}
#endif

INIT_FAIL(fail);
Vprintf("f07puc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[\n] ");
Vscanf("%ld%*[\n] ", &n);
ap_len = n * (n + 1)/2;

/* Allocate memory */
if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
 !(ap = NAG_ALLOC(ap_len, 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_UPPER(i,j).re, &A_UPPER(i,j).im);
  }
  Vscanf("%*[\n] ");
}
else
{
 for (i = 1; i <= n; ++i)
  {
   for (j = 1; j <= i; ++j)
    Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
  }
  Vscanf("%*[\n] ");
}

/* Compute norm of A */
f16udc(order, Nag_OneNorm, uplo_enum, n, ap, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
 Vprintf("Error from f16udc.\n%s\n", fail.message);
 exit_status = 1;
 goto END;
}

/* Factorize A */
f07prc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
 Vprintf("Error from f07prc.\n%s\n", fail.message);
 exit_status = 1;
 goto END;
}

/* Estimate condition number */
f07puc(order, uplo_enum, n, ap, ipiv, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
 Vprintf("Error from f07puc.\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 (ipiv) NAG_FREE(ipiv);
if (ap) NAG_FREE(ap);
return exit_status;

9.2 Program Data
f07puc Example Program Data
4 :Value of N
    ’L’ :Value of UPLO
    (-1.36, 0.00) (-8.87, 0.00)
    ( 1.58, -0.90) ( 1.84, 0.03) (-4.63, 0.00)
    ( 2.21, 0.21) (-1.78, -1.18) ( 0.11, -0.11) (-1.84, 0.00) :End of matrix A

9.3 Program Results
f07puc Example Program Results

Estimate of condition number =  6.68e+00