NAG Library Function Document

nag_zhecon (f07muc)

1 Purpose

nag_zhecon (f07muc) estimates the condition number of a complex Hermitian indefinite matrix $A$, where $A$ has been factorized by nag_zhetrf (f07mrc).

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_zhecon (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_zhecon (f07muc) 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_zhe_norm (f16ucc) to compute $\|A\|_1$ and a call to nag_zhetrf (f07mrc) 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 Arguments

1:  **order** – Nag_OrderType  
    *Input*  
    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:  **uplo** – Nag_UploType  
    *Input*  
    On entry: specifies how $A$ has been factorized.  
    uplo = Nag_Upper  
    $A = PU^HTUPT$, where $U$ is upper triangular.
uplo = Nag_Lower
A = PLDL^H PT, 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 ≥ 0.

4: a[dim] – const Complex
   Input
   Note: the dimension, dim, of the array a must be at least max(1, pda × n).
   On entry: details of the factorization of A, as returned by nag_zhetrf (f07mrc).

5: pda – Integer
   Input
   On entry: the stride separating row or column elements (depending on the value of order) of the matrix in the array a.
   Constraint: pda ≥ max(1, n).

6: ipiv[dim] – const Integer
   Input
   Note: the dimension, dim, of the array ipiv must be at least max(1, n).
   On entry: details of the interchanges and the block structure of D, as returned by nag_zhetrf (f07mrc).

7: anorm – double
   Input
   On entry: the 1-norm of the original matrix A, which may be computed by calling nag_zhe_norm (f16ucc) with its argument norm = Nag_OneNorm. anorm must be computed either before calling nag_zhetrf (f07mrc) or else from a copy of the original matrix A.
   Constraint: anorm ≥ 0.0.

8: 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.

9: fail – NagError *
   Input/Output
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

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

NE_BAD_PARAM
On entry, argument value had an illegal value.

NE_INT
On entry, n = (value).
Constraint: n ≥ 0.
On entry, pda = \langle value\rangle.
Constraint: pda > 0.

**NE_INT_2**

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

**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 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.

**NE_NO_LICENCE**

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.

**NE_REAL**

On entry, anorm = \langle value\rangle.
Constraint: anorm ≥ 0.0.

7  **Accuracy**

The computed estimate rcond is never less than the true value ρ, and in practice is nearly always less than 10^ρ, although examples can be constructed where rcond is much larger.

8  **Parallelism and Performance**

nag_zhecon (f07muc) is not threaded by NAG in any implementation.

nag_zhecon (f07muc) 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.

9  **Further Comments**

A call to nag_zhecon (f07muc) 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_zhetrs (f07msc) 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_dsycon (f07mge).
Example

This example estimates the condition number in the 1-norm (or ∞-norm) of the matrix $A$, where

$$A = \begin{pmatrix}
-1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\
1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\
2.21 + 0.21i & -1.84 + 0.03i & -6.63 + 0.00i & 0.11 + 0.11i \\
3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i 
\end{pmatrix}.$$ 

Here $A$ is Hermitian indefinite and must first be factorized by nag_zhetrf (f07mrc). The true condition number in the 1-norm is 9.10.

Program Text

```c
#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 i, j, n, pda;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv = 0;
    char nag_enum_arg[40];
    Complex *a = 0;

    /* Scalars */
    anorm, rcond;
    i, j, n, pda;
    exit_status = 0;
    uplo;
    fail;
    order;
    /* Arrays */
    ipiv = 0;
    nag_enum_arg[40];
    a = 0;

    INIT_FAIL(fail);
    printf("nag_zhecon (f07muc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n ]");
    #else
        scanf("%*[\n ]");
    #endif
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT"%*[\n ]", &n);
    #else
        scanf("%"NAG_IFMT"%*[\n ]", &n);
    #endif
    #ifdef NAG_COLUMN_MAJOR
        pda = n;
    #else
        pda = n;
    #endif
```
pda = n;
#endif

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

/* Read A from data file */
#ifdef __WIN32
    scanf_s(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n] ", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
* Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; 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
    else
    {
        for (i = 1; i <= n; ++i)
        {
            for (j = 1; j <= i; ++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_zhe_norm (f16ucc).
* 1-norm, infinity-norm, Frobenius norm, largest absolute
* element, complex Hermitian matrix */
nag_zhe_norm(order, Nag_OneNorm, uplo, n, a, pda, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhe_norm (f16ucc).\n");
    exit_status = 1;
    goto END;
}
/* Factorize A */
nag_zhetrf (f07mrc).
* Bunch-Kaufman factorization of complex Hermitian
  * indefinite matrix
*/
nag_zhetrf(order, uplo, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhetrf (f07mrc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Estimate condition number */
/* nag_zhecon (f07muc).
  * Estimate condition number of complex Hermitian indefinite
  * matrix, matrix already factorized by nag_zhetrf (f07mrc)
  */
nag_zhecon(order, uplo, n, a, pda, ipiv, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhecon (f07muc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_machine_precision (x02ajc).
  * The machine precision
*/
if (rcond >= nag_machine_precision)
    printf("Estimate of condition number =\n\n", 1.0/rcond);
else
    printf("A is singular to working precision\n");
END:
NAG_FREE(ipiv);
NAG_FREE(a);
return exit_status;

10.2 Program Data

nag_zhecon (f07muc) Example Program Data
   4 :Value of n
   Nag_Lower :Value of uplo
   (-1.36, 0.00)
   ( 1.58,-0.90) ( -8.87, 0.00)
   ( 2.21, 0.21) ( -1.84, 0.03) (-4.63, 0.00)
   ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00) :End of matrix A

10.3 Program Results

nag_zhecon (f07muc) Example Program Results

Estimate of condition number = 6.68e+00