NAG 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

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

```
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:

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

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

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 $\left\|A^{-1}\right\|_1$.

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
$A = PUDU^{\mathrm{T}}P^{\mathrm{T}}$, where U is upper triangular.

uplo = Nag_Lower
$A = PLDL^{\mathrm{T}}P^{\mathrm{T}}$, 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: $\mathbf{n} \geq 0$.

4: **a**$[dim]$ – const Complex *Input*

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

On entry: details of the factorization of A, as returned by nag_zsytrf (f07nrc).

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

6: **ipiv**$[dim]$ – const Integer *Input*

Note: the dimension, *dim*, of the array **ipiv** must be at least $\max(1, \mathbf{n})$.

On entry: details of the interchanges and the block structure of D, as returned by nag_zsytrf (f07nrc).

7: **anorm** – double *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling nag_zsy_norm (f16ufc) with its argument **norm** = Nag_OneNorm. **anorm** must be computed either **before** calling nag_zsytrf (f07nrc) or else from a **copy** of the original matrix A.

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

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.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle value \rangle$.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

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

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.

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 10ρ, although examples can be constructed where **rcond** is much larger.

8 Parallelism and Performance

nag_zsycon (f07nuc) is not threaded by NAG in any implementation.

nag_zsycon (f07nuc) 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 Users' Note for your implementation for any additional implementation-specific information.

9 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 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 (f07mgc).

10 Example

This example estimates the condition number in the 1-norm (or ∞-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.

10.1 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 <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;

#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);

  printf("nag_zsycon (f07nuc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%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)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
  scanf(" %39s%*[^\n] ", nag_enum_arg);
  /* 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)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
        }
      scanf("%*[^\n] ");
    }

  /* Compute norm of A */
```

```
  /* nag_zsy_norm (f16ufc).
   * 1-norm, infinity-norm, Frobenius norm, largest absolute
   * element, complex symmetric matrix
   */
  nag_zsy_norm(order, Nag_OneNorm, uplo, n, a, pda, &anorm, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zsy_norm (f16ufc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Factorize A */

  /* nag_zsytrf (f07nrc).
   * Bunch-Kaufman factorization of complex symmetric matrix
   */
  nag_zsytrf(order, uplo, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zsytrf (f07nrc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Estimate condition number */
  /* nag_zsycon (f07nuc).
   * Estimate condition number of complex symmetric matrix,
   * matrix already factorized by nag_zsytrf (f07nrc)
   */
  nag_zsycon(order, uplo, n, a, pda, ipiv, anorm, &rcond,
             &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zsycon (f07nuc).\n%s\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 =%11.2e\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_zsycon (f07nuc) Example Program Data
  4                                                          :Value of n
  Nag_Lower                                                  :Value of uplo
 (-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
```

10.3 Program Results

```
nag_zsycon (f07nuc) Example Program Results

Estimate of condition number =   2.06e+01
```