NAG Library Function Document

nag_zppcon (f07guc)

1 Purpose

nag_zppcon (f07guc) estimates the condition number of a complex Hermitian positive definite matrix A, where A has been factorized by nag_zpptrf (f07grc), using packed storage.

2 Specification

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

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

3 Description

nag_zppcon (f07guc) estimates the condition number (in the 1-norm) of a complex Hermitian positive definite 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_zpptrf (f07grc) to compute the Cholesky 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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
$A = U^{\mathrm{H}}U$, where U is upper triangular.

uplo = Nag_Lower
$A = LL^{\mathrm{H}}$, 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: **ap**[*dim*] – const Complex *Input*

Note: the dimension, *dim*, of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n} + 1)/2)$.

On entry: the Cholesky factor of A stored in packed form, as returned by nag_zpptrf (f07grc).

5: **anorm** – double *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling nag_zhp_norm (f16udc) with its argument $\mathbf{norm} = \text{Nag\_OneNorm}$. **anorm** must be computed either **before** calling nag_zpptrf (f07grc) or else from a **copy** of the original matrix A.

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

6: **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.

7: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle\mathit{value}\rangle$ had an illegal value.

NE_INT

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

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_REAL

On entry, $\mathbf{anorm} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{anorm} \geq 0.0$.

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_zppcon (f07guc) 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_zppcon (f07guc) 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_zpptrs (f07gsc) 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_dppcon (f07ggc).

10 Example

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

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive definite, stored in packed form, and must first be factorized by nag_zpptrf (f07grc). The true condition number in the 1-norm is 201.92.

10.1 Program Text

```
/* nag_zppcon (f07guc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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;
  Nag_OrderType order;
  /* Arrays */
  char nag_enum_arg[40];
  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;
#endif

  INIT_FAIL(fail);

  printf("nag_zppcon (f07guc) 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
  ap_len = n * (n + 1) / 2;

  /* Allocate memory */
  if (!(ap = NAG_ALLOC(ap_len, Complex)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  /* Read A from data file */
#ifdef _WIN32
  scanf_s(" %39s%*[^\n] ", nag_enum_arg, (unsigned)_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_UPPER(i, j).re, &A_UPPER(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(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_LOWER(i, j).re, &A_LOWER(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A_LOWER(i, j).re, &A_LOWER(i, j).im);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }
  /* Compute norm of A */
  /* nag_zhp_norm (f16udc).
   * 1-norm, infinity-norm, Frobenius norm, largest absolute
   * element, complex Hermitian matrix, packed storage
   */
  nag_zhp_norm(order, Nag_OneNorm, uplo, n, ap, &anorm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zhp_norm (f16udc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Factorize A */
  /* nag_zpptrf (f07grc).
   * Cholesky factorization of complex Hermitian
   * positive-definite matrix, packed storage
   */
  nag_zpptrf(order, uplo, n, ap, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zpptrf (f07grc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Estimate condition number */
  /* nag_zppcon (f07guc).
   * Estimate condition number of complex Hermitian
   * positive-definite matrix, matrix already factorized by
   * nag_zpptrf (f07grc), packed storage
   */
  nag_zppcon(order, uplo, n, ap, anorm, &rcond, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zppcon (f07guc).\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\n", 1.0 / rcond);
  else {
    printf("A is singular to working precision\n");
  }
END:
  NAG_FREE(ap);
  return exit_status;
}
```

10.2 Program Data

```
nag_zppcon (f07guc) Example Program Data
  4                                                      :Value of n
  Nag_Lower                                              :Value of uplo
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

10.3 Program Results

```
nag_zppcon (f07guc) Example Program Results

Estimate of condition number =   1.51e+02
```