NAG Library Function Document

nag_ztpcon (f07uuc)

1 Purpose

nag_ztpcon (f07uuc) estimates the condition number of a complex triangular matrix, using packed storage.

2 Specification

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

void nag_ztpcon (Nag_OrderType order, Nag_NormType norm, Nag_UploType uplo,
                Nag_DiagType diag, Integer n, const Complex ap[], double *rcond,
                NagError *fail)
```

3 Description

nag_ztpcon (f07uuc) estimates the condition number of a complex triangular matrix $A$, in either the 1-norm or the $\infty$-norm, using packed storage:

$$
\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.
$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if $A$ is singular, the function actually returns an estimate of the reciprocal of the condition number.

The function computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham’s implementation of Hager’s method (see Higham (1988)) to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

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

   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: norm – Nag_NormType

   On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated.

   norm = Nag_OneNorm
   $\kappa_1(A)$ is estimated.

   norm = Nag_InfNorm
   $\kappa_\infty(A)$ is estimated.

   Constraint: norm = Nag_OneNorm or Nag_InfNorm.
3: uplo – Nag_UploType

*Input*

*On entry:* specifies whether $A$ is upper or lower triangular.

uplo = Nag_Upper
- $A$ is upper triangular.

uplo = Nag_Lower
- $A$ is lower triangular.

*Constraint:* uplo = Nag_Upper or Nag_Lower.

4: diag – Nag_DiagType

*Input*

*On entry:* indicates whether $A$ is a nonunit or unit triangular matrix.

diag = Nag_NonUnitDiag
- $A$ is a nonunit triangular matrix.

diag = Nag_UnitDiag
- $A$ is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

*Constraint:* diag = Nag_NonUnitDiag or Nag_UnitDiag.

5: n – Integer

*Input*

*On entry:* $n$, the order of the matrix $A$.

*Constraint:* $n \geq 0$.

6: ap[dim] – const Complex

*Input*

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

*On entry:* the $n$ by $n$ triangular matrix $A$, packed by rows or columns.

The storage of elements $A_{ij}$ depends on the order and uplo arguments as follows:

- if order = Nag_ColMajor and uplo = Nag_Upper,
  - $A_{ij}$ is stored in $ap[(j - 1) \times j/2 + i - 1]$, for $i \leq j$;
- if order = Nag_ColMajor and uplo = Nag_Lower,
  - $A_{ij}$ is stored in $ap[2n - j] \times (j - 1)/2 + i - 1]$, for $i \geq j$;
- if order = Nag_RowMajor and uplo = Nag_Upper,
  - $A_{ij}$ is stored in $ap[(2n - i) \times (i - 1)/2 + j - 1]$, for $i \leq j$;
- if order = Nag_RowMajor and uplo = Nag_Lower,
  - $A_{ij}$ is stored in $ap[(i - 1) \times i/2 + j - 1]$, for $i \geq j$.

If diag = Nag_UnitDiag, the diagonal elements of AP are assumed to be 1, and are not referenced; the same storage scheme is used whether diag = Nag_NonUnitDiag or diag = Nag_UnitDiag.

7: 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.

8: 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.

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.

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_ztpcon (f07uuc) is not threaded by NAG in any implementation.

nag_ztpcon (f07uuc) 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_ztpcon (f07uuc) involves solving a number of systems of linear equations of the form Ax = b or A^Hx = b; the number is usually 5 and never more than 11. Each solution involves approximately 4n^2 real floating-point operations but takes considerably longer than a call to nag_ztptrs (f07usc) 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_dtpcon (f07uec).
10 Example

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

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix},$$

using packed storage. The true condition number in the 1-norm is 70.27.

10.1 Program Text

/ * nag_ztpcon (f07uuc) Example Program.  
*  
* Copyright 2014 Numerical Algorithms Group.  
*  
* Mark 7b revised, 2004.  
* /

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double rcond;
    Integer i, j, n;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    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_ztpcon (f07uuc) Example Program Results\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

    /* Allocate memory */
    if (!ap = NAG_ALLOC(n * n, Complex))
    {
printf("Allocation failure\n");
exiit_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_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
}
#endif
/* Estimate condition number */
/* nag_ztpcon (f07uuc). * Estimate condition number of complex triangular matrix, * packed storage */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztpcon (f07uuc).\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
/* 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(ap);
return exit_status;
}

10.2 Program Data

nag_ztpcon (f07uuc) Example Program Data

<table>
<thead>
<tr>
<th>:Value of n</th>
<th>Nag_Lower</th>
<th>:Value of uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 4.78, 4.56)</td>
<td>( 2.00, -0.30)</td>
<td>(-4.11, 1.25)</td>
</tr>
<tr>
<td>( 2.89, -1.34)</td>
<td>( 2.36, -4.25)</td>
<td>( 4.15, 0.80)</td>
</tr>
<tr>
<td>(-1.89, 1.15)</td>
<td>( 0.04, -3.69)</td>
<td>(-0.02, 0.46)</td>
</tr>
<tr>
<td>( 0.33, -0.26)</td>
<td>:End of matrix A</td>
<td></td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_ztpcon (f07uuc) Example Program Results

Estimate of condition number = 3.74e+01