NAG Library Function Document

nag_zptcon (f07juc)

1 Purpose

nag_zptcon (f07juc) computes the reciprocal condition number of a complex n by n Hermitian positive definite tridiagonal matrix A, using the LDL^{H} factorization returned by nag_zpttrf (f07jrc).

2 Specification

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

```
void nag_zptcon (Integer n, const double d[], const Complex e[],
     double anorm, double *rcond, NagError *fail)
```

3 Description

nag_zptcon (f07juc) should be preceded by a call to nag_zpttrf (f07jrc), which computes a modified Cholesky factorization of the matrix A as

$$A = LDL^{\mathrm{H}},$$

where L is a unit lower bidiagonal matrix and D is a diagonal matrix, with positive diagonal elements. nag_zptcon (f07juc) then utilizes the factorization to compute $\left\|A^{-1}\right\|_1$ by a direct method, from which the reciprocal of the condition number of A, $1/\kappa(A)$ is computed as

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

$1/\kappa(A)$ is returned, rather than $\kappa(A)$, since when A is singular $\kappa(A)$ is infinite.

4 References

Higham N J (2002) *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

5 Arguments

1: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

2: **d**[dim] – const double *Input*

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

On entry: must contain the n diagonal elements of the diagonal matrix D from the LDL^{H} factorization of A.

3: **e**[dim] – const Complex *Input*

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

On entry: must contain the $(n-1)$ subdiagonal elements of the unit lower bidiagonal matrix L. (**e** can also be regarded as the superdiagonal of the unit upper bidiagonal matrix U from the $U^{\mathrm{H}}DU$ factorization of A.)

4: **anorm** – double *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed as shown in Section 10. **anorm** must be computed either **before** calling nag_zpttrf (f07jrc) or else from a **copy** of the original matrix A.

Constraint: **anorm** ≥ 0.0.

5: **rcond** – double * *Output*

On exit: the reciprocal condition number, $1/\kappa_1(A) = 1/\left(\|A\|_1\|A^{-1}\|_1\right)$.

6: **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 ⟨*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 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, **anorm** = ⟨*value*⟩.
Constraint: **anorm** ≥ 0.0.

7 Accuracy

The computed condition number will be the exact condition number for a closely neighbouring matrix.

8 Parallelism and Performance

nag_zptcon (f07juc) is not threaded in any implementation.

9 Further Comments

The condition number estimation requires $O(n)$ floating-point operations.

See Section 15.6 of Higham (2002) for further details on computing the condition number of tridiagonal matrices.

The real analogue of this function is nag_dptcon (f07jgc).

10 Example

This example computes the condition number of the Hermitian positive definite tridiagonal matrix A given by

$$A = \begin{pmatrix} 16.0 & 16.0 - 16.0i & 0 & 0 \\ 16.0 + 16.0i & 41.0 & 18.0 + 9.0i & 0 \\ 0 & 18.0 - \ \ 9.0i & 46.0 & 1.0 + 4.0i \\ 0 & 0 & 1.0 - 4.0i & 21.0 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_zptcon (f07juc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 * UNFINISHED - replace commented out climp calls
 */

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

int main(void)
{
#define CABS(e) sqrt(e.re * e.re + e.im * e.im)

  /* Scalars */
  double anorm, rcond;
  Integer exit_status = 0, i, n;

  /* Arrays */
  Complex *e = 0;
  double *d = 0;

  /* Nag Types */
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_zptcon (f07juc) 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
```

```
  if (n < 0) {
    printf("Invalid n\n");
    exit_status = 1;
    goto END;
  }

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

  /* Read the lower bidiagonal part of the tridiagonal matrix A from */
  /* data file */
#ifdef _WIN32
  for (i = 0; i < n; ++i)
    scanf_s("%lf", &d[i]);
#else
  for (i = 0; i < n; ++i)
    scanf("%lf", &d[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  for (i = 0; i < n - 1; ++i)
    scanf_s(" ( %lf , %lf )", &e[i].re, &e[i].im);
#else
  for (i = 0; i < n - 1; ++i)
    scanf(" ( %lf , %lf )", &e[i].re, &e[i].im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Compute the 1-norm of A */
  anorm = MAX(ABS(d[0]) + CABS(e[0]), CABS(e[n - 2]) + ABS(d[n - 1]));
  for (i = 1; i < n - 1; ++i)
    anorm = MAX(anorm, ABS(d[i]) + CABS(e[i]) + CABS(e[i - 1]));

  /* Factorize A using nag_zpttrf (f07jrc). */
  nag_zpttrf(n, d, e, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zpttrf (f07jrc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Estimate the condition number of A using nag_zptcon (f07juc). */
  nag_zptcon(n, d, e, anorm, &rcond, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zptcon (f07juc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print the estimated condition number */
  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. RCOND = %11.2e\n\n", rcond);

END:
```

```
  NAG_FREE(e);
  NAG_FREE(d);

  return exit_status;
}
```

10.2 Program Data

```
nag_zptcon (f07juc) Example Program Data
    4                                                   : n
   16.0           41.0           46.0           21.0 : diagonal d
 ( 16.0, 16.0) ( 18.0, -9.0) (  1.0, -4.0)             : sub-diagonal e
```

10.3 Program Results

```
nag_zptcon (f07juc) Example Program Results

Estimate of condition number =    9.21e+03
```