NAG Library Function Document

nag_zhf_norm (f16ukc)

1 Purpose

nag_zhf_norm (f16ukc) returns the value of the 1-norm, the ∞-norm, the Frobenius norm, or the maximum absolute value of the elements of a complex Hermitian matrix A stored in Rectangular Full Packed (RFP) format.

2 Specification

```
#include <nag.h>
#include <nagf16.h>
```

```
void nag_zhf_norm (Nag_OrderType order, Nag_NormType norm,
     Nag_RFP_Store transr, Nag_UploType uplo, Integer n, const Complex ar[],
     double *r, NagError *fail)
```

3 Description

Given a complex n by n symmetric matrix, A, nag_zhf_norm (f16ukc) calculates one of the values given by

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \qquad \text{(the 1-norm of } A\text{)},$$

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \qquad \text{(the } \infty\text{-norm of } A\text{)},$$

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \qquad \text{(the Frobenius norm of } A\text{)}, \qquad \text{or}$$

$$\max_{i,j} |a_{ij}| \qquad \text{(the maximum absolute element value of } A\text{)}.$$

A is stored in compact form using the RFP format. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction.

4 References

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee http://www.netlib.org/blas/blast-forum/blas-report.pdf

Gustavson F G, Waśniewski J, Dongarra J J and Langou J (2010) Rectangular full packed format for Cholesky's algorithm: factorization, solution, and inversion *ACM Trans. Math. Software* **37, 2**

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: **norm** – Nag_NormType *Input*

On entry: specifies the value to be returned.

norm = Nag_OneNorm
The 1-norm.

norm = Nag_InfNorm
The ∞-norm.

norm = Nag_FrobeniusNorm
The Frobenius (or Euclidean) norm.

norm = Nag_MaxNorm
The value $\max_{i,j} |a_{ij}|$ (not a norm).

Constraint: **norm** = Nag_OneNorm, Nag_InfNorm, Nag_FrobeniusNorm or Nag_MaxNorm.

3: **transr** – Nag_RFP_Store *Input*

On entry: specifies whether the normal RFP representation of A or its conjugate transpose is stored.

transr = Nag_RFP_Normal
The matrix A is stored in normal RFP format.

transr = Nag_RFP_ConjTrans
The conjugate transpose of the RFP representation of the matrix A is stored.

Constraint: **transr** = Nag_RFP_Normal or Nag_RFP_ConjTrans.

4: **uplo** – Nag_UploType *Input*

On entry: specifies whether the upper or lower triangular part of A is stored.

uplo = Nag_Upper
The upper triangular part of A is stored.

uplo = Nag_Lower
The lower triangular part of A is stored.

Constraint: **uplo** = Nag_Upper or Nag_Lower.

5: **n** – Integer *Input*

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

If $n = 0$, then nag_zhf_norm (f16ukc) returns immediately.

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

6: **ar**[$\mathbf{n} \times (\mathbf{n} + \mathbf{1})/\mathbf{2}$] – const Complex *Input*

On entry: the upper or lower triangular part (as specified by **uplo**) of the n by n Hermitian matrix A, in either normal or transposed RFP format (as specified by **transr**). The storage format is described in detail in Section 3.3.3 in the f07 Chapter Introduction.

7: **r** – double * *Output*

On exit: the value of the norm specified by **norm**.

8: **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 3.2.1.2 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 3.6.6 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 3.6.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

8 Parallelism and Performance

nag_zhf_norm (f16ukc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example reads in the lower triangular part of a symmetric matrix, converts this to RFP format, then calculates the norm of the matrix for each of the available norm types.

10.1 Program Text

```
/* nag_zhf_norm (f16ukc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagf16.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  double r_fro, r_inf, r_max, r_one;
  Integer i, j, n, pda;
  /* Arrays */
  Complex *a = 0, *ar = 0;
  char nag_enum_arg[40];
  /* Nag Types */
  Nag_OrderType order;
  Nag_RFP_Store transr;
  Nag_UploType uplo;
  NagError fail;

#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_zhf_norm (f16ukc) 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
  pda = n;
  if (!(a = NAG_ALLOC(pda * n, Complex)) ||
      !(ar = NAG_ALLOC((n * (n + 1)) / 2, Complex)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  /* Nag_RFP_Store */
#ifdef _WIN32
  scanf_s("%39s ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s ", nag_enum_arg);
#endif
  transr = (Nag_RFP_Store) nag_enum_name_to_value(nag_enum_arg);
  /* Nag_UploType */
#ifdef _WIN32
  scanf_s("%39s  %*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s  %*[^\n] ", nag_enum_arg);
#endif
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
  /* Read upper or lower triangle of matrix A from data file */
  if (uplo == Nag_Lower) {
    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
      }
    }
  }
  else {
    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
      }
    }
  }
  /* Convert complex Hermitian matrix A from full to rectangular full packed
   * storage format (stored in ar) using nag_ztrttf (f01vfc).
   */
  nag_ztrttf(order, transr, uplo, n, a, pda, ar, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ztrttf (f01vfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nNorms of Hermitian matrix stored in RFP format in ar:\n\n");

  /* Get, in turn, the 1-norm, infinity norm, Frobenius norm, and
   * largest absolute element of the Hermitian matrix A stored
   * in rectangular full packed format in ar using nag_zhf_norm (f16ukc).
   */
  nag_zhf_norm(order, Nag_OneNorm, transr, uplo, n, ar, &r_one, &fail);

  if (fail.code == NE_NOERROR) {
    printf("One norm           = %9.4f\n", r_one);
    nag_zhf_norm(order, Nag_InfNorm, transr, uplo, n, ar, &r_inf, &fail);
  }
  if (fail.code == NE_NOERROR) {
    printf("Infinity norm      = %9.4f\n", r_inf);
    nag_zhf_norm(order, Nag_FrobeniusNorm, transr, uplo, n, ar, &r_fro,
                 &fail);
  }
  if (fail.code == NE_NOERROR) {
    printf("Frobenius norm     = %9.4f\n", r_fro);
    nag_zhf_norm(order, Nag_MaxNorm, transr, uplo, n, ar, &r_max, &fail);
  }
  if (fail.code == NE_NOERROR) {
    printf("Maximum norm       = %9.4f\n", r_max);
  }
  else {
    printf("Error from nag_zhf_norm (f16ukc).\n%s\n", fail.message);
    exit_status = 1;
  }

END:
  NAG_FREE(a);
  NAG_FREE(ar);
  return exit_status;
}
```

10.2 Program Data

```
nag_zhf_norm (f16ukc) Example Program Data
  6                                  : n
  Nag_RFP_Normal Nag_Lower           : transr, uplo
  (1.0,1.1)
  (2.0,2.1)  (2.0,2.1)
  (3.0,3.3)  (3.3,3.0)  (3.2,3.0)
  (4.0,4.4)  (4.0,4.3)  (4.0,4.2)  (4.0,4.1)
  (5.0,5.1)  (5.0,5.2)  (5.3,5.0)  (5.0,5.4)  (5.5,5.0)
  (6.9,6.0)  (6.0,6.8)  (6.7,6.0)  (6.0,6.6)  (6.5,6.0)  (6.0,6.4) : matrix A
```

10.3 Program Results

```
nag_zhf_norm (f16ukc) Example Program Results

Norms of Hermitian matrix stored in RFP format in ar:

One norm           =   50.9719
Infinity norm      =   50.9719
Frobenius norm     =   40.3801
Maximum norm       =    9.1439
```