NAG Library Function Document

nag_zhe_norm (f16ucc)

1 Purpose

nag_zhe_norm (f16ucc) calculates the value of the 1-norm, the ∞-norm, the Frobenius norm or the maximum absolute value of the elements of a complex n by n Hermitian matrix.

2 Specification

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

void nag_zhe_norm (Nag_OrderType order, Nag_NormType norm,
    Nag_UploType uplo, Integer n, const Complex a[], Integer pda, double *r,
    NagError *fail)
```

3 Description

Given a complex n by n Hermitian matrix, A, nag_zhe_norm (f16ucc) calculates one of the values given by

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|,$$

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|,$$

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

or

$$\max_{i,j} |a_{ij}|.$$

Note that, since A is symmetric, $\|A\|_1 = \|A\|_\infty$.

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

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:    **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 $\max\limits_{i,j} \left| a_{ij} \right|$ (not a norm).

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

3:    **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.

4:    **n** – Integer   *Input*

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

    If $n = 0$, then **n** is set to zero.

    Constraint: **n** ≥ 0.

5:    **a**[*dim*] – const Complex   *Input*

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

    On entry: the n by n Hermitian matrix A.

    If **order** = 'Nag_ColMajor', A_{ij} is stored in **a**$[(j-1) \times \textbf{pda} + i - 1]$.

    If **order** = 'Nag_RowMajor', A_{ij} is stored in **a**$[(i-1) \times \textbf{pda} + j - 1]$.

    If **uplo** = 'Nag_Upper', the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

    If **uplo** = 'Nag_Lower', the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

6:    **pda** – Integer   *Input*

    On entry: the stride separating row or column elements (depending on the value of **order**) of the matrix A in the array **a**.

    Constraint: **pda** $\geq \max(1, \textbf{n})$.

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 3.6 in the Essential Introduction).

6   Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_INT

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

NE_INT_2

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

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

Not applicable.

9   Further Comments

None.

10   Example

See Section 10 in nag_zpocon (f07fuc) and nag_zhecon (f07muc).