

NAG Library Function Document

nag_dsy_norm (f16rcc)

1 Purpose

nag_dsy_norm (f16rcc) calculates the value of the 1-norm, the ∞ -norm, the Frobenius norm or the maximum absolute value of the elements of a real n by n symmetric matrix.

2 Specification

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

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

3 Description

Given a real n by n symmetric matrix, A , nag_dsy_norm (f16rcc) calculates one of the values given by

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

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

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

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

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 $\max_{i,j} |a_{ij}|$ (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 double *Input*
Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$.
On entry: the n by n symmetric matrix A .
 If **order** = Nag_ColMajor, A_{ij} is stored in **a**[($j - 1$) \times **pda** + $i - 1$].
 If **order** = Nag_RowMajor, A_{ij} is stored in **a**[($i - 1$) \times **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** $\geq \max(1, \mathbf{n})$.
- 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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_INT

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

NE_INT_2

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

NE_INTERNAL_ERROR

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 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_dpocon (f07fgc) and nag_dsycon (f07mgc).
