Module 4.1: nag_mat_norm
Norms of a Matrix

nag_mat_norm provides procedures to compute the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex matrix.

Contents

Introduction

This module contains procedures to compute the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex matrix. It caters for different types of matrices and different storage schemes.

For a general m by n matrix A the norms are given by:

- 1-norm (One-norm): $\|A\|_1 = \max_j \sum_{i=1}^{m} |a_{ij}|$;
- ∞-norm (Infinity-norm): $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$;
- Frobenius or Euclidean norm: $\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$.

If A is symmetric or Hermitian, $\|A\|_1 = \|A\|_\infty$.

Procedure: nag_gen_mat_norm

1 Description

nag_gen_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a general real or complex matrix A.

2 Usage

```
USE nag_mat_norm
[value =] nag_gen_mat_norm(a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

m — the number of rows of the matrix A

n — the number of columns of the matrix A

3.1 Mandatory Argument

a(m, n) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the matrix A.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If norm = '1', 'o' or 'O', the 1-norm, $\|A\|_1$;

if norm = 'i' or 'I', the ∞-norm, $\|A\|_\infty$;

if norm = 'f', 'F', 'e' or 'E', the Frobenius or Euclidean norm, $\|A\|_F$;

if norm = 'm' or 'M', the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: norm = '1'.

Constraints: norm = '1', 'o', 'O', 'i', 'I', 'f', 'F', 'e', 'E', 'm' or 'M'.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document nag_error_handling (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to nag_set_error before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

Procedure: nag_gen_bnd_mat_norm

1 Description

nag_gen_bnd_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a square banded real or complex matrix A.

2 Usage

```
USE nag_mat_norm
[value =] nag_gen_bnd_mat_norm(a, ku  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the matrix A

$k_l \geq 0$ — the number of sub-diagonals in the band matrix A

3.1 Mandatory Arguments

a($k_l + k_u + 1, n$) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the band matrix A; element a_{ij} must be stored in a($k_u + i - j + 1, j$) for $\max(j - k_u, 1) \leq i \leq \min(j + k_l, n)$.

ku — integer, intent(in)

Input: the number k_u of super-diagonals in the band matrix A.

Constraints: ku ≥ 0.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If norm = '1', 'o' or 'O', the 1-norm, $\|A\|_1$;

if norm = 'i' or 'I', the ∞-norm, $\|A\|_\infty$;

if norm = 'f', 'F', 'e' or 'E', the Frobenius or Euclidean norm, $\|A\|_F$;

if norm = 'm' or 'M', the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: norm = '1'.

Constraints: norm = '1', 'o', 'O', 'i', 'I', 'f', 'F', 'e', 'E', 'm' or 'M'.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Procedure: nag_sym_mat_norm

1 Description

nag_sym_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real symmetric, complex Hermitian or complex symmetric matrix A.

The procedure allows conventional or packed storage for A.

2 Usage

```
USE nag_mat_norm
[value =] nag_sym_mat_norm(uplo, a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

2.1 Interfaces

Distinct interfaces are provided for each of the 4 combinations of the following cases:

Real / complex data

Real data: a is of type real(kind=*wp*).

Complex data: a is of type complex(kind=*wp*) and the matrix is Hermitian or symmetric.

Conventional / packed storage (see the Chapter Introduction)

Conventional: a is a rank-2 array.

Packed: a is a rank-1 array.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '$\mathbf{x}(n)$' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the matrix A

3.1 Mandatory Arguments

uplo — character(len=1), intent(in)

Input: specifies whether the upper or lower triangle of A is supplied.

If uplo = 'u' or 'U', the upper triangle is supplied;

if uplo = 'l' or 'L', the lower triangle is supplied.

Constraints: uplo = 'u', 'U', 'l' or 'L'.

a(n,n) **/ a**$(n(n+1)/2)$ — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the matrix A.

Conventional storage (`a` has shape (n,n))

If `uplo` = `'u'`, the upper triangle of A must be stored, and elements below the diagonal need not be set;

if `uplo` = `'l'`, the lower triangle of A must be stored, and elements above the diagonal need not be set.

Packed storage (`a` has shape $(n(n+1)/2)$)

If `uplo` = `'u'`, the upper triangle of A must be stored, packed by columns, with a_{ij} in `a(`$i+j(j-1)/2$`)` for $i \leq j$;

if `uplo` = `'l'`, the lower triangle of A must be stored, packed by columns, with a_{ij} in `a(`$i+(2n-j)(j-1)/2$`)` for $i \geq j$.

Note: if A is complex Hermitian, its diagonal elements must have zero imaginary parts but the procedure does not check for this.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If `norm` = `'1'`, `'o'` or `'O'`, the 1-norm, $\|A\|_1$;

if `norm` = `'i'` or `'I'`, the ∞-norm, $\|A\|_\infty$;

if `norm` = `'f'`, `'F'`, `'e'` or `'E'`, the Frobenius or Euclidean norm, $\|A\|_F$;

if `norm` = `'m'` or `'M'`, the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Note: for a symmetric or Hermitian matrix, ∞-norm = 1-norm.

Default: `norm` = `'1'`.

Constraints: `norm` = `'1'`, `'o'`, `'O'`, `'i'`, `'I'`, `'f'`, `'F'`, `'e'`, `'E'`, `'m'` or `'M'`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
320	The procedure was unable to allocate enough memory.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Procedure: nag_sym_bnd_mat_norm

1 Description

nag_sym_bnd_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real symmetric, complex Hermitian or complex symmetric band matrix A.

2 Usage

```
USE nag_mat_norm
```

```
[value =] nag_sym_bnd_mat_norm(uplo, a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the band matrix A

$k \geq 0$ — the number of super-diagonals or sub-diagonals in the band matrix A

3.1 Mandatory Arguments

uplo — character(len=1), intent(in)

Input: specifies whether the upper or lower triangle of A is supplied.

If uplo = 'u' or 'U', the upper triangle is supplied;

if uplo = 'l' or 'L', the lower triangle is supplied.

Constraints: uplo = 'u', 'U', 'l' or 'L'.

a($k+1, n$) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the band matrix A.

If uplo = 'u', the elements of the upper triangle of A within the band must be stored, with a_{ij} in a($k+i-j+1, j$) for $\max(j-k, 1) \leq i \leq j$;

if uplo = 'l', the elements of the lower triangle of A within the band must be stored, with a_{ij} in a($i-j+1, j$) for $j \leq i \leq \min(j+k, n)$.

Note: if A is complex Hermitian, its diagonal elements must have zero imaginary parts but the procedure does not check for this.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If `norm` = `'1'`, `'o'` or `'O'`, the 1-norm, $\|A\|_1$;

if `norm` = `'i'` or `'I'`, the ∞-norm, $\|A\|_\infty$;

if `norm` = `'f'`, `'F'`, `'e'` or `'E'`, the Frobenius or Euclidean norm, $\|A\|_F$;

if `norm` = `'m'` or `'M'`, the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Note: for a symmetric or Hermitian matrix, ∞-norm = 1-norm.

Default: `norm` = `'1'`.

Constraints: `norm` = `'1'`, `'o'`, `'O'`, `'i'`, `'I'`, `'f'`, `'F'`, `'e'`, `'E'`, `'m'` or `'M'`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Procedure: nag_trap_mat_norm

1 Description

nag_trap_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex trapezoidal matrix A.

2 Usage

```
USE nag_mat_norm
[value =] nag_trap_mat_norm(uplo, a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array **x** must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

m — the number of rows of the matrix A

n — the number of columns of the matrix A

3.1 Mandatory Arguments

uplo — character(len=1), intent(in)

Input: specifies whether A is upper or lower trapezoidal.

If uplo = 'u' or 'U', A is upper trapezoidal;

if uplo = 'l' or 'L', A is lower trapezoidal.

Constraints: uplo = 'u', 'U', 'l' or 'L'.

a(m, n) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the matrix A.

If uplo = 'u', the elements of the upper trapezoidal matrix A must be stored, and elements below the diagonal need not be set;

if uplo = 'l', the elements of the lower trapezoidal matrix A must be stored, and elements above the diagonal need not be set.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If norm = '1', 'o' or 'O', the 1-norm, $\|A\|_1$;

if norm = 'i' or 'I', the ∞-norm, $\|A\|_\infty$;

if norm = 'f', 'F', 'e' or 'E', the Frobenius or Euclidean norm, $\|A\|_F$;

if norm = 'm' or 'M', the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: norm = '1'.

Constraints: norm = '1', 'o', 'O', 'i', 'I', 'f', 'F', 'e', 'E', 'm' or 'M'.

unit_diag — logical, intent(in), optional

Input: specifies whether the supplied diagonal elements of A are to be used or are not referenced and assumed to be unity:

if `unit_diag` = `.false.`, then the diagonal elements are used as supplied;

if `unit_diag` = `.true.`, then the supplied diagonal elements are not used and they are assumed to be unity.

Default: `unit_diag` = `.false.`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

If $m > n$ and `uplo` = `'u'`, the upper trapezium of A becomes a triangular matrix. In this case, we could use `nag_tri_mat_norm` and the appropriate array section of A, i.e.,

```
nag_trap_mat_norm (uplo,a) = nag_tri_mat_norm (uplo,a(:n,:))
```

however, the use of array sections may be more expensive.

A similar situation exists if $m < n$ and `uplo` = `'l'`.

Procedure: nag_tri_mat_norm

1 Description

nag_tri_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex triangular matrix A.

The procedure allows conventional or packed storage for A.

2 Usage

```
USE nag_mat_norm
```

```
[value =] nag_tri_mat_norm(uplo, a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the matrix A

3.1 Mandatory Arguments

uplo — character(len=1), intent(in)

Input: specifies whether A is upper or lower triangular.

If uplo = 'u' or 'U', A is upper triangular;

if uplo = 'l' or 'L', A is lower triangular.

Constraints: uplo = 'u', 'U', 'l' or 'L'.

a(n, n) / **a**($n(n+1)/2$) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the matrix A.

Conventional storage (a has shape (n, n))

If uplo = 'u', the upper triangle of A must be stored, and elements below the diagonal need not be set;

if uplo = 'l', the lower triangle of A must be stored, and elements above the diagonal need not be set.

Packed storage (a has shape $(n(n+1)/2)$)

If uplo = 'u', the upper triangle of A must be stored, packed by columns, with a_{ij} in a($i + j(j-1)/2$) for $i \leq j$;

if uplo = 'l', the lower triangle of A must be stored, packed by columns, with a_{ij} in a($i + (2n-j)(j-1)/2$) for $i \geq j$.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If `norm` = `'1'`, `'o'` or `'O'`, the 1-norm, $\|A\|_1$;

if `norm` = `'i'` or `'I'`, the ∞-norm, $\|A\|_\infty$;

if `norm` = `'f'`, `'F'`, `'e'` or `'E'`, the Frobenius or Euclidean norm, $\|A\|_F$;

if `norm` = `'m'` or `'M'`, the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: `norm` = `'1'`.

Constraints: `norm` = `'1'`, `'o'`, `'O'`, `'i'`, `'I'`, `'f'`, `'F'`, `'e'`, `'E'`, `'m'` or `'M'`.

unit_diag — logical, intent(in), optional

Input: specifies whether the supplied diagonal elements of A are to be used or are not referenced and assumed to be unity:

if `unit_diag` = `.false.`, then the diagonal elements are used as supplied;

if `unit_diag` = `.true.`, then the supplied diagonal elements are not used and they are assumed to be unity.

Default: `unit_diag` = `.false.`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
320	The procedure was unable to allocate enough memory.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Procedure: nag_tri_bnd_mat_norm

1 Description

nag_tri_bnd_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex triangular band matrix A.

2 Usage

```
USE nag_mat_norm
[value =] nag_tri_bnd_mat_norm(uplo, a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the band matrix A

$k \geq 0$ — the number of super-diagonals or sub-diagonals in the band matrix A

3.1 Mandatory Arguments

uplo — character(len=1), intent(in)

Input: specifies whether A is upper or lower triangular.

If uplo = 'u' or 'U', A is upper triangular;

if uplo = 'l' or 'L', A is lower triangular.

Constraints: uplo = 'u', 'U', 'l' or 'L'.

a($k+1, n$) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the band matrix A.

If uplo = 'u', the elements of the upper triangle of A within the band must be stored, with a_{ij} in a($k+i-j+1, j$) for $\max(j-k, 1) \leq i \leq j$;

if uplo = 'l', the elements of the lower triangle of A within the band must be stored, with a_{ij} in a($i-j+1, j$) for $j \leq i \leq \min(j+k, n)$.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If `norm` = `'1'`, `'o'` or `'O'`, the 1-norm, $\|A\|_1$;

if `norm` = `'i'` or `'I'`, the ∞-norm, $\|A\|_\infty$;

if `norm` = `'f'`, `'F'`, `'e'` or `'E'`, the Frobenius or Euclidean norm, $\|A\|_F$;

if `norm` = `'m'` or `'M'`, the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: `norm` = `'1'`.

Constraints: `norm` = `'1'`, `'o'`, `'O'`, `'i'`, `'I'`, `'f'`, `'F'`, `'e'`, `'E'`, `'m'` or `'M'`.

unit_diag — logical, intent(in), optional

Input: specifies whether the supplied diagonal elements of A are to be used or are not referenced and assumed to be unity:

if `unit_diag` = `.false.`, then the diagonal elements are used as supplied;

if `unit_diag` = `.true.`, then the supplied diagonal elements are not used and they are assumed to be unity.

Default: `unit_diag` = `.false.`.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Procedure: nag_hessen_mat_norm

1 Description

nag_hessen_mat_norm is a generic procedure which computes the 1-norm, the ∞-norm, the Frobenius (Euclidean) norm or the element of largest absolute value, of a real or complex upper Hessenberg matrix A.

2 Usage

```
USE nag_mat_norm
[value =] nag_hessen_mat_norm(a  [, optional arguments])
```

The function result is a scalar, of type real(kind=*wp*), containing the required norm.

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '**x**(n)' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

n — the order of the matrix A

3.1 Mandatory Argument

a(n, n) — real(kind=*wp*) / complex(kind=*wp*), intent(in)

Input: the matrix A.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

norm — character(len=1), intent(in), optional

Input: specifies which norm is to be computed and returned via the function name.

If norm = '1', 'o' or 'O', the 1-norm, $\|A\|_1$;

if norm = 'i' or 'I', the ∞-norm, $\|A\|_\infty$;

if norm = 'f', 'F', 'e' or 'E', the Frobenius or Euclidean norm, $\|A\|_F$;

if norm = 'm' or 'M', the element of maximum absolute value, $\max_{ij} |a_{ij}|$ (not strictly a norm).

Default: norm = '1'.

Constraints: norm = '1', 'o', 'O', 'i', 'I', 'f', 'F', 'e', 'E', 'm' or 'M'.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document nag_error_handling (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to nag_set_error before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.

5 Examples of Usage

This procedure is simple to use and no complete example is listed. However, an example is included in the 'Additional Examples' section.

Example 1: The 1-norm of a General Complex Matrix

Finds the 1-norm of a general complex matrix A.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
    PROGRAM nag_mat_norm_ex01

      ! Example Program Text for nag_mat_norm
      ! NAG fl90, Release 4. NAG Copyright 2000.

      ! .. Use Statements ..
      USE nag_examples_io, ONLY : nag_std_in, nag_std_out
      USE nag_mat_norm, ONLY : nag_gen_mat_norm
      ! .. Implicit None Statement ..
      IMPLICIT NONE
      ! .. Intrinsic Functions ..
      INTRINSIC KIND
      ! .. Parameters ..
      INTEGER, PARAMETER :: wp = KIND(1.0D0)
      ! .. Local Scalars ..
      INTEGER :: i, m, n
      ! .. Local Arrays ..
      COMPLEX (wp), ALLOCATABLE :: a(:,:)
      ! .. Executable Statements ..
      WRITE (nag_std_out,*) 'Example Program Results for nag_mat_norm_ex01'

      READ (nag_std_in,*)          ! Skip heading in data file
      READ (nag_std_in,*) m, n

      ALLOCATE (a(m,n))            ! Allocate storage

      READ (nag_std_in,*) (a(i,:),i=1,m)

      WRITE (nag_std_out,*)
      WRITE (nag_std_out,'(1X,a17,2x,f7.3)') 'One-norm of (A) =', &
       nag_gen_mat_norm(a)

      DEALLOCATE (a)               ! Deallocate storage

    END PROGRAM nag_mat_norm_ex01
```

2 Program Data

```
Example Program Data for nag_mat_norm_ex01
  5 4                                                    : Values of m, n
 (-1.34, 2.55) ( 0.28, 3.17) ( 0.00,-2.20) ( 0.72,-0.92)
 (-0.17, 0.00) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 0.26, 1.78) ( 1.32,-1.70) (-0.75, 3.31) (-2.15, 1.19)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67) : End of Matrix A
```

3 Program Results

```
 Example Program Results for nag_mat_norm_ex01

 One-norm of (A) =   15.036
```

Additional Examples

Not all example programs supplied with NAG *fl*90 appear in full in this module document. The following additional examples, associated with this module, are available.

`nag_mat_norm_ex02`
: Finds the norms of a square banded matrix.

`nag_mat_norm_ex03`
: Finds the norms of a symmetric matrix and a triangular matrix.

`nag_mat_norm_ex04`
: Finds the norms of a symmetric band matrix and a triangular band matrix.

`nag_mat_norm_ex05`
: Finds the norms of an upper Hessenberg matrix.

`nag_mat_norm_ex06`
: Finds the norms of a symmetric matrix A, stored in packed storage.

`nag_mat_norm_ex07`
: Finds the norms of an upper trapezoidal complex matrix.