NAG Library Routine Document

F06SAF (ZGEMV)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F06SAF (ZGEMV) computes the matrix-vector product for a complex general matrix, its transpose or its conjugate transpose.

2 Specification

```
SUBROUTINE F06SAF (TRANS, M, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY)
INTEGER                 M, N, LDA, INCX, INCY
COMPLEX (KIND=nag_wp) ALPHA, A(LDA,*), X(*), BETA, Y(*)
CHARACTER(1)            TRANS
```

The routine may be called by its BLAS name ***zgemv***.

3 Description

F06SAF (ZGEMV) performs one of the matrix-vector operations

$$y \leftarrow \alpha Ax + \beta y, \quad y \leftarrow \alpha A^{\mathrm{T}}x + \beta y \quad \text{or} \quad y \leftarrow \alpha A^{\mathrm{H}}x + \beta y,$$

where A is an m by n complex matrix, x and y are complex vectors, and α and β are complex scalars.

If $m = 0$ or $n = 0$, no operation is performed.

4 References

None.

5 Arguments

1: TRANS – CHARACTER(1) *Input*

On entry: specifies the operation to be performed.

TRANS = 'N'
$y \leftarrow \alpha Ax + \beta y$.

TRANS = 'T'
$y \leftarrow \alpha A^{\mathrm{T}}x + \beta y$.

TRANS = 'C'
$y \leftarrow \alpha A^{\mathrm{H}}x + \beta y$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: M – INTEGER *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M ≥ 0.

3: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N ≥ 0.

4: ALPHA – COMPLEX (KIND=nag_wp) *Input*

On entry: the scalar α.

5: A(LDA, $*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the second dimension of the array A must be at least N.

On entry: the m by n matrix A.

6: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F06SAF (ZGEMV) is called.

Constraint: LDA $\geq \max(1, \mathrm{M})$.

7: X($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array X must be at least $\max(1, 1 + (\mathrm{N} - 1) \times |\mathrm{INCX}|)$ if TRANS = 'N' and at least $\max(1, 1 + (\mathrm{M} - 1) \times |\mathrm{INCX}|)$ if TRANS = 'T' or 'C'.

On entry: the vector x.

If TRANS = 'N',

if INCX > 0, x_i must be stored in X$(1 + (i - 1) \times \mathrm{INCX})$, for $i = 1, 2, \ldots, \mathrm{N}$;

if INCX < 0, x_i must be stored in X$(1 - (\mathrm{N} - i) \times \mathrm{INCX})$, for $i = 1, 2, \ldots, \mathrm{N}$.

If TRANS = 'T' or 'C',

if INCX > 0, x_i must be stored in X$(1 + (i - 1) \times \mathrm{INCX})$, for $i = 1, 2, \ldots, \mathrm{M}$;

if INCX < 0, x_i must be stored in X$(1 - (\mathrm{M} - i) \times \mathrm{INCX})$, for $i = 1, 2, \ldots, \mathrm{M}$.

8: INCX – INTEGER *Input*

On entry: the increment in the subscripts of X between successive elements of x.

Constraint: INCX $\neq 0$.

9: BETA – COMPLEX (KIND=nag_wp) *Input*

On entry: the scalar β.

10: Y($*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array Y must be at least $\max(1, 1 + (\mathrm{M} - 1) \times |\mathrm{INCY}|)$ if TRANS = 'N' and at least $\max(1, 1 + (\mathrm{N} - 1) \times |\mathrm{INCY}|)$ if TRANS = 'T' or 'C'.

On entry: the vector y, if BETA = 0.0, Y need not be set.

If TRANS = 'N',

if INCY > 0, y_i must be stored in Y$(1 + (i - 1) \times \mathrm{INCY})$, for $i = 1, 2, \ldots, \mathrm{M}$;

if INCY < 0, y_i must be stored in Y$(1 - (\mathrm{M} - i) \times \mathrm{INCY})$, for $i = 1, 2, \ldots, \mathrm{M}$.

If TRANS = 'T' or 'C',

if INCY > 0, y_i must be stored in Y$(1 + (i - 1) \times \mathrm{INCY})$, for $i = 1, 2, \ldots, \mathrm{N}$;

if INCY < 0, y_i must be stored in Y$(1 - (\mathrm{N} - i) \times \mathrm{INCY})$, for $i = 1, 2, \ldots, \mathrm{N}$.

On exit: the updated vector y stored in the array elements used to supply the original vector y.

11: INCY – INTEGER *Input*

On entry: the increment in the subscripts of Y between successive elements of y.

Constraint: INCY $\neq 0$.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

F06SAF (ZGEMV) is not threaded in any implementation.

9 Further Comments

None.

10 Example

None.