NAG Library Routine Document

F06SCF (ZHEMV)

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

F06SCF (ZHEMV) computes the matrix-vector product for a complex Hermitian matrix.

2 Specification

```
SUBROUTINE F06SCF (UPLO, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY)

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

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

3 Description

F06SCF (ZHEMV) performs the matrix-vector operation

$$y \leftarrow \alpha Ax + \beta y,$$

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

4 References

None.

5 Parameters

1: UPLO – CHARACTER(1) *Input*

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

UPLO = 'U'
The upper triangular part of A is stored.

UPLO = 'L'
The lower triangular part of A is stored.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER *Input*

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

Constraint: N $\geq$ 0.

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

On entry: the scalar α.

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

Note: the second dimension of the array A must be at least max(1, N).

On entry: the n by n Hermitian matrix A.

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

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

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F06SCF (ZHEMV) is called.

Constraint: LDA $\geq$ max(1, N).

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

Note: the dimension of the array X must be at least max(1, 1 + (N − 1) × |INCX|).

On entry: the n-element vector x.

If INCX > 0, x_i must be stored in X(1 + (i − 1) × INCX), for $i = 1, 2, \ldots,$ N.

If INCX < 0, x_i must be stored in X(1 − (N − i) × INCX), for $i = 1, 2, \ldots,$ N.

Intermediate elements of X are not referenced.

7: INCX – INTEGER *Input*

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

Constraint: INCX $\neq 0$.

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

On entry: the scalar β.

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

Note: the dimension of the array Y must be at least max(1, 1 + (N − 1) × |INCY|).

On entry: the n-element vector y, if BETA $= 0$, Y need not be set.

If INCY > 0, y_i must be stored in Y(1 + (i − 1) × INCY), for $i = 1, 2, \ldots,$ N.

If INCY < 0, y_i must be stored in Y(1 − (N − i) × INCY), for $i = 1, 2, \ldots,$ N.

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

10: 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 Further Comments

None.

9 Example

None.