

NAG Library Routine Document

F06TAF

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

F06TAF performs the matrix-vector operation

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

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

2 Specification

```
SUBROUTINE F06TAF (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
```

3 Description

None.

4 References

None.

5 Parameters

- | | | |
|----|---|--------------|
| 1: | UPLO – CHARACTER(1)
<i>On entry:</i> 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.
<i>Constraint:</i> UPLO = 'U' or 'L'. | <i>Input</i> |
| 2: | N – INTEGER
<i>On entry:</i> n , the order of the matrix A .
<i>Constraint:</i> $N \geq 0$. | <i>Input</i> |
| 3: | ALPHA – COMPLEX (KIND=nag_wp)
<i>On entry:</i> the scalar α . | <i>Input</i> |
| 4: | A(LDA,*) – COMPLEX (KIND=nag_wp) array
Note: the second dimension of the array A must be at least $\max(1, N)$.
<i>On entry:</i> the n by n symmetric matrix A . | <i>Input</i> |

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 F06TAF 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 $\max(1, 1 + (N - 1) \times |INCX|)$.
On entry: the n -element vector x .
 If $INCX > 0$, x_i must be stored in $X(1 + (i - 1) \times INCX)$, for $i = 1, 2, \dots, N$.
 If $INCX < 0$, x_i must be stored in $X(1 - (N - i) \times INCX)$, for $i = 1, 2, \dots, N$.
 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 $\max(1, 1 + (N - 1) \times |INCY|)$.
On entry: the n -element vector y .
 If $INCY > 0$, y_i must be stored in $Y(1 + (i - 1) \times INCY)$, for $i = 1, 2, \dots, N$.
 If $INCY < 0$, y_i must be stored in $Y(1 - (N - i) \times INCY)$, for $i = 1, 2, \dots, N$.
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.
