X03ABF – NAG Fortran Library Routine Document

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

X03ABF calculates the value of a ***complex*** scalar product using ***basic precision*** or ***additional precision*** and adds it to a ***complex*** initial value.

2 Specification

```
 SUBROUTINE X03ABF(A, ISIZEA, B, ISIZEB, N, ISTEPA, ISTEPB, CX, DX,
1                  SW, IFAIL)
 INTEGER           ISIZEA, ISIZEB, N, ISTEPA, ISTEPB, IFAIL
 complex           A(ISIZEA), B(ISIZEB), CX, DX
 LOGICAL           SW
```

3 Description

The routine calculates the scalar product of two ***complex*** vectors and adds it to an initial value c to give a correctly rounded result d:

$$d = c + \sum_{i=1}^{n} a_i b_i.$$

If $n < 1$, $d = c$.

The vector elements a_i and b_i are stored in selected elements of the one-dimensional array parameters A and B, which in the (sub)program from which X03ABF is called may be identified with parts of possibly multi-dimensional arrays according to the standard Fortran rules. For example, the vectors may be parts of a row or column of a matrix. See Section 5 for details, and Section 9 for an example.

The products are accumulated in ***basic precision*** and ***additional precision*** depending on the parameter SW.

This routine has been designed primarily for use as an auxiliary routine by other routines in the NAG Fortran Library, especially those in the chapters on Linear Algebra.

4 References

None.

5 Parameters

1: A(ISIZEA) — ***complex*** array *Input*

On entry: the elements of the first vector.

The ith vector element is stored in the array element A($(i-1) \times$ ISTEPA $+ 1$). In the user's (sub)program from which X03ABF is called, A can be part of a multi-dimensional array and the actual argument must be the array element containing the first vector element.

2: ISIZEA — INTEGER *Input*

On entry: the dimension of array A inside the routine.

The upper bound for ISIZEA is found by multiplying together the dimensions of A as declared in the (sub)program from which X03ABF is called, subtracting the starting position and adding 1.

Constraint: ISIZEA $\geq$ (N $-$ 1) $\times$ ISTEPA $+$ 1.

3: B(ISIZEB) — ***complex*** array *Input*

On entry: the elements of the second vector.

The ith vector element is stored in the array element B($(i-1) \times$ ISTEPB $+ 1$). In the (sub)program from which X03ABF is called, B can be part of a multi-dimensional array and the actual argument must be the array element containing the first vector element.

4: ISIZEB — INTEGER *Input*

On entry: the dimension of array B inside the routine.

The upper bound for ISIZEB is found by multiplying together the dimensions of B as declared in the user's (sub)program from which X03ABF is called, subtracting the starting position and adding 1.

Constraint: ISIZEB $\geq$ (N $-$ 1) $\times$ ISTEPB $+$ 1.

5: N — INTEGER *Input*

On entry: the number of elements in the scalar product, n.

6: ISTEPA — INTEGER *Input*

On entry: the step length between elements of the first vector in array A.

Constraint: ISTEPA > 0.

7: ISTEPB — INTEGER *Input*

On entry: the step length between elements of the second vector in array B.

Constraint: ISTEPB > 0.

8: CX — ***complex*** *Input*

On entry: the initial value c.

9: DX — ***complex*** *Output*

On exit: the result d.

10: SW — LOGICAL *Input*

On entry: the precision to be used.

If SW = .TRUE., ***additional precision***.
If SW = .FALSE., ***basic precision***.

11: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, ISTEPA ≤ 0,
or ISTEPB ≤ 0.

IFAIL = 2

On entry, ISIZEA $<$ (N $-$ 1) $\times$ ISTEPA $+$ 1,
or ISIZEB $<$ (N $-$ 1) $\times$ ISTEPB $+$ 1.

7 Accuracy

If the calculation is in ***basic precision*** or ***additional precision***, the result is correct to full implementation accuracy provided that exceptionally severe cancellation does not occur in the summation. If the calculation is in ***basic precision***, such accuracy cannot be guaranteed.

8 Further Comments

The time taken by the routine is approximately proportional to n and also depends on whether ***basic precision*** or ***additional precision*** is used.

9 Example

To calculate the scalar product of the second column of the matrix A and the vector B, and add it to an initial value of $1+i$, where

$$A = \begin{pmatrix} -1 & -i & 1 \\ 2+3i & i & 2i \\ 0 & -1-i & 1-2i \end{pmatrix}, \quad B = \begin{pmatrix} i \\ 1-i \\ -i \end{pmatrix}.$$

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     X03ABF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      PARAMETER        (N=3)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      complex          CX, DX
      INTEGER          I, IFAIL, ISIZEA, ISIZEB, ISTEPA, ISTEPB, J
      LOGICAL          SW
*     .. Local Arrays ..
      complex          A(N,N), B(N)
*     .. External Subroutines ..
      EXTERNAL         X03ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'X03ABF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) ((A(I,J),J=1,N),I=1,N), (B(I),I=1,N)
      CX = (1.0e0,1.0e0)
      ISIZEA = N
      ISIZEB = N
      ISTEPA = 1
      ISTEPB = 1
      SW = .TRUE.
      IFAIL = 0
*
      CALL X03ABF(A(1,2),ISIZEA,B,ISIZEB,N,ISTEPA,ISTEPB,CX,DX,SW,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'Result = ', DX
      STOP
```

```
*
99999 FORMAT (1X,A,'(',F3.0,',',F3.0,')')
      END
```

9.2 Program Data

```
X03ABF Example Program Data
 (-1.0, 0.0) ( 0.0, -1.0) (1.0,  0.0)
 ( 2.0, 3.0) ( 0.0,  1.0) (0.0,  2.0)
 ( 0.0, 0.0) (-1.0, -1.0) (1.0, -2.0)
 ( 0.0, 1.0) ( 1.0, -1.0) (0.0, -1.0)
```

9.3 Program Results

```
 X03ABF Example Program Results

 Result = ( 2., 3.)
```