NAG Library Routine Document

X03ABF

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, SW,    &
                   IFAIL)

INTEGER                 ISIZEA, ISIZEB, N, ISTEPA, ISTEPB, IFAIL
COMPLEX (KIND=nag_wp) A(ISIZEA), B(ISIZEB), CX, DX
LOGICAL                 SW
```

3 Description

X03ABF 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 subroutine from which X03ABF is called may be identified with parts of possibly multidimensional 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 10 for an example.

The products are accumulated in ***basic precision*** or ***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 Library, especially those in the chapters on Linear Algebra.

4 References

None.

5 Parameters

1: A(ISIZEA) – COMPLEX (KIND=nag_wp) 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 your subroutine from which X03ABF is called, A can be part of a multidimensional array and the actual argument must be the array element containing the first vector element.

2: ISIZEA – INTEGER *Input*

On entry: the dimension of the array A as declared in the (sub)program from which X03ABF is called.

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

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

3: B(ISIZEB) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the elements of the second vector.

The ith vector element is stored in the array element $\mathrm{B}((i-1) \times \mathrm{ISTEPB}+1)$. In your subroutine from which X03ABF is called, B can be part of a multidimensional array and the actual argument must be the array element containing the first vector element.

4: ISIZEB – INTEGER *Input*

On entry: the dimension of the array B as declared in the (sub)program from which X03ABF is called.

The upper bound for ISIZEB is found by multiplying together the dimensions of B as declared in your subroutine from which X03ABF is called, subtracting the starting position and adding 1.

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

5: N – INTEGER *Input*

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

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 (KIND=nag_wp) *Input*

On entry: the initial value c.

9: DX – COMPLEX (KIND=nag_wp) *Output*

On exit: the result d.

10: SW – LOGICAL *Input*

On entry: the precision to be used in the calculation.

SW = .TRUE.
additional precision.

SW = .FALSE.
basic precision.

11: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, ISTEPA ≤ 0,
or ISTEPB ≤ 0.

IFAIL $= 2$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

If the calculation is in ***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 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by X03ABF is approximately proportional to n and also depends on whether ***basic precision*** or ***additional precision*** is used.

10 Example

This example calculates 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

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

10.1 Program Text

```
    Program x03abfe

!     X03ABF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x03abf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: n = 3, nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: cx, dx
      Integer                          :: i, ifail, isizea, isizeb, istepa,    &
                                          istepb, j
      Logical                          :: sw
!     .. Local Arrays ..
      Complex (Kind=nag_wp)            :: a(n,n), b(n)
!     .. 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_nag_wp,1.0E0_nag_wp)
      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

99999 Format (1X,A,'(',F3.0,',',F3.0,')')
    End Program x03abfe
```

10.2 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)
```

10.3 Program Results

```
 X03ABF Example Program Results

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