X03AAF – 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

X03AAF calculates the value of a scalar product using ***basic precision*** or ***additional precision*** and adds it to a ***basic precision*** or ***additional precision*** initial value.

2 Specification

```
 SUBROUTINE X03AAF(A, ISIZEA, B, ISIZEB, N, ISTEPA, ISTEPB, C1, C2,
1                  D1, D2, SW, IFAIL)
 INTEGER           ISIZEA, ISIZEB, N, ISTEPA, ISTEPB, IFAIL
 real              A(ISIZEA), B(ISIZEB), C1, C2, D1, D2
 LOGICAL           SW
```

3 Description

The routine calculates the scalar product of two ***real*** 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 X03AAF 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.

Both the initial value c and the result d are defined by a pair of ***real*** variables, so that they may take either ***basic precision*** or ***additional precision*** values.

(a) If SW = .TRUE., the products are accumulated in ***additional precision***, and on exit the result is available either in ***basic precision***, correctly rounded, or in ***additional precision***.
(b) If SW = .FALSE., the products are accumulated in ***basic precision***, and the result is returned in ***basic precision***.

This routine is 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) — ***real*** 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 X03AAF 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 user's (sub)program from which X03AAF is called, subtracting the starting position and adding 1.

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

3: B(ISIZEB) — ***real*** 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 the user's (sub)program from which X03AAF 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 (sub)program from which X03AAF 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: 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: C1 — ***real*** *Input*

9: C2 — ***real*** *Input*

On entry: C1 and C2 must specify the initial value c: $c = \mathrm{C1}+\mathrm{C2}$. Normally, if c is in ***additional precision***, C1 specifies the most significant part and C2 the least significant part; if c is in ***basic precision***, then C1 specifies c and C2 must have the value 0.0. Both C1 and C2 must be defined on entry.

10: D1 — ***real*** *Output*

11: D2 — ***real*** *Output*

On exit: the result d.

If the calculation is in ***additional precision*** (SW = .TRUE.),

$\mathrm{D1} = d$ rounded to ***basic precision***,
$\mathrm{D2} = d - \mathrm{D1}$;

thus D1 holds the correctly rounded ***basic precision*** result and the sum $\mathrm{D1}+\mathrm{D2}$ gives the result in ***additional precision***. D2 may have the opposite sign to D1.

If the calculation is in ***basic precision*** (SW = .FALSE.),

$\mathrm{D1} = d$,
$\mathrm{D2} = 0.0$.

12: SW — LOGICAL *Input*

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

SW = .TRUE., – ***additional precision***;
SW = .FALSE., – ***basic precision***.

13: 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 $< (\mathrm{N}-1) \times \mathrm{ISTEPA} + 1$,
or ISIZEB $< (\mathrm{N}-1) \times \mathrm{ISTEPB} + 1$.

7 Accuracy

If the calculation is an ***additional precision***, the rounded ***basic precision*** result D1 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.

On exit the variables D1 and D2 may be used directly to supply a ***basic precision*** or ***additional precision*** initial value for a subsequent call of X03AAF.

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 1.0 where

$$A = \begin{pmatrix} -2 & -3 & 7 \\ 2 & -5 & 3 \\ -9 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 8 \\ -4 \\ -2 \end{pmatrix}.$$

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.

```
*     X03AAF 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 ..
      real             C1, C2, D1, D2
      INTEGER          I, IFAIL, ISIZEA, ISIZEB, ISTEPA, ISTEPB, J
      LOGICAL          SW
*     .. Local Arrays ..
      real             A(N,N), B(N)
*     .. External Subroutines ..
      EXTERNAL         X03AAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'X03AAF 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)
      C1 = 1.0e0
      C2 = 0.0e0
      ISIZEA = N
      ISIZEB = N
      ISTEPA = 1
      ISTEPB = 1
      SW = .TRUE.
      IFAIL = 0
*
      CALL X03AAF(A(1,2),ISIZEA,B,ISIZEB,N,ISTEPA,ISTEPB,C1,C2,D1,D2,SW,
     +            IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'D1 = ', D1, ' D2 = ', D2
      STOP
*
99999 FORMAT (1X,A,F4.1,A,F4.1)
      END
```

9.2 Program Data

```
X03AAF Example Program Data
   -2   -3    7
    2   -5    3
   -9    1    0
    8   -4   -2
```

9.3 Program Results

```
 X03AAF Example Program Results

 D1 = -5.0 D2 =  0.0
```