NAG Library Routine Document

X03AAF

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

3 Description

X03AAF calculates the scalar product of two ***double precision*** 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 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 ***double precision*** 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 Library, especially those in the chapters on Linear Algebra.

4 References

None.

5 Parameters

1: A(ISIZEA) – ***double precision*** array *Input*

On entry: the elements of the first vector.

The ith vector element is stored in the array element $\mathrm{A}((i-1) \times \mathrm{ISTEPA} + 1)$. In your subroutine 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 the array A as declared in the (sub)program from which X03AAF is called.

The upper bound for ISIZEA is found by multiplying together the dimensions of A as declared in your subroutine from which X03AAF is called, subtracting the starting position and adding 1.

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

3: B(ISIZEB) – ***double precision*** array *Input*

On entry: the elements of the second vector.

The ith vector element is stored in the array element $\text{B}((i - 1) \times \text{ISTEPB} + 1)$. In your subroutine 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 the array B as declared in the (sub)program from which X03AAF is called.

The upper bound for ISIZEB is found by multiplying together the dimensions of B as declared in your subroutine from which X03AAF is called, subtracting the starting position and adding 1.

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

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: $\text{ISTEPA} > 0$.

7: ISTEPB – INTEGER *Input*

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

Constraint: $\text{ISTEPB} > 0$.

8: C1 – ***double precision*** *Input*
9: C2 – ***double precision*** *Input*

On entry: C1 and C2 must specify the initial value c: $c = \text{C1} + \text{C2}$. 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 – ***double precision*** *Output*
11: D2 – ***double precision*** *Output*

On exit: the result d.

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

$\text{D1} = d$ rounded to ***basic precision***;

$\text{D2} = d - \text{D1}$,

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

If the calculation is in ***basic precision*** (SW = .FALSE.), D1 = d;

D2 = 0.0.

12: SW – LOGICAL *Input*

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

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

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

13: 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.

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

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.**

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 $>$ (N $-$ 1) $\times$ ISTEPA + 1,
or ISIZEB $>$ (N $-$ 1) $\times$ 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 X03AAF 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

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 1.0 where

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

9.1 Program Text

```
*     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 ..
      DOUBLE PRECISION C1, C2, D1, D2
      INTEGER          I, IFAIL, ISIZEA, ISIZEB, ISTEPA, ISTEPB, J
      LOGICAL          SW
*     .. Local Arrays ..
      DOUBLE PRECISION 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 = 1.0D0
      C2 = 0.0D0
      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
*
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
```