NAG Toolbox

nag_dot_real_prec (x03aa)

1 Purpose

nag_dot_real_prec (x03aa) 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 Syntax

```
[d1, d2, ifail] = nag_dot_real_prec(a, b, n, istepa, istepb, c1, c2, sw, 'isizea', isizea, 'isizeb', isizeb)
[d1, d2, ifail] = x03aa(a, b, n, istepa, istepb, c1, c2, sw, 'isizea', isizea, 'isizeb', isizeb)
```

3 Description

nag_dot_real_prec (x03aa) calculates the scalar product of two double 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 arguments **a** and **b**, which in the function from which nag_dot_real_prec (x03aa) 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.

Both the initial value c and the result d are defined by a pair of double 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 function is designed primarily for use as an auxiliary function by other functions in the NAG Toolbox, especially those in the chapters on Linear Algebra.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **a**(**isizea**) – REAL (KIND=nag_wp) array

The elements of the first vector.

The ith vector element is stored in the array element **a**$((i-1) \times$ **istepa** $+ 1)$. In your function from which nag_dot_real_prec (x03aa) 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: **b**(**isizeb**) – REAL (KIND=nag_wp) array

The elements of the second vector.

The ith vector element is stored in the array element **b**$((i-1) \times$ **istepb** $+ 1)$. In your function from which nag_dot_real_prec (x03aa) is called, **b** can be part of a multidimensional array and the actual argument must be the array element containing the first vector element.

3: **n** – INTEGER

n, the number of elements in the scalar product.

4: **istepa** – INTEGER

The step length between elements of the first vector in array **a**.

Constraint: **istepa** > 0.

5: **istepb** – INTEGER

The step length between elements of the second vector in array **b**.

Constraint: **istepb** > 0.

6: **c1** – REAL (KIND=nag_wp)

7: **c2** – REAL (KIND=nag_wp)

c1 and **c2** must specify the initial value c: $c =$ **c1** $+$ **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.

8: **sw** – LOGICAL

The precision to be used in the calculation.

sw $= true$
additional precision.

sw $= false$
basic precision.

5.2 Optional Input Parameters

1: **isizea** – INTEGER

Default: the dimension of the array **a**.

The dimension of the array **a**.

The upper bound for **isizea** is found by multiplying together the dimensions of **a** as declared in your function from which nag_dot_real_prec (x03aa) is called, subtracting the starting position and adding 1.

Constraint: **isizea** $\geq$ (**n** $- 1) \times$ **istepa** $+ 1$.

2: **isizeb** – INTEGER

Default: the dimension of the array **b**.

The dimension of the array **b**.

The upper bound for **isizeb** is found by multiplying together the dimensions of **b** as declared in your function from which nag_dot_real_prec (x03aa) is called, subtracting the starting position and adding 1.

Constraint: **isizeb** $\geq$ (**n** $- 1) \times$ **istepb** $+ 1$.

5.3 Output Parameters

1: **d1** – REAL (KIND=nag_wp)
2: **d2** – REAL (KIND=nag_wp)

The result d.

If the calculation is in ***additional precision*** (**sw** $= true$),

d1 $= d$ rounded to ***basic precision***;

d2 $= d -$ **d1**,

thus **d1** holds the correctly rounded ***basic precision*** result and the sum **d1** $+$ **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 $= 0.0$.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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** $+ 1$.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

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 nag_dot_real_prec (x03aa) 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 nag_dot_real_prec (x03aa).

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

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

9.1 Program Text

```
     function x03aa_example

fprintf('x03aa example results\n\n');

% Evaluate d = c + a(:,2).b, where
a = [ -2   -3        7;
       2   -5        3;
      -9    1e-19    0];
b = [ 8;
     -4;
     -2];
n = nag_int(size(a,2));

% c = 1 (in basic precision) + 1e-19 (extra precision)
c1 = 1;
c2 = 1e-20;

% elements are stored contiguously in a and b
istepa = nag_int(1);
istepb = istepa;

% Calculate dodt-product in extended precision
sw = true;

[d1, d2, ifail] = x03aa( ...
                         a(:,2), b, n, istepa, istepb, c1, c2, sw);

fprintf('Accumulated dot-product = %20.15e + %10.2e\n',d1,d2);
```

9.2 Program Results

```
     x03aa example results

Accumulated dot-product = -3.000000000000000e+00 +  -1.90e-19
```