NAG Library Routine Document

F16EAF (BLAS_DDOT)

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

F16EAF (BLAS_DDOT) updates a scalar by a scaled dot product of two real vectors.

2 Specification

```
SUBROUTINE F16EAF (CONJ, N, ALPHA, X, INCX, BETA, Y, INCY, R)
INTEGER            CONJ, N, INCX, INCY
REAL (KIND=nag_wp) ALPHA, X(1+(N-1)*ABS(INCX)), BETA,                    &
                   Y(1+(N-1)*ABS(INCY)), R
```

The routine may be called by its BLAST name ***blas_ddot***.

3 Description

F16EAF (BLAS_DDOT) performs the operation

$$r \leftarrow \beta r + \alpha x^{\mathrm{T}} y$$

where x and y are n-element real vectors, and r, α and β real scalars. If n is less than zero, or, if β is equal to one and either α or n is equal to zero, this routine returns immediately.

4 References

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee http://www.netlib.org/blas/blast-forum/blas-report.pdf

5 Arguments

1: CONJ – INTEGER *Input*

On entry: CONJ is not referenced and need not be set. The presence of this argument in the BLAST standard is for consistency with the interface of the complex variant of this routine.

2: N – INTEGER *Input*

On entry: n, the number of elements in x and y.

3: ALPHA – REAL (KIND=nag_wp) *Input*

On entry: the scalar α.

4: X($1 + (\mathrm{N} - 1) \times |\mathrm{INCX}|$) – REAL (KIND=nag_wp) array *Input*

On entry: the n-element vector x.

If INCX > 0, x_i must be stored in X($(i-1) \times \mathrm{INCX} + 1$), for $i = 1, 2, \ldots, \mathrm{N}$.

If INCX < 0, x_i must be stored in X($(\mathrm{N} - i) \times |\mathrm{INCX}| + 1$), for $i = 1, 2, \ldots, \mathrm{N}$.

Intermediate elements of X are not referenced. If $\alpha = 0.0$ or $\mathrm{N} = 0$, X is not referenced.

5: INCX – INTEGER *Input*

On entry: the increment in the subscripts of X between successive elements of x.

Constraint: INCX $\neq 0$.

6: BETA – REAL (KIND=nag_wp) *Input*

On entry: the scalar β.

7: Y$(1 + (\mathrm{N} - 1) \times |\mathrm{INCY}|)$ – REAL (KIND=nag_wp) array *Input*

On entry: the n-element vector y.

If INCY > 0, y_i must be stored in Y$((i-1) \times \mathrm{INCY} + 1)$, for $i = 1, 2, \ldots, \mathrm{N}$.

If INCY < 0, y_i must be stored in Y$((\mathrm{N} - i) \times |\mathrm{INCY}| + 1)$, for $i = 1, 2, \ldots, \mathrm{N}$.

Intermediate elements of Y are not referenced. If $\alpha = 0.0$ or N $= 0$, Y is not referenced.

8: INCY – INTEGER *Input*

On entry: the increment in the subscripts of Y between successive elements of y.

Constraint: INCY $\neq 0$.

9: R – REAL (KIND=nag_wp) *Input/Output*

On entry: the initial value, r, to be updated. If $\beta = 0.0$, R need not be set on entry.

On exit: the value r, scaled by β and updated by the scaled dot product of x and y.

6 Error Indicators and Warnings

If INCX $= 0$ or INCY $= 0$, an error message is printed and program execution is terminated.

7 Accuracy

The dot product $x^{\mathrm{T}}y$ is computed using the BLAS routine DDOT.

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

8 Parallelism and Performance

F16EAF (BLAS_DDOT) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example computes the scaled sum of two dot products, $r = \alpha_1 x^{\mathrm{T}}y + \alpha_2 u^{\mathrm{T}}v$, where

$$\begin{array}{ll} \alpha_1 = 0.3, & x = (1, 2, 3, 4, 5), \quad y = (-5, -4, 3, 2, 1), \\ \alpha_2 = -7.0, & u = v = (0.4, 0.3, 0.2, 0.1). \end{array}$$

y and v are stored in reverse order, and u is stored in reverse order in every other element of a real array.

10.1 Program Text

```
    Program f16eafe

!     F16EAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: blas_ddot, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: alpha, beta, r
      Integer                          :: conj, i, incx, incy, ix, iy, j, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: x(:), y(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs
!     .. Executable Statements ..
      Write (nout,*) 'F16EAF Example Program Results'
      Write (nout,*)

!     Skip heading in data file.
      Read (nin,*)

!     Accumulate two dot products, set beta=zero initially.
      beta = 0.0_nag_wp

      Do j = 1, 2
!       Read data for dot product.
        Read (nin,*) n
        Read (nin,*) incx, incy
        Allocate (x(1+(n-1)*abs(incx)),y(1+(n-1)*abs(incy)))

        Read (nin,*) alpha

!       Read the vectors x and y and store forwards or backwards
!       as determined by incx (resp. incy).
        If (incx>0) Then
          ix = 1
        Else
          ix = 1 - (n-1)*incx
        End If

        Do i = 1, n
          Read (nin,*) x(ix)
          ix = ix + incx
        End Do

        If (incy>0) Then
          iy = 1
        Else
          iy = 1 - (n-1)*incy
        End If

        Do i = 1, n
          Read (nin,*) y(iy)
          iy = iy + incy
        End Do

!       Compute r = beta*r + alpha*(x^T*y).
!       The NAG name equivalent of blas_ddot is f16eaf.
        Call blas_ddot(conj,n,alpha,x,incx,beta,y,incy,r)

!       Reset beta for accumulation and deallocate x, y.
        beta = 1.0_nag_wp
        Deallocate (x,y)
      End Do
```

```
      Write (nout,99999) r

99999 Format (1X,'Accumulated dot product, r = ',F9.4)
    End Program f16eafe
```

10.2 Program Data

```
F16EAF Example Program Data

   5                                    : first dot product, n
   1    -1                              : incx and incy
   0.3                                  : alpha
   1.0
   2.0
   3.0
   4.0
   5.0                                  : Vector x
  -5.0
  -4.0
   3.0
   2.0
   1.0                                  : Vector y

   4                                    : second dot product, n
  -2    -1                              : incx and incy
  -7.0                                  : alpha
   0.4
   0.3
   0.2
   0.1                                  : Vector x
   0.4
   0.3
   0.2
   0.1                                  : Vector y
```

10.3 Program Results

```
 F16EAF Example Program Results

 Accumulated dot product, r =    0.6000
```