vDSP Vector-to-Vector Arithmetic Operations Reference

Performance > Carbon

2009-01-06

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

vDSP Vector-to-Vector Arithmetic Operations Reference

Framework: Accelerate/vecLib
Declared in vDSP.h

Overview

This document describes the C API for the vDSP functions that receive a vector as input and return a vector as output.

Functions by Task

Testing Bitwise Logical Equivalence

Doing Basic Arithmetic on Real Vectors

Doing Basic Arithmetic on Complex Vectors

Finding Maximum and Minimum Elements

Computing Vector Distance

Interpolating Between Two Vectors

Evaluating Vectors as Polynomials

Applying Pythagoras's Theorem to Vector Elements

Finding a Vector's Extrema

Swapping Elements Between Vectors

Merging Two Vectors

Computing Vector Spectra

Computing the Coherence Function of Two Vectors

Computing the Transfer Function

Doing Recursive Filtering on a Real Vector

Functions

vDSP_deq22

Difference equation, 2 poles, 2 zeros; single precision.

```
void vDSP_deq22 (float * A,
vDSP_Stride I,
float * B,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector; must have at least N+2 elements

I

Stride for A

B

5 single-precision inputs, filter coefficients

C

Single-precision real output vector; must have at least N+2 elements

K

Stride for C

N

Number of new output elements to produce

Discussion

Performs two-pole two-zero recursive filtering on real input vector A. Since the computation is recursive, the first two elements in vector C must be initialized prior to calling vDSP_deq22. vDSP_deq22 creates N new values for vector C beginning with its third element and requires at least N+2 input values from vector A. This function can only be done out of place.

$$C_{nk} = \sum_{p=0}^{2} A_{(n-p)i} B_p - \sum_{p=3}^{4} C_{(n-p+2)k} B_p \qquad \text{n} = \{2, \text{N+1}\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_deq22D

Difference equation, 2 poles, 2 zeros; double precision.

```
void vDSP_deq22D (double * A,
vDSP_Stride I,
double * B,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector; must have at least N+2 elements

I
: Stride for A

B
: 5 double-precision inputs, filter coefficients

C
: Double-precision real output vector; must have at least N+2 elements

K
: Stride for C

N
: Number of new output elements to produce

Discussion

Performs two-pole two-zero recursive filtering on real input vector A. Since the computation is recursive, the first two elements in vector C must be initialized prior to calling vDSP_deq22D. vDSP_deq22D creates N new values for vector C beginning with its third element and requires at least N+2 input values from vector A. This function can only be done out of place.

$$C_{nk} = \sum_{p=0}^{2} A_{(n-p)i} B_p - \sum_{p=3}^{4} C_{(n-p+2)k} B_p \qquad \mathrm{n} = \{2, \mathrm{N}+1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vaam

Vector add, add, and multiply; single precision.

```
void vDSP_vaam (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for A

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real input vector

L
: Stride for `D`

E
: Single-precision real output vector

M
: Stride for `E`

N
: Count ; each vector must have at least `N` elements

Discussion

This performs the operation

$$E_{nm} = (A_{ni} + B_{nj})(C_{nk} + D_{nl}) \qquad n = \{0, N-1\}$$

Multiplies the sum of vectors `A` and `B` by the sum of vectors `C` and `D`. Results are stored in vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vaamD

Vector add, add, and multiply; double precision.

```
void vDSP_vaamD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real input vector

L
: Stride for `D`

E
: Double-precision real output vector

M
: Stride for `E`

N
: Count ; each vector must have at least N elements

Discussion

This performs the operation

$$E_{nm} = (A_{ni} + B_{nj})(C_{nk} + D_{nl}) \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Multiplies the sum of vectors `A` and `B` by the sum of vectors `C` and `D`. Results are stored in vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vadd

Adds vector `A` to vector `B` and leaves the result in vector `C`; single precision.

```
void vDSP_vadd (const float input1[],
vDSP_Stride stride1,
const float input2[],
vDSP_Stride stride2,
float result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion

This performs the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vaddD

Adds vector `A` to vector `B` and leaves the result in vector `C`; double precision.

```
void vDSP_vaddD (const double input1[],
vDSP_Stride stride1,
const double input2[],
vDSP_Stride stride2,
double result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This performs the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vam

Adds vectors `A` and `B`, multiplies the sum by vector `C`, and leaves the result in vector `D`; single precision.

```
void vDSP_vam (const float input1[],
vDSP_Stride stride1,
const float input2[],
vDSP_Stride stride2,
const float input3[],
vDSP_Stride stride3,
float result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This performs the operation

$$D_{nL} = (A_{nI} + B_{nJ})\, C_{nK} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vamD

Adds vectors `A` and `B`, multiplies the sum by vector `C`, and leaves the result in vector `D`; double precision.

```
void vDSP_vamD (const double input1[],
vDSP_Stride stride1,
const double input2[],
vDSP_Stride stride2,
const double input3[],
vDSP_Stride stride3,
double result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This performs the operation

$$D_{nL} = (A_{nI} + B_{nJ})C_{nK} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vasbm

Vector add, subtract, and multiply; single precision.

```
void vDSP_vasbm (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real input vector

L
: Stride for `D`

E
: Single-precision real output vector

M
: Stride for `E`

N
: Count ; each vector must have at least `N` elements

Discussion

This performs the operation

$$E_{nM} = (A_{nI} + B_{nJ})(C_{nK} - D_{nL}) \;, \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Multiplies the sum of vectors `A` and `B` by the result of subtracting vector `D` from vector `C`. Results are stored in vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vasbmD

Vector add, subtract, and multiply; double precision.

```
void vDSP_vasbmD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector

I

Stride for `A`

B

Double-precision real input vector

J

Stride for `B`

C

Double-precision real input vector

K

Stride for `C`

D

Double-precision real input vector

L

Stride for `D`

E

Double-precision real output vector

M

Stride for `E`

N

Count ; each vector must have at least N elements

Discussion

This performs the operation

$$E_{nM} = (A_{nI} + B_{nJ})(C_{nK} - D_{nL}) \ , \qquad n = \{0, N-1\}$$

Multiplies the sum of vectors `A` and `B` by the result of subtracting vector `D` from vector `C`. Results are stored in vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vasm

Vector add and scalar multiply; single precision.

```
void vDSP_vasm (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input scalar

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count ; each vector must have at least N elements

Discussion

This performs the operation

$$D_{nM} = (A_{nI} + B_{nJ})\,c\ ,\qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Multiplies the sum of vectors `A` and `B` by scalar `C`. Results are stored in vector `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vasmD

Vector add and scalar multiply; double precision.

```
void vDSP_vasmD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input scalar

D
: Double-precision real output vector

L
: Stride for `D`

N
: Count ; each vector must have at least N elements

Discussion

This performs the operation

$$D_{nM} = (A_{nI} + B_{nJ})\, c \;, \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Multiplies the sum of vectors `A` and `B` by scalar `C`. Results are stored in vector `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vdist

Vector distance; single precision.

```
void vDSP_vdist (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real output vector

K
: Stride for `C`

N
: Count

Discussion

Performs the operation

$$C_{nk} = \sqrt{A_{ni}^2 + B_{nj}^2} \qquad n = \{0, N\text{-}1\}$$

Computes the square root of the sum of the squares of corresponding elements of vectors `A` and `B`, and stores the result in the corresponding element of vector `C`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vdistD

Vector distance; double precision.

```
void vDSP_vdistD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector

I

Stride for A

B

Double-precision real input vector

J

Stride for B

C

Double-precision real output vector

K

Stride for C

N

Count

Discussion

Performs the operation

$$C_{nk} = \sqrt{A_{ni}^2 + B_{nj}^2} \qquad n = \{0, N\text{-}1\}$$

Computes the square root of the sum of the squares of corresponding elements of vectors A and B, and stores the result in the corresponding element of vector C.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vdiv

Vector divide; single precision.

```
void vDSP_vdiv (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real output vector

K
: Stride for `C`

N
: Count

Discussion

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B_{nJ}} \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Divides elements of vector `A` by corresponding elements of vector `B`, and stores the results in corresponding elements of vector `C`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vdivD

Vector divide; double precision.

```
void vDSP_vdivD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real output vector

K
: Stride for `C`

N
: Count

Discussion

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B_{nJ}} \qquad n = \{0, N\text{-}1\}$$

Divides elements of vector `A` by corresponding elements of vector `B`, and stores the results in corresponding elements of vector `C`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vdivi

Vector divide; integer.

```
void vDSP_vdivi (int * A,
vDSP_Stride I,
int * B,
vDSP_Stride J,
int * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Integer input vector

I

Stride for `A`

B

Integer input vector

J

Stride for `B`

C

Integer output vector

K

Stride for `C`

N

Count

Discussion

Performs the operation

$$C_{nK} = \frac{A_{nI}}{B_{nJ}} \qquad n = \{0, N\text{-}1\}$$

Divides elements of vector `A` by corresponding elements of vector `B`, and stores the results in corresponding elements of vector `C`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_venvlp

Vector envelope; single precision.

```
void vDSP_venvlp (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector: high envelope

I
: Stride for `A`

B
: Single-precision real input vector: low envelope

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

Performs the operation

$$\text{If } C_{nK} > A_{nI} \text{ or } C_{nK} < B_{nJ} \text{ then } D_{nM} = C_{nK}$$
$$\text{else } D_{nM} = 0.0 \qquad n = \{0, N\text{-}1\}$$

Finds the extrema of vector C. For each element of C, the corresponding element of A provides an upper-threshold value, and the corresponding element of B provides a lower-threshold value. If the value of an element of C falls outside the range defined by these thresholds, it is copied to the corresponding element of vector D. If its value is within the range, the corresponding element of vector D is set to zero.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_venvlpD

Vector envelope; double precision.

```
void vDSP_venvlpD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector: high envelope

I

Stride for `A`

B

Double-precision real input vector: low envelope

J

Stride for `B`

C

Double-precision real input vector

K

Stride for `C`

D

Double-precision real output vector

L

Stride for `D`

N

Count

Discussion

Performs the operation

$$\text{If } C_{nK} > A_{nI} \text{ or } C_{nK} < B_{nJ} \text{ then } D_{nM} = C_{nK}$$

$$\text{else } D_{nM} = 0.0 \qquad n = \{0, N\text{-}1\}$$

Finds the extrema of vector C. For each element of C, the corresponding element of A provides an upper-threshold value, and the corresponding element of B provides a lower-threshold value. If the value of an element of C falls outside the range defined by these thresholds, it is copied to the corresponding element of vector D. If its value is within the range, the corresponding element of vector D is set to zero.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_veqvi

Vector equivalence, 32-bit logical.

```
void vDSP_veqvi (int * A,
vDSP_Stride I,
int * B,
vDSP_Stride J,
int * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Integer input vector

I

Stride for `A`

B

Integer input vector

J

Stride for `B`

C

Integer output vector

K

Stride for `C`

N

Count

Discussion

Performs the operation

$$C_{nk} = A_{ni} \bullet XNOR \bullet B_{nj} \qquad \mathrm{n} = \{0, \mathrm{N}-1\}$$

Outputs the bitwise logical equivalence, exclusive NOR, of the integers of vectors `A` and `B`. For each pair of input values, bits in each position are compared. A bit in the output value is set if both input bits are set, or both are clear; otherwise it is cleared.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vintb

Vector linear interpolation between vectors; single precision.

```
void vDSP_vintb (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input scalar: interpolation constant

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

Performs the operation

$$D_{nK} = A_{nI} + C[B_{nJ} - A_{nI}] \qquad n = \{0, N-1\}$$

Creates vector `D` by interpolating between vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vintbD

Vector linear interpolation between vectors; double precision.

```
void vDSP_vintbD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input scalar: interpolation constant

D
: Double-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

Performs the operation

$$D_{nK} = A_{nI} + C[B_{nJ} - A_{nI}] \qquad n = \{0, N\text{-}1\}$$

Creates vector `D` by interpolating between vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vma

Vector multiply and add; single precision.

```
void vDSP_vma (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

This performs the operation

$$D_{nM} = A_{nI} \cdot B_{nJ} + C_{nK} \qquad n = \{0, N-1\}$$

Multiplies corresponding elements of vectors `A` and `B`, add the corresponding elements of vector `C`, and stores the results in vector `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmaD

Vector multiply and add; double precision.

```
void vDSP_vmaD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector

I

Stride for `A`

B

Double-precision real input vector

J

Stride for `B`

C

Double-precision real input vector

K

Stride for `C`

D

Double-precision real output vector

L

Stride for `D`

N

Count

Discussion

This performs the operation

$$D_{nM} = A_{nI} \cdot B_{nJ} + C_{nK} \qquad n = \{0, N-1\}$$

Multiplies corresponding elements of vectors `A` and `B`, add the corresponding elements of vector `C`, and stores the results in vector `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmax

Vector maxima; single precision.

```
void vDSP_vmax (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

I

Stride for `A`

B

Single-precision real input vector

J

Stride for `B`

C

Single-precision real output vector

K

Stride for `C`

N

Count

Discussion

This performs the operation

$$\text{If} \quad A_{nI} \geq B_{nJ} \quad \text{then} \quad C_{nK} = A_{nI} \quad \text{else} \quad C_{nK} = B_{nJ} \qquad n = \{0, N\text{-}1\}$$

Each element of output vector `C` is the greater of the corresponding values from input vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmaxD

Vector maxima; double precision.

```
void vDSP_vmaxD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for A

B
: Double-precision real input vector

J
: Stride for B

C
: Double-precision real output vector

K
: Stride for C

N
: Count

Discussion

This performs the operation

$$\text{If} \quad A_{nI} \geq B_{nJ} \quad \text{then} \quad C_{nK} = A_{nI} \quad \text{else} \quad C_{nK} = B_{nJ} \qquad n = \{0, N-1\}$$

Each element of output vector C is the greater of the corresponding values from input vectors A and B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vmaxmg

Vector maximum magnitudes; single precision.

```
void vDSP_vmaxmg (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real output vector

K
: Stride for `C`

N
: Count

Discussion

This performs the operation

$$\text{If} \quad |A_{nI}| \geq |B_{nJ}| \quad \text{then} \quad C_{nK} = |A_{nI}| \quad \text{else} \quad C_{nK} = |B_{nJ}| \qquad n = \{0, N-1\}$$

Each element of output vector `C` is the larger of the magnitudes of corresponding values from input vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmaxmgD

Vector maximum magnitudes; double precision.

```
void vDSP_vmaxmgD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real output vector

K
: Stride for `C`

N
: Count

Discussion

This performs the operation

$$\text{If} \quad |A_{nI}| \geq |B_{nJ}| \quad \text{then} \quad C_{nK} = |A_{nI}| \quad \text{else} \quad C_{nK} = |B_{nJ}| \qquad n = \{0, N\text{-}1\}$$

Each element of output vector `C` is the larger of the magnitudes of corresponding values from input vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmin

Vector minima; single precision.

```
void vDSP_vmin (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

I

Stride for `A`

B

Single-precision real input vector

J

Stride for `B`

C

Single-precision real output vector

K

Stride for `C`

N

Count

Discussion

This performs the operation

$$\text{If} \quad A_{nI} \leq B_{nJ} \quad \text{then} \quad C_{nK} = A_{nI} \quad \text{else} \quad C_{nK} = B_{nJ} \qquad n = \{0, N\text{-}1\}$$

Each element of output vector `C` is the lesser of the corresponding values from input vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vminD

Vector minima; double precision.

```
void vDSP_vminD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for A

B
: Double-precision real input vector

J
: Stride for B

C
: Double-precision real output vector

K
: Stride for C

N
: Count

Discussion

This performs the operation

$$\text{If} \quad A_{nI} \leq B_{nJ} \quad \text{then} \quad C_{nK} = A_{nI} \quad \text{else} \quad C_{nK} = B_{nJ} \qquad n = \{0, N\text{-}1\}$$

Each element of output vector C is the lesser of the corresponding values from input vectors A and B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vminmg

Vector minimum magnitudes; single precision.

```
void vDSP_vmin (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

I

Stride for A

B

Single-precision real input vector

J

Stride for B

C

Single-precision real output vector

K

Stride for C

N

Count

Discussion

This performs the operation

$$\text{If} \quad |A_{nI}| \le |B_{nJ}| \quad \text{then} \quad C_{nK} = |A_{nI}| \quad \text{else} \quad C_{nK} = |B_{nJ}| \qquad n = \{0, N-1\}$$

Each element of output vector C is the smaller of the magnitudes of corresponding values from input vectors A and B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vminmgD

Vector minimum magnitudes; double precision.

```
void vDSP_vminD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for A

B
: Double-precision real input vector

J
: Stride for B

C
: Double-precision real output vector

K
: Stride for C

N
: Count

Discussion

This performs the operation

$$\text{If} \quad |A_{nI}| \leq |B_{nJ}| \quad \text{then} \quad C_{nK} = |A_{nI}| \quad \text{else} \quad C_{nK} = |B_{nJ}| \qquad n = \{0, N\text{-}1\}$$

Each element of output vector C is the smaller of the magnitudes of corresponding values from input vectors A and B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmma

Vector multiply, multiply, and add; single precision.

```
void vDSP_vmma (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

I

Stride for `A`

B

Single-precision real input vector

J

Stride for `B`

C

Single-precision real input vector

K

Stride for `C`

D

Single-precision real input vector

L

Stride for `D`

E

Single-precision real output vector

M

Stride for `E`

N

Count

Discussion

This performs the operation

$$E_{nM} = A_{nI} \cdot B_{nJ} + C_{nK} \cdot D_{nL} \qquad n = \{0, N-1\}$$

Corresponding elements of `A` and `B` are multiplied, corresponding values of `C` and `D` are multiplied, and these products are added together and stored in `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmmaD

Vector multiply, multiply, and add; double precision.

```
void vDSP_vmmaD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real input vector

L
: Stride for `D`

E
: Double-precision real output vector

M
: Stride for `E`

N
: Count

Discussion

This performs the operation

$$E_{nM} = A_{nI} \bullet B_{nJ} + C_{nK} \bullet D_{nL} \qquad \mathrm{n} = \{0, \mathrm{N}-1\}$$

Corresponding elements of `A` and `B` are multiplied, corresponding values of `C` and `D` are multiplied, and these products are added together and stored in `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vmmsb

Vector multiply, multiply, and subtract; single precision.

```
void vDSP_vmmsb (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real input vector

L
: Stride for `D`

E
: Single-precision real output vector

M
: Stride for `E`

N
: Count

Discussion
This performs the operation

$$E_{nM} = A_{nI} B_{nJ} - C_{nK} D_{nL} \qquad n = \{0, N-1\}$$

Corresponding elements of `A` and `B` are multiplied, corresponding values of `C` and `D` are multiplied, and the second product is subtracted from the first. The result is stored in `E`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vmmsbD

Vector multiply, multiply, and subtract; double precision.

```
void vDSP_vmmsbD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real input vector

L
: Stride for `D`

E
: Double-precision real output vector

M
: Stride for `E`

N
: Count

Discussion

This performs the operation

$$E_{nM} = A_{nI} B_{nJ} - C_{nK} D_{nL} \qquad n = \{0, N-1\}$$

Corresponding elements of `A` and `B` are multiplied, corresponding values of `C` and `D` are multiplied, and the second product is subtracted from the first. The result is stored in `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmsa

Vector multiply and scalar add; single precision.

```
void vDSP_vmsa (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input scalar

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

This performs the operation

$$D_{nK} = A_{nI} \cdot B_{nJ} + C \qquad n = \{0, N-1\}$$

Corresponding elements of A and B are multiplied and the scalar C is added. The result is stored in D.

Availability
Available in Mac OS X v10.4 and later.

Declared In
vDSP.h

vDSP_vmsaD

Vector multiply and scalar add; double precision.

```
void vDSP_vmsaD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
Double-precision real input vector

I
Stride for A

B
Double-precision real input vector

J
Stride for B

C
Double-precision real input scalar

D
Double-precision real output vector

L
Stride for D

N
Count

Discussion
This performs the operation

$$D_{nK} = A_{nI} \cdot B_{nJ} + C \qquad n = \{0, N\text{-}1\}$$

Corresponding elements of A and B are multiplied and the scalar C is added. The result is stored in D.

Availability
Available in Mac OS X v10.4 and later.

Declared In
vDSP.h

vDSP_vmsb

Vector multiply and subtract, single precision.

```
void vDSP_vmsb (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real output vector

L
: Stride for `D`

N
: Count

Discussion

This performs the operation

$$D_{nM} = A_{nI} \cdot B_{nJ} - C_{nK} \qquad n = \{0, N\text{-}1\}$$

Corresponding elements of `A` and `B` are multiplied and the corresponding value of `C` is subtracted. The result is stored in `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmsbD

Vector multiply and subtract; double precision.

```
void vDSP_vmsbD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector

I

Stride for `A`

B

Double-precision real input vector

J

Stride for `B`

C

Double-precision real input vector

K

Stride for `C`

D

Double-precision real output vector

L

Stride for `D`

N

Count

Discussion

This performs the operation

$$D_{nM} = A_{nI} \cdot B_{nJ} - C_{nK} \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Corresponding elements of `A` and `B` are multiplied and the corresponding value of `C` is subtracted. The result is stored in `D`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmul

Multiplies vector `A` by vector `B` and leaves the result in vector `C`; single precision.

```
void vDSP_vmul (const float A[],
vDSP_Stride I,
const float B[],
vDSP_Stride J,
float C[],
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for `A`

B
: Input vector

J
: Address stride for `B`

C
: Output vector

K
: Address stride for `C`

N
: Complex output count

Discussion

This performs the operation

$$C_{nK} = A_{nI} \cdot B_{nJ} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vmulD

Multiplies vector `A` by vector `B` and leaves the result in vector `C`; double precision.

```
void vDSP_vmulD (const double A[],
vDSP_Stride I,
const double B[],
vDSP_Stride J,
double C[],
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Input vector

I

Address stride for A

B

Input vector

J

Address stride for B

C

Output vector

K

Address stride for C

N

Complex output count

Discussion

This performs the operation

$$C_{nK} = A_{nI} \cdot B_{nJ} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vpoly

Vector polynomial evaluation; single precision.

```
void vDSP_vpoly (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N,
vDSP_Length P);
```

Parameters

A

Single-precision real input vector: coefficients

I

Stride for `A`

B

Single-precision real input vector: variable values

J

Stride for `B`

C

Single-precision real output vector

K

Stride for `C`

N

Count

P

Degree of polynomial

Discussion

Performs the operation

$$C_{nK} = \sum_{p=0}^{P} A_{pI} \cdot B_{nJ}^{P-p} \qquad n = \{0, N-1\}$$

Evaluates polynomials using vector `B` as independent variables and vector `A` as coefficients. A polynomial of degree p requires p+1 coefficients, so vector `A` should contain `P`+1 values.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vpolyD

Vector polynomial evaluation; double precision.

```
void vDSP_vpolyD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N,
vDSP_Length P);
```

Parameters

A

Double-precision real input vector: coefficients

I

Stride for `A`

B
: Double-precision real input vector: variable values

J
: Stride for B

C
: Double-precision real output vector

K
: Stride for C

N
: Count

P
: Degree of polynomial

Discussion

Performs the operation

$$C_{nK} = \sum_{p=0}^{P} A_{pI} \bullet B_{nJ}^{P-p} \qquad n = \{0, N\text{-}1\}$$

Evaluates polynomials using vector B as independent variables and vector A as coefficients. A polynomial of degree p requires p+1 coefficients, so vector A should contain P+1 values.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vpythg

Vector pythagoras; single precision.

```
void vDSP_vpythg (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for A

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real input vector

L
: Stride for `D`

E
: Single-precision real output vector

M
: Stride for `E`

N
: Count

Discussion

Performs the operation

$$E_{nM} = \sqrt{(A_{nI} - C_{nK})^2 + (B_{nJ} - D_{nL})^2} \qquad n = \{0, N-1\}$$

Subtracts vector `C` from `A` and squares the differences, subtracts vector `D` from `B` and squares the differences, adds the two sets of squared differences, and then writes the square roots of the sums to vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vpythgD

Vector pythagoras; double precision.

```
void vDSP_vpythgD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real input vector

L
: Stride for `D`

E
: Double-precision real output vector

M
: Stride for `E`

N
: Count

Discussion

Performs the operation

$$E_{nM} = \sqrt{(A_{nI} - C_{nK})^2 + (B_{nJ} - D_{nL})^2} \qquad n = \{0, N\text{-}1\}$$

Subtracts vector `C` from `A` and squares the differences, subtracts vector `D` from `B` and squares the differences, adds the two sets of squared differences, and then writes the square roots of the sums to vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vqint

Vector quadratic interpolation; single precision.

```
void vDSP_vqint (float * A,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N,
vDSP_Length M);
```

Parameters

A
: Single-precision real input vector

B
: Single-precision real input vector: integer parts are indices into A and fractional parts are interpolation constants

J
: Stride for B

C
: Single-precision real output vector

K
: Stride for C

N
: Count for C

M
: Length of A: must be greater than or equal to 3

Discussion

Performs the operation

$$C_{nK} = \frac{A_{\beta-1}[\alpha^2 - \alpha] + A_{\beta}[2.0 - 2.0\alpha^2] + A_{\beta+1}[\alpha^2 + \alpha]}{2}$$

where: $\beta = \max(\mathrm{trunc}(B_{nJ}), 1) \qquad n = \{0, N-1\}$

$\alpha = B_{nJ} - \mathrm{float}(\beta)$

Generates vector C by interpolating between neighboring values of vector A as controlled by vector B. The integer portion of each element in B is the zero-based index of the first element of a triple of adjacent values in vector A.

The value of the corresponding element of C is derived from these three values by quadratic interpolation, using the fractional part of the value in B.

Argument M is not used in the calculation. However, the integer parts of the values in B must be less than or equal to M - 2.

Availability

Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vqintD

Vector quadratic interpolation; double precision.

```
void vDSP_vqintD (double * A,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N,
vDSP_Length M);
```

Parameters

A
: Double-precision real input vector

B
: Double-precision real input vector: integer parts are indices into `A` and fractional parts are interpolation constants

J
: Stride for `B`

C
: Double-precision real output vector

K
: Stride for `C`

N
: Count for C

M
: Length of A: must be greater than or equal to 3

Discussion
Performs the operation

$$C_{nK} = \frac{A_{\beta-1}[\alpha^2 - \alpha] + A_{\beta}[2.0 - 2.0\alpha^2] + A_{\beta+1}[\alpha^2 + \alpha]}{2}$$

where: $\beta = \max(\mathrm{trunc}(B_{nJ}), 1)$ $\qquad n = \{0, \text{N-1}\}$

$\alpha = B_{nJ} - \mathrm{float}(\beta)$

Generates vector `C` by interpolating between neighboring values of vector `A` as controlled by vector `B`. The integer portion of each element in `B` is the zero-based index of the first element of a triple of adjacent values in vector `A`.

The value of the corresponding element of `C` is derived from these three values by quadratic interpolation, using the fractional part of the value in B.

Argument `M` is not used in the calculation. However, the integer parts of the values in `B` must be less than or equal to `M` - 2.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsbm

Vector subtract and multiply; single precision.

```
void vDSP_vsbm (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
Single-precision real input vector

I
Stride for `A`

B
Single-precision real input vector

J
Stride for `B`

C
Single-precision real input vector

K
Stride for `C`

D
Single-precision real output vector

L
Stride for `D`

N
Count

Discussion
Performs the operation

$$D_{nM} = (A_{nI} - B_{nJ})\, C_{nK} \qquad n = \{0, N-1\}$$

Subtracts vector `B` from vector `A` and then multiplies the differences by vector `C`. Results are stored in vector `D`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsbmD

Vector subtract and multiply; double precision.

```
void vDSP_vsbmD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Double for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real output vector

L
: Stride for `D`

N
: Count

Discussion
Performs the operation

$$D_{nM} = (A_{nI} - B_{nJ})\, C_{nK} \qquad \mathrm{n} = \{0, \mathrm{N\text{-}1}\}$$

Subtracts vector `B` from vector `A` and then multiplies the differences by vector `C`. Results are stored in vector `D`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsbsbm

Vector subtract, subtract, and multiply; single precision.

```
void vDSP_vsbsbm (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
float * D,
vDSP_Stride L,
float * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for `A`

B
: Single-precision real input vector

J
: Stride for `B`

C
: Single-precision real input vector

K
: Stride for `C`

D
: Single-precision real input vector

L
: Stride for `D`

E
: Single-precision real output vector

M
: Stride for `E`

N
: Count

Discussion

Performs the operation

$$E_{nM} = (A_{nI} - B_{nJ})(C_{nK} - D_{nL}) \qquad n = \{0, N\text{-}1\}$$

Subtracts vector `B` from `A`, subtracts vector `D` from `C`, and multiplies the differences. Results are stored in vector `E`.

Availability

Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsbsbmD

Vector subtract, subtract, and multiply; double precision.

```
void vDSP_vsbsbmD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
double * D,
vDSP_Stride L,
double * E,
vDSP_Stride M,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input vector

K
: Stride for `C`

D
: Double-precision real input vector

L
: Stride for `D`

E
: Double-precision real output vector

M
: Stride for `E`

N
: Count

Discussion
Performs the operation

$$E_{nM} = (A_{nI} - B_{nJ})(C_{nK} - D_{nL}) \qquad n = \{0, N-1\}$$

Subtracts vector B from A, subtracts vector D from C, and multiplies the differences. Results are stored in vector E.

Availability
Available in Mac OS X v10.4 and later.

Declared In
vDSP.h

vDSP_vsbsm

Vector subtract and scalar multiply; single precision.

```
void vDSP_vsbsm (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
float * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Single-precision real input vector

I
: Stride for A

B
: Single-precision real input vector

J
: Stride for B

C
: Single-precision real input scalar

D
: Single-precision real output vector

L
: Stride for D

N
: Count

Discussion
Performs the operation

$$D_{nK} = (A_{nI} - B_{nJ})C \qquad n = \{0, N-1\}$$

Subtracts vector B from vector A and then multiplies each difference by scalar C. Results are stored in vector D.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsbsmD

Vector subtract and scalar multiply; double precision.

```
void vDSP_vsbsmD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
double * D,
vDSP_Stride L,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for `A`

B
: Double-precision real input vector

J
: Stride for `B`

C
: Double-precision real input scalar

D
: Double-precision real output vector

L
: Stride for `D`

N
: Count

Discussion
Performs the operation

$$D_{nK} = (A_{nI} - B_{nJ})C \qquad n = \{0, N-1\}$$

Subtracts vector `B` from vector `A` and then multiplies each difference by scalar `C`. Results are stored in vector `D`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsub

Subtracts vector `signal1` from vector `signal2` and leaves the result in vector `result`; single precision.

```
void vDSP_vsub (const float input1[],
vDSP_Stride stride1,
const float input2[],
vDSP_Stride stride2,
float result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This performs the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vsubD

Subtracts vector `signal1` from vector `signal2` and leaves the result in vector `result`; double precision.

```
void vDSP_vsub (const float input1[],
vDSP_Stride stride1,
const float input2[],
vDSP_Stride stride2,
float result[],
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This performs the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_vswap

Vector swap; single precision.

```
void vDSP_vswap (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
vDSP_Length N);
```

Parameters

A

Single-precision real input-output vector

I

Stride for A

B

Single-precision real input-output vector

J

Stride for B

N

Count

Discussion

Performs the operation

$$C_{nK} \Leftrightarrow A_{nI} \qquad n = \{0, N\text{-}1\}$$

Exchanges the elements of vectors A and B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_vswapD

Vector swap; double precision.

```
void vDSP_vswapD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
vDSP_Length N);
```

Parameters

A

Double-precision real input-output vector

I

Stride for A

B

Double-precision real input-output vector

J

Stride for B

N

Count

Discussion

Performs the operation

$$C_{nK} \Leftrightarrow A_{nI} \qquad n = \{0, N\text{-}1\}$$

Exchanges the elements of vectors `A` and `B`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_vtmerg

Tapered merge of two vectors; single precision.

```
void vDSP_vtmerg (float * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
float * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

I

Stride for `A`

B

Single-precision real input vector

J

Stride for `B`

C

Single-precision real output vector

K

Stride for `C`

N

Count

Discussion

Performs the operation

$$C_{nK} = A_{nI} + \frac{n(B_{nJ} - A_{nI})}{N-1} \qquad n = \{0, N\text{-}1\}$$

Performs a tapered merge of vectors A and B. Values written to vector C range from element zero of vector A to element N-1 of vector B. Output values between these endpoints reflect varying amounts of their corresponding inputs from vectors A and B, with the percentage of vector A decreasing and the percentage of vector B increasing as the index increases.

Availability
Available in Mac OS X v10.4 and later.

Declared In
vDSP.h

vDSP_vtmergD

Tapered merge of two vectors; double precision.

```
void vDSP_vtmergD (double * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
double * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision real input vector

I
: Stride for A

B
: Double-precision real input vector

J
: Stride for B

C
: Double-precision real output vector

K
: Stride for C

N
: Count

Discussion
Performs the operation

$$C_{nK} = A_{nI} + \frac{n(B_{nJ} - A_{nI})}{N-1} \qquad \text{n} = \{0, \text{N-1}\}$$

Performs a tapered merge of vectors A and B. Values written to vector C range from element zero of vector A to element N-1 of vector B. Output values between these endpoints reflect varying amounts of their corresponding inputs from vectors A and B, with the percentage of vector A decreasing and the percentage of vector B increasing as the index increases.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zaspec

Computes an accumulating autospectrum; single precision.

```
void vDSP_zaspec (DSPSplitComplex * A,
float * C,
vDSP_Length N);
```

Parameters

A
: Input vector

C
: Input-output vector

N
: Real output count

Discussion
`vDSP_zaspec` multiplies single-precision complex vector `A` by its complex conjugates, yielding the sums of the squares of the complex and real parts: (x + iy) (x - iy) = (x*x + y*y). The results are added to real single-precision input-output vector `C`. Vector `C` must contain valid data from previous processing or should be initialized according to your needs before calling `vDSP_zaspec`.

$$C_n = C_n + (Re(A_n))^2 + (Im(A_n))^2 \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zaspecD

Computes an accumulating autospectrum; double precision.

```
void vDSP_zaspecD (DSPDoubleSplitComplex * A,
double * C,
vDSP_Length N);
```

Parameters

A
: Input vector

C
: Input-output vector

N

Real output count

Discussion

`vDSP_zaspecD` multiplies double-precision complex vector `A` by its complex conjugates, yielding the sums of the squares of the complex and real parts: (x + iy) (x - iy) = (x*x + y*y). The results are added to real double-precision input-output vector `C`. Vector `C` must contain valid data from previous processing or should be initialized according to your needs before calling `vDSP_zaspec`.

$$C_n = C_n + (Re(A_n))^2 + (Im(A_n))^2 \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zcoher

Coherence function of two signals; single precision.

```
void vDSP_zcoher (float * A,
float * B,
DSPSplitComplex * C,
float * D,
vDSP_Length N);
```

Discussion

Computes the single-precision coherence function `D` of two signals. The inputs are the signals' autospectra, real single-precision vectors `A` and `B`, and their cross-spectrum, single-precision complex vector `C`.

$$D_n = \frac{[Re(C_n)]^2 + [Im(C_n)]^2}{A_n B_n} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zcoherD

Coherence function of two signals; double precision.

```
void vDSP_zcoherD (double * A,
double * B,
DSPDoubleSplitComplex * C,
double * D,
vDSP_Length N);
```

Discussion

Computes the double-precision coherence function D of two signals. The inputs are the signals' autospectra, real double-precision vectors A and B, and their cross-spectrum, double-precision complex vector C.

$$D_n = \frac{[Re(C_n)]^2 + [Im(C_n)]^2}{A_n B_n} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zcspec

Accumulating cross-spectrum on two complex vectors; single precision.

```
void vDSP_zcspec (DSPSplitComplex * A,
DSPSplitComplex * B,
DSPSplitComplex * C,
vDSP_Length N);
```

Parameters

A

Single-precision complex input vector

B

Single-precision complex input vector

C

Single-precision complex input-output vector

N

Count

Discussion

Computes the cross-spectrum of complex vectors A and B and then adds the results to complex input-output vector C. Vector C should contain valid data from previous processing or should be initialized with zeros before calling `vDSP_zcspec`.

$$C_n = C_n + A_n^* B_n \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zcspecD

Accumulating cross-spectrum on two complex vectors; double precision.

```
void vDSP_zcspecD (DSPDoubleSplitComplex * A,
DSPDoubleSplitComplex * B,
DSPDoubleSplitComplex * C,
vDSP_Length N);
```

Parameters

A

Double-precision complex input vector

B

Double-precision complex input vector

C

Double-precision complex input-output vector

N

Count

Discussion

Computes the cross-spectrum of complex vectors `A` and `B` and then adds the results to complex input-output vector `C`. Vector `C` should contain valid data from previous processing or should be initialized with zeros before calling `vDSP_zcspecD`.

$$C_n = C_n + A_n^* B_n \qquad \mathrm{n} = \{0, \mathrm{N}\text{-}1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zrvadd

Adds real vector `B` to complex vector `A` and leaves the result in complex vector `C`; single precision.

```
void vDSP_zrvadd (DSPSplitComplex * A,
vDSP_Stride I,
const float B[],
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Input vector

I

Address stride for `A`

B

Input vector

J

Address stride for B

C

Output vector

K

Address stride for C

N

Complex output count

Discussion

This performs the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N\text{-}1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zrvaddD

Adds real vector B to complex vector A and leaves the result in complex vector C; double precision.

```
void vDSP_zrvaddD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
const double B[],
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Input vector

I

Address stride for A

B

Input vector

J

Address stride for B

C

Output vector

K

Address stride for C

N

Complex output count

Discussion

This performs the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zrvdiv

Divides complex vector `A` by real vector `B` and leaves the result in vector `C`; single precision.

```
void vDSP_zrvdiv (DSPSplitComplex * A,
vDSP_Stride I,
float * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Discussion
This performs the operation

$$C_{nk} = \frac{A_{ni}}{B_{nj}} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zrvdivD

Divides complex vector `A` by real vector `B` and leaves the result in vector `C`; double precision.

```
void vDSP_zrvdivD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
double * B,
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Discussion
This peforms the operation

$$C_{nk} = \frac{A_{ni}}{B_{nj}} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zrvmul

Multiplies complex vector `A` by real vector `B` and leaves the result in vector `C`; single precision.

```
void vDSP_zrvmul (DSPSplitComplex * A,
vDSP_Stride I,
const float B[],
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for `A`

B
: Input vector

J
: Address stride for `B`

C
: Output vector

K
: Address stride for `C`

N
: Complex output count

Discussion
This peforms the operation

$$C_{nK} = A_{nI} \cdot B_{nJ} \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zrvmulD

Multiplies complex vector `A` by real vector `B` and leaves the result in vector `C`; double precision.

```
void vDSP_zrvmulD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
const double B[],
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for A

B
: Input vector

J
: Address stride for B

C
: Output vector

K
: Address stride for C

N
: Complex output count

Discussion

This performs the operation

$$C_{nK} = A_{nI} \cdot B_{nJ} \qquad \mathrm{n} = \{0, \mathrm{N}-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zrvsub

Subtracts real vector B from complex vector A and leaves the result in complex vector C; single precision.

```
void vDSP_zrvsub (DSPSplitComplex * A,
vDSP_Stride I,
const float B[],
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for A

B
: Input vector

J
: Address stride for B

C
: Output vector

K
: Address stride for C

N
: Complex output count

Discussion

This performs the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zrvsubD

Subtracts real vector B from complex vector A and leaves the result in complex vector C; double precision.

```
void vDSP_zrvsubD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
const double B[],
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for A

B
: Input vector

J
: Address stride for B

C
: Output vector

K
: Address stride for C

N

Complex output count

Discussion

This peforms the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_ztrans

Transfer function; single precision.

```
void vDSP_ztrans (float * A,
DSPSplitComplex * B,
DSPSplitComplex * C,
vDSP_Length N);
```

Parameters

A

Single-precision real input vector

B

Single-precision complex input vector

C

Single-precision complex output vector

N

Count

Discussion

This peforms the operation

$$C_n = \frac{B_n}{A_n} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_ztransD

Transfer function; double precision.

```
void vDSP_ztransD (double * A,
DSPDoubleSplitComplex * B,
DSPDoubleSplitComplex * C,
vDSP_Length N);
```

Parameters

A

Double-precision real input vector

B

Double-precision complex input vector

C

Double-precision complex output vector

N

Count

Discussion

This peforms the operation

$$C_n = \frac{B_n}{A_n} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvadd

Adds complex vectors `A` and `B` and leaves the result in complex vector `C`; single precision.

```
void vDSP_zvadd (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Input vector

I

Address stride for `A`

B

Input vector

J

Address stride for `B`

C

Output vector

K

Address stride for `C`

N

Complex output count

Discussion

This peforms the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvaddD

Adds complex vectors `A` and `B` and leaves the result in complex vector `C`; double precision.

```
void vDSP_zvaddD (DSPDoubleSplitComplex * input1,
vDSP_Stride stride1,
DSPDoubleSplitComplex * input2,
vDSP_Stride stride2,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion

This peforms the operation

$$C_{nK} = A_{nI} + B_{nJ} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvcma

Multiplies complex vector `B` by the complex conjugates of complex vector `A`, adds the products to complex vector `C`, and stores the results in complex vector `D`; single precision.

```
void vDSP_zvcma (const DSPSplitComplex * input1,
vDSP_Stride stride1,
const DSPSplitComplex * input2,
vDSP_Stride stride2,
DSPSplitComplex * input3,
vDSP_Stride stride3,
DSPSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This peforms the operation

$$D_{nL} = A^{*}_{nI} B_{nJ} + C_{nK} \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zvcmaD

Multiplies complex vector `B` by the complex conjugates of complex vector `A`, adds the products to complex vector `C`, and stores the results in complex vector `D`; double precision.

```
void vDSP_zvcmaD (DSPDoubleSplitComplex * input1,
vDSP_Stride stride1,
DSPDoubleSplitComplex * input2,
vDSP_Stride stride2,
DSPDoubleSplitComplex * input3,
vDSP_Stride stride3,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length size);
```

Discussion
This peforms the operation

$$D_{nL} = A^{*}_{nI} B_{nJ} + C_{nK} \qquad n = \{0, N\text{-}1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

vDSP_zvcmul

Complex vector conjugate and multiply; single precision.

```
void vDSP_zvcmul (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Single-precision complex input vector

I
: Stride for `A`

B
: Single-precision complex input vector

J
: Stride for `B`

B
: Single-precision complex output vector

K
: Stride for `B`

N
: Count

Discussion

Multiplies vector `B` by the complex conjugates of vector `A` and stores the results in vector `B`.

$$C_{nK} = A^{*}_{nI} B_{nJ}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvcmulD

Complex vector conjugate and multiply; double precision.

```
void vDSP_zvcmulD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
DSPDoubleSplitComplex * B,
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Double-precision complex input vector

I

Stride for A

B

Double-precision complex input vector

J

Stride for B

B

Double-precision complex output vector

K

Stride for B

N

Count

Discussion

Multiplies vector B by the complex conjugates of vector A and stores the results in vector B.

$$C_{nK} = A^*_{nI} B_{nJ}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvmul

Multiplies complex vectors A and B and leaves the result in complex vector C; single precision.

```
void vDSP_zvmul (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N,
int conjugate);
```

Discussion

Pass 1 or -1 for F, for normal or conjugate multiplication, respectively. Results are undefined for other values of F.

$$Re[C_{nK}] = Re[A_{nI}]Re[B_{nJ}] - F(Im[A_{nI}]Im[B_{nJ}])$$

$$Im[C_{nK}] = Re[A_{nI}]Im[B_{nJ}] + F(Im[A_{nI}]Re[B_{nJ}])$$

$$n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvmulD

Multiplies complex vectors A and B and leaves the result in complex vector C; double precision.

```
void vDSP_zvmulD (DSPDoubleSplitComplex * input1,
vDSP_Stride stride1,
DSPDoubleSplitComplex * input2,
vDSP_Stride stride2,
DSPDoubleSplitComplex * result,
vDSP_Stride strideResult,
vDSP_Length size,
int conjugate);
```

Discussion

Pass 1 or -1 for F, for normal or conjugate multiplication, respectively. Results are undefined for other values of F.

$$Re[C_{nK}] = Re[A_{nI}]\,Re[B_{nJ}] - F\,(Im[A_{nI}]\,Im[B_{nJ}])$$

$$Im[C_{nK}] = Re[A_{nI}]\,Im[B_{nJ}] + F\,(Im[A_{nI}]\,Re[B_{nJ}])$$

$$n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

`vDSP.h`

vDSP_zvsub

Subtracts complex vector B from complex vector A and leaves the result in complex vector C; single precision.

```
void vDSP_zvsub (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A
: Input vector

I
: Address stride for A

B
: Input vector

J
: Address stride for B

C
: Output vector

K

Address stride for C

N

Complex element count

Discussion

This peforms the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N-1\}$$

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zvsubD

Subtracts complex vector B from complex vector A and leaves the result in complex vector C; double precision.

```
void vDSP_zvsubD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
DSPDoubleSplitComplex * B,
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
vDSP_Length N);
```

Parameters

A

Input vector

I

Address stride for A

B

Input vector

J

Address stride for B

C

Output vector

K

Address stride for C

N

Complex element count

Discussion

This peforms the operation

$$C_{nK} = B_{nJ} - A_{nI} \qquad n = \{0, N-1\}$$

Availability
Available in Mac OS X v10.4 and later.

Declared In
`vDSP.h`

Document Revision History

This table describes the changes to *vDSP Vector-to-Vector Arithmetic Operations Reference*.

Date	Notes
2009-01-06	Added more detailed descriptions of function parameters.
2007-06-15	This document describes the V API for the vDSP functions that perform arithmetic operations combining the elements of two vectors

Document Revision History

Index

V