

---

# vDSP Vector-To-Scalar Operations Reference

[Performance > Carbon](#)



2009-01-06



Apple Inc.  
© 2009 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Carbon, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

## vDSP Vector-To-Scalar Operations Reference 5

---

|                          |    |
|--------------------------|----|
| Overview                 | 5  |
| Functions by Task        | 5  |
| Calculating Dot Products | 5  |
| Finding Maximums         | 6  |
| Finding Minimums         | 6  |
| Calculating Means        | 6  |
| Summing Vectors          | 7  |
| Functions                | 8  |
| vDSP_dotpr               | 8  |
| vDSP_dotprD              | 8  |
| vDSP_maxmgv              | 9  |
| vDSP_maxmgvD             | 9  |
| vDSP_maxmgvi             | 10 |
| vDSP_maxmgviD            | 11 |
| vDSP_maxv                | 12 |
| vDSP_maxvD               | 12 |
| vDSP_maxvi               | 13 |
| vDSP_maxviD              | 14 |
| vDSP_meamgv              | 15 |
| vDSP_meamgvD             | 15 |
| vDSP_meanv               | 16 |
| vDSP_meanvD              | 17 |
| vDSP_measqv              | 17 |
| vDSP_measqvD             | 18 |
| vDSP_minmgv              | 19 |
| vDSP_minmgvD             | 19 |
| vDSP_minmgvi             | 20 |
| vDSP_minmgviD            | 21 |
| vDSP_minv                | 22 |
| vDSP_minvD               | 22 |
| vDSP_minvi               | 23 |
| vDSP_minviD              | 24 |
| vDSP_mvessq              | 25 |
| vDSP_mvessqD             | 25 |
| vDSP_rmsqv               | 26 |
| vDSP_rmsqvD              | 27 |
| vDSP_sve                 | 27 |
| vDSP_sveD                | 28 |
| vDSP_svemg               | 29 |
| vDSP_svemgD              | 29 |

vDSP\_svesq 30  
vDSP\_svesqD 31  
vDSP\_svs 31  
vDSP\_svsD 32  
vDSP\_zdotpr 33  
vDSP\_zdotprD 33  
vDSP\_zidotpr 34  
vDSP\_zidotprD 34  
vDSP\_zrdotpr 35  
vDSP\_zrdotprD 35

**Document Revision History 37**

---

**Index 39**

---

# vDSP Vector-To-Scalar Operations Reference

---

|                    |                   |
|--------------------|-------------------|
| <b>Framework:</b>  | Accelerate/vecLib |
| <b>Declared in</b> | vDSP.h            |

## Overview

This document describes the C API for the vDSP functions that receive a vector as input and compute scalars as output. Examples of such operations include calculating the dot product of a vector, or finding the minimum or maximum of a vector.

## Functions by Task

### Calculating Dot Products

[vDSP\\_dotpr](#) (page 8)

Computes the dot or scalar product of vectors *A* and *B* and leaves the result in scalar *C*; single precision.

[vDSP\\_dotprD](#) (page 8)

Computes the dot or scalar product of vectors *A* and *B* and leaves the result in scalar *C*; double precision.

[vDSP\\_zdotpr](#) (page 33)

Calculates the complex dot product of complex vectors *signal1* and *signal2* and leaves the result in complex vector *result*; single precision.

[vDSP\\_zdotprD](#) (page 33)

Calculates the complex dot product of complex vectors *signal1* and *signal2* and leaves the result in complex vector *result*; double precision.

[vDSP\\_zidotpr](#) (page 34)

Calculates the conjugate dot product (or inner dot product) of complex vectors *signal1* and *signal2* and leave the result in complex vector *result*; single precision.

[vDSP\\_zidotprD](#) (page 34)

Calculates the conjugate dot product (or inner dot product) of complex vectors *signal1* and *signal2* and leave the result in complex vector *result*; double precision.

[vDSP\\_zrdotpr](#) (page 35)

Calculates the complex dot product of complex vector *A* and real vector *B* and leaves the result in complex vector *C*; single precision.

[vDSP\\_zrdotprD](#) (page 35)

Calculates the complex dot product of complex vector *A* and real vector *B* and leaves the result in complex vector *C*; double precision.

## Finding Maximums

[vDSP\\_maxv](#) (page 12)

Vector maximum value; single precision.

[vDSP\\_maxvD](#) (page 12)

Vector maximum value; double precision.

[vDSP\\_maxvi](#) (page 13)

Vector maximum value with index; single precision.

[vDSP\\_maxviD](#) (page 14)

Vector maximum value with index; double precision.

[vDSP\\_maxmgv](#) (page 9)

Vector maximum magnitude; single precision.

[vDSP\\_maxmgvD](#) (page 9)

Vector maximum magnitude; double precision.

[vDSP\\_maxmgvi](#) (page 10)

Vector maximum magnitude with index; single precision.

[vDSP\\_maxmgviD](#) (page 11)

Vector maximum magnitude with index; double precision.

## Finding Minimums

[vDSP\\_minv](#) (page 22)

Vector minimum value.

[vDSP\\_minvD](#) (page 22)

Vector minimum value; double precision.

[vDSP\\_minvi](#) (page 23)

Vector minimum value with index; single precision.

[vDSP\\_minviD](#) (page 24)

Vector minimum value with index; double precision.

[vDSP\\_minmgv](#) (page 19)

Vector minimum magnitude; single precision.

[vDSP\\_minmgvD](#) (page 19)

Vector minimum magnitude; double precision.

[vDSP\\_minmgvi](#) (page 20)

Vector minimum magnitude with index; single precision.

[vDSP\\_minmgviD](#) (page 21)

Vector minimum magnitude with index; double precision.

## Calculating Means

[vDSP\\_meanv](#) (page 16)

Vector mean value; single precision.

- [vDSP\\_meanvD](#) (page 17)  
Vector mean value; double precision.
- [vDSP\\_meamgv](#) (page 15)  
Vector mean magnitude; single precision.
- [vDSP\\_meamgvD](#) (page 15)  
Vector mean magnitude; double precision.
- [vDSP\\_measqv](#) (page 17)  
Vector mean square value; single precision.
- [vDSP\\_measqvD](#) (page 18)  
Vector mean square value; double precision.
- [vDSP\\_mvessq](#) (page 25)  
Vector mean of signed squares; single precision.
- [vDSP\\_mvessqD](#) (page 25)  
Vector mean of signed squares; double precision.
- [vDSP\\_rmsqv](#) (page 26)  
Vector root-mean-square; single precision.
- [vDSP\\_rmsqvD](#) (page 27)  
Vector root-mean-square; double precision.

## Summing Vectors

- [vDSP\\_sve](#) (page 27)  
Vector sum; single precision.
- [vDSP\\_sveD](#) (page 28)  
Vector sum; double precision.
- [vDSP\\_svemg](#) (page 29)  
Vector sum of magnitudes; single precision.
- [vDSP\\_svemgD](#) (page 29)  
Vector sum of magnitudes; double precision.
- [vDSP\\_svesq](#) (page 30)  
Vector sum of squares; single precision.
- [vDSP\\_svesqD](#) (page 31)  
Vector sum of squares; double precision.
- [vDSP\\_svs](#) (page 31)  
Vector sum of signed squares; single precision.
- [vDSP\\_svsD](#) (page 32)  
Vector sum of signed squares; double precision.

## Functions

### vDSP\_dotpr

Computes the dot or scalar product of vectors A and B and leaves the result in scalar C; single precision.

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

#### Discussion

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} \cdot B_{nJ} \quad n = \{0, N-1\}$$

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h

### vDSP\_dotprD

Computes the dot or scalar product of vectors A and B and leaves the result in scalar C; double precision.

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

#### Discussion

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} \cdot B_{nJ} \quad n = \{0, N-1\}$$

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h



**vDSP\_maxmgv**

Vector maximum magnitude; single precision.

```
void vDSP_maxmgv (const float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters***A*

Single-precision real input vector. If passed a vector with no elements, this function returns a value of 0 in *C*.

*I*

Stride for *A*

*C*

Output scalar

*N*

Count

**Discussion**

This performs the operation

```
*C = 0;
for (n = 0; n < N; ++n)
    if (*C < abs(A[n*I]))
        *C = abs(A[n*I]);
```

Finds the element with the greatest magnitude in vector *A* and copies this value to scalar *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxmgvD**

Vector maximum magnitude; double precision.

```
void vDSP_maxmgvD (const double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Length N);
```

**Parameters***A*

Double-precision real input vector. If passed a vector with no elements, this function returns a value of 0 in *C*.

*I*

Stride for *A*

*C*

Output scalar

*N*

Count

**Discussion**

This performs the operation

```
*C = 0;
for (n = 0; n < N; ++n)
    if (*C < abs(A[n*I]))
        *C = abs(A[n*I]);
```

Finds the element with the greatest magnitude in vector *A* and copies this value to scalar *C*.**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxmgvi**

Vector maximum magnitude with index; single precision.

```
void vDSP_maxmgvi (float * A,
                  vDSP_Stride I,
                  float * C,
                  vDSP_Length * IC,
                  vDSP_Length N);
```

**Parameters***A*Single-precision real input vector. If passed a vector with no elements, this function returns a value of 0 in *\*C*, and the value returned in *IC* is undefined.*I*Stride for *A**C*

Output scalar

*IC*

Output scalar index

*N*

Count

**Discussion**

This performs the operation

```
*C = 0;
for (n = 0; n < N; ++n)
{
    if (*C < abs(A[n*I]))
    {
        *C = abs(A[n*I]);
        *IC = n*I;
    }
}
```

Copies the element with the greatest magnitude from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the maximum magnitude, *IC* contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxmgviD**

Vector maximum magnitude with index; double precision.

```
void vDSP_maxmgviD (double * A,
                   vDSP_Stride I,
                   double * C,
                   vDSP_Length * IC,
                   vDSP_Length N);
```

**Parameters**

*A*

Double-precision real input vector. If passed a vector with no elements, this function returns a value of 0 in *\*C*. The value returned in *IC* is undefined.

*I*

Stride for *A*

*C*

Output scalar

*IC*

Output scalar

*N*

Count

**Discussion**

This performs the operation

```
*C = 0;
for (n = 0; n < N; ++n)
{
    if (*C < abs(A[n*I]))
    {
        *C = abs(A[n*I]);
        *IC = n*I;
    }
}
```

Copies the element with the greatest magnitude from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the maximum magnitude, *IC* contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxv**

Vector maximum value; single precision.

```
void vDSP_maxv (float * A,
               vDSP_Stride I,
               float * C,
               vDSP_Length N);
```

**Parameters**

*A*  
Single-precision real input vector.

*I*  
Stride for A

*C*  
Output scalar

*N*  
Count

**Discussion**

This performs the operation

```
*C = -INFINITY;
for (n = 0; n < N; ++n)
    if (*C < A[n*I])
        *C = A[n*I];
```

Finds the element with the greatest value in vector A and copies this value to scalar C. If A has length of 0, this function returns -INFINITY.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxvD**

Vector maximum value; double precision.

```
void vDSP_maxvD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length N);
```

**Parameters**

*A*  
Double-precision real input vector

*I*  
Stride for A

*C*  
Output scalar

*N*  
Count

**Discussion**

This performs the operation

```
*C = -INFINITY;
for (n = 0; n < N; ++n)
    if (*C < A[n*I])
        *C = A[n*I];
```

Finds the element with the greatest value in vector A and copies this value to scalar C. If A has length of 0, this function returns -INFINITY.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxvi**

Vector maximum value with index; single precision.

```
void vDSP_maxvi (float * A,
                vDSP_Stride I,
                float * C,
                vDSP_Length * IC,
                vDSP_Length N);
```

**Parameters**

*A*  
Single-precision real input vector.

*I*  
Stride for A

*C*  
Output scalar

*IC*  
Output scalar index

*N*  
Count

**Discussion**

This performs the operation

```
*C = -INFINITY;
for (n = 0; n < N; ++n)
{
    if (*C < A[n * I])
```

```

    {
        *C = A[n * I];
        *IC = n * I;
    }
}

```

Copies the element with the greatest value from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the maximum value, *IC* contains the index of the first instance. If *A* is vector with no elements, this function returns a value of -infinity in *C* and *\*IC* is undetermined.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_maxviD**

Vector maximum value with index; double precision.

```

void vDSP_maxviD (double * A,
vDSP_Stride I,
double * C,
vDSP_Length * IC,
vDSP_Length N);

```

**Parameters**

*A*  
Double-precision real input vector

*I*  
Stride for *A*

*C*  
Output scalar

*IC*  
Output scalar

*N*  
Count

**Discussion**

This performs the operation

```

*C = -INFINITY;
for (n = 0; n < N; ++n)
{
    if (*C < A[n * I])
    {
        *C = A[n * I];
        *IC = n * I;
    }
}

```

Copies the element with the greatest value from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the maximum value, *IC* contains the index of the first instance. If *A* is vector with no elements, this function returns a value of -infinity in *C* and *\*IC* is undetermined.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_meamgv**

Vector mean magnitude; single precision.

```
void vDSP_meamgv (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters**

|          |                                    |
|----------|------------------------------------|
| <i>A</i> | Single-precision real input vector |
| <i>I</i> | Stride for <i>A</i>                |
| <i>C</i> | Output scalar                      |
| <i>N</i> | Count                              |

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} |A_{ni}|$$

Finds the mean of the magnitudes of elements of vector *A* and stores this value in scalar *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_meamgvD**

Vector mean magnitude; double precision.

```
void vDSP_meamgvD (double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Length N);
```

**Parameters**

|          |                                    |
|----------|------------------------------------|
| <i>A</i> | Double-precision real input vector |
| <i>I</i> | Stride for A                       |
| <i>C</i> | Output scalar                      |
| <i>N</i> | Count                              |

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} |A_{ni}|$$

Finds the mean of the magnitudes of elements of vector *A* and stores this value in scalar *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_meanv**

Vector mean value; single precision.

```
void vDSP_meanv (float * A,
                vDSP_Stride I,
                float * C,
                vDSP_Length N);
```

**Parameters**

|          |  |
|----------|--|
| <i>A</i> | Single-precision real input vector. If passed an array of length 0, this function returns a NaN. |
| <i>I</i> | Stride for A   |
| <i>C</i> | Output scalar  |
| <i>N</i> | Count  |

**Discussion**

This performs the operation



$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni}$$

Finds the mean value of the elements of vector A and stores this value in scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_meanvD**

Vector mean value; double precision.

```
void vDSP_meanvD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length N);
```

**Parameters**

*A*  
Double-precision real input vector

*I*  
Stride for A

*C*  
Output scalar

*N*  
Count

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni}$$

Finds the mean value of the elements of vector A and stores this value in scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_measqv**

Vector mean square value; single precision.

```
void vDSP_measqv (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters**

|          |  |
|----------|--|
| <i>A</i> | Single-precision real input vector. If passed an array of length 0, this function returns a NaN. |
| <i>I</i> | Stride for A   |
| <i>C</i> | Output scalar  |
| <i>N</i> | Count  |

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni}^2$$

Finds the mean value of the squares of the elements of vector A and stores this value in scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_measqvD**

Vector mean square value; double precision.

```
void vDSP_measqvD (double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Length N);
```

**Parameters**

|          |                                    |
|----------|------------------------------------|
| <i>A</i> | Double-precision real input vector |
| <i>I</i> | Stride for A                       |
| <i>C</i> | Output scalar                      |
| <i>N</i> | Count                              |

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni}^2$$

Finds the mean value of the squares of the elements of vector A and stores this value in scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minmgv**

Vector minimum magnitude; single precision.

```
void vDSP_minmgv (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters**

*A*

Single-precision real input vector. If passed an array of length 0, this function returns +INF.

*I*

Stride for A

*C*

Output scalar

*N*

Count

**Discussion**

This performs the operation

$$c = |A_0| \quad \text{If } c > |A_{ni}| \quad \text{then } c = |A_{ni}| \quad n = \{1, N-1\}$$

Finds the element with the least magnitude in vector A and copies this value to scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minmgvD**

Vector minimum magnitude; double precision.

```
void vDSP_minmgvD (double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Length N);
```

**Parameters**

|          |                                    |
|----------|------------------------------------|
| <i>A</i> | Double-precision real input vector |
| <i>I</i> | Stride for A                       |
| <i>C</i> | Output scalar                      |
| <i>N</i> | Count                              |

**Discussion**

This performs the operation

$$c = |A_0| \quad \text{If } c > |A_{ni}| \quad \text{then } c = |A_{ni}| \quad n = \{1, N-1\}$$

Finds the element with the least magnitude in vector A and copies this value to scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minmgvi**

Vector minimum magnitude with index; single precision.

```
void vDSP_minmgvi (float * A,
                  vDSP_Stride I,
                  float * C,
                  vDSP_Length * IC,
                  vDSP_Length N);
```

**Parameters**

|           |   |
|-----------|---|
| <i>A</i>  | Single-precision real input vector. If passed a zero vector, this function returns a value of +INF with an indeterminate index. |
| <i>I</i>  | Stride for A  |
| <i>C</i>  | Output scalar   |
| <i>IC</i> | Output scalar index   |
| <i>N</i>  | Count   |

**Discussion**

This performs the operation

$$c = |A_0| \quad \text{If } c > |A_{ni}| \quad \text{then} \quad c = |A_{ni}| \quad n = \{1, N-1\}$$

$$d = 0 \quad \quad \quad d = ni$$

Copies the element with the least magnitude from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the least magnitude, *IC* contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minmgviD**

Vector minimum magnitude with index; double precision.

```
void vDSP_minmgviD (double * A,
vDSP_Stride I,
double * C,
vDSP_Length * IC,
vDSP_Length N);
```

**Parameters**

|           |                                    |
|-----------|------------------------------------|
| <i>A</i>  | Double-precision real input vector |
| <i>I</i>  | Stride for <i>A</i>                |
| <i>C</i>  | Output scalar                      |
| <i>IC</i> | Output scalar                      |
| <i>N</i>  | Count                              |

**Discussion**

This performs the operation

$$c = |A_0| \quad \text{If } c > |A_{ni}| \quad \text{then} \quad c = |A_{ni}| \quad n = \{1, N-1\}$$

$$d = 0 \quad \quad \quad d = ni$$

Copies the element with the least magnitude from real vector *A* to real scalar *C*, and writes its zero-based index to integer scalar *IC*. The index is the actual array index, not the pre-stride index. If vector *A* contains more than one instance of the least magnitude, *IC* contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minv**

Vector minimum value.

```
void vDSP_minv (float * A,
               vDSP_Stride I,
               float * C,
               vDSP_Length N);
```

**Parameters**

*A*

Single-precision real input vector. If passed an array of length 0, this function returns +INF.

*I*

Stride for A

*C*

Output scalar

*N*

Count

**Discussion**

This performs the operation

$$c = A_0 \quad \text{If } c > A_{ni} \quad \text{then } c = A_{ni} \quad n = \{1, N-1\}$$

Finds the element with the least value in vector A and copies this value to scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minvD**

Vector minimum value; double precision.

```
void vDSP_minvD (double * A,
                vDSP_Stride I,
                double * C,
                vDSP_Length N);
```

**Parameters**

*A*

Double-precision real input vector

*I*  
Stride for A

*C*  
Output scalar

*N*  
Count

**Discussion**

This performs the operation

$$c = A_0 \quad \text{If } c > A_{ni} \quad \text{then } c = A_{ni} \quad n = \{1, N-1\}$$

Finds the element with the least value in vector A and copies this value to scalar C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minvi**

Vector minimum value with index; single precision.

```
void vDSP_minvi (float * A,
                vDSP_Stride I,
                float * C,
                vDSP_Length * IC,
                vDSP_Length N);
```

**Parameters**

*A*  
Single-precision real input vector. If passed a zero vector, this function returns a value of +INF with an indeterminate index.

*I*  
Stride for A

*C*  
Output scalar

*IC*  
Output scalar index

*N*  
Count

**Discussion**

This performs the operation

$$c = A_0 \quad \text{If } c > A_{ni} \quad \text{then } c = A_{ni} \quad n = \{1, N-1\}$$

$$d = 0 \quad \text{then } d = ni$$

Copies the element with the least value from real vector  $A$  to real scalar  $C$ , and writes its zero-based index to integer scalar  $IC$ . The index is the actual array index, not the pre-stride index. If vector  $A$  contains more than one instance of the least value,  $IC$  contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_minviD**

Vector minimum value with index; double precision.

```
void vDSP_minviD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length * IC,
                 vDSP_Length N);
```

**Parameters**

$A$   
Double-precision real input vector

$I$   
Stride for  $A$

$C$   
Output scalar

$IC$   
Output scalar

$N$   
Count

**Discussion**

This performs the operation

$$\begin{array}{ll} c = A_0 & \text{If } c > A_{ni} \text{ then } c = A_{ni} \\ d = 0 & d = ni \end{array} \quad n = \{1, N-1\}$$

Copies the element with the least value from real vector  $A$  to real scalar  $C$ , and writes its zero-based index to integer scalar  $IC$ . The index is the actual array index, not the pre-stride index. If vector  $A$  contains more than one instance of the least value,  $IC$  contains the index of the first instance.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h



**vDSP\_mvessq**

Vector mean of signed squares; single precision.

```
void vDSP_mvessq (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters**

|          |                                     |
|----------|-------------------------------------|
| <i>A</i> | Single-precision real input vector. |
| <i>I</i> | Stride for <i>A</i>                 |
| <i>C</i> | Output scalar                       |
| <i>N</i> | Count                               |

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni} \cdot |A_{ni}|$$

Finds the mean value of the signed squares of the elements of vector *A* and stores this value in *\*C*. If *A* is an array of length 0, this function returns a NaN. .

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_mvessqD**

Vector mean of signed squares; double precision.

```
void vDSP_mvessqD (double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Length N);
```

**Parameters**

|          |                                    |
|----------|------------------------------------|
| <i>A</i> | Double-precision real input vector |
| <i>I</i> | Stride for <i>A</i>                |
| <i>C</i> | Output scalar                      |

$N$ 

Count

**Discussion**

This performs the operation

$$C = \frac{1}{N} \sum_{n=0}^{N-1} A_{ni} \cdot |A_{ni}|$$

Finds the mean value of the signed squares of the elements of vector A and stores this value in \*C. If A is an array of length 0, this function returns a NaN.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_rmsqv**

Vector root-mean-square; single precision.

```
void vDSP_rmsqv (float * A,
                vDSP_Stride I,
                float * C,
                vDSP_Length N);
```

**Parameters** $A$ 

Single-precision real input vector. If passed an array of length 0, this function returns a NaN.

 $I$ 

Stride for A

 $C$ 

Single-precision real output scalar

 $N$ 

Count

**Discussion**

This performs the operation

$$C = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} A_{ni}^2}$$

Calculates the root mean square of the elements of A and stores the result in \*C

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_rmsqvD**

Vector root-mean-square; double precision.

```
void vDSP_rmsqvD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length N);
```

**Parameters**

*A*  
Double-precision real input vector

*I*  
Stride for *A*

*C*  
Double-precision real output scalar

*N*  
Count

**Discussion**

This performs the operation

$$C = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} A_{nI}^2}$$

Calculates the root mean square of the elements of *A* and stores the result in *\*C***Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_sve**

Vector sum; single precision.

```
void vDSP_sve (float * A,
              vDSP_Stride I,
              float * C,
              vDSP_Length N);
```

**Parameters**

*A*  
Single-precision real input vector.

*I*  
Stride for A

*C*  
Single-precision real output scalar

*N*  
Count

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}$$

Writes the sum of the elements of A into \*C. If A is a vector with zero elements, this function returns 0 in \*C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_sveD**

Vector sum; double precision.

```
void vDSP_sveD (double * A,
               vDSP_Stride I,
               double * C,
               vDSP_Length N);
```

**Parameters**

*A*  
Double-precision real input vector

*I*  
Stride for A

*C*  
Double-precision real output scalar

*N*  
Count

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}$$

Writes the sum of the elements of A into \*C. If A is a vector with zero elements, this function returns 0 in \*C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svemg**

Vector sum of magnitudes; single precision.

```
void vDSP_svemg (float * A,
                vDSP_Stride I,
                float * C,
                vDSP_Length N);
```

**Parameters**

*A*

Single-precision real input scalar. If passed an value of 0, this function returns 0.

*I*

Stride for *A*

*C*

Single-precision real output scalar

*N*

Count

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} |A_{nI}|$$

Writes the sum of the magnitudes of the elements of *A* into *C*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svemgD**

Vector sum of magnitudes; double precision.

```
void vDSP_svemgD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length N);
```

**Parameters**

|          |                                     |
|----------|-------------------------------------|
| <i>A</i> | Double-precision real input scalar  |
| <i>I</i> | Stride for A                        |
| <i>C</i> | Double-precision real output scalar |
| <i>N</i> | Count                               |

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} |A_n|$$

Writes the sum of the magnitudes of the elements of A into C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svesq**

Vector sum of squares; single precision.

```
void vDSP_svesq (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Length N);
```

**Parameters**

|          |   |
|----------|---|
| <i>A</i> | Single-precision real input scalar. If passed an value of 0, this function returns 0. |
| <i>I</i> | Stride for A  |
| <i>C</i> | Single-precision real output scalar   |
| <i>N</i> | Count   |

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}^2$$

Writes the sum of the squares of the elements of A into C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svesqD**

Vector sum of squares; double precision.

```
void vDSP_svesqD (double * A,
                 vDSP_Stride I,
                 double * C,
                 vDSP_Length N);
```

**Parameters**

*A*  
Double-precision real input scalar

*I*  
Stride for A

*C*  
Double-precision real output scalar

*N*  
Count

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}^2$$

Writes the sum of the squares of the elements of A into C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svs**

Vector sum of signed squares; single precision.

```
void vDSP_svs (float * A,
              vDSP_Stride I,
              float * C,
              vDSP_Length N);
```

**Parameters**

|          |   |
|----------|---|
| <i>A</i> | Single-precision real input scalar. If passed an value of 0, this function returns 0. |
| <i>I</i> | Stride for A  |
| <i>C</i> | Single-precision real output scalar   |
| <i>N</i> | Count   |

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} \cdot |A_{nI}|$$

Writes the sum of the signed squares of the elements of A into C.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_svsD**

Vector sum of signed squares; double precision.

```
void vDSP_svsD (double * A,
               vDSP_Stride I,
               double * C,
               vDSP_Length N);
```

**Parameters**

|          |                                     |
|----------|-------------------------------------|
| <i>A</i> | Double-precision real input scalar  |
| <i>I</i> | Stride for A                        |
| <i>C</i> | Double-precision real output scalar |
| <i>N</i> | Count                               |

**Discussion**

This performs the operation



$$C = \sum_{n=0}^{N-1} A_{nI} \cdot |A_{nI}|$$

Writes the sum of the signed squares of the elements of A into C.

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h

### vDSP\_zdotpr

Calculates the complex dot product of complex vectors signal1 and signal2 and leaves the result in complex vector result; single precision.

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

#### Discussion

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} B_{nJ}$$

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

vDSP.h

### vDSP\_zdotprD

Calculates the complex dot product of complex vectors signal1 and signal2 and leaves the result in complex vector result; double precision.

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

#### Discussion

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} B_{nJ}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zidotpr**

Calculates the conjugate dot product (or inner dot product) of complex vectors `signal1` and `signal2` and leave the result in complex vector `result`; single precision.

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

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}^* B_{nJ}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zidotprD**

Calculates the conjugate dot product (or inner dot product) of complex vectors `signal1` and `signal2` and leave the result in complex vector `result`; double precision.

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

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI}^* B_{nJ}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zrdotpr**

Calculates the complex dot product of complex vector A and real vector B and leaves the result in complex vector C; single precision.

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

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} B_{nJ}$$

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

vDSP.h

**vDSP\_zrdotprD**

Calculates the complex dot product of complex vector A and real vector B and leaves the result in complex vector C; double precision.

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

**Discussion**

This performs the operation

$$C = \sum_{n=0}^{N-1} A_{nI} B_{nJ}$$

**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-Scalar Operations Reference*.

| Date       | Notes   |
|------------|---|
| 2009-01-06 | Minor fixes to pseudocode and function parameter descriptions.        |
| 2008-11-19 | Changed function parameter names to match formulae. Added pseudocode. |

## REVISION HISTORY

### Document Revision History

# Index

---

## V

---

- vDSP\_dotpr function 8
- vDSP\_dotprD function 8
- vDSP\_maxmgv function 9
- vDSP\_maxmgvD function 9
- vDSP\_maxmgvi function 10
- vDSP\_maxmgviD function 11
- vDSP\_maxv function 12
- vDSP\_maxvD function 12
- vDSP\_maxvi function 13
- vDSP\_maxviD function 14
- vDSP\_meamgv function 15
- vDSP\_meamgvD function 15
- vDSP\_meanv function 16
- vDSP\_meanvD function 17
- vDSP\_measqv function 17
- vDSP\_measqvD function 18
- vDSP\_minmgv function 19
- vDSP\_minmgvD function 19
- vDSP\_minmgvi function 20
- vDSP\_minmgviD function 21
- vDSP\_minv function 22
- vDSP\_minvD function 22
- vDSP\_minvi function 23
- vDSP\_minviD function 24
- vDSP\_mvessq function 25
- vDSP\_mvessqD function 25
- vDSP\_rmsqv function 26
- vDSP\_rmsqvD function 27
- vDSP\_sve function 27
- vDSP\_sveD function 28
- vDSP\_svemg function 29
- vDSP\_svemgD function 29
- vDSP\_svesq function 30
- vDSP\_svesqD function 31
- vDSP\_svs function 31
- vDSP\_svsD function 32
- vDSP\_zdotpr function 33
- vDSP\_zdotprD function 33
- vDSP\_zidotpr function 34
- vDSP\_zidotprD function 34
- vDSP\_zrdotpr function 35
- vDSP\_zrdotprD function 35