
vDSP Matrix 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 Matrix Operations Reference 5

Overview	5
Functions by Task	5
Multiplying Real Matrices	5
Transposing a Matrix	5
Copying a Submatrix	5
Multiplying Complex Matrices	6
Functions	6
vDSP_mmov	6
vDSP_mmovD	7
vDSP_mmul	8
vDSP_mmulD	9
vDSP_mtrans	9
vDSP_mtransD	10
vDSP_zmma	10
vDSP_zmmaD	11
vDSP_zmms	12
vDSP_zmmsD	13
vDSP_zmmul	14
vDSP_zmmulD	14
vDSP_zmsm	15
vDSP_zmsmD	16

Document Revision History 19

Index 21

vDSP Matrix Operations Reference

Framework:	Accelerate/vecLib
Declared in	vDSP.h

Overview

This document describes the C API for the matrix arithmetic operations available in vDSP. It provides functionality for multiplying and transposing real or complex matrices.

Functions by Task

Multiplying Real Matrices

[vDSP_mmu1](#) (page 8)

Performs an out-of-place multiplication of M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; single precision.

[vDSP_mmu1D](#) (page 9)

Performs an out-of-place multiplication of M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; double precision.

Transposing a Matrix

[vDSP_mtrans](#) (page 9)

Creates a transposed matrix C from a source matrix A; single precision.

[vDSP_mtransD](#) (page 10)

Creates a transposed matrix C from a source matrix A; double precision.

Copying a Submatrix

[vDSP_mmov](#) (page 6)

The contents of a submatrix are copied to another submatrix.

[vDSP_mmovD](#) (page 7)

The contents of a submatrix are copied to another submatrix.

Multiplying Complex Matrices

[vDSP_zmma](#) (page 10)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , adds the product to M -by- N matrix C , and stores the result in M -by- N matrix D ; single precision.

[vDSP_zmmaD](#) (page 11)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , adds the product to M -by- N matrix C , and stores the result in M -by- N matrix D ; double precision.

[vDSP_zmms](#) (page 12)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , subtracts M -by- N matrix C from the product, and stores the result in M -by- N matrix D ; single precision.

[vDSP_zmmsD](#) (page 13)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , subtracts M -by- N matrix C from the product, and stores the result in M -by- N matrix D ; double precision.

[vDSP_zmmu1](#) (page 14)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B and stores the results in an M -by- N matrix C ; single precision.

[vDSP_zmmu1D](#) (page 14)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B and stores the results in an M -by- N matrix C ; double precision.

[vDSP_zmsm](#) (page 15)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , subtracts the product from M -by- P matrix C , and stores the result in M -by- P matrix D ; single precision.

[vDSP_zmsmD](#) (page 16)

Performs an out-of-place complex multiplication of an M -by- P matrix A by a P -by- N matrix B , subtracts the product from M -by- P matrix C , and stores the result in M -by- P matrix D ; double precision.

Functions

vDSP_mmov

The contents of a submatrix are copied to another submatrix.

```
void
vDSP_mmov (float * A,
           float * C,
           vDSP_Length NC,
           vDSP_Length NR,
           vDSP_Length TCA,
           vDSP_Length TCC);
```

Parameters

- A
Single-precision real input submatrix
- C
Single-precision real output submatrix

<i>NC</i>	Number of columns in A and C
<i>NR</i>	Number of rows in A and C
<i>TCA</i>	Number of columns in the matrix of which A is a submatrix
<i>TCC</i>	Number of columns in the matrix of which C is a submatrix

Discussion

The matrices are assumed to be stored in row-major order. Thus elements $A[i][j]$ and $A[i][j+1]$ are adjacent. Elements $A[i][j]$ and $A[i+1][j]$ are TCA elements apart.

This function may be used to move a subarray beginning at any point in a larger embedding array by passing for A the address of the first element of the subarray. For example, to move a subarray starting at $A[3][4]$, pass $\&A[3][4]$. Similarly, the address of the first destination element is passed for C

NC may equal *TCA*, and it may equal *TCC*. To copy all of an array to all of another array, pass the number of rows in *NR* and the number of columns in *NC*, *TCA*, and *TCC*.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_mmovD

The contents of a submatrix are copied to another submatrix.

```
void vDSP_mmovD (double * A,
                double * C,
                vDSP_Length NC,
                vDSP_Length NR,
                vDSP_Length TCA,
                vDSP_Length TCC);
```

Parameters

<i>A</i>	Double-precision real input submatrix
<i>C</i>	Double-precision real output submatrix
<i>NC</i>	Number of columns in A and C
<i>NR</i>	Number of rows in A and C
<i>TCA</i>	Number of columns in the matrix of which A is a submatrix
<i>TCC</i>	Number of columns in the matrix of which C is a submatrix

Discussion

The matrices are assumed to be stored in row-major order. Thus elements $A[i][j]$ and $A[i][j+1]$ are adjacent. Elements $A[i][j]$ and $A[i+1][j]$ are TCA elements apart.

This function may be used to move a subarray beginning at any point in a larger embedding array by passing for A the address of the first element of the subarray. For example, to move a subarray starting at $A[3][4]$, pass $\&A[3][4]$. Similarly, the address of the first destination element is passed for C.

NC may equal TCA, and it may equal TCC. To copy all of an array to all of another array, pass the number of rows in NR and the number of columns in NC, TCA, and TCC.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_mmul

Performs an out-of-place multiplication of M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; single precision.

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

Discussion

This performs the operation

$$C_{(mN+n)K} = \sum_{p=0}^{P-1} A_{(mP+p)I} \cdot B_{(pN+n)J} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameters A and B are the matrixes to be multiplied. I is an address stride through A. J is an address stride through B.

Parameter C is the result matrix. K is an address stride through C.

Parameter M is the row count for both A and C. Parameter N is the column count for both B and C. Parameter P is the column count for A and the row count for B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_mmulD

Performs an out-of-place multiplication of M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; double precision.

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

Discussion

This performs the operation

$$C_{(mN+n)K} = \sum_{p=0}^{P-1} A_{(mP+p)I} \cdot B_{(pN+n)J} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameters A and B are the matrixes to be multiplied. I is an address stride through A. J is an address stride through B.

Parameter C is the result matrix. K is an address stride through C.

Parameter M is the row count for both A and C. Parameter N is the column count for both B and C. Parameter P is the column count for A and the row count for B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_mtrans

Creates a transposed matrix C from a source matrix A; single precision.

```
void vDSP_mtrans (float * A,
                 vDSP_Stride I,
                 float * C,
                 vDSP_Stride K,
                 vDSP_Length M,
                 vDSP_Length N);
```

Discussion

This performs the operation

$$C_{(mN+n)K} = A_{(nM+m)I} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameter A is the source matrix. I is an address stride through the source matrix.

Parameter C is the resulting transposed matrix. K is an address stride through the result matrix.

Parameter M is the number of rows in C (and the number of columns in A).

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_mtransD

Creates a transposed matrix C from a source matrix A; double precision.

```
void vDSP_mtransD (double * A,
                  vDSP_Stride I,
                  double * C,
                  vDSP_Stride K,
                  vDSP_Length M,
                  vDSP_Length N);
```

Discussion

This performs the operation

$$C_{(mN+n)K} = A_{(nM+m)I} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameter A is the source matrix. I is an address stride through the source matrix.

Parameter C is the resulting transposed matrix. K is an address stride through the result matrix.

Parameter M is the number of rows in C (and the number of columns in A).

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmma

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, adds the product to M-by-N matrix C, and stores the result in M-by-N matrix D; single precision.

```
void vDSP_zmma (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
DSPSplitComplex * D,
vDSP_Stride L,
vDSP_Length M,
vDSP_Length N,
vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = C_{(rN+q)K} + \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J}$$

$$0 \leq r < M, \quad 0 \leq q < N$$

Parameters A and C are the matrixes to be multiplied, and C the matrix to be added. I is an address stride through A. J is an address stride through B. K is an address stride through C. L is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmmaD

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, adds the product to M-by-N matrix C, and stores the result in M-by-N matrix D; double precision.

```
void vDSP_zmmaD (DSPDoubleSplitComplex * A,
vDSP_Stride I,
DSPDoubleSplitComplex * B,
vDSP_Stride J,
DSPDoubleSplitComplex * C,
vDSP_Stride K,
DSPDoubleSplitComplex * D,
vDSP_Stride L,
vDSP_Length M,
vDSP_Length N,
vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = C_{(rN+q)K} + \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J}$$

$$0 \leq r < M, \quad 0 \leq q < N$$

Parameters A and C are the matrixes to be multiplied, and C the matrix to be added. I is an address stride through A. J is an address stride through B. K is an address stride through C. L is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmms

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, subtracts M-by-N matrix C from the product, and stores the result in M-by-N matrix D; single precision.

```
void vDSP_zmms (DSPSplitComplex * A,
               vDSP_Stride I,
               DSPSplitComplex * B,
               vDSP_Stride J,
               DSPSplitComplex * C,
               vDSP_Stride K,
               DSPSplitComplex * D,
               vDSP_Stride L,
               vDSP_Length M,
               vDSP_Length N,
               vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J} - C_{(rN+q)K}$$

$$0 \leq r < M, \quad 0 \leq q < N$$

Parameters A and B are the matrixes to be multiplied, and C the matrix to be subtracted. I is an address stride through A. J is an address stride through B. K is an address stride through C. L is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmmsD

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, subtracts M-by-N matrix C from the product, and stores the result in M-by-N matrix D; double precision.

```
void vDSP_zmmsD (DSPDoubleSplitComplex * A,
                vDSP_Stride I,
                DSPDoubleSplitComplex * B,
                vDSP_Stride J,
                DSPDoubleSplitComplex * C,
                vDSP_Stride K,
                DSPDoubleSplitComplex * D,
                vDSP_Stride L,
                vDSP_Length M,
                vDSP_Length N,
                vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J} - C_{(rN+q)K}$$

$$0 \leq r < M, \quad 0 \leq q < N$$

Parameters A and B are the matrixes to be multiplied, and C the matrix to be subtracted. I is an address stride through A. J is an address stride through B. K is an address stride through C. L is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmmul

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; single precision.

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

Discussion

This performs the operation

$$C_{(mN+n)K} = \sum_{p=0}^{P-1} A_{(mP+p)I} \cdot B_{(pN+n)J} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameters A and B are the matrixes to be multiplied. I is an address stride through A. J is an address stride through B.

Parameter C is the result matrix. K is an address stride through C.

Parameter M is the row count for both A and C. Parameter N is the column count for both B and C. Parameter P is the column count for A and the row count for B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmmulD

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C; double precision.

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

Discussion

This performs the operation

$$C_{(mN+n)K} = \sum_{p=0}^{P-1} A_{(mP+p)I} \cdot B_{(pN+n)J} \quad n = \{0, N-1\} \text{ and } m = \{0, M-1\}$$

Parameters A and B are the matrixes to be multiplied. I is an address stride through A. J is an address stride through B.

Parameter C is the result matrix. K is an address stride through C.

Parameter M is the row count for both A and C. Parameter N is the column count for both B and C. Parameter P is the column count for A and the row count for B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmsm

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, subtracts the product from M-by-P matrix C, and stores the result in M-by-P matrix D; single precision.

```
void vDSP_zmsm (DSPSplitComplex * A,
vDSP_Stride I,
DSPSplitComplex * B,
vDSP_Stride J,
DSPSplitComplex * C,
vDSP_Stride K,
DSPSplitComplex * D,
vDSP_Stride L,
vDSP_Length M,
vDSP_Length N,
vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = C_{(rN+q)K} - \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J}$$

Parameters A and B are the matrixes to be multiplied, and C is the matrix from which the product is to be subtracted. aStride is an address stride through A. bStride is an address stride through B. cStride is an address stride through C. dStride is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

vDSP_zmsmD

Performs an out-of-place complex multiplication of an M-by-P matrix A by a P-by-N matrix B, subtracts the product from M-by-P matrix C, and stores the result in M-by-P matrix D; double precision.

```
void vDSP_zmsmD (DSPDoubleSplitComplex * A,
                vDSP_Stride I,
                DSPDoubleSplitComplex * B,
                vDSP_Stride J,
                DSPDoubleSplitComplex * C,
                vDSP_Stride K,
                DSPDoubleSplitComplex * D,
                vDSP_Stride L,
                vDSP_Length M,
                vDSP_Length N,
                vDSP_Length P);
```

Discussion

This performs the operation

$$D_{(rN+q)L} = C_{(rN+q)K} - \sum_{p=0}^{P-1} A_{(rP+p)I} B_{(pN+q)J}$$

Parameters A and B are the matrixes to be multiplied, and parameter C is the matrix from which the product is to be subtracted. aStride is an address stride through A. bStride is an address stride through B. cStride is an address stride through C. dStride is an address stride through D.

Parameter D is the result matrix.

Parameter M is the row count for A, C and D. Parameter N is the column count of B, C, and D. Parameter P is the column count of A and the row count of B.

Availability

Available in Mac OS X v10.4 and later.

Declared In

vDSP.h

Document Revision History

This table describes the changes to *vDSP Matrix Operations Reference*.

Date	Notes
2009-01-06	Includes vDSP_mmov and vDSP_mmovD.
2007-06-15	New document that describes the C API for the vDSP functionality for matrix operations.

REVISION HISTORY

Document Revision History

Index

V

- `vDSP_mmov` [function 6](#)
- `vDSP_mmovD` [function 7](#)
- `vDSP_mmul` [function 8](#)
- `vDSP_mmulD` [function 9](#)
- `vDSP_mtrans` [function 9](#)
- `vDSP_mtransD` [function 10](#)
- `vDSP_zmma` [function 10](#)
- `vDSP_zmmaD` [function 11](#)
- `vDSP_zmms` [function 12](#)
- `vDSP_zmmsD` [function 13](#)
- `vDSP_zmmul` [function 14](#)
- `vDSP_zmmulD` [function 14](#)
- `vDSP_zmsm` [function 15](#)
- `vDSP_zmsmD` [function 16](#)