vDSP Matrix 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 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_mmul` (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_mmulD` (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_zmmul (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_zmmulD (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

NC

Number of columns in `A` and `C`

NR

Number of rows in `A` and `C`

TCA

Number of columns in the matrix of which `A` is a submatrix

TCC

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

A

Double-precision real input submatrix

C

Double-precision real output submatrix

NC

Number of columns in `A` and `C`

NR

Number of rows in `A` and `C`

TCA

Number of columns in the matrix of which `A` is a submatrix

TCC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

Document Revision History

Index

V