Math Vector Library Machine Learning Extension Reference Manual (C/C++) DD-00004-010

Jan Adelsbach

January 26, 2024

Contents

1 About this Guide

1.1 Legal Information

1.2 Feedback and Contact

For feedback on this document, please use the following email address:
techpubs@adelsbach-research.eu

Please include the page number or a link to the page.

For general contact details, please visit https://adelsbach-research.eu/contact.

1.3 Introduction

This manual describes the *Application Programming Interface* (API) of the *Math Vector Library Machine Learning Extension* for the C and C++ programming language families.

1.4 Audience for This Guide

The audience of this guide is assumed to be C or C++ programmers who understand the basic concepts of at least one of the aforementioned programming languages.

Familiarity with pointer based arrays in C/C++ is strongly recommended.

1.5 How to Use This Guide

This guide first describes some general programming details of the library and then documents each function individually.

The documentation for each function applies both the *single* and *double* precision versions. The former can be differentiated by a suffix letter `f`.

1.6 Conventions Used in This Guide

x
: Normal math typesetting represents a normal variable.

$\mathbf{x}$
: Bold math typesetting represents a vector.

`Mono`
: Monospace typesetting represents C function names, variables or data types.

2 Overview

2.1 Introduction

The *Math Vector Library Machine Learning Extension* is an extension to the *Math Vector Library* which provides high-performance function library with vectorized versions of standard mathematical functions. The *Machine Learning* extension builds upon the infrastructure of the *Math Vector Library* to provide performance tuned primitive functions for machine learning applications.

The functions can operate both on dense and strided vectors, the latter can be supplied individually for result and operand vectors. Stride only executes the function on every n-th element leaving the elements in between untouched.

This manual describes the *Application Programming Interface* (API) of the machine learning extension functions.

2.2 Thread Safety

All routines in the library are completely thread-safe, as long as the data supplied in arguments is exclusive to the current thread.

2.3 SIMD/SPMD Unit Usage

This library makes excessive use of *Single Instruction Multiple Data* (SIMD) or *Single program Multiple Data* (SPMD) style extensions of the respective processor platform. It thereby abides by the standard system calling conventions when utilizing such.

2.4 Performance Characteristics

All subroutines in this library have a performance characteristic of $O(n)$. The routines may have different execution profiles depending upon the arguments supplied.

3 Utility

3.1 `mvecmlver` - Version query

```
#include <mvecml.h>

void mvecmlver(int *major, int *minor);
```

Queries the version of the library and stores the *major* and *minor* version numbers in the respective arguments.

3.1.1 Parameters

`MAJOR` - **INTEGER** *EXIT:* The major version number of the library.

`MINOR` - **INTEGER** *EXIT:* The minor version number of the library

4 Activation functions

4.1 `vmlsigm` - Sigmoid activation

```
#include <mvecml.h>

void vmlsigm (int n, double *y, int incy, const double *x, int incx);
void vmlsigmf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the Sigmoid activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{1+e^{-\mathbf{x}}}$$

4.1.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.2 vmlsigmd - Sigmoid activation (first derivative)

```
#include <mvecml.h>

void vmlsigmd (int n, double *y, int incy, const double *x, int incx);
void vmlsigmdf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the Sigmoid activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{1+e^{-\mathbf{x}}}\left(1-\frac{1}{1+e^{-\mathbf{x}}}\right)$$

4.2.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.3 vmlrelu - ReLU activation

```
#include <mvecml.h>

void vmlrelu (int n, double *y, int incy, const double *x, int incx);
void vmlreluf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the ReLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \begin{cases} 0 & \text{where } \mathbf{x} \le 0 \\ x & \text{where } \mathbf{x} > 0 \end{cases}$$

4.3.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \ge 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.4 vmlrelud - ReLU activation (first derivative)

```
#include <mvecml.h>

void vmlrelud (int n, double *y, int incy, const double *x, int incx);
void vmlreludf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the ReLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \begin{cases} 0 & \text{where } \mathbf{x} < 0 \vee \mathbf{x} = 0 \\ 1 & \text{where } \mathbf{x} > 0 \end{cases}$$

The case where $\mathbf{x} = 0$ is normally undefined, for proper behavior in the application scenario of machine learning this function substitutes 0 as a result on those elements to handle this condition gracefully.

4.4.1 Parameters

`N` **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.5 vmltanh - Tanh activation

```
#include <mvecml.h>

void vmltanh (int n, double *y, int incy, const double *x, int incx);
void vmltanhf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the TanH activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}$$

4.5.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.6 vmltanhd - Tanh activation (first derivative)

```
#include <mvecml.h>

void vmltanhd (int n, double *y, int incy, const double *x, int incx);
void vmltanhdf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the TanH activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq 1 - \left(\frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}\right)^2$$

4.6.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.7 vmlsplu - Softplus activation

```
#include <mvecml.h>

void vmlsplu (int n, double *y, int incy, const double *x, int incx);
void vmlspluf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the Softplus activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \ln(1 + e^{\mathbf{x}})$$

4.7.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.8 vmlsplud - Softplus activation (first derivative)

```
#include <mvecml.h>

void vmlsplud (int n, double *y, int incy, const double *x, int incx);
void vmlspludf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the Softplus activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{1+e^{-\mathbf{x}}}$$

4.8.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.9 vmlgauss - Gaussian activation

```
#include <mvecml.h>

void vmlgauss (int n, double *y, int incy, const double *x, int incx);
void vmlgaussf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the Gaussian activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq e^{-\mathbf{x}^2}$$

4.9.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.10 vmlgaussd - Gaussian activation (first derivative)

```
#include <mvecml.h>

void vmlgaussd (int n, double *y, int incy, const double *x, int incx);
void vmlgaussdf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the Gaussian activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq -2\mathbf{x}e^{-\mathbf{x}^2}$$

4.10.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.11 vmlgelu - GELU activation

```
#include <mvecml.h>

void vmlgelu (int n, double *y, int incy, const double *x, int incx);
void vmlgeluf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the GELU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{2}\mathbf{x}\left(1+\operatorname{erf}\left(\frac{\mathbf{x}}{\sqrt{2}}\right)\right)$$

4.11.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.12 vmlgelud - GELU activation (first derivative)

```
#include <mvecml.h>

void vmlgelud (int n, double *y, int incy, const double *x, int incx);
void vmlgeludf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the GELU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{2}\mathbf{x}\left(1+\operatorname{erf}\left(\frac{\mathbf{x}}{\sqrt{2}}\right)\right)+\frac{\mathbf{x}}{\sqrt{2}\sqrt{\pi}}e^{-\mathbf{x}^2/2}$$

4.12.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.13 vmlsilu - SiLU activation

```
#include <mvecml.h>

void vmlsilu (int n, double *y, int incy, const double *x, int incx);
void vmlsiluf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the SiLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} = \frac{\mathbf{x}}{1 + e^{-\mathbf{x}}}$$

4.13.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.14 vmlsilud - SiLU activation (first derivative)

```
#include <mvecml.h>

void vmlsilud (int n, double *y, int incy, const double *x, int incx);
void vmlsiludf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the SiLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} = \frac{1 + e^{-\mathbf{x}} + xe^{-\mathbf{x}}}{(1 + e^{-\mathbf{x}})^2}$$

4.14.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.15 vmlssgn - Softsign activation

```
#include <mvecml.h>

void vmlssgn (int n, double *y, int incy, const double *x, int incx);
void vmlssgnf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the Softsign activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{\mathbf{x}}{|\mathbf{x}| + 1}$$

4.15.1 Parameters

`N` **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.16 vmlssgnd - Softsign activation (first derivative)

```
#include <mvecml.h>

void vmlssgnd (int n, double *y, int incy, const double *x, int incx);
void vmlssgndf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the Softsign activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{|\mathbf{x}|}{(\mathbf{x}^2+1)|\mathbf{x}|+2\mathbf{x}^2}$$

4.16.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.17 vmlmish - MISH activation

```
#include <mvecml.h>

void vmlmish (int n, double *y, int incy, const double *x, int incx);
void vmlmishf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the MISH activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \tanh(\log(e^{\mathbf{x}} + 1))\mathbf{x}$$

4.17.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.18 vmlmishd - MISH activation (first derivative)

```
#include <mvecml.h>

void vmlmishd (int n, double *y, int incy, const double *x, int incx);
void vmlmishdf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the MISH activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \tanh(\log(e^{\mathbf{x}}+1)) + \frac{\mathbf{x}e^{\mathbf{x}}\left(\frac{1}{\cosh(\log(e^{\mathbf{x}}+1))}\right)^2}{e^{\mathbf{x}}+1}$$

4.18.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.19 vmlatan - Arctan activation

```
#include <mvecml.h>

void vmlatan (int n, double *y, int incy, const double *x, int incx);
void vmlatanf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the arc-tangent activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \tan^{-1}(\mathbf{x})$$

4.19.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.20 vmlatand - Arctan activation (first derivative)

```
#include <mvecml.h>

void vmlatand (int n, double *y, int incy, const double *x, int incx);
void vmlatandf(int n, float *y, int incy, const float *x, int incx);
```

Given an input vector **x** and a result vector **y** this function computes the first derivative of the arc-tangent activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \frac{1}{\mathbf{x}^2 + 1}$$

4.20.1 Parameters

N **- INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y **- ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY **- INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

X **- ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX **- INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.21 vmlprelu - PReLU activation

```
#include <mvecml.h>

void vmlprelu (int n, double *y, int incy, double a, const double *x, int incx);
void vmlpreluf(int n, float *y, int incy, float a, const float *x, int incx);
```

Given an input vector **x**, a result vector **y** and a constant α this function computes the PReLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \begin{cases} \alpha\mathbf{x} & \text{where } \mathbf{x} < 0 \\ \mathbf{x} & \text{where } \mathbf{x} \geq 0 \end{cases}$$

4.21.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`A` - **REAL** *ENTRY:* Constant α.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.22 vmlprelud - PReLU activation (first derivative)

```
#include <mvecml.h>

void vmlprelud (int n, double *y, int incy, double a, const double *x, int incx);
void vmlpreludf(int n, float *y, int incy, float a, const float *x, int incx);
```

Given an input vector **x**, a result vector **y** and a constant α this function computes the first derivative of the PReLU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \begin{cases} \alpha & \text{where } \mathbf{x} < 0 \\ 1 & \text{where } \mathbf{x} \geq 0 \end{cases}$$

4.22.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`A` - **REAL** *ENTRY:* Constant α.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.23 vmlselu - SELU activation

```
#include <mvecml.h>

void vmlselu (int n, double *y, int incy, double l, double a, const double *x, int incx);
void vmlseluf(int n, float *y, int incy, float l, float a, const float *x, int incx);
```

Given an input vector **x**, a result vector **y** and constants λ and α this function computes the SELU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \lambda \begin{cases} \alpha(e^{\mathbf{x}} - 1) & \text{where } \mathbf{x} < 0 \\ \mathbf{x} & \text{where } \mathbf{x} \geq 0 \end{cases}$$

4.23.1 Parameters

`N` - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

`Y` - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

`INCY` - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

`L` - **REAL** *ENTRY:* Constant λ.

`A` - **REAL** *ENTRY:* Constant α.

`X` - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

`INCX` - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

4.24 vmlselud - SELU activation (first derivative)

```
#include <mvecml.h>

void vmlselud (int n, double *y, int incy, double l, double a, const double *x, int incx);
void vmlseludf(int n, float *y, int incy, float l, float a, const float *x, int incx);
```

Given an input vector **x**, a result vector **y** and constants λ and α this function computes the first derivative of the SELU activation function of the values in the **x** vector and stores the result in the **y** vector.

$$\mathbf{y} \doteq \lambda \begin{cases} \alpha e^{\mathbf{x}} & \text{where } \mathbf{x} < 0 \\ 1 & \text{where } \mathbf{x} \geq 0 \end{cases}$$

4.24.1 Parameters

N - **INTEGER** *ENTRY:* Number of elements of **x** and **y**.
CONSTRAINT: $n \geq 1$.

Y - **ARRAY OF REAL** *EXIT:* Result vector **y**.
CONSTRAINT: Must contain $n \times$ incy elements.
CONSTRAINT: Must not overlap with array **x**.

INCY - **INTEGER** *ENTRY:* Stride for the vector **y**.
CONSTRAINT: incy > 0.

L - **REAL** *ENTRY:* Constant λ.

A - **REAL** *ENTRY:* Constant α.

X - **ARRAY OF REAL** *ENTRY:* Input vector **x**.
CONSTRAINT: Must contain $n \times$ incx elements.
CONSTRAINT: Must not overlap with array **y**.

INCX - **INTEGER** *ENTRY:* Stride for the vector **x**.
CONSTRAINT: incx > 0.

5 Acknowledgements

Parts of this product include or are derived from code under the following licences:

```
/*
 * Copyright (c) 2003, Steven G. Kargl
 * Copyright (c) 2005, 2011 David Schultz <das@FreeBSD.ORG>
 * Copyright (c) 2005 Bruce D. Evans and Steven G. Kargl
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

/*
 * Copyright (c) 2008 Stephen L. Moshier <steve@moshier.net>
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */

------------------------------------------------------------------------
Copyright © 2005-2020 Rich Felker, et al.
Copyright © 2013 Szabolcs Nagy

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```