Math Vector Library Installation and Users Guide
DD-00001-001

Jan Adelsbach

February 16, 2025

Contents

1 About this Guide **3**
1.1 Legal Information . . . 3
1.2 Feedback and Contact . . . 3
1.3 Introduction . . . 3
1.4 Audience for This Guide . . . 3
1.5 Conventions Used in This Guide . . . 3

2 System Requirements **4**

3 Installation and Deinstallation **5**
3.1 UNIX, Linux and BSD . . . 5
3.1.1 Verifying the distribution (*optional*) . . . 5
3.1.2 Installation . . . 5
3.1.3 Updating to a Patch Release . . . 5
3.1.4 Deinstallation . . . 5
3.2 Microsoft Windows . . . 6
3.2.1 Verifying the distribution (*optional*) . . . 6
3.2.2 Installation . . . 6
3.2.3 Updating to a Patch Release . . . 6
3.2.4 Deinstallation . . . 6

4 Usage **7**
4.1 Header Files . . . 7
4.2 Compiling against the Library (C/C++) . . . 7
4.3 Compiling against the Library (Fortran) . . . 7

5 Library Variants **8**

6 Extensions **9**
6.1 Machine Learning Extension . . . 9
6.1.1 Header Files . . . 9
6.1.2 Libraries . . . 9
6.2 Advanced Trigonometry Extension . . . 9
6.2.1 Header Files . . . 9
6.2.2 Libraries . . . 9

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

The *Math Vector Library* is a high-performance function library for standard mathematical functions operating on vectors with optional stride.

Included in the distribution is also the *Math Scalar Library* which provides an implementation of the C standard math library `libm` using the same implementations as the Math Vector Library.

The *Math Vector Machine Learning Extension* provides high-performance primitives for machine learning applications.

1.4 Audience for This Guide

The audience of this guide is assumed to be the system administrators or users installing the *Math Vector Library* distribution onto their system.

Usage of the Linux command line and basic system operations is assumed.

1.5 Conventions Used in This Guide

`Mono`
: Monospace typesetting represents commands or file paths.

Italic
: Italic typesetting refers to abbreviations or further literature.

2 System Requirements

Around 200MB of free disk space are required for installing the distribution for a single platform.

In order to extract the distribution on Linux an extraction tool for `.tgz` (TAR-GZIP) archives or for `.zip` (ZIP) archives is required. This can be the standard system tools `tar(1)` and `gunzip(1)` and or `unzip(1)`.

On Windows you need the ability to extract `.zip` (ZIP) archives, this is normally included by default and the archive can be opened in the explorer using a double-click.

The specific system requirements in terms of platform, processor features and operating system are dependent on the distribution. See the information in the product distribution portal.

3 Installation and Deinstallation

3.1 UNIX, Linux and BSD

3.1.1 Verifying the distribution (*optional*)

Verifying the integrity of the distribution archive can be used to ensure the integrity of the distribution archive in that the distribution has not been tampered in a potential malicious manner with during transit.

Checksum files using the MD5 and SHA256 algorithms are provided for every product, namely the files `MD5SUM` or `SHA256SUM`. These can be obtained in the product distribution portal or by contacting support.

NOTE: Both of these files must be re-downloaded each time a new distribution is released, as they only contain the checksums of the distributions released up until they were downloaded.

Assuming one or both of the aforementioned files are in the same directory as the distribution archive the integrity of the latter can be verified using:

MD5

```
md5sum -ignore-missing -c MD5SUM
```

SHA256

```
sha256sum -ignore-missing -c SHA256SUM
```

Any of these commands should show the filename of the distribution archive followed by an OK. If this is not the case re-download the distribution archive or contact customer support. If the output is empty, the checksum for the distribution is not present in the file(s).

3.1.2 Installation

The distribution is delivered as a `gzip(1)` compressed `tar(1)` archive. The distribution is simply extracted.

In order to install the software extract the distribution archive matching the hardware and operating system platform into `/opt/adelsbach` using a deviation of the following commands:

```
mkdir -p /opt/adelsbach
tar xvf mvec10.xxxx.pppp.vvvv.tgz /opt/adelsbach/
```

Where `xxxx` is the operating system, `pppp` is the platform and `vvvv` is the variant. The `mvec10` corresponds to version 1.0 of the distribution.

This will extract the distribution under `/opt/adelsbach/mvecXY` where `XY` are the major and minor version numbers. As such this allows multiple major and minor versions to be simultaneously installed on the same system.

After installation ensure the file permissions are set accordingly using `chmod(1)` and or `chown(1)`, such that eligible users have read and execute permission to the distribution directory.

3.1.3 Updating to a Patch Release

A patch release is usually issued for minor improvements or bug-fixes, these have the same exact same file and directory names as the respective major and minor version of the distribution.

In order to install a patch release follow the instructions in 3.1.2, thereby overwriting the files of the existing distribution installation to be updated.

The file `VERSION` inside the distribution directory of the product should then match the patch release version.

3.1.4 Deinstallation

To deinstall the product, delete the directory `/opt/adelsbach/mvecXY` where `XY` are the major and minor version numbers. This can be accomplished using:

```
rm -rf /opt/adelsbach/mvecXY
```

3.2 Microsoft Windows

3.2.1 Verifying the distribution (*optional*)

Verifying the integrity of the distribution archive can be used to ensure the integrity of the distribution archive in that the distribution has not been tampered in a potential malicious manner with during transit.

Checksum files using the MD5 and SHA256 algorithms are provided for every product, namely the files `MD5SUM` or `SHA256SUM`. These can be obtained in the product distribution portal or by contacting support.

NOTE: Both of these files must be re-downloaded each time a new distribution is released, as they only contain the checksums of the distributions released up until they were downloaded.

Assuming one or both of the aforementioned files are in the same directory as a distribution archive, for example `mvec10.win.amd64.avx.zip` the integrity of the latter can be verified using:

MD5

```
CertUtil -hashfile mvec10.win.amd64.avx.zip MD5
```

SHA256

```
CertUtil -hashfile mvec10.win.amd64.avx.zip SHA256
```

Both of these should print a checksum that equals the one found in the `MD5SUM` or `SHA256SUM` for the given distribution filename.

3.2.2 Installation

The distribution is delivered as a ZIP archive. The distribution is simply extracted.

In order to install the software extract the distribution archive matching the hardware and operating system platform into `C:\Program Files\Adelsbach`.

This will extract the distribution under `C:\Program Files\Adelsbach\mvecXY` where `XY` are the major and minor version numbers. As such this allows multiple major and minor versions to be simultaneously installed on the same system.

3.2.3 Updating to a Patch Release

A patch release is usually issued for minor improvements or bug-fixes, these have the same exact same file and directory names as the respective major and minor version of the distribution.

In order to install a patch release follow the instructions in 3.2.2, thereby overwriting the files of the existing distribution installation to be updated.

The file `VERSION` inside the distribution directory of the product should then match the patch release version.

3.2.4 Deinstallation

To deinstall the product, delete the directory `C:\Program Files\Adelsbach\mvecXY` where `XY` are the major and minor version numbers.

4 Usage

4.1 Header Files

All functions are declared in two C/C++ header files:

`mvec.h` **and** `mvec64.h` Declarations of the vector oriented functions of the *Math Vector Library*.

`msca.h` Declarations of the scalar oriented functions of the *Math Scalar Library*.

All of these header files are standalone and do not require the inclusion of any external header files, unless complex number functionality is required, in which case the standard system header `complex.h` needs to be included before any of the headers above.

4.2 Compiling against the Library (C/C++)

In order to use the product, the header search path of the C/C++ compiler needs to be set to the `include` directory inside the distribution directory.

In order to link with the library the library search path needs to include `lib` or `lib64` directories inside the distribution directory and the application needs to be linked against the respective library used. See 5 for the variants of the library available.

This can be accomplished as follows for the following compilers:

GNU C/C++ Compiler, LLVM clang/clang++, Portland Group Compiler, IBM XL C/C++
For the include search path use the `-I` flag, such as: `-I/opt/adelsbach/mvecXY/include`.
For the library search path use the `-L` flag, such as: `-I/opt/adelsbach/mvecXY/lib` or `-I/opt/adelsbach/mvecXY/lib64` for 64bit systems.
For linking against `libmvec` use the `-l` flag, such as: `-lmvec`.

Microsoft Visual Studio
Right-click in your Solution Exporer and Select **Properties**.
Add the folder the distribution was extracted to under **Additional Library Directories**.
Add the library to link against under **Linker -> Input -> Additional Dependencies**.
Add the header file folder under **C/C++ -> Additional Include Directories**.

4.3 Compiling against the Library (Fortran)

The Fortran subroutine entry points are contained in the same library files as the C/C++ version, as such the same compiler and linker instructions as provided in 4.2 apply.

By default it is assumed that the Fortran compiler and linker use suffix "_" function names and use the same calling conventions as C/C++. This is the default Fortran call specification on most platforms and supported compilers.

5 Library Variants

Standard Library - `mvec`

The standard *Math Vector Library* and *Math Scalar Library* represent the standard, single threaded variants of the library.

Debugging Library - `mvec_dbg`

The standard distribution of the *Math Vector Library* contains debugging versions of both the shared and static library. These are suffixed with `_dbg`.

The debugging versions contain full debugging symbols and support for breakpoints.

Multi-threaded Library - `mvec_mp`

The standard distribution of the *Math Vector Library* contains a multi-threaded version of the library on platforms where supported. This version uses the OpenMP standard to introduce multi-threading into the library functions, as such any application linked against the *Math Vector Library* in its multi-threaded version also needs to link against a OpenMP runtime library. The *Math Scalar Library* is not available in the multi-threaded version. The following can be used to link against this version of the library:

GNU C/C++ Compiler

Use the following flags to link against the GNU OpenMP Runtime (GOMP) `-lmvec_mp -lgomp`.

LLVM clang/clang++

Use the following flags to link against the OpenMP Runtime (omp) `-lmvec_mp -lomp`.

GPGU Accelerated Library - `mvec_gpgpu`

This version of the library makes use of *General Purpose GPU* (GPGPU) accelerators for executing functions on large vectors. Please refer to current platform support, as not all GPGPU accelerators and or processor architectures or operating systems may be supported.

6 Extensions

6.1 Machine Learning Extension

By default the distribution ships with the *Machine Learning Extension*, which features high-performance primitives for deep learning applications.

6.1.1 Header Files

All C/C++ declarations of the *Machine Learning Extension* are declared in the header file `mvecml.h` and their 64bit counterparts in `mvecml64.h`. The header files are stand-alone and do not require the inclusion of any other header files.

6.1.2 Libraries

The functions of the *Machine Learning Extension* are implemented inside the library `mvecml`. A debug version `mvecml_dbg` and an OpenMP based shared multiprocessing version `mvecml_mp` are also available, as is a GPGPU version `mvecml_gpgu` on specific platforms.

6.2 Advanced Trigonometry Extension

The default distribution contains the *Advanced Trigonometry Extension*. The latter implements various specialized trigonometry functions, as well as the standard trigonometry functions with different units than radians.

For sake of completness, this extension also provides the standard trigonometry functions also found in the main library but with the extension function name prefix.

6.2.1 Header Files

All C/C++ declarations of the *Advanced Trigonometry Extension* are declared in the header file `mvecatrig.h` and their 64bit counterparts in `mvecatrig64.h`. The header files are stand-alone and do not require the inclusion of any other header files.

6.2.2 Libraries

The functions of the *Advanced Trigonometry Extension* are implemented inside the library `mvecatrig`. A debug version `mvecatrig_dbg` and an OpenMP based shared multiprocessing version `mvecatrig_mp` are also available, as is a GPGPU version `mvecatrig_gpgu` on specific platforms.