# ViennaMath 1.0.0

User Manual

Institute for Microelectronics
Gußhausstraße 27-29 / E360
A-1040 Vienna, Austria

Institute for Analysis and Scientific Computing
Wiedner Hauptstraße 8-10 / E101
A-1040 Vienna, Austria/Europe

*Author and Project Head:*

Karl Rupp

Institute for Microelectronics
Vienna University of Technology
Gußhausstraße 27-29 / E360
A-1040 Vienna, Austria/Europe

| | |
|---|---|
| Phone | +43-1-58801-36001 |
| FAX | +43-1-58801-36099 |
| Web | `http://www.iue.tuwien.ac.at` |

Institute for Analysis and Scientific Computing
Vienna University of Technology
Wiedner Hauptstraße 8-10 / E101
A-1040 Vienna, Austria

| | |
|---|---|
| Phone | +43-1-58801-10101 |
| Web | `http://www.asc.tuwien.ac.at` |

# Contents

# Introduction

Answer the *why another FEM code?* question.

# Chapter 1

# Installation

This chapter shows how ViennaMath can be integrated into a project and how the examples are built. The necessary steps are outlined for several different platforms, but we could not check every possible combination of hardware, operating system, and compiler. If you experience any trouble, please write to the mailing list at

viennamath-support@lists.sourceforge.net

## 1.1 Dependencies

- A recent C++ compiler (e.g. GCC version 4.2.x or above and Visual C++ 2008 are known to work)
- CMake [1] as build system (optional, but recommended for building the examples)

## 1.2 Generic Installation of ViennaMath

Since ViennaMath is a header-only library, it is sufficient to copy the viennamath/ folder either into your project folder or to your global system include path. On Unix based systems, this is often /usr/include/ or /usr/local/include/.

On Windows, the situation strongly depends on your development environment. We advise to consult the documentation of the compiler on how to set the include path correctly. With Visual Studio this is usually something like C:\Program Files\Microsoft Visual Studio 9.0\VC\include and can be set in Tools -> Options -> Projects and Solutions -> VC++-Directories.

## 1.3 Building the Examples and Tutorials

For building the examples, we suppose that CMake is properly set up on your system. The other dependencies are listed in Tab. 1.1.

| File | Purpose |
|---|---|
| `tutorial/access.cpp` | Shows the use of `access` |
| `benchmarks/sparse-custom.cpp` | Benchmarks for a sparse storage scheme with custom object identification |

Table 1.1: Overview of the examples in the `examples/` folder

### 1.3.1 Linux

To build the examples, open a terminal and change to:

```
$> cd /your-ViennaMath-path/build/
```

Execute

```
$> cmake ..
```

to obtain a Makefile and type

```
$> make
```

to build the examples. If desired, one can build each example separately instead:

```
$> make access              #builds the access tutorial
$> make dense               #builds dense storage benchmark
```

Speed up the building process by using jobs, e.g. `make -j4`.

### 1.3.2 Mac OS X

The tools mentioned in Section 1.1 are available on Macintosh platforms too. For the `GCC` compiler the Xcode [2] package has to be installed. To install `CMake`, external portation tools such as Fink [3], DarwinPorts [4], or MacPorts [5] have to be used.

The build process of `ViennaMath` is similar to Linux.

### 1.3.3 Windows

In the following the procedure is outlined for `Visual Studio`: Assuming that an `OpenCL` SDK and `CMake` is already installed, Visual Studio solution and project files can be created using `CMake`:

- Open the `CMake` GUI.
- Set the `ViennaMath` base directory as source directory.
- Set the `build/` directory as build directory.
- Click on 'Configure' and select the appropriate generator (e.g. `Visual Studio 9 2008`)

- Click on 'Generate' (you may need to click on 'Configure' one more time before you can click on 'Generate')

- The project files can now be found in the ViennaMath build directory, where they can be opened and compiled with Visual Studio (provided that the include and library paths are set correctly, see Sec. 1.2).

# Chapter 2

# Basic Types

## 2.1 Types Evaluated at Runtime

## 2.2 Types Evaluated at Compiletime

# Chapter 3

# Advanced Features

## 3.1 Function Symbols

## 3.2 Integration Symbols

## 3.3 LaTeX Output

## Chapter 4

# Benchmark Results

Rerun the HPC 2010 benchmarks. Shouldn't be too far off the hand-tuned version.

# Chapter 5

# Library Internals

Maybe describe something here (probably not enough time?)

# Chapter 6

# Design Decisions

Every now and then a programmer is faced with the question “Why have you implemented it that way?”. This is then the chance to outline the various pros and cons of different approaches and assign appropriate weights such that the approach taken turns out to be the best. In this chapter the various design considerations are presented and the reasons why the library is the way it is are given. Remaining questions or inputs for further improvements should be sent to the mailing list at

`viennamath-support@lists.sourceforge.net`

## 6.1 Something

# Change Logs

### Version 1.0.0

First release

# Chapter 7

# License

# Bibliography

[1] "CMake." [Online]. Available: http://www.cmake.org/

[2] "Xcode Developer Tools." [Online]. Available: http://developer.apple.com/technologies/tools/xcode.html

[3] "Fink." [Online]. Available: http://www.finkproject.org/

[4] "DarwinPorts." [Online]. Available: http://darwinports.com/

[5] "MacPorts." [Online]. Available: http://www.macports.org/