# Software Development Tools and Methods
# Practical session 41st Weeks

October 9th

**A report is needed for this practical session. You have to upload it on GitLab before October 23th**

## Contents




**Abstract**

The aim of this practical work is to be able to use some development tools. It is not necessary to write many code in this lab. When it is necessary to do so, it is mainly to use/call tools (Boost libraries, Eigen3, etc.) There is no algorithmic difficulty. However, you have to configure / use many tools (in particular cmake).

Think to use the tutorials listed in the Annex or those you find online.


## 1 General presentation

It is recommended to use the examples of the course (`Codes` Folder)

The Libraries and Tools to be used in this session are : `git`, `CMake`, `Boost.Test`, `Boost.Timer`, `Boost.uBlas`, `Eigen3` and `taskset`.

We have seen together some of these tools. Others will be discovered using the online tutorials and examples.

**You will need to give a report (which will be marked) in latex/pdf format and commit it in GitLab before October 23th.** Each pair of students will create a sub-directory `NAME1_NAME2` in the `Reports` repository containing their files `.tex` and `.pdf`.

## 2 CMake, Eigen3, Boost Tools

1. Create a project with CMake in a new sub-directory of your `HOME`.
2. Download the last version of Eigen3 library in your project folder (https://bitbucket.org/eigen/eigen/downloads). Add it to your project using as an external one. You can also download directly Eigen3 using `wget` :

```
wget http://bitbucket.org/eigen/eigen/get/3.2.10.tar.gz
```

   You can add Eigen3 as an external projet by using these commands in your own CMakeLists.txt

```
include(ExternalProject)
ExternalProject_Add(
  Eigen
  # we suppose here that your project is named TP
  # Library Eigen3 is stored in the sub-folder eigen
  SOURCE_DIR ${TP_SOURCE_DIR}/eigen
  INSTALL_COMMAND echo "Skipping install"
)
```

3. Add the eigen directory using `INCLUDE_DIRECTORIES()`.
4. Create a folder tests and write some Eigen3 tests that should be checked. These tests have to use `Boost.Test`. Use `enable_testing()` and `add_test()`.
5. Test the performances of Eigen3 for some linear algebra elementary operations (for example matrices/matrices product or matrices/vectors product).
   (a) Use `Boost.Timer` to measure the time cost. Think to use `cpu_timer` and `cpu_times elapsed_times` with these headers (see `Boost.Timer` tutorial).

```
#include <boost/timer/timer.hpp>
using namespace Eigen;
using namespace boost::timer;
```

   You will need also to link your program with `system` and `timer` components of Boost (Do it like the example seen in course for example).
   (b) Add the benchmark to list of tests.
   (c) Check the execution time for different matrices/vectors sizes. (which bounds and steps you use ?)
   (d) Compare the results according to the used options to optimize your code. Check the type of sse extension supported by your computer (`cat /proc/cpuinfo`)
   - `-g`
   - `-O2`
   - `-O3`
   - `-O2 -msse2` or other level option sse
   - `-O3 -msse2` or other level option sse
   - ....
6. Write the same performance tests but using `boost.ublas` instead of Eigen3.

```
#include <boost/numeric/ublas/matrix.hpp>
using namespace boost::numeric::ublas;
```

7. The computers have 4 processors (cat /proc/cpuinfo). You can select which processor you can use thanks to The `taskset` tool (see `man taskset`). Test the execution of your benchmarks using 1, 2, 3 or 4 processors. What do you observe? Explain your results.

## 3 The report

You will have only to give the sources LaTeX file and the compiled pdf file. This document has to contain a description of your work with its different steps, choices and your comments.

You will include the listings of your codes and the results of their executions and also the contents of your CMakeLists. To do so, you will use the LaTeX package `listings` :

```
\usepackage{hyperref}
\usepackage{listings}
\hypersetup{colorlinks,pdffitwindow,linkcolor=blue}
\definecolor{lbcolor}{rgb}{0.95,0.95,0.95}
\definecolor{cblue}{rgb}{0.,0.0,0.6}
\definecolor{lblue}{rgb}{0.1,0.1,0.4}
\lstset{language=C++,showspaces=false,showstringspaces=false,
        captionpos=t,literate={»}{\ensuremath{»}}1,mathescape}
\lstset{basicstyle=\small\ttfamily}
\lstset{lineskip=-2pt}
\lstset{keywordstyle=\color{red}\bfseries}
\lstset{emph={inline},emphstyle=\color{red}\bfseries}
\lstset{commentstyle=\ttfamily\color{cblue}}
\lstset{backgroundcolor=\color{lbcolor},rulecolor=}
\lstset{frame=single,framerule=0.5pt}
\lstset{belowskip=\smallskipamount}
\lstset{aboveskip=\smallskipamount}

\begin{lstlisting}[language=c++]
class A {};
\end{lstlistings}

The documentation can be found in
\begin{lstlisting}
/usr/share/doc/texlive-doc/latex/listings-ext/listings-ext.pdf
/usr/share/doc/texlive-doc/latex/oberdiek/listingsutf8.pdf
```

## 4 Online Documentation

In addition to course notes, you can use online documentation, for example

**eigen3**

- Site http://eigen.tuxfamily.org/index.php?title=Main_Page
- Documentation http://eigen.tuxfamily.org/dox/

**cmake**

- Site http://www.cmake.org/

**Boost.Test**

- Site http://www.boost.org/doc/libs/1_46_1/libs/test/doc/html/index.html
- Boost.Test Tutorial http://www.alittlemadness.com/2009/03/31/c-unit-testing-with-boostt
- CMake and Boost Tutorials http://iborco.blogspot.com/2008/06/boost-with-cmake.html

**Boost.Timer**

- Site http://www.boost.org/doc/libs/1_46_1/libs/timer/index.html
- Tutorial http://unitedsoft.ch/boost/libs/timers/boost_timers/tutorial.html

**Boost.uBlas**

- Site http://www.boost.org/doc/libs/1_46_1/libs/numeric/ublas/doc/index.htm
- Example from documentation

```
#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m (3, 3);
    vector<double> v (3);
    for (unsigned i = 0; i < std::min (m.size1 (), v.size ()); ++ i) {
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;
        v (i) = i;
    }

    vector<double> res1 (3);
    vector<double> res2 (3);
    matrix<double> res3 (3, 3);
    res1 = prod(m,v);
    res2 = prod(v,m);
    res3 = prod(m,m);
    std::cout << res1 << std::endl;
    std::cout << res2 << std::endl;
    std::cout << res3 << std::endl;
}
```