Advanced Methods for Scientific Computing (AMSC)
Lecture title: Introduction

Luca Formaggia

MOX
Dipartimento di Matematica
Politecnico di Milano

A.Y. 2025/2026

General Information

Lecturer: Prof. Luca Formaggia (luca.formaggia@polimi.it)
Assistant: Dr. Paolo Baioni (paolojoseph.baioni@polimi.it)
Tutor: Marco Scarpelli (marco.scarpelli@polimi.it)
Reception Hours: Wednesday 14.15 – 16.15 (on appointment)

Lectures and Laboratories are held on
Thursday from 10.15 to 12.00 room 3.1.6 and Friday from 10.15 to 12.00 in room 9.0.3

The slots used by Laboratory sessions will be indicated at lecture.

Lectures and laboratory sessions are streamed and recorded. Recordings will be available only to registered students.

General Information

The course consists of lectures, laboratory sessions with the use of your computer and a written test possibly followed by an oral examination, or, optionally, a project on a topic agreed with the instructor's.

During the course we will create groups of students to work on selected problems.

Group work is not evaluated, it is part of the learning process.

On line resources

- The page of the courses on line of Politecnico, WeBeep will be used as exchange point. I will put there a copy of the slides and other material, including a bibliography;
- A git repository on github has been set up for the course Examples (we will do a brief introduction to git);
- Another git repository on github has been set up for the course Laboratory Sessions;
- Lecture recordings. Through the usual channels of Politecnico di Milano (only for registered students).

There is also a youtube playlist with material prepared for the companion courses for Mathematical Engineering (PACS), a bit outdated.

Operative system

We will refer to the Linux operative system, the Unix-based operative based on the kernel originally developed by Linus Torvald and the GNU project of the Free Software Foundation.

To make the system more uniform we adopt the environment module system. Instructions on different ways to install Linux and the modules on your PC are contained on the course WeBeep site under the section Material.

The module system is not necessary if you have a native Linux system (like Ubuntu) with an updated compiler and you install the relevant packages whenever needed.

Downloading the Examples

- If you do not have already an account on GitHub, create one going to the GitHub site and following the instructions;
- Once logged in, click on the icon on the top right, and choose *Settings*;
- On the left frame, choose *SSH and PGK Keys* and add your public rsa key following the instructions.

Generate ssh key (RSA)

The RSA keys are used to ensure secure communication. On a unix system to generate them you do

- Type `ssh-keygen -t rsa`. It will ask where to store the keys, use the default by pressing return.
- If you are paranoic, give a passphrase to protect your private key. If you think to use the keys for something more critical than this course, set the passphrase.
- When finished, you will have a nice but useless picture on the terminal, and in the `.ssh` folder (note the dot at the beginning) you find the file `id_rsa.pub` whose content should be copied verbatim into the GitHub site, as previously described.

Don't touch the file `id_rsa` (the private key), and don't give it to anybody. You may use other key types (like dsa), but rsa is fine.

Downloading the Examples

One you registered on github and you have provided your public ssh key you can do:

```
mkdir <name> # create a directory
cd <name>
git clone --recursive \
           git@github.com:HPC-Courses/AMSC-CodeExamples.git
cd AMSC-CodeExamples
```

and READ THE INSTRUCTIONS IN THE README.md FILE and those in `Examples/README.md` file.

Note: you may also download the code using the https protocol, but using ssh is better.

Examples: user setup

To update the content (do it frequently!).

```
cd   AMSC-CodeExamples
git pull --recurse-submodules
```

you can do the pull from any folder under the git repo `AMSC-CodeExamples`

the `recurse-submodule` option is necessary since I use git submodules to keep track of externally maintaned parts.

Another possibility (for more experienced students)

You may also fork the `AMSC-CodeExamples` repository in your git account. With forking you have your own copy that you can keep updated via a sync on GitHub.

You can play with the Examples as you wish. However, I suggest you to make a local branch for your experiments. Less possibilities of making a mess. Or, better, if you make a mess you just delete the local branch....

If you find a bug, or improve an example, you may submit the changes via pull request!.

The main folders of the Examples

The Examples are organised in different directories.
In (almost) all directories a `README.md` files contains the description of the content. In the main `Example` folder you have also the file `CONTENT.md` with an overall description.

Overlook of Examples folder

The directory `Examples` consists of several subdirectories:

- `include`, where the header files used by more than one examples are stored;
- `lib`, where libraries used by more than one example are stored;
- `src`, where the actual examples are stored.

In each directory under `src` there is a *Makefile*: typing `make` compiles the example; `make doc` produces a documentation in the subdirectory *doc*, `make clean` does a cleanup ; `make distclean` cleans also the documentation and the possible produced libraries; `make install` installs libraries in `Examples/lib` and header files in `Examples/include`.

In some cases the compiling instructions may be different: read the local `README.md` file.

Submodules

Parts of the Examples are kept in other git repositories, since they are forked from software made and updated by others.

The `--recursive` in `git clone` will also download the submodules and if you use`git pull --recurse-submodules` you keep them updated. If you want to keep them updated with the remote repo (or you have forgotten the `--recursive` when cloning the repo) do

```
git submodule update --recursive --remote --merge
```

But, in fact, I have created a script, `install-git-submodules.sh`, for the purpose.

How to compile the examples

Copy `Makefile.user` in `Makefile.inc` and change `PACS_ROOT` to the directory where the Examples reside (the same directory of `Makefile.user`).

Run `./setup.sh` to compile some basic stuff. If you want go to the folder `Extras` and run `./install_extras.sh` (it takes some time, get a coffee!). You need to have cmake installed.

To compile a specific example, you go in the directory and type `make`. But first read the README.md file!

Laboratory sessions

Labs are in another git repo. To get it:

```
git clone git@github.com:HPC-Courses/AMSC-Labs.git
```

or

```
git clone https://github.com/HPC-Courses/AMSC-Labs.git
```

and do

```
git pull
```

to keep it updated.

Compilers

We use as reference compiler the gnu compiler (g++), at least version 10.0. It is normally provided with any Linux distribution. Check the version with `g++ -v`.

Another very good compiler is clang++, of the LLVM suite, downloadable from llvm.org. Use version 15.0 or higher. You may find it in most Linux distributions (on Ubuntu you install it with `sudo apt-get install clang`). With respect to the gnu compiler it gives better error messages (and it is sometimes faster).

We will stick to C++ standard, so in principle any compiler which complies to the standard should be able to compile the examples.

On line C++ references

There are quite al lot of in-line references about C++. The main ones (in my opinion) are:

- www.cppreference.com: a very complete reference site. It is my preferred one!. A little technical sometimes, but you find everything!
- www.cplusplus.com: another excellent *on-line reference* on C++ with many examples, adjourned to the new standards.
- Wikipedia is also a useful source of information.

Use the web to find answers!

Development tools

The use of IDEs (Integrated Development environment) may help the development of a software. I often use Eclipse and provided by several Linux distributions), but other very well known IDEs are Visual Studio and CLion.
Visual Studio and Clion can be integrated with copilot for free if you have registered to GitHub with the official Polimi email and you declare you are a student.
All examples illustrated in the course will contain a Makefile to ease compilation process.

Of course, it is not compulsory to use an IDE (but it helps). A good editor may be sufficient. Good editors are Atom, emacs, vim and gedit. They all support syntax highlighting. nano is another lightweight texteditor.

copilot and chatGPt or other LLM

Large languuage models like chatGPt (copilot is specialised on programming) can speedup coding considerably if you know what you are doing and, more importantly, if you understand what they are doing.

I encourage you to use them, but with a bit of salt. Often they produce bugged code, sometimes subtly wrong[1], or inefficient code.

So they cannot spare you knowing the programming language well!

[1] I have asked chatGPT for a code for Simpson quadrature, it gave me a perfect code... but for the trapezoidal rule!

A note on the language used in the course

The course is given in English. Please forgive my mistakes, bad pronunciation and typos.
I suggest a book for those of you who wish to write the project report or the thesis in English. It contains also a lot of hints on the use of LaTeX.

N.J. Higham, Handbook of Writing for the Mathematical Sciences, Second Edition, SIAM, ISBN: 978-0-89871-420-3, 1998.

Get it, it is really plenty of good advice (and nice quotations).

A note on the author: Nicholas, J. Higham, is a well known mathematician who passed away very recently. He is famous for his works on the accuracy and stability of numerical algorithms. He has contributed software to LAPACK and the NAG library, and to several pieces of code currently included in MATLAB.