Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 1

HS 22
Dr. R. Käppeli

This first exercise sheet is probably the most time consuming as it sets up the environment you will use during the whole semester (and for many students beyond this semester!). We will do our best to help you getting started!

Problem 1.1 Install core tools

Install a `C++` compiler. If you already have a working development environment you can feel free to use that.
As our programs get larger and more complicated, we will need an efficient way to handle libraries and dependencies. We will use the build systems CMake[1] and GNU Make[2].
A version control system is essential to keep track of changes and for collaborative software development. We will use git[3]. There are editors which integrate git support and also GUI clients available such as SmartGIT[4].

Linux

Most common is the GNU compiler[5] for Linux, installed from your distribution's package repository. Equivalently, install CMake and make as well as git. On Ubuntu, this can be done with:

```
sudo apt install build-essential cmake git
```

macOS

For macOS we recommend to use a package management system such as Brew[6] or MacPorts[7]. For example with Brew installed, you get all the necessary software with:

```
brew install git gcc make cmake
```

Windows 10/11

There are two options:

- Installing the Windows Subsystem for Linux (WSL)[8] to get a Linux shell, and then install the core tools there (see **Linux**). Within the WSL Linux shell, your Windows files are available under `/mnt/c/` (see also here[9] for more information).
- Installing MSYS2[10], Git for Windows[11] and CMake for Windows. Within the Linux shell, your Windows files are available under `/c/` (adapt to your setup).

[1]http://www.cmake.org/
[2]http://www.gnu.org/software/make/
[3]http://git-scm.com/
[4]http://www.syntevo.com/smartgithg/
[5]http://gcc.gnu.org/
[6]https://brew.sh/
[7]https://www.macports.org/
[8]https://docs.microsoft.com/en-us/windows/wsl/about
[9]https://learn.microsoft.com/en-us/windows/wsl/faq#how-do-i-use-a-windows-file-with-a-linux-app-
[10]https://www.msys2.org/
[11]https://gitforwindows.org/

Problem 1.2 Cloning the lecture repository

Log into the ETH GitLab website[12], and clone the lecture repository[13]. By pulling this repository every week, you can always get the latest lecture notes, exercise sheets and solutions.

Create a new repository, which you will use to work on and hand in the exercises.
To avoid re-typing your password every time you push or pull from your repository, you can follow this guide[14] to set up SSH keys on your account.

Problem 1.3 Compilation and execution

As a warm up make sure you can compile (`c++ -o main main.cpp`) and run (`./main`) the following `main.cpp` program.

```
#include <iostream>

using namespace std;

int main() {
  cout << "Hello ETH students." << endl;
  return 0;
}
```

Problem 1.4 Unix-shell tutorial

Everything worth repeating is worth automating. Work through the shell-tutorial[15] at least up to and including the chapter on "Pipes and Filters".

Problem 1.5 Machine epsilon

Write a program to determine the floating-point precision on your machine. This is called machine epsilon[16].

Problem 1.6 Simpson numerical integration

The 1-dimensional Simpson integration approximates the function $f(x)$ by a parabola $\tilde{f}(x)$ in each bin stretching from x to $x + \Delta x$. For that one needs 3 function values at x, $x + \Delta x/2$ and $x + \Delta x$. The integral over the interpolating parabola $\tilde{f}(x)$ gives

$$\int_x^{x+\Delta x} \mathrm{d}x\, f(x) \approx \int_x^{x+\Delta x} \mathrm{d}x\, \tilde{f}(x) = \frac{\Delta x}{6}\Big[f(x) + 4f(x+\Delta x/2) + f(x+\Delta x)\Big]\,. \tag{1}$$

In order to numerically integrate a function from a to b you discretize it to N bins and use the interpolation formula within each bin. If you use regular a mesh (equally sized bins) with bin size $\Delta x = (b-a)/N$ then the complete formula for Simpson integration is

$$\begin{aligned}\int_a^b \mathrm{d}x\, f(x) \;&\approx\; \frac{\Delta x}{6}\Big[f(a) + 4f(a+\Delta x/2) + 2f(a+\Delta x) + 4f(a+3\Delta x/2) + \ldots \\ &\qquad + \ldots + 2f(b-\Delta x) + 4f(b-\Delta x/2) + f(b)\Big] + O\left(N^{-4}\right)\,. \end{aligned} \tag{2}$$

[12] https://gitlab.ethz.ch/
[13] https://gitlab.ethz.ch/pt1_hs22/lecture
[14] https://docs.gitlab.com/ce/ssh/README.html
[15] https://swcarpentry.github.io/shell-novice/
[16] https://en.wikipedia.org/wiki/Machine_epsilon

Write a program to implement the following numerical integration using Simpson's rule

$$\int_0^\pi dx \sin(x) \; . \tag{3}$$

Hint for testing/debugging: The Simpson integration should integrate polynomials up to the 3rd order exactly with any number of bins. So you may for instance integrate $\int_0^1 \mathrm{d}x \, x(1-x) = 1/6$ with $N = 1, 2, 3, 10$ bins for testing purposes.