Programming Techniques for Scientific Simulations Exercise 1

Problem 1.1 Account for D-PHYS workstations

Follow the ISG guidelines[1] on how to get an account. This is needed in order to use the workstations in the exercise class and to access the GitLab[2] system.

Problem 1.2 Unix-shell tutorial

Everything worth repeating is worth automating. Work through the shell-tutorial[3].

Problem 1.3 Install core tools

Install a `C++` compiler. Most common are the GNU compiler[4] for Linux, Xcode[5] with command line tools for Mac OS X, and MSVC[6] for Windows (available with an MSD-NAA account, which can be obtained through IDES[7] for ETH students).

As our programs get larger and more complicated, we will need an efficient way to keep track of libraries and dependencies. Install CMake[8] and also GNU Make[9] if you use Linux.

To keep track of changes and for collaborative coding a version control system is essential. Install git[10]. In addition to the command line, there are also GUI clients available such as SmartGIT[11]

Problem 1.4 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;
}
```

[1] https://admin.phys.ethz.ch/newaccount/
[2] https://gitlab.phys.ethz.ch/
[3] http://software-carpentry.org/v4/shell/index.html
[4] http://gcc.gnu.org/
[5] https://developer.apple.com/xcode/
[6] http://microsoft.com/visualstudio/
[7] https://ides.ethz.ch/
[8] http://www.cmake.org/
[9] http://www.gnu.org/software/make/
[10] http://git-scm.com/
[11] http://www.syntevo.com/smartgithg/

Problem 1.5 Machine epsilon

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

Problem 1.6 Simpson numerical integration

The 1-dimensional Simpson integration approximates the function by a parabola 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

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

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

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

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 2^{nd} order precisely 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.