Introduction to Computational Physics

PHYS 210 2016 Course Description

Location: Henn 205

PHYS 210 101	**Lecture**	**1**	**Tue Thu**	**12:30**	**13:00**
PHYS 210 L1A	**Laboratory**	**1**	**Tue Thu**	**13:00**	**15:00**
PHYS 210 102	**Lecture**	**1**	**Tue Thu**	**15:30**	**16:00**
PHYS 210 L1B	**Laboratory**	**1**	**Tue Thu**	**16:00**	**18:00**

Instructor: **Ludovic Van Waerbeke (Henn330), email:** phys210@phas.ubc.ca

Lab Assistants:
Colby DeLisle, email: cdelisle@phas.ubc.ca
Evan Thomas, email: zucchini@phas.ubc.ca
Xiruo Yan, email: xyan@phas.ubc.ca

Colby DeLisle: Tue & Thu 13:00-15:00 and 16:00-18:00
Evan Thomas: Tue & Thu 13:00-15:00
Xiruo Yan: Tue & Thu 16:00-18:00

Instructor + Lab Assists mailing list: phys210instructors@phas.ubc.ca

What is Computational Physics?

From wikipedia https://en.wikipedia.org/wiki/Computational_physics

"**Computational physics** is the study and implementation of numerical analysis to solve problems in **physics** for which a quantitative theory already exists. Historically, **computational physics** was the first application of modern computers in science, and is now a subset of **computational science**."

For the **large** majority of physical situations, an analytical solution does not exist. For instance none of the following situations have an analytical solution:

-The gravitational interaction between 3 or more bodies is space
-The development of turbulence in the atmosphere (weather forecast, climate predictions)
-Designing a building that can resist earthquakes
-Designing an airplane

-Chemistry of complex (or not so complex) molecules
-Particle interactions at high energy in a particle accelerator
-etc…

However, at the same time, the underlying fundamental physical laws of **any** phenomena are quite simple and well understood. Unfortunately, under realistic situations, like those above, they cannot be solved analytically so one has to rely on **numerical calculations** in order to obtain a solution. Let's put it this way: complexity emerges from simple fundamental laws, the consequence is that simple solutions you learn at school are not applicable in most realistic situations.

The techniques of numerical computation you will learn in this class can be used for **anything**: physics, mathematics, engineering, economics, sociology, biology, medicine, finance, arts (e.g. music and instrument design), etc…

In this class, you will learn a **language; PHYS210 is nothing else but a language class**, a language that will allow you to command a computer to do some tasks in order to perform the numerical calculations you want. A computer is the opposite of a human brain, it is the most stupid thing you can think of: it does what you tell it to do, period. Moreover, a computer can process very quickly a **huge** amount of information, which is ideal to get quickly a numerical solution to complex problems, once the problem is well "taught" to the computer. The way you tell a computer what to do is by learning the language it was programmed to understands. There are **MANY** of such languages, called programming languages (https://en.wikipedia.org/wiki/Programming_language). In this class we will focus on one particular language, **python**. For history about python read here: https://en.wikipedia.org/wiki/Python_(programming_language).

Python is currently the most versatile high level programming language; it is used in all scientific areas a lot and also in non scientific fields. It is intuitive, open-source (i.e. free and continuously developed by the community of users) and very powerful (i.e. it can do complex tasks quickly with a minimum number of commands). Learning programming in python requires that you learn programing concepts that are shared by any programming languages. We can draw a parallel with human languages: most of them share a common underlying logical structure, but they differ in the writing, pronunciation & grammar. For us, python is the language (it contains vocabulary and grammar), and the programming concepts form the underlying logical structure common to many programming languages. In this class, you will learn the language (python) and the concepts of programing through examples. Doing one without the other would be a nearly useless academic exercise, it would be like learning music without playing an instrument. The point of this class is that you develop skills you can apply to practical situations right now and in the future.

So, what does it mean to “learn a programming language” (python in our case)? You will generally face two sorts of problems as a scientist: problems where you have made some measurements, i.e. some data from an experiment, and you want to test a particular theory, see if it is in agreement with the data. For instance, imagine you have data describing the motion of planets in the sky and you want to test if Newton’s law of gravity is right and to what precision. For this type of problems, you will “teach” the computer to solve Newton’s law so that it returns you some predictions as to how the planets are supposed to move in the sky given how they move now, and you will compare to your data. Then you also have problems where you want to use physical laws which you know are correct to predict the outcome of a certain physical situation. For instance, as a climate scientist, you want to predict the future of climate for the next thousand years. To do that you have to solve the fluid equations that govern the complex system air +clouds+temperature gradient+water over a large period of time. Only a computer can do this calculation. Solving these problems require the development of the following basic skills:

-be able to read data from a file on a computer and load the value in a program
-be able to generate simulated data
-be able to write data to a file on a computer
-be able to manipulate the data inside a program, do calculations with the data
-be able to visualize your results and/or your calculations

In this class you will learn how to do all of this with python.

Most of you are unfamiliar with the idea of a programing language. You indeed never got a chance to “see” the language hiding behind your smart phone applications, or behind your Windows “click and run” favorite software, or even behind a simple web page. But it is there!!! this is called a **source code**, or **program**. Learning programming literally means learning how to write a source code, i.e. working behind the comfortable window-based environment you are used to. In this class you will spend most of your time coding, i.e. writing python code, text based programs to execute certain tasks.

There is no better way of learning programing than practicing, it is like music and language learning. There is no textbook for this class as the amount and quality of documentation available online is limitless. I will however post on the connect web site all relevant information/documentation/examples we will use in class.

Course Organisation

Lectures/Labs:

Each group of students will meet twice a week (Tue and Thu) with the instructor and TAs, each meeting is 2.5h duration total. The class location is the computers room Hennings 205.

<u>Students are expected to read before class the lecture material posted on connect.</u> During the first half hour I will review the reading material, which will consists in discussing in more details the content of the reading material through examples. The examples are lines of code/command lines I will type and run on my laptop for everyone to see how it really works. Students will be able to follow what happens on my screen from its image projected on the front wall and/or using the online meeting tool www.join.me (I will explain in class how this works).

During this ~half hour lecture, students will be able to edit their own version of the reading material (which they access from connect in a word .docx format) so they can take notes of anything we say and discuss in class. It is a convenient way of taking notes, you can of course takes written notes on paper if you prefer.

During the next 2 hours, students will do the lab work, which consists in solving the problems presented in lab using the material learned that day and before. Two TAs will constantly be on site to help you with the labs.

Marking:

There is no exam for this course. 100% of marking is from lab work. As we approach the middle of the term, lab exercises will become more challenging and the time left for lecturing will decrease. The lab exercises are due before the next lab unless otherwise specified.

Work remotely:

You will be able to work from home, e.g. to complete your lab work. Whether you use a Windows, OS X or Linux Operating System (OS), there is a solution for you to connect to the server hosting your work at UBC. Additionally, Hennings 205 is always open and students can come use the computers any time of the day.

Office hours:

With 5 hours of lab contact per week, there is no formal additional office hours. However, students can write to the instructor and TAs to seek help outside lab hours. The course instructors (which includes me+TAs only) email is phys210instructors@phas.ubc.ca

A face-to-face meeting with the instructor or TAs is also possible by email appointment (see emails addresses above).

Textbook:

There is no textbook for this class. Internet is an infinite source of information when it comes to computing languages, especially with python! I will massively use internet as a source of written
material in the lecture notes.

Every year the UBC Physsoc organizes free Python courses. Check the following website to see when the next one happens:

http://physsoc.phas.ubc.ca/news/pythonlatex-workshops/