Physics 261: Computational Physics — Syllabus

Paul A. Nakroshis

Spring 2021

Physics 261: Computational Physics I is an introductory course on scientific programming using the Python programming language. We will work primarily using Jupyter Notebooks, which allow you to mix text, graphics, mathematical equations (using LaTeX) and, of course Python code, in your reports. By the end of the semester, you should feel comfortable using Python to solve everyday problems, from data analysis to numerical simulation. Along the way, you will gain experience with the linux terminal, and logging into a JupyterHub server.

Introduction

Suppose you take an introductory calculus-based physics course; you will have learned all about Newton's laws of motion and solved many problems with an analytical approach. However, one problem you will not have solved is a simple problem of throwing a ball while including air resistance. One of the reasons you didn't solve this problem is that it's impossible to solve analytically, since air resistance is a non-linear force[1] However, adding air resistance which is proportional to the square of the particle's velocity, while impossible to solve analytically, is not so complicated to solve computationally.

[1] Actually, it's even worse—air resistance is not really a simple function of velocity at all; for low velocities, one can approximate the air resistance as linear, and but as the speed increases, it's not even correct to treat it as a simple function of the velocity to a fixed power.

So, one of the great uses of computation is to be able to solve problems via computer that are difficult or impossible to solve with pen and paper. Such problems can range from solving non-linear differential equations, to systems which involve random processes (for instance, diffusion, or disease propagation), and even the everyday work of processing and visualizing data. In this class, I will introduce you to the numerical python library (numpy) and Matplotlib (useful for visualizing data and functions), and build up your facility with the Python language so that you will walk away with a toolset that you can use throughout your scientific career.

You will notice that this course is being offered as a *Laboratory* course; since 2006, the course has been a 3 credit lecture course, but I wanted to increase the time available in class for helping you debug programs, and also, we wanted to make the course a more comprehensive introduction to the toolkit scientists need for quantitative work.

Background

For over a hundred years, physics departments around the world have taught undergraduate physics by emphasizing the solution of prob-

lems which admit of an analytical solution. However, in recent decades, computer power has increased significantly and a modern laptop has the speed that far exceeds that of a mainframe computer from decades past.

However, given that most undergraduates now own a laptop computer, it makes no sense to restrict our teaching to analytically solvable questions. Many physics departments now have a computing requirement, or even their own computational physics course, but few go beyond this.

At USM, we are modifying our curriculum to include computing throughout our major's degree. The first step in this is to teach computational physics every spring semester, so that entering freshmen have the needed experience to use computation throughout all their upper level courses.

Because computing is important to many disciplines—engineering, linguistics, business, mathematics, statistics, biology, chemistry for example—our hope is that our computational physics course can serve as a computing course that many majors can use to satisfy their degree's computing requirement.

Skills/Learning Outcomes

By the end of the semester, my expectation is that you will gain experience in the following:

1. The Bash Shell (basic commands like ls, cd, mv, rmdir, etc)
2. Logging into a JupyterHub server and using the Jupyter Lab and Jupyter Notebbook interfaces
3. Learning how to install and maintain a working python distribution on your own computer.
4. Uploading data, python scripts, and .ipynb files to the server
5. Using Github for version control
6. Learning the basics data types, and control structures in Python 3.x
7. Reading in and writing data files in multiple formats
8. Become proficient in using Markdown syntax and incorporating basic LaTeX code to typeset mathematics.
9. Become proficient in technical written communication.
10. Learn how to use the Euler, Euler-Cromer, Verlet, and Runge-Kutta methods for solving ordinary homogeneous and inhomogeneous differential equations.

11. Using simulation to solve random systems (random walks in 1 and 2 dimensions, percolation theory)

12. Use python to read in data, make technical plots, perform fits to the data, as well as the uncertainties in the fit parameters.

13. Simulate the motion of simple systems such as projectile motion, planetary motion (with and without air resistance)

14. Explore the behavior of chaotic systems such as a driven non-linear oscillator, or the logistic map

15. Work together on a team to solve a larger programming project, all coordinated using version control.

16. Walk away from class with a computational toolset that can be used for all further coursework, and provides a framework of understanding relevant to a STEM Career and further scientific work.

Required/optional Textbooks

Requires:
A Student's Guide to Python for Physical Modeling, updated edition, by Jesse M. Kinder and Philip Nelson, 2018.
ISBN 978-0-691-18057-1
Optional:
Effective Computation in Physics, A Field Guide to Research with Python, by Anthony Scopatz, Kathryn Huff, 2016, O'Reilly Media
ISBN: 978-1491901533

Atendance Policy

I expect that everyone will be at every class except in extenuating circumstances. In every class, you will likely be working together on a team, and your presence in class is important. If I find that you are missing class too often (i.e. more than three times), you can expect that I will talk with you and that you will likely receive a lower grade for the course, or asked to leave if this repeated absence is coupled with poor quality work written work.

Outside Help/Student Hours

I have a laboratory (room 252 science) and an office cubicle in the C-wing on the second floor. In general, if my office or laboratory door is open, I am happy to help you, so feel free to stop in and ask questions.

I have set aside several hours where I will make a point to be in my office. Please take advantage of my willingness to help you! I can be much more effective one-on-one than I can in front of the whole class.

Grading

I am unfortunately contractually obligated to submit a grade for every student. Grading is subjective, and I do my best to submit a grade which represents my sense of your level of understanding and your level of improvement in understanding for the course.

This course will consist of shorter **assignments** (more toward the beginning of the course) and longer **projects** which will require a formal report. In addition, there may be occasional quizzes, and likely group reports. You will get regular feedback on each assignment and each report, as well as on your final written report. In attempt to make this quantitative, Table **??** shows the grading scheme (late assignments and project reports **will not** be accepted) for the course:

Item	Points
Attendance $\times$ Perceived Effort	100
Assignments* & Projects*	600
Quizzes	100
Final Project	200
TOTAL	1000

Table 1: Grading Scheme for the course. *Late assignments and projects **will not be accepted**.

Items to include in Formal Reports

Some homework will be relatively straightforward exercises; others will be longer projects. On these more formal reports, here are guidelines for your submitted report.

Introduction

The introduction should give an overview of the problem and an indication of where it fits into the subject of physics.

Physics & Numerical Method

Describe the background physics of the problem, and detail the algorithm that is used to solve the problem. This section should also list relevant snippets of your code to show how it is implemented. A full listing of your code should always be attached as the last section of your report.

Verification

Tell what you did to verify that the program gives correct results; this typically involves showing that your code gives reasonable results for

simple cases where an analytic solution is known or obvious. Generally speaking there should be more than one test used to verify program integrity.

Results

Present the results of running the program to demonstrate the behavior of the system under different circumstances. Results might be presented in graphical form or as tables, as appropriate. Be sure that results that are presented are labeled properly, so that the reader can figure out what has been calculated and what is being displayed. Make sure that all figures and tables should have descriptive captions.

Conclusion

Present a discussion of the physical behavior of the system based on your simulations and answer any special questions posed in the assignment.

Instructor Contact Information

Paul A. Nakroshis
Dept of Physics
Lab: Room 252 Science Building
Office: 2nd Floor C-Wing, Science Building
Portland Campus
EMail: pauln at maine dot edu
Web: portlandphysics.me
Github: https://github.com/paulnakroshis
Laboratory: 207-780-4158
Office: 207-228-8045
Student Hours: by appointment

This document & other resources

Should you lose this syllabus, an electronic .pdf version of this file (with clickable hyperlinks) is available online at the course homepage which can be found at
http://people.usm.maine.edu/pauln/physics261.html.
Information about other physics courses can be found at the Physics Department Homepage.