Physics 256 – Introduction to Computational Physics – Fall 2021

Instructor:

Dr. Juan M. Vanegas
E-mail: jvanegas@uvm.edu
Office: E205, Innovation Hall
Office hours: W 1:30 pm – 3:30 pm, or by appointment

Schedule:

Lectures: TR 8:30 – 9:45 am, Lafayette L210
Midterm exam: TBA
Final exam: TBA

Prerequisites:

Phys 152 or Phys 125 (or equivalent), and Math 121 (Vector Calculus)

Course textbook:

Computational Physics, 2nd edition (2005, Pearson).
Authors: Nicholas Giordano and Hisao Nakanishi.
ISBN: 978-0-1314-6990-7

Additional references:

Learning IPython for Interactive Computing and Data Visualization, Second Edition
(2015, Packt Publishing)
Author: Cyrille Rossant
ISBN: 978-1-78398-698-9
http://proquest.safaribooksonline.com/9781783986989?uicode=uvermont

Computational Physics: Problem Solving with Python, Third Edition
(2015, Wiley-VCH)
Authors: Rubin H. Landau, Manuel J. Paez, and Cristian C. Bordeianu
ISBN: 978-3-527-41315-7

Shell Programming in Unix, Linux and OS X, Fourth Edition
(2016, Addison-Wesley Professional)
Author: Stephen G. Kochan and Patrick Wood.
ISBN: 978-0-13-449600-9
http://proquest.safaribooksonline.com/9780134496696

Grading:

- Weekly homework assignments 30 %
- Midterm exam 20 %
- Final exam 20 %
- Semester programming project 20 %
- In-class attendance/participation 10 %

In-class work and homework:

This class has a large participatory programming component. You will learn by attacking problems in real-time in class, either in groups or by yourself. Please come prepared to lectures with a laptop or other suitable portable computing device. Students are encouraged to work together, but you must write and turn in your own code. Do NOT share your code with other students in the class. Academic dishonesty will not be tolerated! No credit will be given for late assignments.

Midterm and final exams

The exams for this class are **pen and paper**, no computers involved! The point of these tests is to assess your knowledge of the methods used to solve the computational problems, and strategies for converting complex mathematical equations into pieces that the computer can solve.

Graduate Credit

Graduate students enrolled in the course will be required to answer additional homework problems as well as choose a more advanced topic for their final project.

Course plan (order and specific topics may change):

1. Introduction to python and Jupyter notebooks.
2. Summations, loops, and cancelation errors.
3. Error assessment and introduction to Numpy.
4. Statistical and continuous radioactive decay.
5. Air resistance and projectile motion.
6. Simple harmonic motion.
7. Chaotic motion and the chaotic pendulum.
8. The logistic map.
9. Advanced visualization using Matplotlib and Scipy.
10. Random walks and the diffusion equation.
11. Random numbers.
12. Root finding and optimization.
13. Electrostatic potentials in 2 and 3 dimensions.

14. Numerical quadrature and magnetic fields.
15. Ising model, mean field theory and Metropolis-Monte Carlo.
16. Waves on a string.
17. Spectral analysis and Fourier Transforms.
18. Linear regression and data fitting.
19. Integration, Simpson's rule and Monte Carlo.
20. Introduction to finite element methods in python with FEniCS.

Semester Project Guidelines

The semester project is worth 20% of the total semester grade (10% for the project itself and 10% for the presentation).

Example topics:

- Solving the Schrodinger equation using Monte Carlo methods
- Simulation of n-body systems interacting through gravitational forces
- Diffusion limited aggregation in 2 and 3 dimensions
- Solving stochastic differential equations for population dynamics
- Gravitational field on the surface of an asteroid
- Mean squared displacement and diffusion dynamics
- Molecular dynamics simulations
- Ising model

Programming project:

- Grading:
    - Legible, well documented code that **runs** - 5%
    - Brief write up explaining the basic problem, the computational approach used to solve it, and Data visualization showing expected results - 5%
- The programming component of the semester project should be considerately more than for a typical homework assignment.
- If possible I would like to be able to run your code and see that it runs and produces expected results. For projects where the code is written in languages other than python or requiring other packages, please provide enough evidence that your code works (input/output). If you wrote a modification/extension to somebody else's code, please make sure to clearly label your own contributions.

Project Presentation:

- Grading:

- Introduce the problem clearly, explain the methods used, and how the problem was solved 5%
- Briefly explain how your code works, what are the expected results, and visualization of results 5%