Computational Modules for the Physical Sciences

Instructor's Guide

Introduction

These modules, developed under a grant from the TIDES (Teaching to Increase Diversity and Equity in STEM) project of the Association of American Colleges & Universities, are designed to teach the basics of algorithmic thinking and computer programming to students in the physical sciences using the Python programming language. It is hoped that the set of modules will work as a largely independent learning experience in basic programming skills for science students; of course, the set also could serve as the core content or a supplement to the main content of a formal course. In fact, they were tested out on and refined with feedback from students who used them in a formal computational methods course (PHYS 350) offered at Bryn Mawr in the spring of 2015. (That course was run essentially as a computational lab: students were asked to do readings in the modules ahead of class time, and they worked on the embedded exercises during class. Lecturing was kept to a very bare minimum. Help was available from both an instructor and a TA, but the students usually looked to their classmates for help, and that was encouraged.)

A secondary goal of the set of modules is to attract to computing and computational science students from groups typically underrepresented in those fields. One approach to that goal is to give students an appreciation of the wide range of individuals who have contributed to the development of computational science, or who have used it in interesting or important ways. This approach is implemented in each module by the inclusion of a scientist profile, most of which were developed by the students in PHYS 350. A second approach, which was employed at regular intervals in that course, was to ask students a "reflection question" meant to encourage them to think about how science and computation was relevant to them and could serve their personal goals. We asked the students in the course at the end of the semester to submit an "e-portfolio" including all their work in the course: module exercise solutions, their scientist profile, responses to reflection questions, and term project write-ups. The guidelines we provided the students regarding the profiles, and the reflections questions/prompts we asked them to respond to (as well as ones added more recently), are provided at the end of this document.

The set of modules consists of 17, numbered 00-16, but Module 1 – an introduction to programming in Python – is divided into three parts, labeled 01A, 01B, 01C. Below, you will find brief overviews of the topics covered in each module, with estimated times for student completion. A detailed table of contents for the modules, a dependency tree, and a list of scientists profiled in the modules are provided later in this document. (Most of the earlier modules are based on chapters in the outstanding textbook *Computational Physics*, by Mark Newman.)

Overview of the Modules and Suggestions on Their Use

Module 0 provides motivation for the modules and an introduction to computing, as well as many useful resources; as a result, it is strongly recommended that students read it. There are no associated exercises, so this module is a quick read. This module exists only as a pdf file.

Modules 01A, 01B and 01C are essential, as they provide a student with the knowledge of Python they will need to work through the subsequent modules. Part A takes students roughly 3-5 hours to read and work through the seven exercises; Part B has four exercises and takes 2-4 hours, Part C has five exercises and takes approximately 3-5 hours to complete. Since they provide the foundation for the students' future computational work, this trio of notebooks should not be rushed through.

Module 02 introduces the topics of numerical errors and computational speed. Not a lot of this material is used later, so if time is limited students could be asked simply to skim this module (which does not contain any exercises). This module takes around 2-4 hours for students to read and work through the Breakpoints – quick checks of the student's understanding.

Module 03 represents the start of the development of scientific computation, introducing Euler's method in the context of one-dimensional motion (with both constant and varying forces). As with the three parts of Module 01, this module takes about 3-6 hours to complete, and it serves as a gentle introduction to code development.

Module 04 introduces the topic of finite difference methods as the numerical approach to differentiation. Both linear and higher-order approximations to the first derivative are presented, and higher-order derivatives are discussed briefly, as is interpolation. This module also takes 3-6 hours to complete.

Module 05 covers numerical integration, including the trapezoidal rule, Simpson's method, higher-order methods, and Gaussian quadrature. It also briefly discusses numerical integration over infinite ranges, and multiple integrals. This module takes 4-6 hours to complete.

Module 06 describes the numerical solution of linear equations, using the techniques of Gaussian elimination with back-substitution, and LU decomposition, also briefly discussing other decomposition methods. This module is estimated to take 3-6 hours for completion, depending in part on the student's familiarity with techniques of linear algebra.

Module 07, which may be thought of as an extension of Module 06, introduces the QR decomposition method for finding the eigenvalues and eigenvectors of a matrix. Several applications are presented, including finding principal axes of inertia of a solid body, solving for the normal modes of coupled harmonic oscillators, and determining the eigenenergies of a

quantum system. It is estimated that this module will take 5-8 hours for students to complete, again depending in part on their familiarity with linear algebra.

Module 08 presents techniques for analyzing data. In the main part of the notebook, linear least-squares fitting is discussed. Appendices present the basics of singular value decomposition and principal components analysis. The module also introduces the Pandas library for data analysis, as well as Bokeh for data visualization. This module was written, in part, to address students' interest in learning about "big data." Since that topic means very different things to workers in different disciplines, this module is deliberately shallower than others, but it addresses topics more likely to be relevant to undergraduate science students. This module takes roughly 3 hours to complete.

Module 09 presents the numerical treatment of Fourier analysis. Students who have little or no familiarity with the topic might need to do some background reading before attempting the module, as it runs through the underlying mathematics quickly. The module introduces the Fourier series and then transitions to the discrete Fourier transform. Two-dimensional transforms and the discrete cosine transform are covered briefly. The fast Fourier transform is presented with most of the details omitted. The module takes approximately 4-6 hours to complete, not counting time that may be required for background reading.

Module 10 covers the numerical solution of differential equations, starting with first-order equations of one variable. The Euler's and Runge-Kutta methods are presented. Second-order equations of one variable are covered next. The module concludes with a discussion of boundary value problems solved using the shooting and relaxation methods. The time estimate for completion of this module is 3-5 hours.

Module 11 discusses the solution of partial differential equations, looking at both boundary-value and initial-value problems, and introduces the finite difference, Jacobi, Gauss-Seidel, forward-time centered-space, and Fourier transform methods. The estimated completion time is 2-4 hours.

Module 12 presents random numbers in numerical computations, and the basics of the Monte Carlo method. That method is applied to a simple integration problem, for which the mean value method, importance sampling, and the transformation method are introduced. Next, Monte Carlo simulations are discussed, and the Markov Chain Monte Carlo method is described. These techniques are applied to the Ising model, and simulated annealing is briefly presented. This module takes 3-6 hours for completion.

Module 13 is an overview of symbolic computing in Python. It covers basic evaluation of symbolic expressions, symbolic computation of derivatives and integrals, evaluation of limits, power series expansions, finding roots of equations, solutions to simultaneous equations, the

solution of both ordinary and partial differential equations, and matrix operations. The module should be quick to read through, and there is only one exercise to provide a little practice on some of the operations.

Module 14 is a brief introduction to object-oriented programming, applied to the physical problem of masses connected by springs. A specific application is shown to oscillations of a CO_2 molecule. The approximate completion time for this module is 1-2 hours.

Module 15 presents the basics of parallel computing and introduces several Python packages for parallel computing: multiprocessing (and its Jupyter notebook-friendlier variant, multiprocess), concurrent.futures, joblib, ipyparallel, and dask. (Brief mention also is made of the cuda and numba packages used for parallel computing with GPUs.) Two example tasks are parallelized using some or all of the five packages: squaring a set of numbers (an "embarrassingly parallel" task) and finding prime numbers. Note that the behavior of the five packages is somewhat platform-dependent, and the two examples might not work in all packages on every machine. This module contains only one exercise. Reading through the module and working the examples should take 1-2 hours, but the exercise could take several hours, especially if students did not complete Mastery Exercise #3 in Module 3 – modeling a particle bouncing inside a 2-D box.

Module 16 is an introduction to machine learning with a focus on artificial neural networks (ANNs). Three packages for implementing ANNs are introduced – PyTorch, TensorFlow, and scikit-learn – and each is used to perform the standard task of categorizing handwritten digits. Only multilayer feedforward networks with perceptrons are presented, as this field is extremely broad and one module could not possibly cover the full range of approaches to ANNs, not to mention machine learning more generally. Estimated reading time is 1-1.5 hours. The exercise is very similar to the example analyzed in the module but using a different standard data set. It should take about an hour or less.

IMPORTANT: All the modules have embedded images that are automatically loaded into the notebooks if the images are stored in the right location. They must be in a folder/directory named Images located in the same folder/directory as the modules themselves. The Images.zip file contains those images. A couple of the modules also make use of supplementary files, located in the Supplements folder in the PICUP folder.

If you have questions about the modules or find errors in them, please contact Mark Matlin, mmatlin AT brynmawr.edu.

Scientist Profile Instructions and List of Scientist Profiles

Profile Instructions

Our objective for the Profile Collection is to gather contemporary (last ~50 years) stories of individuals who have contributed to the development and use of computing techniques in the sciences. We would like the collection to showcase the achievements of people from a diversity of backgrounds and the ways in which their stories inspire us. Particularly, those who may have overcome challenges or whose accomplishments may have been overlooked. Please address these questions in your profiles:

- Who is the person and what is their context? Tell us about their education background, i.e., where and what did they studied, and something about their career path.
- How does their work connect to computational techniques and/or scientific computing?
- What is inspirational about their story? Please prepare two depictions of your profile:
- A written text document with a visual element and a description that includes, but is not limited to, the answers to the three questions above. Please include any references that you found on the profiled individual. Length ~250 words.
- A single slide to support a 5-minute presentation of your profile. Be creative! And feel free to include material beyond the three questions if you like.

List of Profiles

Module 00: None
Module 01: None
Module 02: Ada Lovelace
Module 03: Jean Bartik
Module 04: James McLurkin
Module 05: Annie Jane Easley
Module 06: Ellen Ochoa
Module 07: Mark Dean
Module 08: Jeannette Wing
Module 09: Margaret Hamilton
Module 10: Fred Begay
Module 11: Grace Hopper
Module 12: Luis von Ahn
Module 13: Chieko Asakawa
Module 14: Barbara Liskov
Module 15: Anousheh Ansari
Module 16: Alan Turing

Reflection Prompts

One-time prompts (all but the last would be best administered early in the use of the modules)

- Values affirmation: Write a brief statement about 2-3 things that you value and care about, and why.

- Growth mindset: Do you think that mental and academic abilities are mostly something we are born with or something that we can develop? Please watch this 11-minute video https://www.youtube.com/watch?v=pN34FNbOKXc and write about what you take away from it. (Embedded in Module 3.)

- Growth mindset: What feedback would you give someone on a task they performed that would help to promote a growth mindset?

- Describe a time when you overcame a struggle in learning (e.g., an academic idea, a physical skill, a musical instrument). Write it up in a paragraph that you would be willing to share (anonymously, if you wish) with someone else working through these modules.

- In what ways do you think learning computer programming might be useful to you in the future?

- What connections do you see between the modules and other classes/research activities/ interests of yours?

Prompts included in most of the modules

- Which components of this module did you find you were easily able to work through, and why do you think they were especially easy for you?

- Which components of this module did you find more difficult to work through, and why do you think they were challenging?

- When you got stuck, what did you do to get unstuck? Could this or similar actions be helpful if you get stuck in future work?

- What do you understand more deeply about this material?

- What questions or uncertainties remain for you regarding this material?

Table of Contents of the Modules

Module 2: Numerical Errors and Computational Speed

- Numerical Errors
- Computational Speed and Big-O Notation
- Vectorization
- Profiling

Module 3: Iterative Methods

- One-dimensional Motion without Drag
- Two-dimensional Projectile Motion without Drag
- Two-dimensional Motion with Drag

Module 4: Differentiation and Interpolation

- Definition of the Derivative
- From Differences to the Derivative
- Higher-order Approximations to the First Derivative
- Higher-order Derivatives
- Interpolation

Module 5: Integration

- The Trapezoidal Rule
- Simpson's Method
- Choosing the Number of Steps, N
- Higher-order Methods
- Gaussian Quadrature
- Comparison of Integration Methods
- Integration over Infinite Ranges
- Multiple Integrals

Module 6: Solution of Linear Equations

- Gaussian Elimination with Back-Substitution
- LU Decomposition
- Other Decompositions

Module 7: Eigenequations

- Eigenvalues and Eigenvectors
- Applications
  - Principal Axes of Inertia
  - Coupled Harmonic Oscillators
  - Energies of a Quantum System

Module 8: Analyzing Data

- Linear Least-Squares Fitting
- An Introduction to Pandas for Data Analysis
- Visualizing Data with Bokeh
- Appendix 1: Singular-value Decomposition
- Appendix 2: Principal Components Analysis

Module 9: Fourier Analysis

- The Fourier Series
- The Discrete Fourier Transform
- Two-Dimensional Fourier Transforms
- The Discrete Cosine Transform
- The Fast Fourier Transform

Module 10: Differential Equations

- First-Order Equations of One Variable
  - Euler's Method
  - Runge-Kutta Method
- Second-Order Equations of One Variable
- Boundary Value Problems
  - The Shooting Method
  - The Relaxation Method

Module 11: Partial Differential Equations

- Partial Differential Equations
  - Boundary Value Problems
  - Initial Value Problems
- Other Methods

Module 12: Monte Carlo Methods

- Random Numbers in Numerical Computation
- Monte Carlo Integration
  - The Mean Value Method
  - Importance Sampling
  - The Transformation Method
- Monte Carlo Simulations
  - The Ising Model
  - Simulated Annealing

Module 13: Symbolic Computing in Python

- Starting with `sympy`
- Evaluation
- Derivatives

- Integrals
- Limits
- Power Series Expansions
- Equation Roots
- Simultaneous Equations
- Differential Equations
- Matrix Operations

Module 14: A Brief Introduction to Object-Oriented Programming

- Why Use Object-Oriented Programming?
- The Idea Behind Object-Oriented Programming
- A Simple Example
- A More Sophisticated Example
  - Two Balls Connected by One Spring
  - Three Balls Connected by Two Springs
    - An Atomic Example
    - A Simplification

Module 15: Parallel Computing

- Introduction
- The `multiprocessing` (`multiprocess`) package
- The `concurrent.futures` package
- The `joblib` package
- The `ipyparallel` package
- The `dask` package
- `cuda` and `numba`

Module 16: Machine Learning

- Basics of Neural Networks
  - What is a Neural Network?
  - Neuron Inputs and Outputs
  - The Learning Process
- `PyTorch`
- `TensorFlow`
- `scikit-learn`

Module Dependencies