Course Syllabus

Computational Astrophysics

ASTR 615
Spring 2023

Dr. Massimo Ricotti
ricotti@umd.edu

Class Meets
Tu&Th 12:30–13:45 pm

Where:
ATL 3426

Office Hours
TBD

Prerequisites
ASTR615: permission of Astronomy Dept.

Course Communication
Any time-sensitive information will be delivered by *ELMS* announcement. You can reach out via *ELMS* or by email to discuss questions, absences, or accommodations. For help in writing professional emails, visit ter.ps/email.

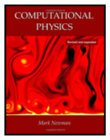
Learning Outcomes

This course introduces computational methods relevant to both observers and theorists engaged in modern astrophysics research. Although the skills learned are applicable to many areas of study, most examples will be drawn from astrophysics problems that demand numerical approaches to solve.

After successfully completing this course, you will be able to:

1. Solve computational astrophysics problems using Python.
2. Define a computational project in terms of its requirements, input and output data, and format for delivery of results.
3. Work collaboratively on a project by assigning and sharing tasks and critically evaluating the work of others.
4. Evaluate, in written and oral presentation form, a major computational astrophysics method.

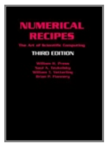
Required Resources



Mark Newman *Computational Physics*
First Edition (Revised and Expanded, 2013).
ISBN: 978-148014551-1

Course website: https://umd.instructure.com/courses/1285644

Optional Resources



W.H. Press et al. *Numerical Recipes*
Third Edition (2007).
ISBN: 978-052188068-8

Also, free resources found online by a google search or equivalent.

Campus Policies

It is our shared responsibility to know and abide by the University of Maryland's policies that relate to all courses, which include topics like:

- Academic integrity
- Student and instructor conduct
- Accessibility and accommodations
- Attendance and excused absences
- Grades and appeals
- Copyright and intellectual property

Please visit www.ugst.umd.edu/courserelatedpolicies.html for the Office of Undergraduate Studies' full list of campus-wide policies and follow up with me if you have questions.

Overview

This course provides practical knowledge for solving complex astrophysical problems using computers. We will adopt Python as our computing language and follow the textbook *Computational Physics* closely. Python is the language of choice for many astronomers (and physicists) and is ranked #1 in IEEE Spectrum's *The Top Programming Languages 2020*. The course begins with a background to Python, including the numerical library NumPy and visualization packages Matplotlib and VPython. Following this, we cover a variety of topics relevant to computational astrophysics in turn: generating random numbers, integration and differentiation, function solving, estimating model parameters, Fourier transforms, parallel computing, deep learning, and solving ordinary and partial differential equations. We will use the gravitational *N*-body problem and fluid dynamics as practical examples of the last two subjects. See the Schedule for all topics covered and the Appendix for the course design rationale.

Activities, Learning Assessments, and Expectations for Students

Most classes will have readings and short online quizzes assigned beforehand. These assignments will be posted on ELMS-Canvas (hereafter ELMS) no later than the previous class. You are expected to join each class prepared to discuss the readings and to work on related problems in a collaborative manner. A large portion of your course grade will be based on these pre- and in-class activities. Pre-class quiz solutions will be revealed on ELMS after 5 pm on the day of class. Where applicable, in-class exercises will be graded using the rubric shown on ELMS with the corresponding assignment. Most in-class activities will require use of an internet-capable computer. Please ask before using symbolic solvers like Wolfram|Alpha and Mathematica for pre- and in-class exercises (although feel free to check your answers afterwards).

Many activities, both during class and for homework assignments, will be done in teams of 2 or 3 students each. Teams will be assigned after the first class. Normally team activities shall not be carried out by just 1 student, except in special circumstances with my prior approval. Learning in teams is often more effective than learning alone, particularly when students have a range of relevant backgrounds (which is likely for this course) but does come with important responsibilities. You must strive to do an equal share of the work and to support your teammate(s) in their learning. Any issues should be addressed to me. Grades assigned to teams are assigned equally to team members, however I reserve the right to lower the grade of persistently uncooperative members. For in-class activities, absent team members will receive a zero score unless the absence is excused.
The major learning assessments will be 5 homework assignments, 3 in-class timed quizzes, and activities related to the ASTR615 Project (see below). There are no exams in this course. All homework must be submitted prior to class on the day it is due or be assessed an automatic 20% penalty. Late homework must be submitted by the beginning of the following class to receive credit. Late penalties may be waived under extenuating circumstances (see the University course-related policies above).

The homework assignments consist largely of designing, running, and analyzing the results of computer code. **We will be using the Python programming language in this course, but since a previous version of this class was exclusively in C, I am open to discussing also on an individual basis, using C instead of python for your assignments and/or exercises.** A basic requirement is that coding assignments turned in for assessment must be runnable "out of the box," meaning, they should be well-documented and have no special software dependencies that I would have to resolve, unless a special exception is granted in advance. As part of the collaborative learning in this course, teams will exchange their assignments for testing prior to final submission to me (a worksheet will be provided for this purpose). It is permissible to use existing algorithms, either by typing them up or importing an applicable package, as part of a solution to an assignment, but, where warranted, sources must be cited, with appropriate comments in the code.

We will be using GitHub for working collaboratively on and submitting homework assignments. More information about this—including the full grading rubric for homework assignments, and the peer-review worksheet—is on ELMS (under *Pages → Homework Information*). You may also find the free-to-use Repl.it platform (repl.it) or Visual Studio Code Live Share convenient for coding with your teammate(s).

Regarding absences, your 2 lowest pre-class scores will be dropped automatically. A liberal excused-absence policy (no documentation required) will be used for up to 2 missed in-class activities, except for activities related to the ASTR615 Project. Documentation is required to excuse any additional graded items, in accordance with the University policies referenced above.

The in-class timed quizzes are meant to serve as low-bar assessments of individual coding and problem-solving capability. They will require use of an internet-capable computer.

The ASTR615 Project consists of a written report and an oral presentation given by students taking ASTR615 for credit (see next section). More information about the ASTR615 Project will be provided later in the course.

The course schedule is maintained on ELMS. In the event of a prolonged university closing, or an extended absence from the university, adjustments to the course schedule and assignment deadlines will be made based on the duration of the closing and the specific dates missed.

Personal Devices

As indicated above, you will need an internet-capable computer in order to work collaboratively on in-class assignments. You may also use any other applicable personal device, such as a tablet or smartphone, to assist with meeting the objectives of this course. You are expected to remain on task during class and to limit distractions to both yourself and your fellow students.

Grades

Your grade is determined by your performance on the learning assessments in the course. Grades will not be curved. If earning a particular grade is important to you, please speak with me at the beginning of the semester so that I can offer some helpful suggestions for achieving your goal.
All assessment scores will be posted on the course ELMS page. I am happy to discuss your grades with you, and if I have made a mistake, I will correct it promptly. Any formal grade disputes must be submitted in writing and within one week of receiving the grade.
The following table shows the weights for the various learning assessments in this course.

Assessment	Weight
Homework (5 assignments)	35%
In-class Timed Quizzes (3)	10%
In-class Activities	20%
Pre-class Activities (most classes), 2 lowest scores dropped	20%
Final term project	15%

Multiple graded items of the same type within each category above have the same weight (for example, if there are *N* homework assignments, each is worth 1/*N* of your total homework grade).

Grades are always rounded to the nearest integer, with 0.5 rounding up to 1.

The following boundaries will be used to compute letter grades:

+	97%	+	87%	+	77%	+	67%		
A	**93%**	**B**	**83%**	**C**	**73%**	**D**	**63%**	**F**	**< 60%**
–	90%	–	80%	–	70%	–	60%		

For this course, letter grades correspond to the University's marking system.

There may need to be some adjustment to scores depending on the class average. However, any adjustment will be to lower the grade boundaries given above, never to raise them.

There may be occasional opportunities for extra credit. These will be announced in class or via ELMS. These opportunities apply to everyone in the course; there will be no special extra-credit opportunities for individual students.

Course Evaluation

Your participation in the evaluation of courses through CourseEvalUM is a responsibility you hold as a student member of our academic community. Your feedback is confidential and important to the improvement of teaching and learning at the University as well as to the tenure and promotion process. Please go to the website to complete your evaluations during the final two weeks of the semester. By completing all of your evaluations each semester, you will be able to access online the evaluation reports for the hundreds of courses for which 70% or more students submitted their evaluations. You will be reminded about this toward the end of the semester.

A Safe Learning Environment

The campus is meant to be a safe place to learn, free from harassment and intimidation of any kind. If you have experienced any form of harassment as a member of the University community, you should contact the Office of Civil Rights & Sexual Misconduct on campus. Please be aware that faculty are required by law to report any instance of sexual misconduct brought to their attention. For confidential assistance, contact CARE. Also, the Equity, Diversity, and Inclusion committee in the Department of Astronomy maintains a useful resource page.

In this course, all viewpoints and backgrounds are welcome, but everyone should feel empowered to call out problematic behavior, including my own, and in teams.

Names/Pronouns and Self Identifications

The University of Maryland recognizes the importance of a diverse student body, and we are committed to fostering inclusive and equitable classroom environments. I invite you, if you wish, to tell us how you want to be referred to both in terms of your name and your pronouns (she/her, he/him, they/them, etc.).

The pronouns someone indicates (mine are he, him, his) are not necessarily indicative of their gender identity. Visit trans.umd.edu to learn more.

Additionally, it is your choice whether to disclose how you identify in terms of your gender, race, class, sexuality, religion, and dis/ability, among all aspects of your identity (e.g., should it come up in classroom conversation about our experiences and perspectives) and should be self-identified, not presumed or imposed. I will do my best to address and refer to all students accordingly, and I ask you to do the same for all of your fellow Terps.

Additional Help

Taking personal responsibility for your own learning means acknowledging when your performance does not match your goals and doing something about it. I hope you will come talk to me so that I can help you find the right approach to success in this course, and I encourage you to visit tutoring.umd.edu to learn more about the wide range of campus resources available to you. If you would like to improve your writing and communication skills, consider scheduling an appointment with the campus Writing Center. For help in writing professional emails, visit ter.ps/email. You should also know there are a wide range of resources to support you with whatever you might need (see go.umd.edu/assistance), and if you just need someone to talk to, visit counseling.umd.edu or one of the many other resources on campus. Most services are free to use because you have already paid for them, and **everyone needs help**…all you have to do is ask for it.

Schedule

Below is the tentative schedule for this course, including the event or due dates for the major learning assessments. Any changes will be announced on our ELMS page.

##	Date	Topic	Major Assessment
01	Thu Jan 26	Introduction	
02	Tue Jan 31	The Command Line	
03	Thu Feb 2	Python Basics	
04	Tue Feb 7	NumPy	
05	Thu Feb 9	Visualization	
06	Tue Feb 14	Homework 1 Peer Review	Quiz 1
07	Thu Feb 16	Accuracy and Speed	
08	Tue Feb 21	Random Numbers	Homework 1
09	Thu Feb 23	Integrals	
10	Tue Feb 28	Monte Carlo Integration; Derivatives	
11	Thu Mar 2	Linear Equations	
12	Tue Mar 7	Homework 2 Peer Review	Quiz 2
13	Thu Mar 9	Nonlinear Equations	
14	Tue Mar 14	Least-Squares Fitting	Homework 2
15	Thu Mar 16	Ordinary Differential Equations 1	
	Tue Mar 21	*No Class (Spring Break)*	
	Thu Mar 23	*No Class (Spring Break)*	
16	Tue Mar 28	Ordinary Differential Equations 2	

17	Thu Mar 30	N-body Techniques 1	Project Prospectus
18	Tue Apr 4	N-body Techniques 2 (& Homework 3 Peer Review)	
19	Thu Apr 6	Partial Differential Equations 1	
20	Tue Apr 11	Partial Differential Equations 2	Homework 3
21	Thu Apr 13	Fluid Dynamics 1	Project Draft
22	Tue Apr 18	Fluid Dynamics 2 (& Homework 4 Peer Review)	
23	Thu Apr 20	Fourier Transforms 1	
24	Tue Apr 25	Fourier Transforms 2	Homework 4
25	Thu Apr 27	Deep Learning	Project Final
26	Tue May 2	Parallel Computing 1	
27	Thu May 4	Homework 5 Peer Review	Quiz 3
28	Tue May 9	Parallel Computing 2	
29	Thu May 11	ASTR615 Project Presentations	Homework 5*

Copyright Notice

The material in this class and the class structure has been adapted from Derek Richardson's Astr415/615 class. Class materials may not be reproduced for anything other than personal use without permission.

Appendix: Course Design Choices

A course on computational astrophysics can be many things. My goal is to build student confidence in tackling the kinds of numerical problems likely to be faced in the field by introducing fundamental tools and techniques of the discipline. Still, there are many design choices to be made for accomplishing this goal. Here is a short rundown on my current choices, given in FAQ form.

What is this course *not*?

This is not a computer science course. We will use computer code as a tool. You are encouraged to follow best coding practices to the extent possible, and guidance will be provided. Even better is to use object-oriented programming techniques, such as classes and inheritance, but these more-advanced approaches will not be taught in this course, except in passing.

Why Python?

Python remains the most popular modern programming language and is used increasingly in astrophysics. The language is versatile, easy to learn, and portable. For astrophysics, it is essential to include the popular and well-maintained add-on packages NumPy (numerical Python), SciPy (scientific Python), and Matplotlib (graphing). Although C and FORTRAN (and to a lesser extent C++), along with proprietary languages like IDL and MATLAB, are used in astrophysics, these are needed these days mostly for performance (C/C++) or legacy reasons. Speaking of legacy code, there is a lot of astrophysics-related Python 2 code out there, but you should be learning Python 3, as NumPy, SciPy, and Matplotlib are no longer being updated for Python 2.

Why use the command line?

Most computational astrophysics applications, including data reduction tools, require familiarity with the command line, which these days typically means having a Linux-like interface to the machine operating system. Although modern IDEs (Integrated Development Environments) provide high-level functionality that can hide the command line, you will be at your most efficient and versatile by learning a handful of powerful shell commands.

Why Git?

Most modern collaborative coding projects use some form of command-line-based version-control system, and these days the most popular is Git (a successor to solutions like CVS and Subversion). Although it may be overkill in our course, Git is a convenient way of synchronizing course projects in one place that also simplifies peer review and grading, without having to email files around (for example). As a bonus, it provides a backup in case your work is lost or corrupted. Having some experience with Git could be an advantage if you continue on in a computation-oriented field of study.

Why GitHub?

On its own, Git is a local solution to version control, so sharing your work beyond your local environment requires also setting up a server. GitHub is one of several free Git-based cloud solutions that simplify sharing by providing a universally accessible Git server (others include Bitbucket and SourceForge; GitHub is the most widely used, which motivated its selection for this course). As a bonus, Git cloud servers provide off-site backup of your coding projects.

Why *Computational Physics*?

I find this textbook strikes the right balance between techniques and applications without going into excessive detail, so it remains readable. As a bonus, it uses Python 3 exclusively for implementations. The most recent edition is from 2013, so it lacks recent Python development. Where appropriate, I have added “MR’s Take” notes for each reading from the textbook that point out some of these issues. The much-denser *Numerical Recipes* (3rd Edition) is a good reference for deeper algorithm discussions.

Why these particular topics?

We cannot cover all possible computational astrophysics topics in a single course. The choices offered are a compromise between essential elements and interesting new developments. We focus on ODEs and the *N*-body problem since they are my own specialty, and on PDEs and fluid dynamics because they are a natural extension of ODEs that lead into a rich area of current study in astrophysics.

Why are the classes flipped?

Students come into this course with a wide range of computing background. By reading in advance of class, you can budget your time with the preparatory material in a way that works for you. Time in class is then spent mostly doing practical exercises, with both myself and the TA there to assist. The best way to become comfortable with coding is to practice it as much as possible, which is what we will do.

Why are we in teams?

In many instances working in a team can improve individual understanding as you collectively arrive at a solution to a problem. It also makes coding and debugging more efficient. Finally, teamwork builds important skills you may need in your future career.

Addendum by MR: Is a Computational Astrophysics Class in C forever gone?

I have taught Computational astrophysics for undergraduates (ASTR 415) and graduate students (ASTR 615) for many years now. My version of this class was entirely in C (with python only used for plotting). This class is a revamped version of the old class fully using python. The material has been adapted and developed by Derek Richardson.
I have always been resistant to adopting only python for this class as in python you can solve most problems with a few lines of code and it's hard to see what is inside the python functions, while in C (Numerical Recipes in C, that is free online was the book of reference) you can see the source code and how the algorithm works. So, even if you use a pre-written C function/subroutine, you can make a connection between the code/exercise and the theory/math behind the algorithm.
It is however undeniable that most of the problems in astrophysics (unless you run supercomputer simulations) can be easily solved using python. Also, most students have some previous exposure to python, while it is becoming increasingly rare for Astronomy student to already have had experience with C. That said, during my lass class in C (ASTR 415, Fall 2022) I had the following feedback: while I had the feeling that most student initially hated to have to learn a new language (C, that is significantly harder to learn than python), by the end of the class most of them appreciated doing the effort and felt that they learn a lot. Several students also strongly encouraged me to not switch to python for this class.

I am happy to try using python for this class, but if you feel that you want to be challenged in using C, I can try to accommodate a hybrid class in C/python. This is something I have never tried before and it may not work well. But I am willing to have a conversation, get feedback from you, and accommodate the class structure according to your preference.