This is a Draft. More details soon.

Course goals:

Mathematical modeling is the very heart of applied mathematics. Presented with a new problem, we need to figure out how to formulate it in mathematical terms; how to decide if the formulation is correct; and how to use the formulation to do something useful. Applied Mathematics has been described as a â€œno holds barredâ€ competition to solve problems and get to the truth. Recent events from the rise of chatbots to the management of the Covid pandemic have mathematical modeling at their core: In the case of chatbots, we finally have arrived at a model of language that is sufficiently robust that we can reliably generate it â€“ this is fundamentally an advance of modeling, though its instantiation was enabled by incredible technology; in the case of covid, mathematical models presaged the intensity of the pandemic and helped to manage it â€“ yet in many ways they also failed in making accurate enough predictions to be useful.

Mathematical modeling is hard. It requires that we creatively invent equations, and that the equations can be solvable and useful. Doing it well is an art form. **Evaluating models** â€“ testing how well they work â€“ is extremely important.

Arguably the major advances that have occurred in mathematical modeling over the last decade are advances in systematic evaluation and iterative improvement of models.

The goal of this class is to make you into a mathematical modeler, using everything that you have learned to create models of things that are of interest to you. The mathematics that can be applied is vast, and so as part of becoming a modeler, you need to learn to be comfortable quickly assessing mathematics that you do not yet know â€“ and then figuring out what you need for the problem at hand. At the same time, sophisticated modeling in 2024 requires software engineering -- at a level that is beyond the AM undergraduate curriculum. To address this, we are now introducing a **hybrid undergraduate and graduate class** (Applied Math 115/215) where the undergrad component will focus on the essentials of mathematical modeling, and the graduate class will *also* build upon this to learn how to implement these models using modern software engineering skills (which we will teach in additional lecture each week to the graduate students). Our hope is that this hybrid class will end up as a community where people with different skills can work together and make amazing models of interesting things.

To get us to this point, the class will survey examples of areas where mathematical models are important: we will simultaneously focus on *interesting questions* (what is the probability that the best team wins the world series? Should a fast food restaurant offer a discount to their customers? How many people can the earth support?), while at the same time learning mathematical ideas that are generally useful. Fundamentally, this course approaches a number of problems *without the prejudice* of trying to apply a particular method of solution.

The examples we will study, along with associated questions we might address,include:

- **Sports Analytics:** When do we expect a person to break the 2 hour mark in a marathon? What is the optimal pace to take at different parts of a race? Given the draw of a tournament (tennis, march madness, etc), and everything you know about the teams and players, please predict who will win. Assign probabilities. Evaluate potential solutions in a mathematically same way.
- **Random Walks, Diffusion and Markov processes**. Models of stock and option pricing. Random walks on graphs: Page rank and web search. Diffusion in the atmosphere: Pollution forecasting, and attributing carbon emission.
- **Predicting elections**. Can we predict in advance the outcome of an election given the properties of the voters? Can we devise advertising strategies to affect the outcome?
- **Queuing problems.** Design the algorithm for elevators in a large building with many people. Design the algorithms for traffic lights. Design Amazonâ€™s logistics chains. Should a fast food restaurant offer discounts to attract customers?
- **Create a Hedge Fund.** Given the data for stock or other asset prices over time, create a mathematical model that powers a notional hedge fund. Evaluate this model against performance of similar funds in the past and assess how well are you doing.d
- **Simple models of climate**. Why is the climate of the earth what it is? Can we estimate in simple terms when and if and how it will change? What do actual climate models do?
- **Prediction Markets** Choose a question in a prediction market (eg https://www.predictit.org/). Formulate a mathematical model to answer the question. Enter. Assess in advance how well you think you will do.

Course format:

During the course, you will work on theoretical and practical content. The theoretical aspect will be covered during lectures. The practical content will be covered during guided sections, homework, and projects. The undergraduate class (AM115) will focus on the fundamentals of modeling, with coding in Python notebooks. The graduate class (AM215) will meld this with an introduction to software engineering: the art of systematically evaluating state-of-the-art models requires more sophisticated programming skills.

We will run the undergraduate and graduate classes together, with the goal of making them synergistic with each other. While both classes will focus on the essentials of mathematical modeling, the graduate class will also teach the essentials of software engineering needed to become a better modeler. There will be opportunities for members of the graduate and undergraduate classes to collaborate. Yet, the expectations and requirements of the two classes are different.

For AM115:

The course will consist of two lectures each week. There will also be a weekly section, as explained below.

Section: Computation and computational exercises are at the heart of this class. To cover the practical aspect of the course, we will have a **mandatory** section each week. Guided by a TF, you will solve a Jupyter notebook exercise that illustrates the modeling theme of the week. You will learn how to put into practice the computational tools introduced during lectures.

For AM215:

The course will consist of two lectures each week. Two on mathematical modeling (Tue and Thr) and one on Software Development (Fri). There will also be a section, as explained below.

Section: Sections for AM215 students will be similar to those of 115, yet will go deeper into computational methodologies. Guided by a TF, the students will work on Jupyter Notebook exercises, as well as other tools, such as bash scripts, Git and python scripts. Attendance will be **mandatory**.

Note: Students of AM115 are also welcome to attend the AM215 extra lecture! The software engineering that underlies excellent mathematical modeling is at the heart of many remarkable advances we have seen in recent years.

Typical enrollees:

For AM115:

Advanced undergraduates in quantitative subjects. Prerequisites are:

- The students should be comfortable with linear algebra, calculus, and basic probability theory, at the level of Math 21a/b and Stat110.
- Basic programming knowledge is assumed. A background in Python will be helpful but not necessary.

For AM215:

This is a core class for master students of Data and Computational Science. Apart from the math requisites of AM115, students should be familiar with programming at the level of CS50.

When is the course typically offered?

The hybrid AM115/AM215 class is offered in the Fall. AM115 is also taught in the spring, though without the graduate component.

Assignments and grading:

Sections

Each section will be graded on attendance and completion equally. One point will be assigned on attendance, taken by the TF. And one point will be after grading the lab.

In this course, completion is interpreted as a solution to the problems. It might not be completely correct, as long as the proposed solution's overall idea complies with the course's contents.

Attempts to solve the problem will be given full points. The interpretation of the quality of the solution will be the responsibility of the TFs and, ultimately, the instructors.

Problem Sets

There will be four homework assignments that focus on basic problem-solving skills. Students are expected to work in groups of up to three students.

Assignments for AM115 will consist of Jupyter Notebooks. AM215 will be different and include other deliverables, such as scripts and/or libraries.

Mini-projects

There will be two mini-projects. These projects will be carried out in three-person teams. The Teaching Staff will define the topics around subjects treated during class. Students are allowed to propose their ideas within the scope of the class and subject to approval. The Teaching Staff will provide support when needed.

For AM115

Projects will consist of a â€œscripted partâ€, in which you will explore a topic according to a set of instructions (like a homework). And an unscripted part, where you extend the topic in some direction of interest.

A brief report of **up to** ten pages is expected from each team; often, these might be easier to write as a Jupyter Notebook so you can incorporate modeling elements with your logic. *Contributions from each group member should be clearly noted in the report.*

For AM215

Projects will be unscripted, and the team is expected to plan and implement their solution on a GitHub repository hosted within the AM215 Organization. A brief PDF report of **up to** ten pages is expected from each team, along with the corresponding documentation in the code. All the work must be done within the GitHub repository. *Contributions from each group member should be clearly noted in the report.*
Students working on AM215 projects are expected to work on a library public to the entire class within the AM215 GitHub organization.
Students of AM115 are encouraged to use the AM215 libraries and interact with the development team using GitHub tools. Such interactions include but are not limited to, requesting features, requesting a specific API, or suggesting integrations with other third-party libraries.

Final project

Each group will carry out a final project investigating a new model or carrying out a significant extension of a model discussed in class. Student creativity in evaluating, selecting, and refining the topic is an important part of the project evaluation. During the course, graded project presentations will be conducted. Students are expected to receive feedback and implement the suggested changes.

For AM215

Students are expected to address any problem within the scope of the class. The main deliverable for the final project is a Python library with documentation, tests, examples, and all the software development techniques addressed during lectures.

Grading

Grades will be assigned individually according to the following table:

Item	Percentage
Problem sets	20%
Mini-projects	30% (15% each)
Sections	20%
Final project	30%

Late policy

Homework

Homework submissions are accepted before the assignment deadline. You have three late days at your disposal that can be consumed for late submissions. Two consecutive late days can be used most of the time for homework assignments.
In cases where the submission requires more than one type of submission, such as a Gradescope submission and a commit to a GitHub repository, the latest date will be considered for the late-day calculation.

It is your responsibility to plan your work ahead and submit it on time. If you have consumed all your late days and have another late submission, it is to your benefit to still submit the work. We assume the Harvard Honor Code for all late submissions in case solutions are already posted.

If you have a **verifiable medical condition** or other special circumstances that interfere with your coursework, please inform the instructors **as soon as possible**.

No late submissions are accepted for sections or project submissions.

Grading errors

If you think there is an error in your assignment grading, you can submit a regrade request through the Gradescope platform.
You will have **one week** to submit your regrade request. Exceptional requests outside this period are evaluated on a per-student basis.

Collaboration policy

You are welcome to discuss the course material and homework with others in order to understand it better. Any work submitted as your own without properly citing the original author(s) is considered plagiarism. Failure to follow the academic integrity and dishonesty guidelines outlined in the Harvard Student Handbook will have an adverse effect on your final grade. This includes the removal of copyright notices in code. You may not submit the same or similar work to this course that you have submitted or will submit to another without permission. The teaching staff may use tools to compute correlations between submitted work.

Use of AI Models

Purpose of Policy:

This policy outlines the acceptable use of AI models, including but not limited to ChatGPT, in completing assignments for this course.

Policy Guidelines:

1. **Original Work:** Students are expected to complete assignments using their original thoughts and interpretations. AI models can be used to help understand concepts, generate ideas, or learn about different perspectives, but they should not write or complete assignments for students.

2. **Collaboration with AI:** Students may use AI models for brainstorming or generating preliminary ideas, but the final work submitted must be substantially their own. Students should be able to explain their reasoning, logic, and conclusions without relying on the model's output.

3. **Ethical Considerations:** Students are encouraged to approach the use of AI with ethical considerations in mind, including issues related to privacy, bias, and authenticity.

Consequences for Non-Compliance:

Failure to adhere to this policy may result in academic penalties as outlined in the course's academic integrity policy.

Questions and Clarifications:

If students have questions about the appropriate use of AI models in an assignment, they should consult the course instructor or teaching assistants before proceeding.

Please refer to the University's policy for further information.

Accessibility

If you have a documented disability (physical or cognitive, temporal or permanent) that may impair your ability to complete assignments or otherwise participate in the course and satisfy course criteria, please contact the Disability Access Office (DAO). They will authorize us to help you with corresponding accommodations.

Diversity Statement

All participants in this class are expected to foster empathy and respect towards each other. This includes instructors, TF, or students. The motivation to take this course shall be to experience the joy of learning in an environment that allows for diverse thoughts, perspectives, and experiences and honors your identity, including race, gender, class, sexuality, religion, ability, etc. Any constructive feedback for improving the class environment is welcome, and we encourage you to reach out to the Teaching Staff with any concerns you may have. If you prefer to speak with someone outside of the course, you may find helpful resources at the Office for Equity, Diversity, Inclusion, and Belonging (OEDIB).