EE 194/BIO 196: Modeling, simulating and optimizing biological systems Spring 2018

Class meeting time: TuTh 4:30-5:45
Homework review: there will be an optional session every week in the computer lab that is dedicated to helping with the programming homework assignments. We will pick the date/time during the first class period.
Instructor: Joel Grodstein. Office hours: 1 hour before class or by appointment.
Teaching assistant: Cassandra Donatelli. Office hours Wed 9:15-10:15 in SEC 350, and Tue 10-11am at 200 Boston Ave (with advance notice)

Prerequisites: junior status or above

***Course Overview*:**

The class will cover different ways to model biological systems, and then use computers to simulate or optimize the models. Our examples will mainly be drawn from work in population modeling, bio-electricity and soft robotics being done at Tufts labs. We will implement the models in Python, and thus will spend roughly the first third of the semester learning Python (largely through the above examples).

***Course Objectives*:**

Upon completion of the course, students will be familiar with:

1. The uses of modeling in biology; e.g., modeling to make predictions; to validate existing hypotheses; to come up with new hypotheses; to aid our understanding of natural phenomena.
2. The art of modeling: high level vs. low level, top-down vs. bottom-up, stochastic vs. deterministic, lumped vs. continuous objects, discrete vs. continuous time; the effects of all these on simulating & optimizing.
3. Ways to simulate models
4. Simple optimization techniques: exhaustive enumeration, genetic algorithms, greedy heuristics
5. Enough computer programming (Python) to implement all of the above. Note that this is not a computer-programming course; the level of rigor will be substantially less than (e.g.,) COMP 11 or COMP 15, and more comparable to BIO 40.

Why should you care?

- Biology increasingly seems to be studying areas whose complexity warrants computer modeling.
- Even students who have not historically considered themselves to be adept at quantitative work and at using computers are now seeing both those skills becoming more necessary. We will assume no programming background; the course will be self-contained in that regard.

Rough course sequence:

1. Introduction to modeling (level of detail, top-down/bottom-up, linear/nonlinear, lumped/continuous; stochastic/deterministic; cont/disc time)
2. Python: variables and loops
3. Population modeling
4. Python: matrices, if/then, functions

5. Differential equations: what they mean intuitively (without too much math)
6. Kinetic proofreading
7. Soft robotics
8. Evolutionary algorithms
9. Bacterial chemotaxis
10. Bioelectricity and morphogenesis

The programming and modeling topics above will actually be somewhat intermixed with the other topics, rather than being standalone units.

***Programming Assignments**:*

There will be at roughly seven programming assignments throughout the term. As noted below, the coding for all assignments must be completed individually. However, you can (and should) talk about ideas with other students.

Final project or exam:

The course will have a final project. You may work in teams of 1-3 people, and choose any project that is related to the course materials. The examples that we will go over during the semester are only a small exposure to the topics of population modeling, bioelectricity and soft robotics; it is very easy to go into any of them more deeply. If you are interested in one of these fields, this is an excellent opportunity. Here are a few suggestions:

- *Kinetic proofreading*: first, learn more about the up-to-date molecular biology of translation, using the references given in class. Then, using the same chemical-reaction framework you used on HW #4, build the reactions for it and simulate it.
- *Soft robotics*: there are numerous possibilities.
    - A pure Python project: improve our animation software so that it draws the Manduca body segments moving vertically when they bend.
    - Learn other optimization algorithms and see how they compare with genetic algorithms. E.g., simulated annealing or lookahead-greedy algorithms.
    - Try to improve our fitness function enough that a natural, realistic Manduca crawl indeed becomes the fastest behavior. You could start by taking the physical constants in the model from the Manduca literature (which we can provide to you), or by adding penalties to the fitness function for irresponsible behaviors.
- *Chemotaxis*: Learn about the chemical reactions that E.coli actually use in chemotaxis, and implement them in our chemical-reaction framework.
- *Bioelectricity*: improve upon the planaria model we used in class. For example:
    - We never coded the formation of a two-headed planaria. You should be able to do this (though I've never tried) fairly easily.
    - Our homework used flux-vs.-voltage functions. Change this to actually instantiate ion channels. I'll code the ion channels for you, but you have to figure out how many ion channels to instantiate in each cell, which kind of ion channels to use, and what their switching functions are. You may want to use a genetic algorithm or other optimization algorithm to help you. Note that this is very similar to an active research project currently underway at Tufts.

Resources:

- *An Introduction to Systems Biology: Design Principles of Biological Circuits*, Uri Alon. An excellent introduction to the field of systems biology, with a bit of synthetic biology and morphogenesis thrown in. Particularly, Chapter 2 is a nice introduction to modeling a cell; Chapter 7 covers chemotaxis, Chapter 8 covers morphogenesis and Chapter 9 covers kinetic proofreading. The book is on reserve at Tisch.
- *Plant and Animal Populations, Methods in Demography* by Thomas Ebert. An excellent introduction to population biology. It is on reserve at Tisch.
- There are numerous books on Python programming. Two that are available free online are
  - http://greenteapress.com/thinkpython2/thinkpython2.pdf
  - https://automatetheboringstuff.com
- There are numerous websites with Python practice problems. A few are:
  - https://www.w3resource.com/python-exercises
  - http://www.practicepython.org
  - http://codingbat.com/python
  - http://usingpython.com/python-programming-challenges

Grade Formula

As with the course material, the number of assignments (as well as the existence of a final project or final exam) will be adapted to what best works for our mix of students. The grading formula will adapt with it. One possible mix might be:

- Programming assignments – 75%
- A few quizzes – 15%
- Class participation – 10%

Late Assignments:

Late assignments will be penalized by 10% per day. Any extensions due to extenuating circumstances (illness or family emergencies) must be arranged ahead of time with the instructor before the original due date.

Collaboration policy

Learning is a creative process. Individuals must understand problems and discover paths to their solutions. During this time, discussions with friends and colleagues are encouraged—you will do much better in the course, and at Tufts, if you find people with whom you regularly discuss problems. But those discussions should take place in English, not in code. If you start communicating in code, you're breaking the rules. When you reach the coding stage, therefore, group discussions are no longer appropriate. Each program, unless explicitly assigned as a pair problem, must be entirely your own work. Do not, under any circumstances, permit any other student to see any part of your program, and do not permit yourself to see any part of another student's program. In particular, you may not test or debug another student's code, nor may you have another student test or debug your code. (If you can't get code to work, consult a teaching assistant or the instructor.) Using another's code in any form or writing code for use by another violates the University's academic regulations. Do not, under any circumstances, post a public question to Piazza that contains any part of your code. Private questions directed to the instructors are OK. Suspected violations will be reported to the University's Judicial Officer for investigation and adjudication. Be careful! As described in the handbook on academic integrity, the penalties for violation can be severe. A single bad decision made in a moment of weakness could lead to a permanent blot on your academic record. The same standards apply to all homework assignments; work you submit under your name must be entirely your own work. Always acknowledge those with whom you discuss problems! Suspected violations will be reported to the University's Judicial Officer for investigation and adjudication. Again, be careful

Additional resources

Tufts University values the diversity of our students, staff, and faculty, and recognizes the important contribution each student makes to our unique community. Tufts is committed to providing equal access and support to all qualified students through the provision of reasonable accommodations so that each student may fully participate in the Tufts experience. If you have a disability that requires reasonable accommodations, please contact the Student Accessibility Services office at Accessibility@tufts.edu or 617-627-4539 to make an appointment with an SAS representative to determine appropriate accommodations. Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision. Tufts and the teaching staff strive to create a learning environment that is welcoming to students of all backgrounds. If you feel unwelcome for any reason, please let us know so we can work to make things better. You can let us know by talking to anyone on the teaching staff. If you feel uncomfortable talking to members of the teaching staff, consider reaching out to your academic advisor, the department chair, or your dean.