Title

Biophysics software for interdisciplinary education and research

Journal

American Journal of Physics, 82(5)

ISSN

0002-9505

Author

Deutsch, JM

Publication Date

2014-05-01

DOI

10.1119/1.4869198

Peer reviewed

Biophysics software for interdisciplinary education and research

J. M. Deutsch*

Department of Physics, University of California, Santa Cruz CA 95064

Abstract

Biophysics encompasses many disciplines, and so transcends the knowledge and skills of the individual student; its instruction therefore provides formidable challenges. This paper describes educational materials that were developed by the author and have been used successfully in an interdisciplinary course on biophysics, taken by undergraduates from a variety of disciplines. Projects were devised on topics that ranged from x-ray diffraction to the Hodgkin-Huxley equations. They are team-based and strongly encourage collaboration. Extensive use is made of software written in Python/SciPy, which was modified to explore a large range of phenomena. This modified software can also be used in lectures, in the teaching of more traditional biophysics courses, and in research.

I. INTRODUCTION

Many of the most active and productive areas in science are highly interdisciplinary in nature. Research in such areas is carried out by teams, with each member being expert in a specific field. Such teams have discovered new phenomena and created new technologies that would have been impossible within the confines of a single discipline. An example is the development of massively parallel gene sequencing,[1] which involves expertise in molecular biology, biochemistry, soft condensed matter physics, optics, and computer engineering. Collaboration within such teams needs to be highly coordinated in order to produce successful outcomes.

Educational methods have had difficulty keeping up with this shift of research towards interdisciplinary collaboration. In the vast majority of courses now being taught, learning takes place within a single discipline, and students are not trained to deal with the challenges involved in collaboration across diverse fields. To address this issue, the author has developed a course that teaches students from different disciplines how to collaborate in a way that mirrors much of the research that is currently being performed in biophysics. The course has now been taught three times, and an evaluation of its efficacy, including student assessments, will be addressed in a future publication. This paper focuses on the materials and software tools that were developed for the course, providing information that should be useful to other educators interested in reproducing various of its aspects. In addition, the course involves a large number of useful simulations that can be used separately in both instruction and research; these simulations are available as an online supplement to this article.[2]

The course spans ten weeks, with approximately four hours of lectures and typically three hours of discussion sections per week. There is an additional week at the end of the quarter during which students give presentations. The number of students in the course has ranged from 12 to 21. Those who enroll come from a wide range of disciplines, and are mostly upper-division students. The bulk of their work involves completing several projects per week; these projects involve collaboration within interdisciplinary teams of students. Each team is usually composed of three students, with the members of the team chosen to provide as much interdisciplinary breadth as possible. A typical team might consist, for example, of a physics major, a biochemistry major, and biomolecular engineering major. Typically, each student is responsible for work in his or her area of expertise, such as computer coding,

mathematical and physical analysis, biology, and biochemistry. The projects are designed to enable the students to exchange knowledge and approaches to scientific problems, and so to promote interdisciplinary interaction.

The projects use software written in Python/SciPy, an environment that has become increasingly popular in recent years.[3,4] SciPy is a set of libraries and data structures in the Python programming language that are designed to facilitate scientific computation. The software developed for this course serves multiple functions in teaching students how to carry out different aspects of research, and to collaborate effectively. This software is often used to perform simulations that act as proxies for experiments. In addition, SciPy code is employed in data analysis, specifically for the reading, fitting, and plotting of data. Most projects involve either programming or some code modification. Visual Python[5] is also used in this course. It displays three dimensional graphics, such as of macromolecules, with a minimum of code required to do the visualization.

The use of this software allows students to gain substantial theoretical and intuitive understanding of biophysical processes through active learning. The course has been highly successful, and has had a major impact on many of the students who have taken it. It has often influenced their career choices, for example, towards biophysics, or software development, and it has obtained consistently high student evaluations.

II. USES OF MATERIAL

The main purpose of this paper is to introduce instructors to the biophysics material that was developed for the course by the author. The material develops a range of skills: the understanding of biological processes, physical laws, mathematical analysis, and programming, and it also promotes interdisciplinary collaboration. It can be used in a variety of educational settings, as discussed below.

The course projects are typically associated with one or more computer programs written in SciPy. The software is sometimes presented to students in incomplete form, allowing them to fill in important steps that are necessary to complete the code. At other times, the software is supplied to students in a completely working state. However, completed versions of all the programs are available to instructors upon request. The programs can potentially be used in the following ways:

1. **Lecture demonstrations.** The most straightforward use of this material is in lecture demonstrations that use the software. A variety of physical and biophysical processes can be simulated, and the relatively simple Python source code is easily modified, even during lectures. For example, in a lecture, it is illuminating to do an analytic derivation and then to test the derived formulae numerically, by running the appropriate code. The author has found that the use of this software has made theoretical discussion much more tangible and has led to many interactions with students during lectures. SciPy code often appears similar to pseudocode, so that the main gist of the underlying algorithm is understandable, even to students that do not know how to program.

2. **Lower division courses.** With the use of completely working code, lower division students can benefit from many of the simulations presented here. They can be used outside of lectures by students who have installed SciPy and Visual Python on their computers. The ability to visualize a problem and to use this software interactively enables the student to develop an intuitive understanding of concepts (such as protein folding) that is far higher than can be obtained from abstract descriptions alone.

3. **Existing biophysics courses.** Existing biophysics courses are taught predominantly to students who are expected to have a common core of scientific knowledge. For example, a course in biophysics might cater to biophysics graduate students, or it might be intended for physics seniors. The materials developed here can be adapted to fit into the requirements of such courses. For physics or engineering students, many questions in projects that require considerable biological knowledge should probably be omitted. Conversely, a more biologically oriented audience would be unable to answer questions that require too detailed a knowledge of physics, and correspondingly, such questions should be left out.

4. **Research.** Many of the programs that have been developed are also useful for researchers, because they provide simple and easily modifiable software for use in addressing a large number of problems. The author has used several of these programs in his own research, and has also corresponded with other researchers who have used some of this software, for example, code that simulates the Langevin dynamics.

III. TYPES OF PROJECTS

The projects that have been used in the course span a wide range of topics in biophysics, involving the physical bases of biophysics techniques, such as x-ray crystallography, and theoretical concepts such as Brownian motion in a potential.

Students are typically required to submit several projects in each week. Most projects require completion within a week, though several require two weeks to complete. Some of the projects are tailored to the research interests of the individual students. For example, the "Distribution of ATP and mitochondria in axons" is a research topic of current interest to William Saxton, on the biology faculty at UCSC, and a team of students taking the course has had meetings with him to determine how to model this process most usefully.

Also, most projects have one or more SciPy programs associated with them. However, some are purely analytical, and one involves additional software (Foldit). Aside from the software component, the course requires that the students address physical and biological aspects of various problems.

In order to be teachable within a space of ten weeks, the course is built up from three fundamental mathematical and physical ideas: Fourier transforms, diffusion, and molecular simulations. Once these are mastered, they are used in a variety of contexts to leverage the understanding of a wide range of different problems.

The projects are described below in nearly chronological order, so as to explain in more detail how a fairly substantial territory is covered in a way that is accessible to teams of students with very diverse backgrounds. A full list of the projects is furnished in Appendix A. The plots and simulations shown here are only a small subset of those available to the students, and are displayed as they would appear to them.

The course consists primarily of nine modules, each lasting one week. During the last two weeks, the students prepare for their final projects.

Module 1: Introduction to SciPy

The course begins with students setting up SciPy on their individual computers and learning the basics of reading input and writing output in SciPy. The specific focus of this module is on image processing. The images that are used were encoded similarly to those

in an x-ray diffraction experiment. The data can be decoded using several operations. One of these operations is the two-dimensional inverse Fourier transform. Figure 1 shows an example of an initial image that needs to be decoded, with the decoding process involving this transform. The real part of the Fourier transform has been processed in such a way as to allow for reasonable quality when inverted, despite the small amount of information used to encode each point. The students study the SciPy code that was used to encode the image, and are given a software file containing skeleton code and hints. They need to complete this code in order to decode the image and display it. The decoded image should be recognizable to a life science student; for example, it might be the image of a neuron.

Module 2: Techniques for structure determination

The next set of projects explores the mathematics and physics of x-ray diffraction and tomography, both of which make heavy use of Fourier transforms. Students are furnished with computer-generated files that contain data similar to that obtained from tomography or diffraction experiments, and they are required to decode this data so as to determine the underlying structure using a process similar to that used in real research situations. This is distinct from typical homework assignments that concentrate more on the internal details of a method, and do not closely mimic actual research procedures.

Module 3: Diffusion

The concept of diffusion is introduced. Physics majors are generally quite familiar with the differential equations that describe this process; however the underlying connection with random motion is not usually explained to them. In general, two approaches have been used in the analysis of diffusion. One comes from the perspective of differential equations, and the other uses a stochastic description to introduce the idea of a random walk. Both of these approaches are visualized using software. By these means, the connection between these two approaches becomes far clearer, and in addition the students are introduced to the concept of random molecular motion. Since there are a large number of biological applications that involve diffusive processes, this also helps to maintain the interest of students in the biological disciplines.

For example, one important technique, "fluorescence recovery after photobleaching"[6] (FRAP), is nicely illustrated by computer simulation (Fig. 2). A region is photobleached, and fluorescent molecules diffuse back into this region so that a changing fluorescence intensity results. The project involving FRAP induces students to consider how a combination of scaling and simulation can provide quantitative predictions for experiments. This kind of analysis, involving the combination of scaling with simulation, is rarely taught, but can be appreciated here without the need for sophisticated mathematics. It provides students with a much better intuitive understanding of diffusion than is provided in the usual approach of solving the corresponding partial differential equations using analytical means.

Module 4: Morphogenesis

Most subsequent problems in the course are leveraged from knowledge of Fourier transforms, diffusion, and simulations. In this module, various aspects of morphogenesis are discussed, and patterns such as those produced by reaction-diffusion equations are now made understandable by modifying the diffusion code developed in Module 3 to include more than one kind of diffuser, as well as nonlinear interactions between different chemical species. For example, the model introduced by Turing,[7] and further developed by Murray,[8] that gives rise to patterns on the skin of animals can be modeled numerically (Fig. 3), and this produces a variety of different morphologies, such as stripes and spots. The generation of these patterns can be understood by adjusting the parameters and boundary conditions of the model.

The most important lesson to be learned from this approach is how complexity emerges from simple physical processes. Biology students are able to investigate to what extent such principles are applicable to specific problems. Other problems with biological relevance, such as dendritic growth and diffusion-limited aggregation[9] (DLA), are also taught, and simulations of these can be modified by students so as to explore their behavior and their relation to biology. For example, there has been much experimental work relating DLA (Fig. 4) to the morphology of bacterial colonies.[10,11] Students not only learn how such simulations work, but through the process of collaboration they can critically assess evidence supporting this mechanism in biological systems.

Module 5: Optical traps, noise, and correlations

Instead of using differential equations to understand diffusion, a more basic approach can be taken that exploits the inherently random nature of the underlying motion; that is, the Langevin equation.[12] This approach initiates the student to the study of the dynamics of particles in liquids, taking their thermal motion into account. To understand such problems, correlations and power spectra are crucial, and these concepts also relate back to Fourier transforms. Particles in optical traps[13] provide an excellent illustration of these principles, as well as being used in active areas of biophysical research. A simple example that can be easily simulated is a particle tethered by a spring in a viscous liquid acted on by random thermal noise (Fig. 5). This is perhaps the most difficult part of the course to understand from a mathematical perspective, and the projects employ simulations of these systems, so as to obtain data as if they were derived from actual experiments. By analyzing these data by means of Fourier transforms and correlation functions[12] students can, for the most part, grasp the underlying physics quite well. This problem of random motion in a potential, is also related to the more difficult problem of bacterial chemotaxis; for example, how bacteria move to regions of higher glucose concentration despite being too small to detect chemical gradients effectively.

Module 6: Macromolecular dynamics

The problems considered so far have involved only a few degrees of freedom, so the next logical step is to consider systems with many degrees of freedom. There are a large number of biophysical systems of this kind, and these relate to the dynamics of macromolecules. An excellent way to understand complex biomolecules is through simulation, and several examples of systems involving biomolecules are presented in this module using the same underlying code base.

For example, the electrophoresis of DNA is modeled as a semi-flexible chain traversing a network of rigid obstacles. The dynamics of this system show semi-periodic behavior[14] that can easily be seen in simulations visualized in three dimensions—for example using Visual Python,[5] as shown in Fig. 6. This simulation leads to an understanding of pulsed field electrophoresis, which has been instrumental in the separation of very long DNA chains.

Further, using this code base, it is possible to understand a different problem: force versus displacement in a stretched DNA chain. This relationship has been measured experimentally using optical traps. More generally, recent experiments involving the stretching of macromolecules that have complex internal states, such as RNA, have employed simulations that have considerably increased the understanding of their structure. This technique, "single molecule force spectroscopy,"[15] can be understood by simulations, as illustrated in Fig. 7.

In this module, a project is introduced in which a molecule has two groups with a mutual attraction, so that when the molecule is pulled by its ends, it has metastable behavior that gives rise to hysteresis loops. In another project, the students are given experimental data on the stochastic transitions between multiple internal states of a biomolecule, and analyzing the details of these transitions provides useful information about molecular conformations. Other projects have also been developed, including the translocation of linear macromolecules through a nanopore (see Fig. 8), which is another active area of research.[16]

Module 7: Proteins

Proteins constitute a widely studied class of macromolecules, so a number of projects involving proteins have been developed for this course. These involve a variety of important physical phenomena, including the coil-globule transition, the helix-coil transition, and the determination of the folded state of a protein. This problem of "protein folding" is studied for simplified models such as the so-called "HP" model of Dill and Chan[17] (see Fig. 9). This is a good model to simulate using Monte Carlo methods, leading to an understanding of the energy landscape of this system. The landscape is highly complex even when studied in two dimensions, so one project in this module is devoted to this case.

A powerful variation of Monte Carlo uses parallel tempering, which forms the basis of another project in this module. To speed up these simulations, C code is embedded in Python code using Weave,[18] which introduces some of the more computer-oriented students to C programming. Another project in this module involves learning to use a piece of software that folds proteins called Foldit; this is freely available[19] and is useful for elucidating many of the subtle interactions seen in proteins. The software is in the form of a computer game that has become very popular, and some of its players have produced excellent protein structures.[20]

Module 8: Molecular machines and effects of solvent

Some of the more subtle features of macromolecular interaction are electrostatic, and these can be described approximately in some cases by the nonlinear Poisson-Boltzmann equation.[21] In this module students learn how to solve this equation, and the algorithm that is employed closely resembles that used in earlier projects in Module 4 (see Fig. 10). There are also three projects involving biological motors. The first two are minimal models of myosin proteins;[22] these can be understood by simulations incorporating chain stiffness, plus binding and unbinding from a linear array of sites. Students change model parameters such as binding angles and chain stiffness, to determine how these changes affect the characteristics of the motor. Two snapshots from such a simulation are shown in Fig. 11. In the course, different teams design motors with different parameters, and the teams run a race to determine which motor works best. Another problem that is currently being studied is that of cytoplasmic streaming in drosophila oocytes.[23,24] This is modeled by a variation of the DNA code base that was introduced in Module 6. It leads to interesting behavior, such as a breaking of chiral symmetry, and collective organization of microtubules to advect fluid chaotically through the cell. Figure 12 shows a snapshot of a microtubule moving as a rotating helix, due to the action of kinesin motors that walk upwards along its surface.

Module 9: Introduction to neurons

The final module consists of an introduction to the biophysics of the neuron. The Hodgkin-Huxley equations[25] are implemented in code, and this greatly elucidates how action potentials are generated and propagate. For example, the propagation of a spike is shown in Fig. 13. Complementary to this is the high-level organization of neural networks. Using code similar in spirit to that used earlier in the course to study the helix coil transition, the mechanism for associative memory proposed by Hopfield is implemented.

IV. REQUIRED STUDENT LEVEL OF KNOWLEDGE

As described in the Introduction, teams consisting of three students are formed at the beginning of the course, with the goal of providing sufficient diversity of knowledge within each team to be able to tackle the assigned projects effectively.

A. Software

For each team, there is normally one student who has had a certain amount of exposure to programming. The physics majors at UCSC are required to take introductory programming, and often students from other majors, particularly in the engineering disciplines, are quite knowledgeable programmers. The software is provided to the students in a variety of states; sometimes as fully functioning code, at other times with a few missing lines that need to be filled in, and rarely where more extensive modifications are needed.

The software platform was chosen to follow a middle ground between two very different pedagogical uses of computers in education. The first scenario is exemplified by numerical methods courses. In these, simulation code is generally developed in a highly efficient language, such as C, C++, Fortran, or Java, so that students learn how to carry out state-of-the-art simulations using a variety of numerical techniques, such as Monte Carlo or molecular dynamics simulations. Such courses require that students have a reasonably advanced knowledge of applied mathematics, and the ability to code at an intermediate level in one of the above languages. The other scenario is to teach a course using a platform that already has a graphical user interface (GUI) programmed in, so that students navigate the interface to make changes in parameters. This latter scenario limits the flexibility in what can be simulated, though it is accessible to a much broader audience. This approach was pioneered more than two decades ago,[26] and excellent biophysics software of this type has recently been developed.[27]

The software in the present course stands at neither of these two extremes, recognizing that the majority of physics and engineering majors have not taken a numerical methods course, but nevertheless have a fair degree of programming proficiency. At the same time, code even with the most flexible GUI interface cannot come close to the flexibility attainable using source code, such as is employed in this course. This is why SciPy, which is used in the course, is so popular within the research community, and is also being developed extensively for instructional use by others.[28] It allows fairly complex problems to be addressed with a minimum of programming overhead, and also has advanced graphics capabilities so as to allow data to be displayed in a variety of ways. However, a fair amount of learning is still necessary to become proficient using these programming tools, and the course was designed to enable the students to achieve this proficiency. Indeed, students have often listed the

acquisition of these programming skills as a major benefit of taking the course. At the beginning of the course, the code is at a fairly basic level, and eschews the use of high-level constructs that might be unfamiliar to many of the students. As the course progresses, the programs become more advanced, and some C code is integrated into the Python source code through Weave[18] so as to increase its efficiency.

In general, the student teams handle the software component of the projects very well. In most cases students need only to change parameters in the code, and they are able to follow the structure of the code sufficiently to be able to make these changes. In fact, many biology and biochemistry students become quite proficient at coding through taking the course. In addition, the students are given a considerable amount of choice as to which projects they work on. Therefore, teams that are less interested in software development generally choose projects that do not require much programming. On the other hand, some students are very interested in the programming aspect, and occasionally produce impressive modifications of the code with which they are furnished, or write their own code. This flexibility in coding emphasis fosters enthusiasm among the students, who often end up carrying out work at an unexpectedly advanced level.

The most difficult part of this course lies in the initial installation of SciPy and Visual Python on the students' own computers. These are both complex pieces of software that undergo constant development. The operating systems that work best for this purpose are Linux and Windows, while Macintosh computers are the most problematic. Some students are unable or unwilling to upgrade their Mac operating systems, so that it is more difficult for them to obtain a working version of SciPy. However, the latest releases of SciPy from Enthought[29] have been found to work well, and are available for both the Mac and Windows operating systems.

There are still some issues with Visual Python. These include problems with graphics cards, incompatibilities between the libraries provided by Visual Python and those that come with Linux, and conflicts between Enthought and the Mac version of Visual Python. Nevertheless, the vast majority of students taking the course have been able to obtain a working system that enabled them to modify and run the necessary code. In addition, the author has been able to get the software to run on versions of all three operating systems; however, given the nature of software, installation is a constantly moving target.

B. Knowledge of biology and biochemistry

Because the course is focused on problems stemming from biology, each team needs to include at least one member with a fair amount of biological sophistication. Yet many of the steps that are involved in completing projects for the course also require computer coding and a knowledge of physics. To balance the efforts equitably among the team members, projects were created so that biological contributions were necessary for their successful completion. For example, Module 1 requires a small amount of code modification in order to decode the Fourier transform of an image. The object is to determine what several encoded images (such as shown in Fig. 1) actually represent. The images are taken from biology or biochemistry, and physics majors may find it difficult to identify them correctly, so the completion of the module requires input from a biology student. This kind of interaction between the different disciplines has been found to improve the cohesion of the teams.

The biologically oriented questions are of a somewhat different character than is typically found in biology courses. They are often more open-ended, and allow students from different backgrounds to interchange their perspectives on these questions. Further, certain biophysics articles often need to be understood before undertaking quantitative analysis, and these articles are much more comprehensible to students in biology and biochemistry than to those in physics. For example, the project involving fluorescence recovery after photobleaching, which is a common biophysical technique, is not normally taught to physics majors. The biology students can then translate papers on this method into wording that can be understood by physics students. Further, problems often require students to use their knowledge of biology to come up with situations where a physical process, such as absorption of diffusers, is relevant, and to determine to what extent such physical considerations are applicable. These aspects of the course foster the interaction between the different disciplines.

Because there is a fair amount of choice of projects in the course, teams can choose problems that are in line with their individual interests. For example, there are problems that are popular with neuroscience students, such as magnetic resonance imaging (MRI). Further, there are specific problems, such as involve the translocation of a chain through a nanopore, or the Poisson-Boltzmann equation, that are of interest to students who have carried out laboratory research in these areas. Students in these disciplines have been able

to use their expertise to collaborate more effectively with other team members who are more software or mathematically oriented, to the mutual benefit of the team as a whole.

V. COURSE MATERIALS

The website for the course is http://physweb.ucsc.edu/drupal/courses/physics-180-spring-2013. and the contents of the projects are also available as supplementary information on EPAPS. The code provided to the students is, in some cases, deliberately incomplete. Instructors can email the author (josh@ucsc.edu) to obtain a full set of Python source code and data files. Appendix A provides a full list of the projects used in the course.

Appendix: List of projects

1. Introduction
    (a) Using SciPy and inverse Fourier-transforming images
2. Imaging techniques
    (a) Reconstructing a 2D structure from x-ray data
    (b) Introduction to tomography
    (c) Analytic methods in diffraction
    (d) Introduction to NMR (two-week project)
    (e) Image processing
3. Diffusion
    (a) One dimension
        i. Binomial distribution, diffusion, central limit theorem
        ii. Mean time to capture by diffusion in one dimension
        iii. Steady-state solution of the diffusion equation in one dimension
        iv. Comparing numerical and analytic solutions to the diffusion equation
    (b) Three dimensions

i. Probability of capture by a spherical absorber by diffusion

ii. Diffusion to a disk-like absorber

iii. Diffusion through many circular apertures in a planar barrier

(c) Using FRAP to determine a diffusion coefficient

(d) Kirby Bauer antibiotic testing

(e) Distribution of ATP and mitochondria in axons

4. Morphogenesis

(a) Reaction diffusion and biological patterns

(b) Flow lines in Murray's model of pattern formation

(c) Morphogenesis by the mechanism of diffusion limited aggregation

(d) Origins of dendritic growth

5. Dynamics with thermal motion

(a) Correlation functions and power spectra

(b) Brownian motion in an optical trap

(c) Brownian motion of a free particle

(d) Modeling bacterial chemotaxis

6. Dynamics of macromolecules

(a) Motion of DNA during gel electrophoresis

(b) Force on ends of DNA chain

(c) Translocation of a linear macromolecule through a nanopore

(d) Determining molecular states from force data

(e) Hysteresis in single molecule force spectroscopy

(f) Force on a freely jointed chain

7. Understanding protein interactions and folding

(a) Foldit

(b) Coil globule transition

(c) Helix coil transition

(d) 2D HP model

(e) Folding time for the HP model

(f) Using parallel tempering to fold a protein

(g) Effect of charges in solution

8. Motor proteins

(a) Myosin V simulation race

(b) Myosin 2 Olympics

(c) Cytoplasmic streaming in drosophila oocytes

9. Membrane potentials and the Hodgkin-Huxley equations

10. The Hopfield model

ACKNOWLEDGMENTS

The author thanks Dr. Barbara Goza and Hee-Sun Lee for useful discussions. He thanks Onuttom Narayan for useful suggestions. He especially would like to thank Michelle V. Mai for her critical insights in making the material more relevant to life-science majors. He thanks Diana Deutsch for a very thorough and critical reading of the manuscript. This material is based upon work supported by the National Science Foundation under Grant CCLI DUE-0942207.

---

* josh@ucsc.edu

[1] D. A. Wheeler, *et al.*, "The complete genome of an individual by massively parallel DNA sequencing," Nature **452**, 872–877 (2008).

[2] Online supplement to this article, [URL TO BE INSERTED BY AIP].

[3] SciPy project website, `<http://scipy.org>` .

[4] T. E. Oliphant, "Python for Scientific Computing," Comp. Sci. Eng. **9**, 10–20 (2007).

[5] Visual Python project website, <http://vpython.org>.

[6] D. Axelrod, D. Koppel, J. Schlessinger, E. Elso and W. Webb, "Mobility measurement by analysis of fluorescence photobleaching recovery kinetics," Biophys. J. **16**, 1055–1069 (1976).

[7] M. A. Turing, "The Chemical Basis of Morphogenesis," Phil. Trans. R. Soc. London B **237**, 37–72 (1952).

[8] J. D. Murray, *Mathematical Biology: I. An Introduction* (Springer, New York, 1993).

[9] T. A. Witten and L. M. Sander, "Diffusion limited aggregration, a kinetic critical phenomenon," Phys. Rev. Lett. **47**, 1400–1403 (1981).

[10] T. Matsuyama and M. Matsushita, "Fractal Morphogenesis by a Bacterial Cell Population," Crit. Rev. Microbiology **19**, 117–135 (1993).

[11] Y. Kozlovsky, I. Cohen, I. Golding, and E. Ben-Jacob, "Lubricating bacteria model for branching growth of bacterial colonies," Phys. Rev. E **59**, 7025–7035 (1999).

[12] F. Reif, *Fundamentals of Statistical and Thermal Physics* (McGraw-Hill, New York, 1965), Section 15.8.

[13] D. G. Grier, "A revolution in optical manipulation," Nature **424**, 810–816 (2003).

[14] J. M. Deutsch, "Theoretical studies of DNA during gel electrophoresis," Science **240**, 922–924 (1988).

[15] M. T. Woodside, C. Garciá-Garciá, and S. M. Block, "Folding and unfolding single RNA molecules under tension," Current Opinion in Chemical Biology **12**, 640–646 (2008).

[16] S. Winters-Hilt, W. Vercoutere, V. S. DeGuzman, D. Deamer, M. Akeson, and D. Haussler, "Highly accurate classification of Watson-Crick basepairs on termini of single DNA molecules," Biophys J. **84**, 967–976 (2003).

[17] K. F. Lau and K. A. Dill, "Theory for protein mutability and biogenesis," Proc. Natl. Acad. Sci. **87**, 638–642 (1990).

[18] Weave reference guide, <http://docs.scipy.org/doc/scipy/reference/tutorial/weave.html>.

[19] Foldit project website, <http://fold.it/portal/>.

[20] F. Khatib *et al.*, "Crystal structure of a monomeric retroviral protease solved by protein folding game players," Nature Struct. Mol. Biol. **18**, 1175–1177 (2011).

[21] C. Holm "Polyelectrolytes-Theory and Simulation," in *Soft-Matter Characterization*, Vol. 2, edited by R. Borsali and R. Pecora (Springer, New York, 2008), p. 295.

[22] A. R. Dunn and J. A. Spudich, "Dynamics of the unbound head during myosin V processive translocation," Nature Struct. Mol. Biol. **14**, 246–248 (2007).

[23] L. R. Serbus, B. J. Cha, W. E. Theurkauf, and W. M. Saxton, "Dynein and the actin cytoskeleton control kinesin-driven cytoplasmic streaming in Drosophila oocytes," Development **132**, 3743–3752 (2005).

[24] J. M. Deutsch, M. E. Brunner, and W. M. Saxton, "The mechanics of a microscopic mixer: microtubules and cytoplasmic streaming in Drosophila oocytes," <http://arxiv.org/abs/1101.2225> (2011).

[25] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," J. Physiol. **117**, 500–544 (1952).

[26] T. F. Weiss, G. Trevisan, E. B. Doering, D. M. Shah, D. Huang, and S. I. Berkenblit, "Software for Teaching Physiology and Biophysics," J. Sci. Ed. Tech. **1**, 259–274 (1992).

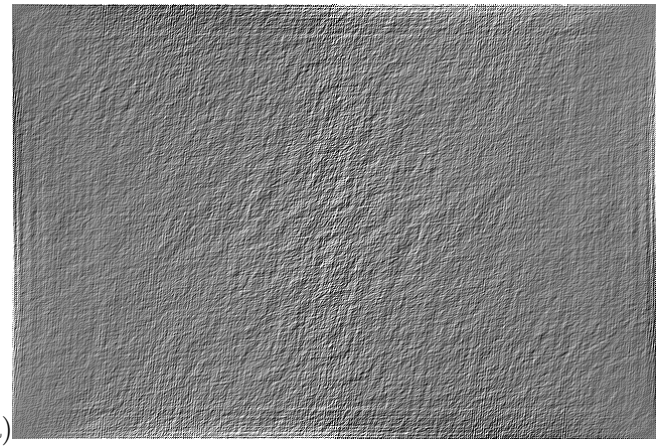
[27] R. Tinker and Q. Xie, "Applying Computational Science to Education: The Molecular Workbench Paradigm," Comp. Sci. Eng. **10**, 24–27 (2008).

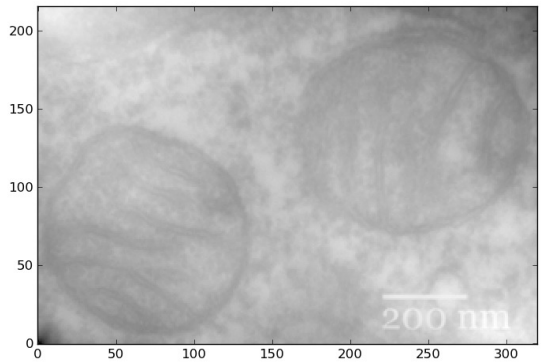
[28] C. R. Myers and J. P. Sethna, "Python for Education: Computational Methods for Nonlinear Systems," Comp. Sci. Eng. **9**, 75–79 (2007).

[29] Enthought home page, <http://enthought.com>.

FIGURE CAPTIONS

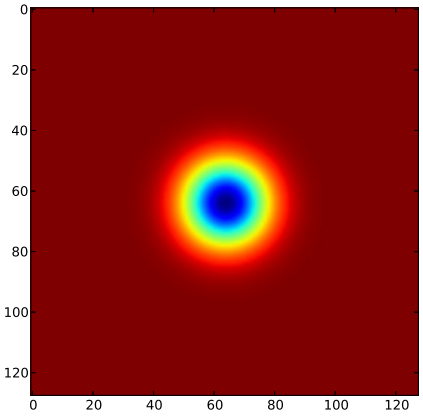
(a)



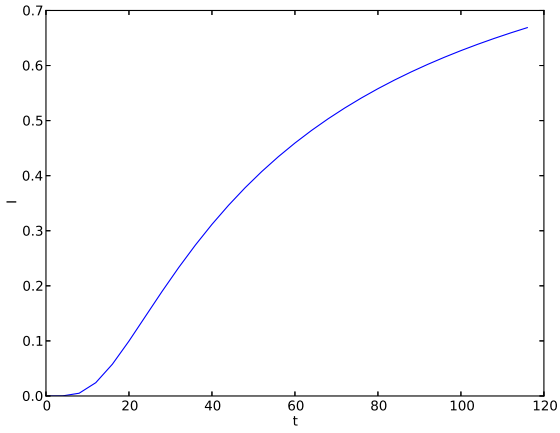


(b)

FIG. 1. (a) An image that requires decoding. The pixel intensities are related to the Fourier transform of the original image. The decoding process introduces the student to an important idea in x-ray diffraction: inverting Fourier transform data can provide much structural information. (b) The image after the decoding process that shows mitochondria in the mammalian lung.



(a)



(b)

FIG. 2. Enhanced online [URL TO BE INSERTED BY AIP]. A simulation of fluorescence recovery after photobleaching (FRAP). A circular area is photobleached and a simulation of the evolution of density is displayed. (a) A snapshot of fluorescence intensity during the simulation. (b) A plot of intensity as a function of time at the center of the circle.

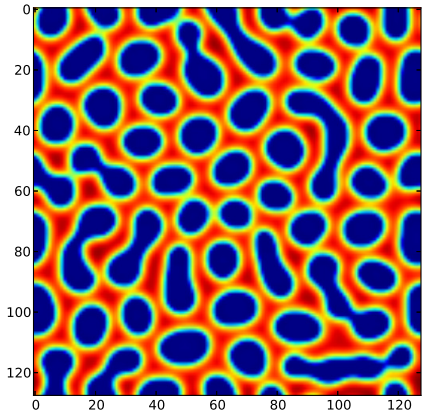


FIG. 3. Enhanced online [URL TO BE INSERTED BY AIP]. A simulation of Turing's model,[7,8] which describes the evolution of spots on the skin of animals.

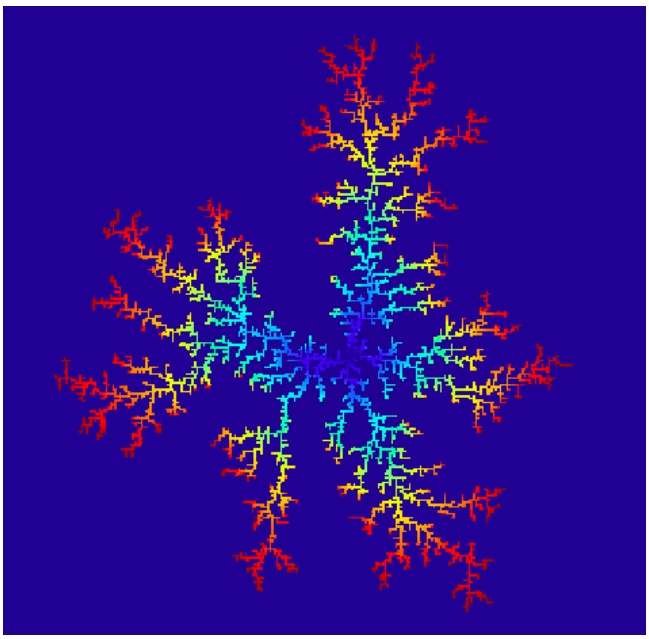


FIG. 4. Enhanced online [URL TO BE INSERTED BY AIP]. A simulation of diffusion limited aggregation. This process is important in understanding many physical phenomena, including the morphology of bacterial colonies.

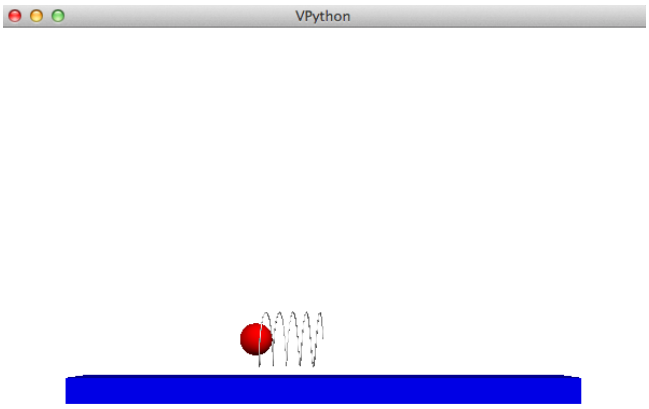


FIG. 5. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a simulation of the Langevin equation, which models a particle in a highly viscous fluid coupled to a spring. This model is the same mathematically as that used to understand noise in optical traps. The graphics for this simulation were easily implemented using Visual Python.[5]

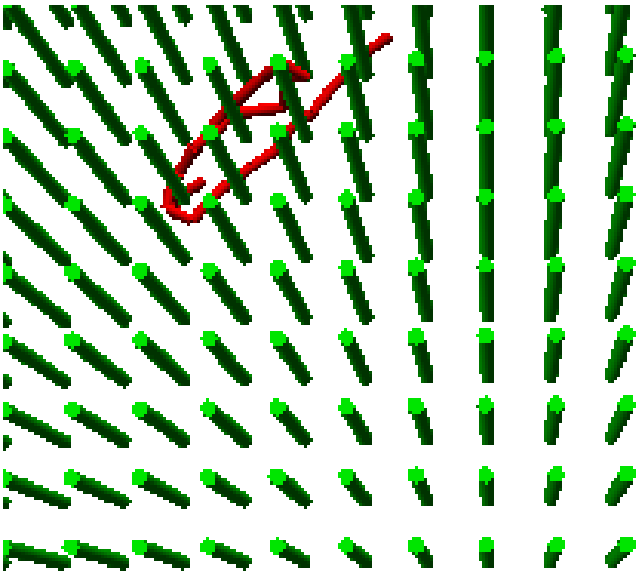


FIG. 6. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a simulation of electrophoresis of DNA traversing an array of nanopillars, shown using Visual Python.[5]

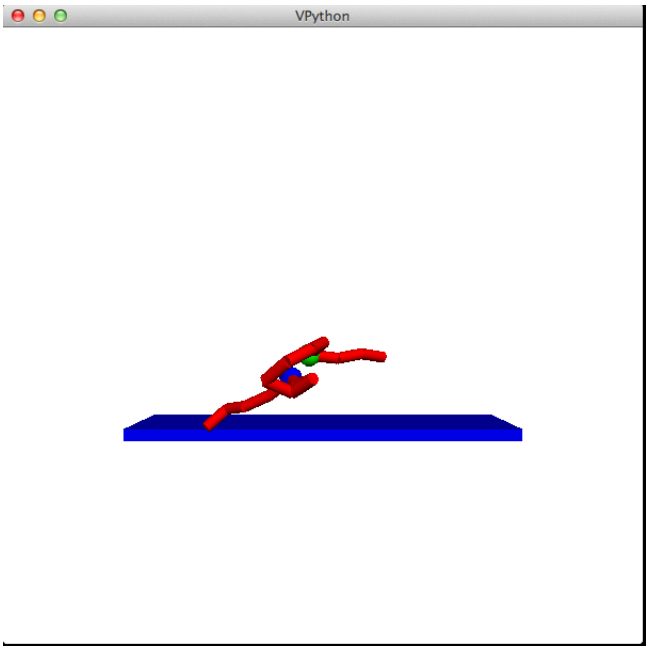


FIG. 7. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a simulation of a molecule with two metastable internal states, used to study single molecule force spectroscopy. An external force is applied to the chain ends, and this force is slowly increased.

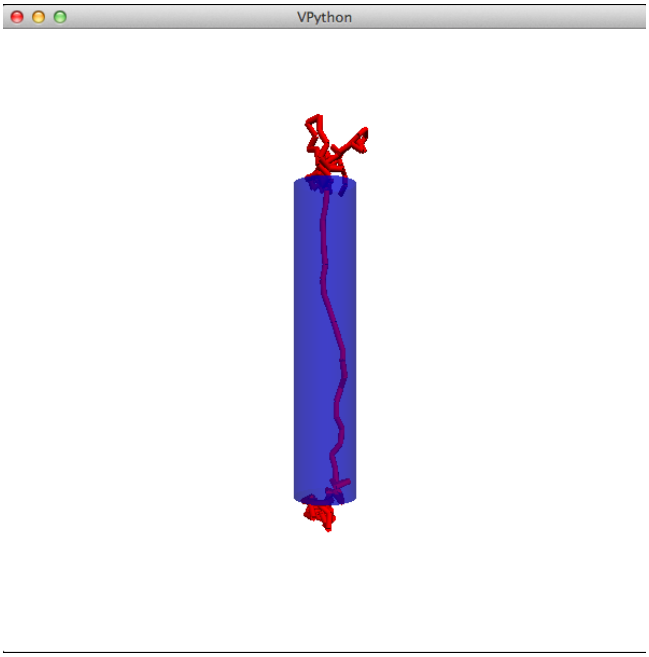


FIG. 8. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a simulation of a macromolecule translocating through a pore in a membrane, visualized using Visual Python.[5]

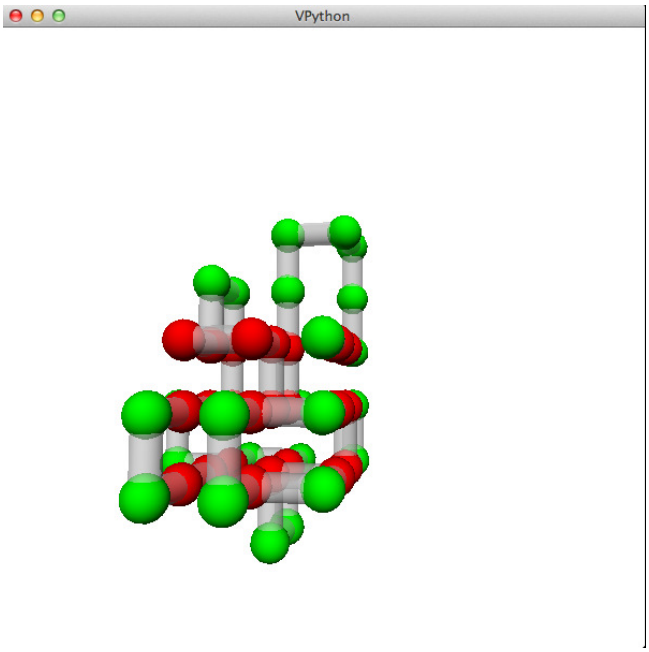


FIG. 9. A snapshot of the folding of a protein on a lattice, for the HP model, using Monte Carlo and simulated annealing.

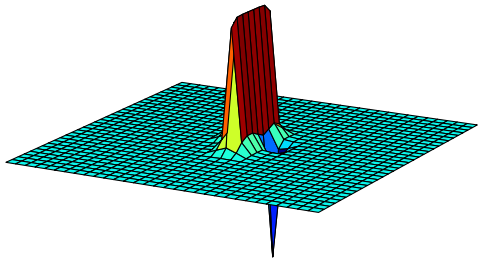


FIG. 10. A numerical calculation of the potential around a collection of charges in two dimensions, according to the nonlinear Poisson-Boltzmann equation.

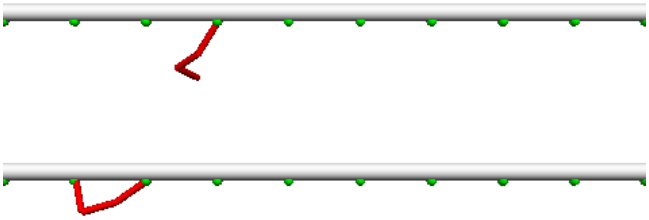


FIG. 11. Enhanced online [URL TO BE INSERTED BY AIP]. Two snapshots of a minimal simulation of myosin V on actin, capturing the essential physics of motor motion. The motion is similar to hand-over-hand motion used on monkey bars.

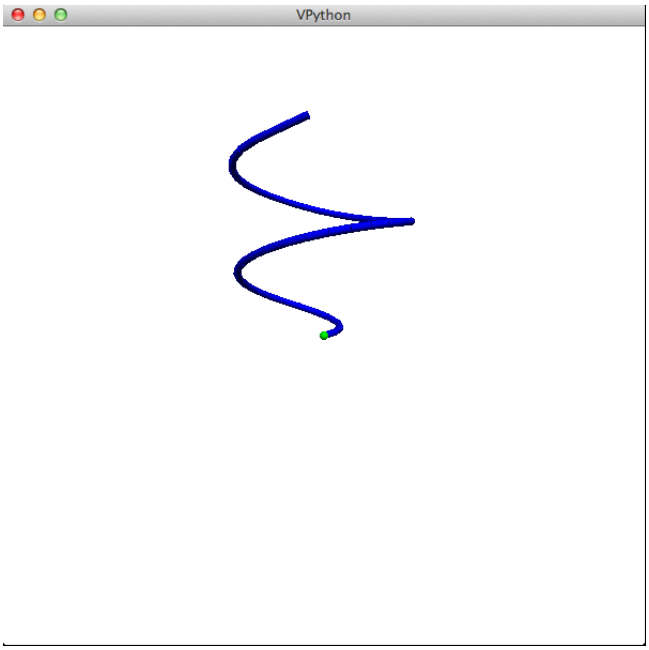


FIG. 12. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a microtubule moving inside a Drosophila oocyte. The rotating helical motion is a consequence of the force generated by kinesin motors walking along microtubule tracks.

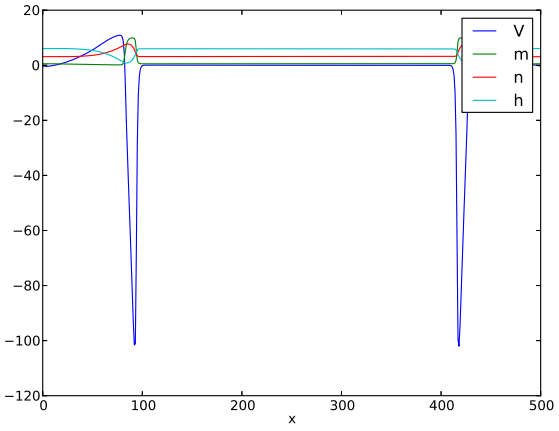


FIG. 13. Enhanced online [URL TO BE INSERTED BY AIP]. A snapshot of a simulation of action potentials travelling down an axon as described by the Hodgkin-Huxley equations.