COMPUTATIONAL BIOLOGY INSTRUCTION AT THE UNIVERSITY OF WASHINGTON CENTER FOR BIOENGINEERING

E. SWANSON and T. P. LYBRAND
University of Washington, Molecular Bioengineering Program, Box 351750, Seattle, WA 98195-1750 USA

Computational tools are rapidly becoming an essential component of molecular biology research. However, there is as yet relatively little attention paid to computational biology in the standard curricula of most biology programs. We describe here some of the graduate computational biology courses we have developed in the Center for Bioengineering at the University of Washington, with special emphasis on instructional methods and approaches that appear to work well.

1 Introduction

Computation has become a major research tool for most scientific disciplines.[1] The role of high-performance computing in the physical sciences and engineering disciplines is well established. The quantitative definition of high-performance computing is somewhat arbitrary, and varies from one discipline to the next. In astrophysics, theoretical chemistry, or computational fluid dynamics, high-performance computing often implies gigaflop (1 gigaflop = 1 billion floating point operations per second) calculation speeds. In other disciplines, computation speeds available from current RISC workstations qualify fully as high-performance computing. In some cases, fundamental properties of matter can be calculated as accurately as they can be measured with state-of-the-art instrumentation. As early as 1968, Kolos and Wolniewicz performed extremely accurate quantum mechanical calculations for the dissociation energy of molecular hydrogen. Their result, in disagreement with the best experimental measurements of that time, was later shown to be correct to within approximately 0.0002 percent.[2] Thanks to advances in numerical methods and computer resources, airplanes or boats can be designed entirely on the computer, without resort to expensive scale models or testing. For example, Boeing proudly advertises their new 777 jumbo jet as the first plane to be designed "entirely on computers".[3] America'a Cup yachts (and many other marine vessels) are designed primarily on computer, with little or no scale model or prototype building prior to actual boat construction.[4] In other engineering design applications, model construction and testing has been abandoned completely in favor of computer simulation, coupled with interactive computer graphics. The Caterpillar heavy equipment manufacturer now designs the operator cabs for their machinery using supercomputer simulations and a virtual reality graphics projection system developed at the National Center for Supercomputing Applications.[5] The virtual reality system consists of a room that is designed to allow computer graphics images to be projected on the ceiling, floor, and three walls, giving the "passenger" the impression of total immersion in a computer-generated environment (in this case, the cab of a large tractor). Design changes may be made almost instantly, via instructions to the simulation program running on the supercomputer. Thus, the effect of placing the operator controls in a different location, making the windows 5 percent smaller, or moving the driver's seat 2 inches closer to the windshield can be accessed immediately. Architectural design is now done primarily on computers, and is usually coupled with three-dimensional interactive graphics to permit architects and clients to "walk through" a building during the design phase.[6] As in equipment design, the consequences of design characteristics can be evaluated immediately, and changes can be made and evaluated interactively. The motivations for use of computer-based design in these applications are simple: costs and design times are reduced significantly over traditional approaches that involve construction of physical models and prototypes. Computational "experiments" can also often be used to explore conditions or situations that are unsafe or unattainable in the laboratory. Computed-aided design and engineering is practical in many disciplines because computational power and algorithms are now adequate to represent the essential physics of the systems in question (e.g., air flow over a jet fuselage). Extensive comparisons of computational results with experimental data have verified the accuracy and reliability of computer modeling methods for these engineering and physical sciences applications.[1] Because high-performance computing is well established in the physical and engineering sciences, training in computational techniques is now routine in most engineering and physical science curricula.

In contrast, high-performance computing has only recently emerged as a major research tool in most biological disciplines. While high-performance computing is now essential for tasks ranging from sophisticated sequence analysis to detailed biomolecular modeling, and prospects for the future appear nearly unlimited[7], there is still relatively little attention paid to computational biology in biology education. Few biology programs at present include much formal course work or training in computation methods as part of a general curriculum. As a result, there is a great demand for computational biology instruction. This need is filled partially by numerous workshops offered by professional societies, software vendors, and major computing facilities, but these workshops can serve only a limited number of people in a cost-effective way. It appears there is a clear need to incorporate computational biology training in standard biology curricula in a more formal manner. This training needs to address not only the use of powerful sequence analysis or molecular modeling tools, but also access to and use of databases and other useful biology resources now readily available on the World Wide Web. We describe here our efforts to establish appropriate computational biology courses in the Center for Bioengineering at the University of Washington, and our experiences to date with these courses in a general graduate curriculum.

2 Course Content and Implementation

2.1 Computational Biology Instruction

In the Center for Bioengineering at the University of Washington, our mission is to train interdisciplinary scientists who are able to use tools of physical and engineering sciences, including high-performance computing tools, to study biological molecules and systems. However, our computational biology courses serve a much larger audience, including students and faculty from throughout the medical school.

Our core computational biology course provides students with an introduction and overview for a wide range of computational techniques. In the course, there is heavy emphasis on molecular modeling and simulation (reflecting the specialization and focus of our faculty and staff), but protein sequence analysis and use of biological databases and other network-accessible resources is also prominent. The course is targeted to first and second year graduate students, and it is expected that students have reasonable backgrounds in chemistry and biochemistry upon enrollment. No attempt is made to provide "remedial" instruction in these areas. No prior experience with computational tools, or even basic computing, is assumed, however. The course consists of lectures, recitation sessions, and laboratory work. Lectures are used to present basic concepts and brief review of appropriate material from chemistry and biochemistry. The lecture material is supplemented with extensive reading assignments from the scientific literature. Recitation sessions are used primarily to discuss current literature, and to explore the ways computational biology techniques are applied in "real world" research projects, and how computational methods may be used in conjunction with experimental studies. In the recitation sessions, students are expected to take an active role in topic choice and discussion, in much the same spirit as a graduate-level journal club. The laboratory is by far the most important component of the course. In laboratory, students become proficient in the use of molecular graphics software, structural and sequence analysis tools, and network-based information search and retrieval.

Students are also exposed to more sophisticated molecular modeling methods, such as molecular dynamics. The laboratory exercises are divided into separate modules, each of which emphasizes particular skills and techniques. The modules must be completed in sequence, as each subsequent module requires that students draw upon skills and techniques learned in the previous modules. At the beginning of the course, short modules (1 week) are devoted to computer basics such as file management and text editor usage (this module is for novice computer users only) and use of network resources such as databases or computational tools accessible through the World-Wide-Web or Gopher connections. The students then progress to a series of modules (2-3 weeks) that introduce interactive molecular graphics techniques. These modules include exercises that require students to analyze protein structures using geometrical measurements, surface property calculations, etc. Molecular mechanics techniques, including energy minimization and molecular dynamics, are normally introduced in the next module (2-3 weeks). A typical exercise included with this module is a protein-small molecule or protein-protein docking project. The docking exercises require that the students couple their newly learned molecular mechanics techniques with the interactive graphics skills acquired in the previous module. We also include exercises that allow students to perform detailed numerical analyses for a protein molecular dynamics simulation. The students do not run the actual MD simulation as that would be impractical with 20 students in the course, but each student is required to perform all the necessary setup and preparation needed to run a detailed protein MD simulation. Then, a module that covers sequence analysis and alignment methods, secondary structure prediction techniques, and homology modeling tools is presented (2 weeks). The exercises in this module involve sequence database searches, sequence identification and classification, etc. After completion of this module, a final lab project is assigned. This modular approach to lab exercises allows students to proceed at their own pace (within reason). For example, someone quite familiar with general computer skills and molecular graphics software (techniques emphasized in the early modules) can move quickly into more detailed molecular modeling and sequence analysis exercises. Techniques and methods taught in each module are presented in the context of real computational biology problems, rather than some artificial task. For example, the sequence analysis module exercises require that students take an unknown sequence (i.e., a sequence newly reported in the literature) and search databases via WWW or other network tools to locate similar sequences and classify the unknown sequence in terms of protein family and probable function. The final laboratory project combines most of the techniques the students have learned during the course in an exercise designed to mimic (in greatly abbreviated form) a typical dissertation research project. A representative exercise from recent years is a protein homology modeling task. Students were given a sequence of a soluble, globular protein of unknown three-dimensional structure, and instructed to generate a 3D model for this protein using whatever tools seemed appropriate. In this final exercise, students are required to plan their research protocol and choose methods without any assistance from the instructional staff. A final report, written in the form of a research article, forms the basis for the laboratory grade. Grading is based primarily on the soundness and feasibility of their research strategy, execution of techniques, and, to a lesser extent, quality of the final three-dimensional model generated. The final module generally requires 3-4 weeks of relatively intense laboratory work. In addition to the final lab exercise, the course grade is also based on a written final exam (which tests for understanding of basic concepts and terminology), and two written reports. One report is an in-depth review of a specific computational technique, and the other report is an abbreviated NIH research proposal that integrates a significant computational component in the research plan. Most students choose some aspect of their dissertation project as the topic for this assignment.

This course has been extremely successful since its inception three years ago, and routinely attracts more enrollees than we can accommodate. As the course has evolved, we have continued to place greater emphasis on the laboratory sessions. In laboratory, the students gain a much greater appreciation for the capabilities and limitations of computational tools than would ever be possible in lecture or journal club formats. Upon completion of the laboratory modules, it is expected that students will be able to use one or more molecular graphics programs proficiently, and that they will have a rudimentary working knowledge of various molecular mechanics and sequence analysis techniques. Our experiences to date indicate that these are reasonable expectations for first and second year graduate students. The only disadvantages of our teaching approach are logistical in nature: it is extremely expensive and labor-intensive to teach a laboratory-based computational biology course. The instructional staff spends a great deal of time working with students one-on-one in the laboratory. It is also quite time-consuming to prepare appropriate laboratory exercises, much more so than preparation of a series of lectures, for example. It is essential that the laboratory be equipped adequately to provide a "real" computational biology research environment for the students. In fact, our laboratory is used as a research facility when classes are not in session. It is presently equipped with ten high performance interactive graphics workstations (including stereo viewing hardware and ample memory and local disk capacity) and two large, multi-processor compute and file servers. The laboratory is also equipped with a wide range of commercial and academic software for molecular modeling and sequence analysis. When we first began to teach the course, we utilized primarily commercial software such as Insight and Discover from Biosym. [8] We based this decision on the belief that the elaborate graphical user interfaces and tight integration of compute features in the commercial packages would make it easier for novice users to perform the lab exercises. In recent years, we have relied increasingly on academic software, much of it developed in our research group, as our primary teaching tools. We routinely use molecular graphics packages like Midas [9], PSSHOW [10] and MD-Display [11], and molecular mechanics programs like AMBER. [12] We discovered that many students were focusing too much on the elaborate interface features in the commercial packages. Our academic software packages lack sophisticated user interfaces for the most part, and require that students make detailed choices regarding input parameters and options before they can run a calculation. We feel this compels the students to understand the methods they use in more detail, and to think more carefully about how the calculations are performed (including more critical consideration as to whether a given calculation should even be performed). To insure that students have ample access to resources, course enrollment is restricted to 20 students per term. This enrollment cap insures that each student has personal access to a workstation for as much as ten hours each week. When we first started teaching this course, we often had students work in pairs in the laboratory sessions, but this proved undesirable. Students did not tend to "explore" the capabilities of techniques and software as much when they worked in teams, and often one member of each pair would become a passive observer. For these students, the laboratory exercises became little more than demonstration sessions conducted by their lab partners.

While this course comprises the core of our computational biology curriculum, several other computational biology courses are offered. Most focus on biosystems modeling (e.g., cardiovascular and microcirculation simulation, pharmacokinetic modeling, etc.) and medical imaging analysis and reconstruction. Many of these courses have adopted our basic teaching approach, namely, a heavy emphasis on laboratory exercises as opposed to lecture sessions.

2.2 Computational Biology as an Instructional Tool

As mentioned above, our teaching mission in the Center for Bioengineering is to train interdisciplinary scientists for biological and biomedical research. We have a diverse student population; most students enter the program with physics, mathematics, or engineering backgrounds, and little formal course work in biology or biochemistry. Many of these physical science and engineering students do not respond well in traditional biology courses. While many biology disciplines, such as protein crystallography, physiology, and genetics, are quantitatively rigorous, many other biology fields are still taught primarily as descriptive subjects. The lack of quantitative rigor contained in many biology courses seriously diminishes their enthusiasm for biology. We have attempted to develop courses that present basic biology concepts and material from a more quantitative and physics-based perspective, and we have begun to use various computational biology tools to facilitate the presentation of course material. Basic protein structure-function lectures are presented not in a classroom, but in the molecular graphics laboratory, and proteins are presented as mechanical devices, to be examined in the context of an engineering structural analysis problem. Evolutionary relationships among proteins can best be illustrated by having students actually perform multiple sequence alignments and generate simple phylogenetic trees. DNA replication and transcription are presented as standard information storage and retrieval problems, in much the same manner as would be taught in an electrical engineering course on information storage systems. In short, biological systems are presented as an assembly of molecular machinery, fully interpretable (at least in principle) using basic laws of mechanics and thermodynamics, and each important concept is presented in terms of mathematical equations and models, if possible. This change in perspective has had a dramatic impact for many of our students from engineering and physical sciences backgrounds. They respond more enthusiastically in class, and grasp the basic course material more quickly and completely than in "traditional" biology courses. This reflects to a great extent the tradition in engineering and physical sciences to make detailed quantitative measurements and to understand systems through construction of models. These "cultural attributes" of engineers and physical scientists interface nicely with the philosophy and methodology of computational biology. While our fledgling "Biology for Physicists and Engineers" course series is probably not appropriate for all students, it has certainly been quite successful in our department thus far. Our favorable experience with application of computational biology tools to help teach basic biology concepts, and similar experiences with computational chemistry tools to aid chemistry education[13], encourage us that computational biology research tools may have a much broader applicability in general biology education.

3 Summary

We have developed introductory courses to train graduate students and staff in the theory and application of computational techniques for biological research. We have enjoyed considerable success to date, as evidenced by the general popularity of the courses and the eagerness of students to use techniques learned in the course in their own research projects. Our experience clearly shows that extensive laboratory exercises greatly enrich the class. We have also discovered, somewhat serendipitously, that computational biology tools can in some cases be useful teaching aids in basic biology classes, especially for students from physics or engineering backgrounds. This discovery parallels recent observations in undergraduate chemical education, where it has been observed that molecular modeling and computational chemistry tools can enhance the learning experience substantially.

Acknowledgments

The course development described here has been supported in part by a grant from the Whitaker Foundation to the University of Washington, Center for Bioengineering.

References

1. W.J. Kaufmann and L.L. Smarr *Supercomputing and the Transformation of Science*, (W.H. Freeman, New York, 1993).
2. W. Kolos and L. Wolniewicz, *J. Chem. Phys.* **49**, 404 (1968).
3. D. Stover, *Popular Science* **244(6)**, 78 (1994).
4. J.A. Schofield, *Design News* **50(7)**, 54 (1995).
5. *NCSA Access* **7(2)**, (1993)
6. D.J. Mahoney, *Comp. Graphics World* **17(6)**, 22 (1994)
7. J.C. Wooley and M.N. Varma, *Basic Life Sci.* **63**, 1 (1994)
8. Insight II, Biosym Technologies, San Diego, CA.
9. T.E. Ferrin, C.C. Huang, L.E. Jarvis, and R. Langridge, *J. Mol. Graphics* **6**, 13 (1988)
10. E. Swanson, "PSSHOW: Silicon Graphics 4D version", Seattle, WA, 1990.
11. T. Callahan, E. Swanson, and T.P. Lybrand, *J. Mol. Graphics*, submitted.
12. D.A. Pearlman, D.A. Case, J.C. Caldwell, G.L. Seibel, U.C. Singh, P. Weiner, and P.A. Kollman, AMBER 4.0, University of California, San

Francisco, 1991.
13. R.L. DeKock, J.D. Madura, F. Rioux, and J. Casanova, in *Reviews in Computational Chemistry*, Vol. 4, ed. K.B. Lipkowitz and D.B. Boyd (VCH, New York, 1994)