AAPT Topical Conference Summary

Computational Physics for Upper Level Courses

Davidson College, July 27-28 2007

Overview

Over sixty physicists, and a stray mathematician or two, met at Davidson College July 27-28, 2007 to discuss the past, present, and future of computational physics in the undergraduate curriculum. The two-day conference was organized as a satellite meeting of the American Association of Physics Teachers Summer Meeting held in Greensboro, NC.

The meeting began with greetings from Davidson College Physics Professor and Chair of the Organizing Committee Wolfgang Christian, Davidson College Dean of Faculty and Vice President for Academic Affairs Clark Ross, and American Association of Physics Teachers Executive Officer Toufic Hakim. Christian, Ross, and Hakim set the tone for the conference in their statements on the conference focus, the College's mission, and the role of this conference in the mission of AAPT.

The conference consisted of three kinds of sessions: invited talks, contributed posters, and informal (yet organized) discussions. The invited talks provided introductions to many aspects of computational physics in the undergraduate+e curriculum. These ranged from using computation in research (Sprott), using computation to teach students techniques that are important for large scale problems without having to carry out large scale computations (Gottlieb), using computation with undergraduates (Bug, Esquembre, Heyer, and Shiflet) and general talks on the need for computation in the curriculum (Chonacky , Landau, and Swendsen).

Invited Talks

Norman Chonacky, Editor-in-Chief of *Computing in Science and Engineering*, gave an overview of the role of scientific computing in physics and other fields the current state of the integration of computational physics into the curriculum. He then stressed the need to rethink computational physics education. Norman suggested that there was a case for urgency in the determination of the present issues of computational physics education, motivated in part by data about what physics graduates actually do. AIP employment surveys indicate that the greatest difference between preparation and employment expectation was in the use of computational physics training. In addition, he suggested that we should revisit the way we think about physics, shifting from purely analytic methods to a mix of analysis and numerical techniques. Moreover, we need to recognize the difference between computational science and scientific computation. Norman also reported results from a recent survey in *Computing in Science and Engineering* in which, based on 250 responses, there appeared to be a mismatch between values and practice in computational physics. Nevertheless, several stable paradigms for integration of computational physics in the undergraduate curriculum emerged from the survey. As a result, building a computational physics community, begun in earnest by groups such as the Shodor Foundation, CiSE, and the TeraGrid Project, is extremely important. Reforms in computational physics appear to be easier to implement at the upper-undergraduate level. He also recognized that computation practice differs across disciplines and that, for sustainability, crossing disciplines is important.

Robert Swendsen began by asking how we can use computation to teach students physics. He then gave examples of physics visualizations, but concluded that the actual power of computation is having the students program. He indicated that during a computational tutorial (using Monte Carlo technique) for a broad audience, successful completion of the tutorial was in the inverse order from what one might intuitively expect: undergraduates were more successful than graduate students, post-doctoral fellows, and professors. This was due to the fact that the more advanced the participants were in their education, the less they were able to think numerically. He then asked, what are we doing now to teach computational physics? He pointed out that his institution, Carnegie Mellon, requires 6 analytical methods courses and an introduction to computer science course. Two computational physics electives are offered but computational physics is not required. The curriculum is determined by history and change is slow. (Swendsen pointed out that what we still call modern physics is the physics of the 1920s). He concluded that computation will be the basis for scientific progress in the 21st century and that we are still preparing our students to solve the problems of the 1960s. He reiterated that change is easier at the upper-undergraduate level, but he thought that it is necessary to change at the lower level as well.

Francisco Esquembre described his efforts to integrate computer modeling into the undergraduate curriculum. Modeling can help students' understanding of science and he argued that every college graduate should be familiar with computer modeling, but not necessarily be an expert programmer. He stated that we should be teaching numerical analysis and general programming skills through computer modeling. Problems in adopting this approach are that the standard computer science approach requires high-level programming and does not include scientific structures, computer graphics, GUIs and specific examples from areas other than computer science. There is no balance between technique and science, between too specific and too advanced. A possible solution is a tool that makes scientific programming easy (designs and builds simulations at a very high level, takes care of computer-specific tasks, and provides powerful interaction and visualization capabilities). Esquembre then described his own free and open-source tool called Easy Java Simulations (Ejs) and showed examples of how he uses it to teach computer modeling.

Angela Shiflet described how her computational physics curriculum allows her to reach out to science students from all fields. Angela began by describing the SIAM Working

Group’s determination that the common computational curriculum content components should be: simulations and modeling, programming and algorithms, applied mathematics, scientific visualization, numerical analysis, application domain content, team-based projects, presentation, and research or professional experience. Angela then described how she uses computational science and modeling in her curriculum. Angela reported on several of her undergraduate computational students who had very successful external internships.

Amy Bug described how she integrates computation and research into the curriculum at Swarthmore. She began putting her program and school into context with the quote, “a liberal arts education trains students to do nothing and prepares students to do everything.” She then highlighted the interdisciplinary nature of computational physics stating that it was a combination of computational physics, applied mathematics, and physics. Computational physics can (1) build intuition about the physical world (experiment, theory, and computation), (2) build generic problem-solving skills (model building), (3) help students explore and visualize families of solutions, both analytic and numerical, and (4) enlarge the number of accessible problems and find novel behavior.

Rubin Landau described how computational physics could serve as a focus for physics education. Rubin began by briefly describing his computational physics career which started in graduate school in 1966, and computation education since 1988, including establishing one of the first undergraduate degrees in computational physics. In his experience, one approach is not best for all situations. We need computational physics because it changes how we do science and what we do with increased computing power. Rubin concluded that as physics educators we have two choices: we must either change to include computation or continue to teach physics as a classic static discipline, like classic Greek. Rubin states that we must also have computational physics education, not just computation plus physics education. The benefit of computational projects is they are research-like and this is indispensible in the job market given that only 22% of physics majors “remain in physics.” Rubin concluded that students learn by doing and that traumatic experiences (like code failure) are educational. Most education is learning words and simple concepts; instead, a hands-on approach, where confusion is often the first step to understanding, is more effective.

Discussions

Equally as thought provoking as the invited talks were the posters and the organized discussions. Participants ranged from those with little or no experience with computation in the curriculum, to those that have successfully integrated computation into the curriculum, and every shade in between. This wide range of experience was evident in the diverse opinions that were generated by the discussions. Despite the diverse opinions about specifics, there were several general themes that all the participants agreed needed to be addressed.

Faculty members are reluctant to change, especially in areas where faculty members themselves are not experts. Given this reluctance, the mechanism for how we recommend and make change is important and needs to be as inclusive as possible. In addition, the recommendations must also take into account our students' widely varied backgrounds, from no computational experience to a great deal.

The approach to planning and implementing change should take these issues into account and can take either a top-down or a bottom-up approach. Departments may be more willing to change their upper-level courses first (top-down) or more willing to change their introductory courses first (bottom-up). The approach should be based on local expertise and departmental buy-in. Without departmental buy-in it is impossible to sustain change and reforms will most likely wither and die. The ultimate goal of these approaches is to develop computation across the curriculum, maintaining connections among courses from the beginning.

We need exemplars of successful computational physics programs from small schools to large universities. Physics education research can/should be used to help promote the use of computation in courses (hands-on learning, interactive engagement, research/project based learning). Specifically:

- Visualization tools (plotting functions, etc.) can be transferred to upper-level courses even when the course instructor doesn't use computation. The students should know to how to use these tools even if they are not required. Ideally, students should have consistency in using programming across the physics curriculum, even in "non-computational" classes.
- Some of the simpler tools (Physlets, Ejs, etc.) can be used even without a computational course as background. We encourage and support our fellow faculty to use such tools in their courses.
- Capstone research projects can invite students to use the basic skills that they learned in earlier computational courses to do something much more sophisticated. For these projects we want students to make progress in the quickest way possible. This often means they are forced to use the computational language and tools that they have already. Students should, however, use the tool that best suits the goals of the research and makes the computations easy to do so they can focus on the physics. Independent of the programming language or package used, the most important thing is to get students thinking about physics in terms of models.
- Should we fix on a single language/tool or should we try to introduce several tools to our students? Multiple tools may be expensive and may be hard to maintain. It might be best to focus on a single (or a few) tool(s), but faculty members may want to use their own favorite tools. This is even more of a problem across departments. Uniformity (using a single platform) may be best so that students can really master a single programming language or platform. There is also some

sense that students should be exposed to at least two computational languages/tools. Learning a second language allows students to identify important commonalities in the various implementations of the tools. It also forces the student to learn more about the first language as the student compares the two.

- Computation should be incorporated into advanced laboratory courses. Most advanced laboratories use computer data acquisition and analysis. Additionally, students can compare experimental results to the results of a computer model.
- Students should gain experience using a document preparation system, such as LaTeX.
- Students with a diversity of computational backgrounds can work in groups to bring various tools to bear on a project.
- Introducing computation early and in non-computational courses can help to change the culture in the department and get other faculty to acknowledge the usefulness of computation.
- Can we convince students to learn programming just by expecting them to know it? If the expectations are there, students may be quite capable of learning programming (at least at a rudimentary level) on their own.
- Students rarely come to college with programming skills, although some may be very familiar with programming and numerical work on their graphical calculators. The absence of experience may be good since they won't have bad programming habits either.
- In upper-level physics courses rapid prototyping is important as a preview for full coding. Coding helps students learn hardware and software limitations. In general students should progress from simple to advanced software packages. Pedagogical issues should be addressed along the way. Participants expressed concern about students' blindly using computational packages, since we want students to move beyond black box approaches.
- Do we want to build enthusiasm or build skills? Presumably for a higher-level course we want to build skills. Is there any value in using black-box tools? What level of "black-boxness" is best? Is it best to use a blend of bare-bones programming along with some black-box stuff? Currently it is important to know what's under the hood in simulations, but that may not be true in the future. The concern with this, though, is that we need to validate our programs, but that may not be possible in sophisticated programs without understanding the code. Sometimes "getting under the hood" may get students bogged down and keep them from learning the important concepts. For high-performance computing you can use MATLAB and "brown-box" coding, because the detailed coding occurs in a very low-level language. In many situations, students may not need programming, but will need to understand the algorithms (at a detailed level) that are used in the simulations they run. For classical mechanics, for example, an

intermediate approach using Ejs may be good. Some programming is required, but much of the work is done for them.

- Courses: Most participants recommended that a specialized computational course should be required. Students must learn to code to understand the details of the algorithms, but it is unreasonable to expect students to write really sophisticated programs. Some students seem to favor constructing programs from scratch. They need the experience of writing a program from nothing – but don't necessarily need to do this for every single computation. The programming requirements may depend on the student's background if the class is interdisciplinary: physics majors should code, but maybe other majors don't need to.

Model syllabi and course integration suggestions:

1) Just expect students to pick up programming on their own.
2) Full-scale computational physics program with multiple dedicated computation courses as well as some integration into the rest of the curriculum.
3) Insert computation into the existing course structure.
4) A single specialized course at a lower level that then supports computation in regular courses at the upper level. How late is too late? Sophomore year seems to be the latest time that would work effectively.
5) A specialized course at a low level and another at a high level to do more sophisticated work.
6) Push to get programming and computation into the high school curriculum. Then we can do more at the college level. Students ought to learn computation when they learn problem-solving.

Invited Talks

Integrating Computation and Research into the Liberal Arts Physics Curriculum
Amy Bug (Swarthmore College)

Motivation for Curricular Reform from Computational Applications to Physics, Other Sciences, and Engineering
Norman Chonacky (CISE editor)

Integrating Computer Modeling into the Curriculum
Francisco Esquembre (University of Murcia, Spain)

One Lattice Gauge Theorist's Perspective on Important Skills and Concepts for Computational Physics Courses
Steven Gottlieb (Indiana University)

Computation in Physics Education; Why, What and How?
Rubin Landau (Oregon State University)

Simple Models of Complex Chaotic Systems
J. Clint Sprott (University of Wisconsin)

What students (and professors) can learn about physics with a computer
Robert Swendsen (Carnegie-Mellon University)

Reaching Out to Science Students with Computational Science and Modeling
Angela Shiflet (Wofford College)

Bringing Computation to Life
Laurie Heyer (Davidson College).

Posters

OSP Quantum Mechanics: Interactive Computer-Based Curricular Material
Mario Belloni and Wolfgang Christian, Department of Physics, Davidson College

Eddy Current Simulations Using the Receding Image Method
Yao Liu, Columbia University
John Belcher, Massachusetts Institute of Technology

Combining Computational Physics with Video Analysis in Tracker
Douglas Brown, Department of Physics, Cabrillo College

Computational Course Projects and Undergraduate Research
Brian Clark, Department of Physics, Illinois State University

Computation in the Lawrence Physics Curriculum
David M. Cook, Department of Physics, Lawrence University

Computation in Classical Mechanics
Javier E. Hasbun, Department of Physics, University of West Georgia

Moving Reluctant Students and Faculty over the Computational Algebra Hurdle
Randall S. Jones and Joseph Ganem, Loyola College in Maryland

System Dynamics Markup Language: An open source modeling toolkit
David Joiner, New Jersey Center for Science and Technology Education, Union NJ, USA

Dissemination of Animation-Based Curricular Materials: BQ Database
William F. Junkin III and Anne J. Cox, Eckerd College

WebTOP: Three Dimensional Interactive Simulations and Activities for Teaching and Learning Waves and Optics
Taha Mzoughi, Kennesaw State University and John T. Foley, Mississippi State University

The Monte Carlo Method in Undergraduate Statistical Mechanics
Kelly Roos, Bradley University

Visualizing Tensors
Rob Salgado, Syracuse University

Creating a Computational and Visualization Laboratory
Jim Socacki, James Madison University

Finite Matrix Approximations in Computational Quantum Mechanics
Paul Stanley, Beloit College

Computational Methods in Applied Physics at Bethel University
Keith Stein, Bethel University

Computational Physics at Berry College, In and Out of the Classroom
Todd Timberlake, Berry College

Integrating Computation into the Curriculum with Applications to Nonlinear Dynamics
Brian Utter, James Madison University

Computer Simulations - An Introduction to Scientific Research
Katharina Vollmayr-Lee, Bucknell University

Computational Quantum Mechanics
J Jay Wang, University of Massachusetts Dartmouth

Integrating Undergraduate Research into the Computational Curriculum
Jie Zou, Eastern Illinois University

Organizing Committee

Wolfgang Christian (Chair)

Norman Chonacky

Rubin Landau

Robert Hilborn

Jan Tobochnik

This conference was sponsored by the American Association of Physics Teachers (AAPT) and Computing in Science and Engineering (CISE). Partial funding for this conference is being provided by Davidson College, Computing in Science and Engineering, the Shodor Foundation, the TeraGrid project, and NSF grant DUE- 442581.

Computational Physics for Upper Level Courses Conference Attendees