Computational Physics

A Better Model for Physics Education?

Computational physics provides a broader, more balanced, and more flexible education than a traditional physics major. Moreover, presenting physics within a scientific problem-solving paradigm is a more effective and efficient way to teach physics than the traditional approach.

A dramatic increase in computational power and use over the past decade has driven rapid advances in science, technology, and education. However, these advances' long-range effect on undergraduate physics education remains an open question. In the past, educators were content to have undergraduates view scientific computations as "black boxes" and wait until graduate school to learn what was inside. Yet, our increasing reliance on computers makes this less viable today and much less likely to be viable in the future. In response, schools have started to develop *computational–physics education*, in which the dash indicates a union of computation and physics on pretty much equal footing as individual courses or formal programs. Another response, which we can call *computational physics–education*, views the computer as a tool to advance physics education, without questioning what goes on inside the black box.

In this article, I suggest that computational–physics education presents the path forward because it integrates the tools and results from research into education and, by using research-rich experiences to stimulate and activate students, is based on solid educational principles. Think of it as "physics education *with* research" instead of "physics education research," which seemingly focuses on students' ability and failure to learn various physics concepts.

This article will provide evidence of the need for change in physics education—specifically, it reviews both the number and types of programs presently in existence and the intellectual content of such programs.

Need for Change in Physics Education?

A little more than 10 years ago, the American Institute of Physics (AIP) presented evidence of the need for change in the "standard model" of undergraduate physics education.[1] It surveyed physics majors five years after graduation, asking them which aspects of their education were most valuable in their current employment. The results (shown in Figure 1) indicate that for graduates whose primary field of employment is engineering, mathematics, or science, the three most important skills are scientific problem solving, information synthesis, and mathematical skills. Moreover, these skills are also highly important to graduates who find employment related to software, with this latter group having a high need for computer programming and software development.

1521-9615/06/$20.00 © 2006 IEEE
Copublished by the IEEE CS and the AIP

RUBIN LANDAU
Oregon State University

Not long after the AIP conducted its study, the US National Science Board issued a report about the importance of mathematics and computer skills.[2] The NSB found that only 35 percent of mathematics and computer science (CS) bachelors work in the same fields as their degrees, whereas an even smaller number (22 percent) of physical and biological science bachelors work in the same fields as their degrees. A similar trend exists at the doctorate level (74 percent versus 52 percent). One conclusion we might draw from these data is that requiring students to spend more of their time on physics, in an effort to get them to understand it the way professors do, deprives them of learning other things that could be equally or more important to their educations. This might also be moving physics into the class of mature, classical subjects that are less relevant to society's needs.

The US President's Information Technology Advisory Committee (PITAC) and other groups have stated that CS departments alone can't meet the growing need for computer professionals (although predictions of the needed numbers vary with the economy). The most recent PITAC report goes so far as to view computational science unequivocally as a "third pillar" of scientific inquiry (accompanying experiment and theory) and that "computational science is now indispensable to the solution of complex problems in every sector [because] advances in computing and connectivity make it possible to address problems previously deemed intractable or beyond imagination. Yet, despite the great opportunities and needs, universities and the Federal government have not effectively recognized the strategic significance of computational science in either their organizational structures or their research and educational planning" (www.nitrd.gov/pitac/reports/).

The computational science referred to here is the mother discipline to computational physics (CP), the subject of this article. As we view computational science in Figure 2a, we see that this multidisciplinary subject combines an application (physics), applied mathematics, and CS in the course of solving realistic scientific problems (Figure 2b). As indicated in the PITAC report, the use of computation and simulation has now become prevalent and essential to science.

Although diagrams such as Figure 2a have become visual clichés, they're particularly relevant here. First, as CP has matured, we've come to realize that it represents more than just the overlap of physics, CS, and math in a Venn diagram. CP is also a bridge that connects the three disciplines

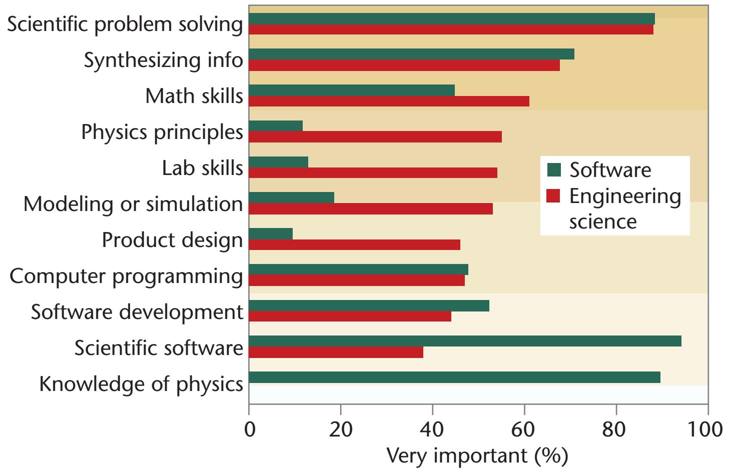
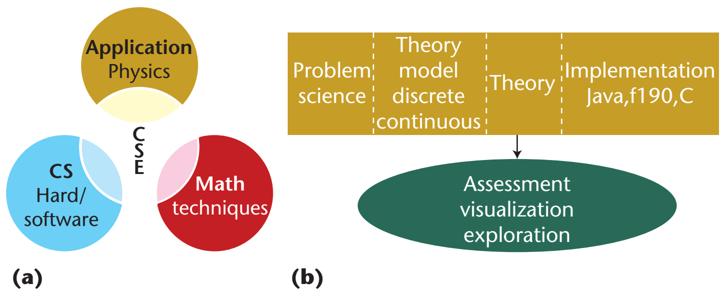


Figure 1. Importance of knowledge and skills five to seven years after graduation. The light bars are ordered by importance for physics majors whose primary field of employment is engineering, mathematics, or science; the dark bars show results for graduates employed in software development or programming. (Data courtesy of the American Institute of Physics.)



Application
Physics
CSE
CS
Hard/ software
Math techniques
(a)
Problem science
Theory model discrete continuous
Theory
Implementation Java,f190,C
Assessment visualization exploration
(b)

Figure 2. A new plan. (a) The three disciplines of computational physics and a bridge among them. (b) The scientific problem-solving paradigm used in educational materials.

even though it contains core elements of its own, such as computational tools and methods.[3] Second, as physics has matured and physics education research and computational physics–education have tended to focus inward on the traditional concepts of physics and mathematical physics, they've moved away from the center of Figure 2a and away from the problem-solving paradigm of science in Figure 2b. In contrast, CP's commonality of tools and shared problem-solving mindset draws it closer to other computational sciences and outward to address a broad range of new problems. Additionally, researchers have found that incorporating technology within the problem-solving paradigm is a more effective way to teach science and technology than focusing directly on individual components.[3]

Table 1. Results from surveys of undergraduate programs in the computational sciences.

	BS/BA degree programs (22)	Undergraduate minor, concentration, track, emphasis, option focus (21)	Foreign programs
Computational physics	1. Houghton College	1. Abilene Christian	1. Trinity College Dublin
	2. Illinois State Univ.	2. North Carolina State Univ.	
	3. Oregon State Univ.	3. Pennsylvania State Univ., Erie	
	4. State Univ. of NY, Buffalo	4. Univ. Arkansas	
	5. Christopher Newport		
Computational science	1. Stanford Univ.	1. Capital	1. Australian Nat'l Univ.
	2. State Univ. of NY, Brockport	2. Clark	2. Kanazawawa Univ. Japan
	3. Stevins Inst. Tech.	3. Old Dominion	3. Nat'l Univ. Singapore
	4. Univ. California, Berkeley	4. Rensselaer Polytechnic Inst.	4. Univ. Calgary
	5. Salve Regina	5. Univ. Erlangen-Nurnberg	
	6. Syracuse	6. Univ. Waterloo	
		7. Univ. Wisconsin, Eau Claire	7. Utrecht Univ.
		8. Univ. Wisconsin, LaCrosse	
		9. Univ. Wisconsin Madison	
		10. Wittenberg	
		11. Wofford College	
Computational biology	1. Carnegie Mellon	1. Univ. California, Merced	
	2. Univ. Pennsylvania	2. Center Computational Biology (Colorado)	
Computational mathematics	1. Arizona State Univ.	1. Princeton	
	2. City Univ. NY, Brooklyn	2. San Diego State Univ.	
	3. Michigan State Univ.	3. Univ. Central Florida	
	4. Missouri State Univ.	4. Univ. Nebraska, Lincoln	
	5. Rice Univ.		
	6. Rochester Inst. Tech.		
	7. Seattle Pacific Univ.		
	8. Saginaw Valley State Univ.		
	9. San Jose State Univ.		
	10. Univ. Chicago		
	11. Univ. Illinois, Chicago		

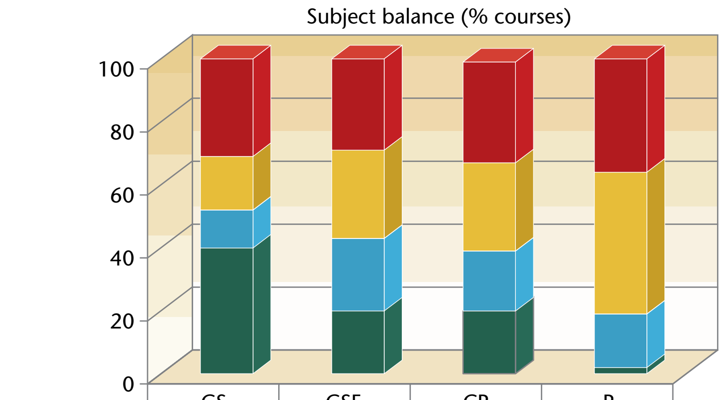
Computational Science Degree Programs

A bachelor's degree in any of the computational sciences is rare. Table 1 lists all the undergraduate computational science programs in the US, as determined by outside surveys,[4,5] with some updates. Note that the table shows only active undergraduate programs and excludes some programs that appear to be straightforward dual-degree programs without the bridge courses to draw the disciplines together. Although computation is finding its place within many disciplines and courses, the need for multidisciplinary computational science programs still appears to exist.

Currently, no professional organization of computational scientists or accreditation body decides the proper content for a "computational" education. However, the US National Science Foundation- (NSF-) supported Computational Science Curriculum Virtual Institute has been working on this, as have study groups with US Department of Energy sponsorship (see www.eotepic.org/moodle/course/category.php?id=8). Although we might presume that agreeing on such a standard would entail endless academic discussions, in practice, the balance of subjects for the existing computational programs appears quite similar, at least in a broad sense. Along with a coauthor,[5] I came to this conclusion after analyzing the published sample curricula of different programs and then calculating the average percent of the total curriculum dedicated to the broad categories of computing, mathematics, applications, and other subjects. Figure 3, taken from our work, compares BS programs in

CS, computational science and engineering (CSE), CP, and physics. (Although these numbers are the averages of several such programs; the exact values of the percentages might not have high significance because the number of programs is small, and we had to make some subjective decisions about which category to use for specific courses.) The left column shows strong computing (green) but weak application (red) components in the CS degree; the right column shows strong application but weak computing components in the physics degree. We see that an undergraduate degree in CP has a similar balance to one in CSE—namely, approximately equal weights for mathematics and computing (roughly 20 percent) and a higher weight for application (roughly 28 percent). This is a fairly uniform balance among components and, as expected, a CP or CSE degree contains less physics than a physics degree and less computing than a computational science degree.

The numbers appear to confirm our impression (prejudice?) that regular physics undergraduates might not be learning enough about computation, and that regular CS undergraduates might not be learning enough about math and science. In addition, although we haven't done any surveys, we've seen some physics curricula get even more imbalanced, possibly as a consequence of physics educators' efforts to eliminate students' misconceptions from the start or to deepen students' understanding of the mathematical foundations of physics. We suggest that a better way to eliminate misconceptions might be to apply physics to realistic problems.[6] The next section describes the steps involved and the materials needed to affect such an approach.



	CS	CSE	CP	P
🟥 Other	31	29	32	36
🟨 Application	17	28	28	45
🟦 Math	12	23	19	17
🟩 Comp	40	20	20	2

Figure 3. Degree programs. BS degree programs in (from left to right) computer science (CS), computational science and engineering (CSE), computational physics (CP), and physics (PH) exhibit a fairly uniform balance among components in terms of the average percent of the total curriculum dedicated to (from bottom to top) computing, mathematics, application, and other topics.

Computational Physics Pedagogy and Subjects

After two years in administrative processing, the Oregon State Board of Higher Education approved a bachelor's degree in computational physics in October 2001, separate from a traditional physics degree (www.physics.oregonstate.edu/~rubin/CPUG/). Although the first students didn't enter our program at Oregon State University (OSU) until late 2002, we had our first graduate, a transfer student, in 2003 and expect five four-year graduates in June 2006, for a grand total of 10. Some of the CP graduates are also dual majors with physics, math, or computational science, but approximately half are students who were drawn to the computational program and who wouldn't otherwise be at OSU or in physics. These numbers are small, but they're a good start. Our classes are well attended because the introductory classes are required of all physics majors, and other students (including graduate students) take the upper-level classes. Additionally, the OSU College of Science is in the process of starting programs in computational mathematics and computational biology, in which case, we'll share (and modify) our courses with those programs.

Table 2 shows a sample of OSU's CP curriculum. We've built it up course-by-course over time, as we've proposed, developed, and modified new courses while simultaneously teaching them. The computer classes (in bold) are distributed throughout all years of study. This idea started in 1989 with a senior-graduate-level two-term course in CP. We published the materials developed for that course in 1996.[7] Simultaneous with the completion of the text was an early exploration into the use of the newly developing World Wide Web to provide multisensory enhancements to the text. We've extended the course by another term with additional study of partial differential equations, wavelets, fluid dynamics, molecular dynamics, filtering, matrix libraries, visualization tools, object-oriented programming, and nonlinear dynamics. These new materials, and rewritten versions of the old, will appear in *A Survey of Computational Physics*,[8] to be published in late 2006 with simultaneous Spanish and Korean versions.

Table 2. Sample curriculum for the BS in computational physics degree showing course numbers and credits (1 credit = 10 class hours).

	Fall	Winter	Spring
Fresh (46)	Differential Calculus	**Scientific Computing I**	**Intro Computer Science I**
	General Chemistry	Integral Calculus	Vector Calculus I
	Fitness/Writing I	General Chemistry	General Phys
	Perspective	Perspective	Fitness/Writing I
	CP/CS Seminar		
Sophomore (45)	**Intro Computer Science II**	**Discrete Math**	**Scientific Computing II**
	Writing II	Infinite Series and Sequences	**Linear Algebra**
	Vector Calculus II	General Physics	Applied Differential Equations
	General Physics	Perspective	Intro Modern Physics
Junior (44)	**CP Simulations I**	**CP Simulations II**	Periodic Systems
	CP Seminar	**Data Structures**	Classical/Quantum Mechanics
	Intro Probability	Waves in 1D	Energy and Entropy
	Oscillations	Quantum Measurement	Biology Perspective/Elective
	Static Vector Fields	Central Forces	
	Writing III/Speech	Elective/Perspective	
Senior (45)	**Numerical Linear Algebra**	**Advanced CP Lab**	**Thesis**
	Electromagnetism	**Social & Ethical CS**	**Interact Multimedia**
	Mathematical Methods	Electives	CP Seminar
	Elective	Synthesis	Electives

CP = computational physics; CS = computer science; computer-intensive courses are shown in bold.

In 1997, we introduced a one-quarter Introductory Scientific Computing course designed to provide first- and second-year students with the computational tools they'd need throughout their undergraduate careers. Princeton University Press published these materials in 2005 as *A First Course in Scientific Computing*.[9] (The change in publishers reflects the difficulties that for-profit publishers have when marketing multidisciplinary books.) In recognition of the widespread disagreement over which computing tools lower-division college students should learn, the paper text covered Maple and Java, whereas the accompanying CD contained essentially identical texts in Mathematica and Fortran90, along with the associated notebooks, worksheets, programs, and data sets. Combined, *A First Course in Scientific Computing* and *A Survey of Computational Physics* pave a continuous computational path throughout the undergraduate curriculum.

The least-developed component of our curriculum is the advanced computational laboratory. In it, senior CP students experiment with computer simulations taken from graduate research projects. We modified the scientific descriptions and actual computer simulations to make the research experience accessible to undergraduate students in a short time (in contrast to the people-years required to develop the research codes originally). The students get the codes running, investigate some suggested problems, make some modifications, and then compare the results to those published in the literature. For many students, this is their first experience with truly large programs, old-fashioned Fortran, and articles in the scientific literature.

A key component of the CP program is getting students actively engaged with projects (as if each were an original scientific investigation) in a large number of areas. In this way, students experience the excitement of their personal research, get familiar with several different approaches, acquire confidence in making a complex system work for them, and continually build on their accomplishments. We've found the project approach to be flexible, and it encourages students to take pride in their work and creativity. It produces significant learning, even though we might be "teaching with our mouths shut,"[10] and it works well for independent study or distance learning.

The materials and classes we've built along the way reflect our own rules of education, which are personal observations gleaned from decades of teaching:

- Most of education is learning what the words mean; the concepts are usually simple if only you can understand what is being said.

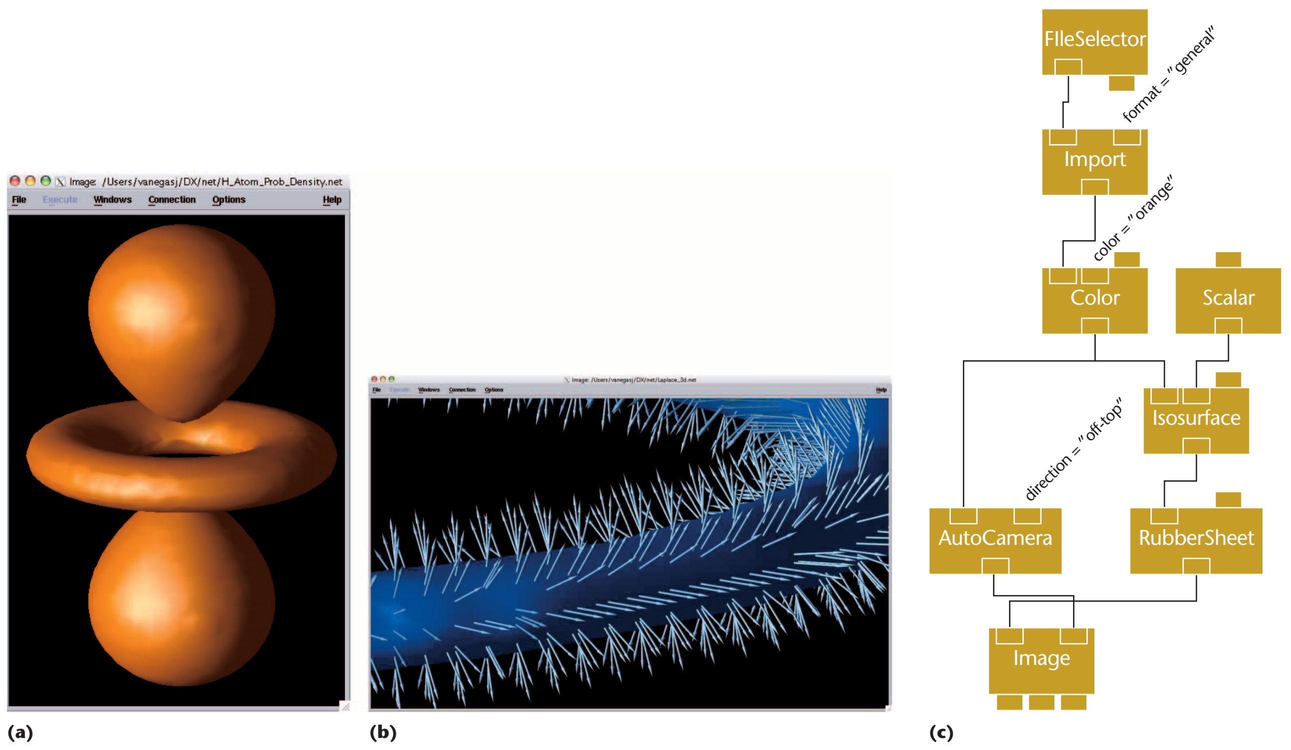


Figure 4. Example visualizations produced with OpenDX. (a) The 3D state of hydrogen, (b) an equipotential surface for a toroidal capacitor with the resulting electric field, and (c) the visual program that produced the hydrogen visualization.

- Confusion is the first step to understanding.
- Traumatic experiences tend to be educational.
- Scholarly and pedagogical presentations are often designed to either impress the audience with the presenter's brilliance and depth or to make the materials appear simple and logical (we opt for the latter).

We also adhere to the science, technology, engineering, and mathematics (STEM) education knowledge base as summarized in seven principles of good practice for undergraduate learning":[11]

- Encourage student–faculty contact.
- Develop student reciprocity and cooperation.
- Encourage active learning.
- Give prompt feedback.
- Emphasize the importance of spending time on a task instead of skimming it.
- Communicate high expectations.
- Respect diverse talents and ways of learning.

We run all our computation classes with a combination of lectures and "over the shoulder" labs. The students work on and discuss their projects with an instructor and then write them up as "executive summaries" with sections for the problem, equations, algorithms, code listings, visualizations, discussion, and critique. The emphasis is professional, much like reporting to a workplace manager.

Visualizations are important for all the classes, and we teach the use of Maple, Mathematica, PtPlot, gnuplot, AceGr, and OpenDX for 2D, 3D, and animated plots (all are open source except for Maple and Mathematica). Figure 4 shows samples of the types of plots possible with OpenDX, a professional-level package that can handle large, multidimensional data sets.

Figure 5 shows the actual topics covered in CP classes and their connections to other subjects as a concept map. This map is essentially a fleshed-out version of Figure 2a and was produced as part of the EPIC collaboration. On the left, we see hardware and software components from CS; in the middle, applied mathematics algorithms; and on the right, physics applications. Table 3 lists specific subjects covered in the individual courses, with an asterisk indicating Web-available enhancements.

The Next Step

Although people don't typically view the Web as a good teaching medium for general physics or as ap-

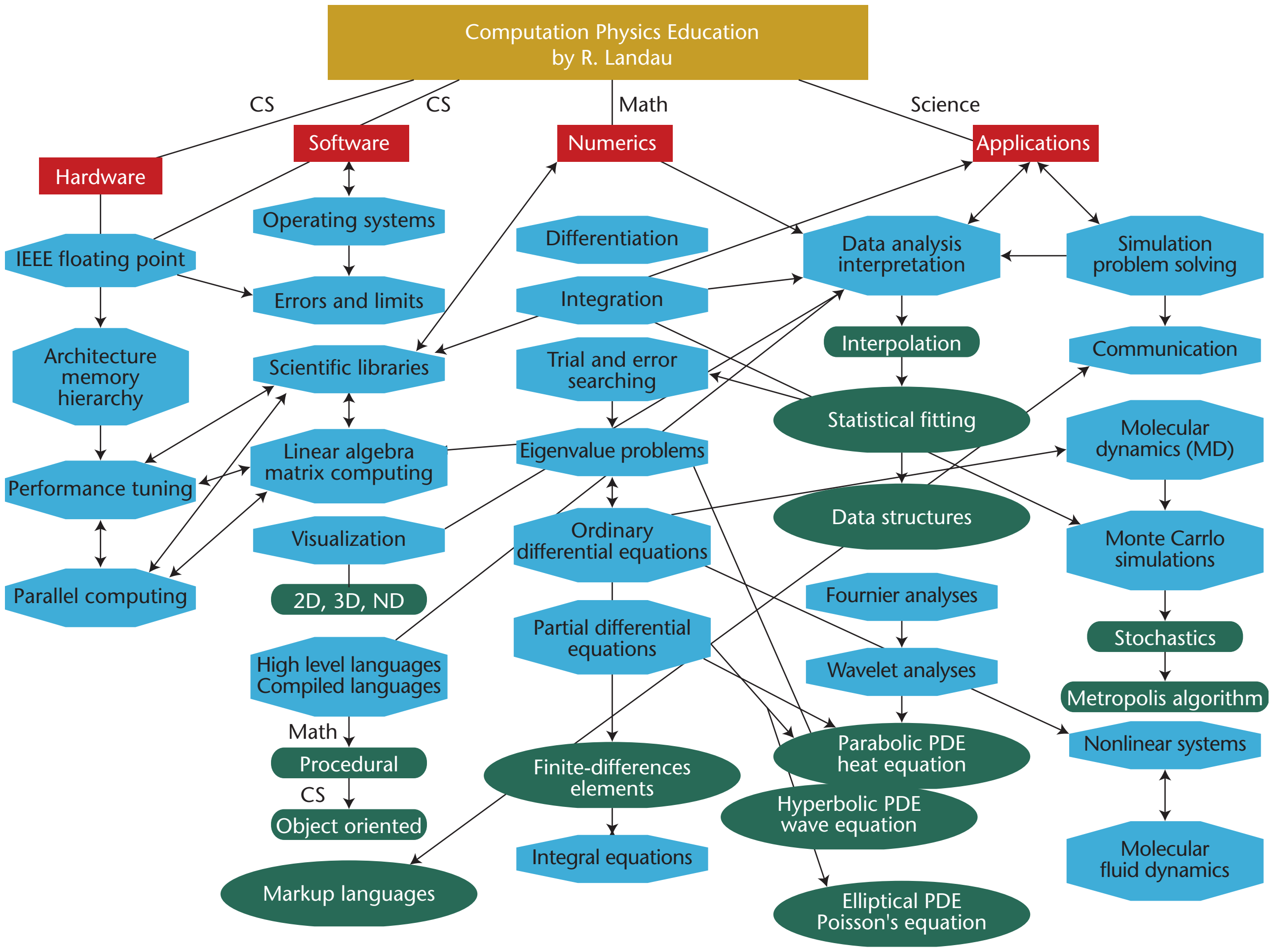
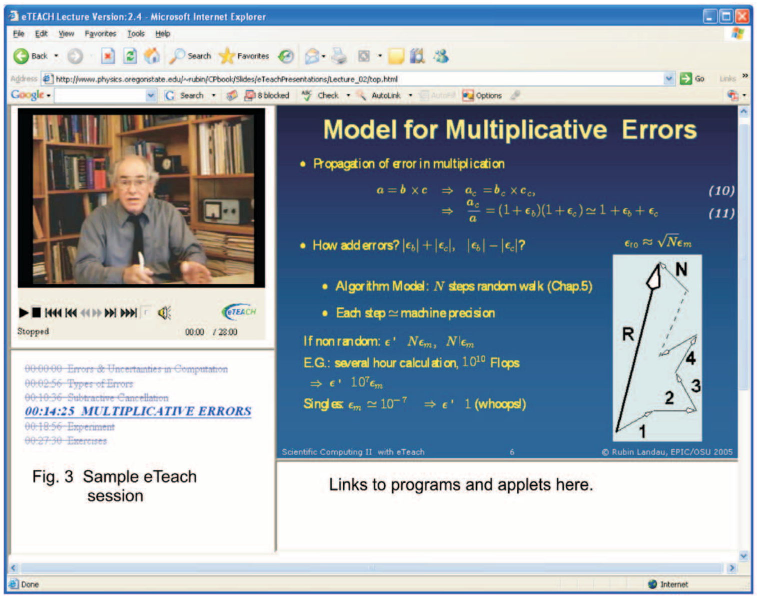


Figure 5. Concept map. This fleshed out version of Figure 2a shows hardware and software components from computer science, applied mathematics algorithms, and physics applications.

propriate for students with weak self-discipline or limited motivation, the best way to learn scientific computing is sitting at a computer in trial-and-error mode.[12] Furthermore, the Web is an ideal environment for computational science: projects are always in a centralized place for students and faculty to observe, codes are there to run or modify, and interactive visualizations can be quite striking with 3D, color, sound, and animation. The educational materials that our research group and Manuel J. Paez's group at the University of Antioquia, Colombia, developed contain several Web enhancements that provide alternate viewing modes, ideally to improve students' understanding of complex and abstract materials.[13]

I have a long-standing interest in advancing the level of digital science and mathematics books by using multimodal and interactive elements. Specifically, I want to see hybrid instruments that incorporate a tutoring approach to teach objective materials and computer simulations to develop more tacit understanding. Equations in the text should have direct interactions with Maple and Mathematica (possible with XML or MathML), Java and Fortran codes should be directly executable from the text, and the figures should have multiple, interactive layers that promote learning at multiple levels. Not only would this benefit disabled students, but it would also let any reader use a variety of senses to understand the materials.

As part of my work with EPIC, we're converting some of our conventional courses into electronic forms appropriate for distant delivery. Although you can find numerous computational science tutorials, applications, applets, and reading materials on the Web, there is little in the way of complete courses. This ongoing work is part of a larger effort to disseminate OSU courses and a first step in


eTEACH Lecture Version:2.4 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
http://www.physics.oregonstate.edu/~rubin/CPbook/Slides/eTeachPresentations/Lecture_02/top.html
Stopped 00:00 / 28:00
Done Internet

Model for Multiplicative Errors

- Propagation of error in multiplication

$$a = b \times c \;\Rightarrow\; a_c = b_c \times c_c, \quad (10)$$
$$\Rightarrow \frac{a_c}{a} = (1+\epsilon_b)(1+\epsilon_c) \simeq 1 + \epsilon_b + \epsilon_c \quad (11)$$

- How add errors? $|\epsilon_b| + |\epsilon_c|, \quad |\epsilon_b| - |\epsilon_c|$? $\epsilon_{ro} \approx \sqrt{N}\epsilon_m$
  - Algorithm Model: N steps random walk (Chap.5)
  - Each step $\simeq$ machine precision

If non random: $\epsilon' \;\; N\epsilon_m, \;\; N!\epsilon_m$
E.G.: several hour calculation, 10^{10} Flops
$\Rightarrow \epsilon' \;\; 10^7\epsilon_m$
Singles: $\epsilon_m \simeq 10^{-7} \;\; \Rightarrow \epsilon' \;\; 1$ (whoops!)

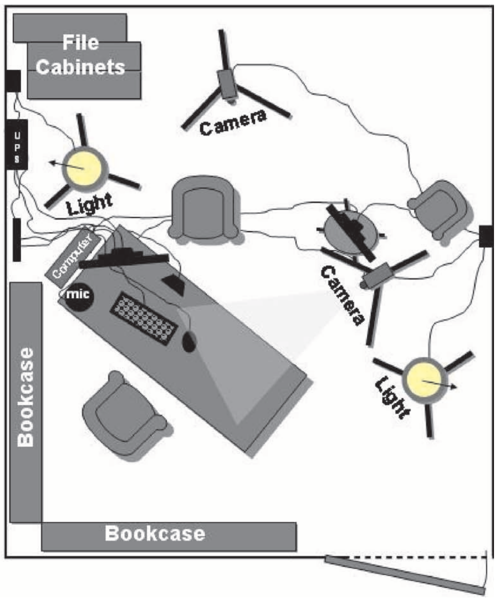
Scientific Computing II with eTeach 6 © Rubin Landau, EPIC/OSU 2005

~~00:00:00 Errors & Uncertainties in Computation~~
~~00:02:56 Types of Errors~~
~~00:10:36 Subtractive Cancellation~~
00:14:25 MULTIPLICATIVE ERRORS
~~00:18:56 Experiment~~
~~00:27:30 Exercises~~

Fig. 3 Sample eTeach session

Links to programs and applets here.

(a)



(b)

Figure 6. Online computational science course. (a) Recorded using eTeach, this screenshot shows a frame from the video, an interactive table of contents, a synchronized and animated slide, and links to programs and applets. (b) The "studio" in which the course was filmed shows the layout of cameras, lights, prompter (dual monitor), and desk, all in a standard faculty office.

establishing a national repository of university-offered undergraduate courses and modules in computational science. The repository would be a collection from various pioneering programs throughout the country, which together would cover the entire field of computational science at various levels. Once in place, these courses could offer other schools a way to include modern computation and multidisciplinary studies into their own curricula without having to hire specialists in the field or develop courses of their own or set up their own computational labs with all the requisite hardware and software for what is often a small number of students in any one location.

Figure 6a shows a screenshot from one of our first recordings done with eTeach (http://eteach.engr.wisc.edu/newEteach/home.html). It shows an image of the screen from the first lesson produced. The combination of video, slides, and table of contents is streamed over the Web or from a CD and viewed in a standard browser. It's in the informal format of a student questioning a professor in his or her office, with an actual office used as the studio (Figure 6b). The multimedia document contains

- a video and sound frame that replaces the lecture but serves a similar purpose of summarizing the materials, highlighting its major features, and placing it in a broader context;
- a PowerPoint (or other) slide frame that automatically synchronizes with the progressing office hour and other items on the slide;
- an frame with external links that lead to interactive applets, course materials, and quizzes, all synchronized automatically with the discussion; and
- a dynamic table of contents frame that highlights the titles of the lecture topics as they change, allowing jumps to any portion of the lecture at any time.

The eTeach materials use standard Web technologies such as the Windows Media Player and Internet Explorer to maintain a high degree of universality and multimedia interactivity. The system was developed and is in use at the University of Wisconsin. We will start using it, or a similar system, in 2007.

Beginnings are challenging. Although only time will judge the viability of programs such as ours, they do appear to attract new students and to provide them with broad preparation for future career choices. Based on personal experience, we advocate this type of program as a model that keeps physics relevant to a changing society yet teaches it better and with fewer credits.

Acknowledgments

This work is supported in part by a US National Science Foundation Grant and Engaging People in Cyber Infrastructure (EPIC). I thank Manuel Paez, Bob Panoff, Christian Bordeianu, Sally Haerer, Robyn Wangberg, Hans Kowallik, Scott Clark, and Kyle Augustson.

References

1. *Skills Used Frequently by Physics Bachelors in Selected Employment Sectors*, tech. report, Am. Inst. of Physics Education and Employment Statistics Division, 1995.
2. *Science and Engineering Indicators*, tech. report, US Nat'l Science Board, 1996, chapters 2–3.
3. O. Yasar et al., "A Computational Technology Approach to Education," *Computing in Science & Eng.*, vol. 8, no. 3, 2006, pp. 76–81.
4. C.D. Swanson, *Computational Science Education Survey*, tech. report, Krell Inst., Nov. 2003; www.krellinst.org/services/technology/CSE_survey/.
5. O. Yasar and R.H. Landau, "Elements of Computational Science and Engineering Education," *SIAM Rev.*, vol. 45, no. 4, 2003, pp. 787–805.
6. R. Root-Bernstein, *Discovering*, Random House, 1989.
7. R.H. Landau and M.J. Paez, *Computational Physics, Problem Solving with Computers*, John Wiley, 1997; www.physics.regonstate.edu/~rubin/CPbook.
8. R.H. Landau, M.J. Paez, and C.C. Bordeianu, *A Survey of Computational Physics*, 2nd ed., Wiley-VCH, 2006; www.physics.oregonstate.edu/~rubin/CPbook/II/.
9. R.H. Landau, *A First Course in Scientific Computing*, Princeton Univ. Press, 2005; www.physics.oregonstate.edu/~rubin/IntroBook/.
10. D.L. Finkel, *Teaching with Your Mouth Shut*, Boynton/Cook, 2000.
11. A.W. Chickering and Z.F. Gamson, "Implementing the Seven Principles: Technology as Lever," *Am. Assoc. of Higher Learning Bulletin*, Oct. 1996, pp. 3–6.
12. P. Davis, "How Undergraduates Learn Computer Skills: Results of a Survey and Focus Group," *Technological Horizons in Education J.*, vol. 26, Apr. 1999, p. 69.
13. C. Dede et al., *Multisensory Immersion as a Modeling Environment for Learning Complex Scientific Concepts, Computer Modeling and Simulation in Science Education*, N. Roberts, W. Feurzeig, and B. Hunter, eds., Springer-Verlag, 1999, pp. 382–389.

***Rubin Landau** is a professor of physics at Oregon State University, where he also directs the university's BS degree program in computational physics. His research interests include theoretical/computational subatomic particle physics, high-performance computing, and computational science education. Landau has a BS from Cornell University and a PhD from the University of Illinois. He serves on the executive committee of the American Physical Society's Division of Computational Physics. Contact him at rubin@physics.oregonstate.edu.*