9-2006

Using Computational Methods to Reinvigorate an Undergraduate Physics Curriculum

Jaime R. Taylor
Marshall University, Jaime.Taylor@marshall.edu

B. A. King III

Recommended Citation
J. Taylor and A. King, "Using Computational Methods to Reinvigorate an Undergraduate Physics Curriculum," Computing in Science & Engineering (special issue Computation in Physics Courses) 8, 38-43, 2006.

Using Computational Methods to Reinvigorate an Undergraduate Physics Curriculum

Austin Peay State University's Department of Physics and Astronomy has reinvigorated its physics program by adding a required computational methods class and small computational components to classes across its curriculum. A front-to-back problem-management approach has required a change in the way the department assesses students' performance.

Computational techniques have been in use in physics since World War II. Today, the ever-increasing complexity of the problems scientists face has in turn increased the need for numerical computation across the sciences, from protein folding to molecular modeling, from plasma dynamics to particle physics. Nonetheless, undergraduate science programs across the US have been slow to include such computational methods in their curricula. Unfortunately, this is even true in physics, a fact that's especially disturbing considering the speed with which such techniques have been integrated into engineering curricula.

One of the most oft-cited reasons for not including computational methods in undergraduate physics curriculum is that, because there's so much other material to cover, computational physics just "won't fit." This can be especially true at institutions with smaller programs, in which adding new classes can be problematic whether there are too few students to take them or too few faculty to teach them. Another possible problem is an institution- or state-imposed limit on the number of classes that a degree program can require. Nonetheless, we do students a disservice by leaving the topic out entirely.

During the 1990s, Austin Peay State University's (APSU's) Department of Physics and Astronomy suffered from low enrollment and attrition. With the help of the Department of Mathematics, the physics department faculty decided to make the changes necessary to reinvigorate its struggling major program. We found that with minor curricular changes in the traditional physics core topics and some interdisciplinary cooperation, we were able to successfully address computational physics at the undergraduate level by adding a single computational methods course and incorporating small computational components in classes across our curricula.

Reinvigoration

By late 1999, APSU's physics department had fallen into stagnation, with no major changes to the curriculum in at least 20 years. As a result, the number of students majoring in physics was low—nine majors total in the 1999–2000 academic year. We graduated even fewer, averaging slightly more than one graduate per year for the years 1995 to 2000, with only a small fraction of students continuing on to graduate school.

To develop a thriving program, the faculty de-

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

JAIME R. TAYLOR AND B. ALEX KING III
Austin Peay State University

cided it should make curriculum changes with two main goals in mind:

- increasing the success rate of students in junior- and senior-level classes and
- including in the new curriculum a focus on practical skills that have immediate applicability to graduate research or the job market.

The second goal's main objective was to ensure student experience with computational and experimental physics and to provide students with front-to-back problem-management skills that would help them excel in research or industry. These targets mesh well with the needs of a student entering what the American Institute of Physics defines as a professional master's degree program,[1]

> "…[O]ne that addresses the current needs of the economy as well as the needs of its students by providing both fundamental knowledge and specialized skills. Fundamental knowledge is the foundation on which students build throughout their working lives while specialized skills help advance their careers immediately by enabling them to smoothly transition into the work place upon graduation."

We addressed these new curriculum goals by introducing three new classes:

- Theoretical Methods, a bridge course that students take in their sophomore year to prepare for the mathematical rigors of junior- and senior-level coursework;
- Experimental Methods, a project-based course taken in either the sophomore or junior year that focuses on data acquisition and instrument control; and
- Computational Methods, another project-oriented course that focuses on programming and numerical methods.

Due to the state-imposed limitations on the number of courses we could require a student to take (no more than 120 semester hours for any program, almost 50 of which are core requirements), we needed to redevelop our curriculum with a minimal change in the number of hours required. Consequently, we dropped the second semesters of both Mechanics and Electricity & Magnetism from the major and incorporated topics from these and other physics fields into the three methods courses. (It's important to note that the faculty also implemented many extracurricular changes as suggested in related research,[2] but those are outside this article's scope.)

Computational Tools

In the past 10 years or so, considerable effort has gone into developing and studying the inclusion of computational methods into the undergraduate physics curriculum.[3–6] The tools we selected to augment the APSU physics program closely mirror those used at Lawrence University.[5] We'd like to say that this set evolved in a predefined way, but we included each element in the program as we perceived a need or as funding became available (in the case of some software). As we added elements, we modified the curriculum accordingly. The tools we included are

- LaTeX,
- Mathematica,
- Matlab, and
- C++/Fortran.

LaTeX actually serves a twofold purpose. We place considerable emphasis on preparing and presenting results in all our upper-level laboratories and projects, and LaTeX is an integral part of preparing a professional manuscript. Also, by introducing LaTeX early on, it familiarizes students with compiling and debugging a set of commands much like a compiled programming language.

Considerable effort has gone into developing and studying the inclusion of computational methods into the undergraduate physics curriculum.

Mathematica, as an algebraic solver, is another tool that's accessible at an intermediate level (for algebraic or calculus problems), but the more powerful packages can help tackle more advanced problems as well (ordinary and partial differential equations, nonlinear problems, and so on). Mathematica and Maple are the two algebraic solvers in most widespread use. We chose Mathematica because the math and physics faculty were most familiar with its syntax, and APSU had a preexisting license for it.

Although it's true that numerical computations (such as finite-difference techniques for ordinary and partial differential equations) can be done with Mathematica, this wasn't the driving force in its development. Matlab and IDL, on the other hand, were developed specifically for crunching numbers

and viewing the results. Both packages provide the convenience of a fast development environment (with many included tools) and use an interpreted language so that students can focus on understanding the numerical techniques and the physics rather than the programming. Matlab grew out of the desire to put a GUI on the front end of Fortran packages such as LINPACK and EISPACK. IDL was developed to provide NASA scientists, particularly astronomers, with an environment to view and mathematically manipulate images. For this reason, IDL is used heavily in the astronomical community, whereas applied physicists and engineers use Matlab almost exclusively. Because most APSU physics graduates go on to graduate school in applied physics or engineering, Matlab was our natural choice.

Languages such as Matlab are excellent for fast development, but they carry the usual speed penalty of an interpreted language. Although the ever-increasing speed of computing hardware has mitigated this effect somewhat, it's still important for students to be able to address problems using a compiled language because of its use in most high-performance computing environments—even though Matlab and Mathematica now have distributed tools as well. Also, learning a compiled language and the concepts that come with it helps students gain a better understanding of how fast development environments work. C++ is by far the most commonly used modern (object-oriented) language, and Fortran is still in wide use in the scientific community due to legacy code, again, making them obvious choices.

In addition to these tools, we also chose two other package environments. Although these aren't traditional programming languages, they require many of the same skills as traditional programming. LabView instrument control and data acquisition are a necessity in experimental physics. Although Matlab can perform these tasks, LabView is the de facto industry standard for General Purpose Interface Bus (GPIB) instrument control and data acquisition. We use this product extensively in the Experimental Methods class.

Although no standard electronic simulation software package exists, most are based on the Simulation Program with Integrated Circuit Emphasis (SPICE), which was released by the Electronics Research Laboratory at the University of California, Berkeley. We use the Multisim package from Electronics Workbench in our electronics course.

Computational Curriculum

Clearly, we were only able to incorporate a small number of the necessary computational tools into a single computational methods course, so we had to cover the rest of them elsewhere. Just as most traditional physics programs have a logical progression in their conceptual and mathematical complexity, so too should the order in which students are introduced to computational tools and the intricacy of their application. To this end, we decided to cover the more accessible instruments in the sophomore- and junior-level core physics classes. The experience then culminates in the senior-level Computational Methods course. Table 1 lists the computational components and the classes in which we use them in our curriculum.

Physics majors are encouraged to take Introduction to Programming (C++) either in their freshman year or in the first semester of their sophomore year, to introduce them to programming concepts. The Modern Physics course and its required laboratory are the first classes students take past their freshman University Physics sequence. We introduce Mathematica with classroom examples in the Modern Physics lectures. These consist primarily of things such as a derivation of the relativistic velocity addition equations, 1D analytical solutions of Schrödinger's equation, and visualization of the spherical harmonics during the solution of the hydrogen atom. The first step in developing students' front-to-back problem-solving skills is to give them a definite problem to solve, such as a predefined laboratory assignment. We have them submit their laboratory reports in LaTeX (which has the added benefit of strengthening their conceptual understanding of coding) and present one laboratory to the class using Microsoft PowerPoint.

In the spring of their sophomore year, we advise students to take Theoretical Methods. To familiarize them with the Matlab environment, we introduce it as a tool to visualize analytic solutions of damped and forced harmonic motion. Matlab is also used to analyze the convergence of Fourier series solutions arising from separable solutions to partial differential equations. In an effort to unify the students' sophomore year, the APSU mathematics faculty agreed to move their first-semester Differential Equations class to the spring semester and made curricular changes to the course. Students are required to complete a final project of their choosing that must be solved using Mathematica, written up in LaTeX, and presented in PowerPoint.

Past the sophomore year, the order in which students are introduced to other applications of these instruments is less structured. We've included them

Table 1. Austin Peay State University's physics-major program courses.

Year	Course	Tools incorporated
Freshman	CSci 1010 Intro. to Programming I	C++
Sophomore	Phys 3700 Modern Physics	Mathematica, LaTeX
	Phys 3005 Theoretical Methods	Matlab
	CSci 2010 Intro. to Programming II*	C++
	Math 3120 Differential Equations I	Mathematica, LaTeX
Junior/Senior	Phys 3030 Electricity & Magnetism	Matlab
	Phys 3050 Electric Circuits†	Electronics Workbench
	Phys 3550 Experimental Methods	LabView
	Phys 3800 Quantum Mechanics	Mathematica
	Phys 4000 Computational Methods††	Matlab, C++, Fortran
	Phys 4300 Image Processing*	Matlab
	CSci 3005 Object Oriented Programming*	C# using ASP.NET
	Math 3130 Differential Equations II*	Mathematica, LaTeX
	Math 4450 Math Models*	Mathematica
	Math 4460 Applied Math*	Fortran
	Math 4670 Numerical Analysis*	Fortran

Unmarked classes are required of all physics majors

** Courses required for students interested in our computational methods emphasis*

† Physics electives

†† The only new class added to the preexisting physics curriculum to address computational physics

in some of the required courses such as Electricity & Magnetism and Quantum Mechanics as well as electives such as Image Processing and Special Topics. In junior- and senior-level physics courses, we typically replace one exam with an extended one-to-two-week project utilizing either Matlab or Mathematica, a written report in LaTeX, and a classroom PowerPoint presentation. These projects can be individual assignments or group assignments (two or three students working together), depending on the class. Exposure to both types is essential; the first enhances the student's ability to manage a problem from beginning to end, and the second teaches them to work within the type of group dynamic they're likely to experience in graduate school or on the job.

When incorporating computational methods into preexisting physics courses, it was necessary to compromise more traditional approaches. For example, students in the first semester of a traditional Electricity & Magnetism course typically spend a great deal of time covering boundary-value problems. Unlike solving Schrödinger's equation for the hydrogen atom, which yields closed-form solutions, the solutions to the vast majority of boundary-value problems in electrostatics yield infinite series of Bessel functions or spherical harmonics. Although these solutions are exact, they're nonetheless infinite, and in problems containing discontinuities or singularities, the solutions don't converge rapidly and are thus of limited usefulness. Furthermore, these solution techniques are only valid for simple geometries, and unlike the solutions for the hydrogen atom, they yield little in the way of a deeper conceptual understanding of the physics at hand. We decided to more concisely cover traditional solution techniques by reducing the minutiae, which let us include a numerical treatment of boundary-value problems. Unfortunately, there isn't an undergraduate-level text on electrodynamics that incorporates numerical techniques—an area in which physics textbooks trail engineering textbooks.

Students that want to pursue graduate work or a career with an emphasis in computational methods must take additional courses in mathematics and computer science during their junior and senior years. They must take Math Models (which incorporates Mathematica), Numerical Methods (where Fortran is used), and Applied Mathematics (which also uses Fortran). Additionally, we encourage these students to take a course in object-oriented programming, which uses C#.

Computational Methods Course

Students typically take the Computational Methods course in the last half of their senior year, so it serves as a capstone for the curriculum (although

outstanding students sometimes take it in their junior year). By this time, students have programming experience in C++ as well as modeling complex physics and mathematics problems using Matlab and Mathematica. This prior experience ensures that the students are prepared to develop numerical techniques for application to a variety of physics problems.

Several texts cover advanced problem solving with examples across the breadth of physics, although they all tend to fall into two main types: those that focus on using Mathematica's algebraic capabilities[7,8] or numerical capabilities[9] to attack advanced problems and those that focus on compiled languages.[10–12] A few also focus on Java,[13,14] but Java isn't particularly suited to high-performance or distributed computing, so it's of limited usefulness to students who want to pursue computational physics in graduate school. The situation is different in engineering, where Matlab is frequently the focus to the exclusion of other tools.[15,16]

The ability to follow a complex problem from conception to conclusion is the most important thing we can teach students to make them successful.

Clearly, choosing a text is difficult. We decided that because of the number of our majors who were interested in pursuing graduate-level engineering, additional experience with Matlab was necessary, but not at the expense of neglecting compiled languages (which are necessary for most high-performance computing applications). In addition, a shorter text would be better because the course is taught in a single semester. In the end, Alejandro Garcia's *Numerical Methods for Physics*[17] was our ideal choice. This text includes examples with extensive prepackaged code, all of which is available in Matlab, C++, and Fortran.

Most of the weekly homework assignments in this class require the students to modify preexisting codes to fit their needs. This is an important skill for students (and one they don't cover elsewhere) because real-world problems almost always start with an existing code base. It also lets us cover the material more quickly. Essentially, we can discuss the entire text in a single semester, including ordinary and partial differential equations (both explicit and implicit finite-difference techniques), matrix methods, nonlinear systems, fast Fourier techniques, and numerical integration. An additional unit that covers genetic algorithms is generally included at the end of the semester, after we've covered the text.

Rather than using traditional exams, this class employs two three-week projects that are typically extensions of problems suggested in the text, although students are free to choose any problem they like. Two recent examples include stable orbits in a three-body problem and a simple model of ground resonance in articulated rotor aircraft.

Assessment

Ten years ago, the APSU physics faculty were concerned that students were less interested in learning the underlying concepts of physics and more interested in "getting the right answer." In an effort to combat this trend, we turned to Eric Mazur's *Peer Instruction*, in which he states, "exams determine the way in which students study."[18] We modified the exams in our calculus-based University Physics class so that 40 percent of each exam is made up of conceptual questions in the style of David Halliday, Robert Resnick, and Jearl Walker.[19] The remaining 60 percent are traditional pencil-based problems. Due to this change in our assessment method, we observed that students were more successful in their transition to upper-division coursework.

In revising the upper-level curriculum, we determined that the ability to follow a complex problem from conception to conclusion is the most important thing we can teach students to make them successful. We again realized that proper assessment would be key to achieving this goal. Hence, we must evaluate our students based on these abilities, but traditional in-class or take-home exams don't work in many cases. Learning front-to-back problem-management skills is a time-consuming process. Accordingly, we've woven smaller projects throughout the curriculum, which prepare students for the computational methods capstone course. To ensure that students assign these projects appropriate weight, they're a significant portion of the grade for their respective classes.

This revised curriculum resulted in a dramatic increase in the number of physics majors at APSU and a shift in their postgraduate career paths. By the 2005–2006 academic year, we had more than 60 physics majors, and between 2004 and 2006, we graduated 12 students. Ten of these

graduates received graduate assistantships in fields such as medical physics, biophysics, astronomy, mechanical engineering, electrical engineering, materials science, and atmospheric science. Of the two students who opted not to go to graduate school in the sciences, one is working in industry as a mid-level engineer and pursuing a master's degree in management, and the other has joined the US Navy.

Of course, there's always room for improvement. We haven't yet sufficiently integrated these tools into our Intermediate Mechanics class nor have we been able to sufficiently incorporate the use of Unix or Linux as a computing platform (the platform that students are most likely to encounter in graduate school). We also hope to be able to include a parallel and distributed computing component in the curriculum in the near future; we are pursuing external funding to support such an effort. CiSE

Acknowledgments

We thank the faculty of the mathematics and computer science departments at Austin Peay State University for their support. In particular, we thank William Glunt, Samuel Jator, and Nell Rayburn for the modifications they have made in their courses to help further our students' computational skills. This work has been supported by funding from the Tennessee Space Grant Consortium.

References

1. S.D. Norton, P.W. Hammer, and R. Czujko, *Mastering Physics for Non-Academic Careers*, Am. Inst. of Physics Report, 2001; www.aip.org/professionalmasters/mstphys.pdf.
2. R.C. Hilborn, R.H. Howes, and K.S. Krane, *Strategic Programs for Innovation in Undergraduate Physics Project Report*, Am. Assoc. of Physics Teachers Report, 2003; www.aapt.org/Projects/upload/SPIN-UP-Final-Report.pdf.
3. R.H. Landau, H. Kowallik, and M.J. Paez, "Web-Enhanced Undergraduate Course and Book for Computational Physics," *Computers in Physics*, vol. 12, no. 3, 1998, pp. 240–247; www.aip.org/cip/pdf/landau.pdf.
4. R.L. Spencer, "Teaching Computational Physics as a Laboratory Sequence," *Am. J. Physics*, vol. 73, no. 2, 2005, pp. 151–153.
5. D.M. Cook, "Computation in Undergraduate Physics: The Lawrence Approach," Am. Physical Soc., 25 Mar. 2004; www.lawrence.edu/dept/physics/ccli/talkaps304.html.
6. R.H. Landau, "Computational Physics for Undergraduates: The CPUG Degree Program at Oregon State University," *Computing in Science & Eng.*, vol. 6, no. 2, 2004, pp. 68–75.
7. G. Baumann, *Mathematica for Theoretical Physics*, 2nd ed., Springer, 2005.
8. R.L. Zimmerman and F.I. Olness, *Mathematica for Physics*, 2nd ed., Addison-Wesley, 2002.
9. D. Dubin, *Numerical and Analytical Methods for Scientists and Engineers Using Mathematica*, Wiley-Interscience, 2003.
10. P.L. DeVries, *A First Course in Computational Physics*, John Wiley & Sons, 1994.
11. N.J. Giordano and H. Nakanishi, *Computational Physics*, 2nd ed., Prentice Hall, 2005.
12. R.H. Landau and M.J. Paez, *Computational Physics: Problem Solving With Computers*, Wiley-Interscience, 1997.
13. T. Pang, *An Introduction to Computational Physics*, 2nd ed., Cambridge Univ. Press, 2006.
14. H. Gould, J. Tobochnik, and W. Christian, *An Introduction to Computer Simulation Methods: Applications to Physical Systems*, 3rd ed., Addison-Wesley, 2006.
15. T.L. Harman, J.B. Dabney, and N.J. Richert, *Advanced Engineering Mathematics with Matlab*, 2nd ed., Thomson Engineering, 1999.
16. S.C. Chapra, *Applied Numerical Methods with Matlab for Engineering and Science*, McGraw-Hill, 2004.
17. A.L. Garcia, *Numerical Methods for Physics*, 2nd ed., Prentice Hall, 2000.
18. E. Mazur, *Peer Instruction*, Prentice Hall, 1997, p. 16, 23.
19. D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*, 7th ed., John Wiley & Sons, 2005.

Jaime R. Taylor *is professor and chair in the Department of Physics and Astronomy at Austin Peay State University. His research interests include computational physics, evolutionary algorithms, and image processing. Taylor has a PhD in engineering science from the University of Tennessee Space Institute. He is a member of the American Association of Physics Teachers, the Tennessee section of the American Association of Physics Teachers, and the Tennessee Academy of Sciences. Contact him at taylorjr@apsu.edu.*

B. Alex King III *is an associate professor in the Department of Physics and Astronomy at Austin Peay State University. His research interests include computational physics, genetic algorithms, and high-energy physics. King has a PhD in physics from the University of Illinois at Chicago. He is a member of the American Physical Society, the American Association of Physics Teachers, and the Tennessee section of the American Association of Physics Teachers. Contact him at kinga@apsu.edu.*