Computation in undergraduate physics: The Lawrence approach

David M. Cook*
Lawrence University, Department of Physics, Appleton, Wisconsin 54912

The physics program at Lawrence University has introduced sophisticated computational techniques throughout its curriculum. Distinguishing features of the Lawrence approach include a focus on flexible, general purpose computational packages, application to theory and experiment, extensive use for preparing reports, and distribution throughout the curriculum. Most importantly, computation is introduced early enough so that students subsequently use computers independently on their own initiative. A required sophomore course in computational mechanics provides a uniform orientation to symbolic and numerical tools, and an elective junior/senior course in computational physics is offered. Student use of computational resources in independent studies and summer research experiences and positive comments from recent graduates provide evidence of the success and value of these curricular inclusions.

I. INTRODUCTION

Computers have been used to support the teaching of physics for at least four decades.[1] Until recently most applications focused on introductory courses, in which students typically supply parameters to instructor-written programs or access web-based resources, although elementary programming is sometimes included.[2] As valuable as these applications are, they should no longer constitute the entire exposure of undergraduate physics majors to computation. Several recent surveys[3] on the post-graduation activities of undergraduate physics majors support the conclusion that curricula must be modified to assure that graduates are familiar with the capabilities and limitations of a variety of tools, acquainted with an assortment of standard algorithms applied in physical contexts, and skilled in using computers not only for learning but also for doing physics.

Fortunately, attention is increasingly being paid to computation in the entire physics curriculum. Numerous texts focusing on computational physics[4] and some aimed at specific courses but with a computational component[5] are available, though each of necessity picks a particular software package, which constrains its adoption to contexts in which the chosen package is used. In the last several years, a group now calling itself the Partnership for Computing in Undergraduate Physics[6] has embarked on a systematic attempt to bring together those who have been striving in isolation to bring computation into the undergraduate curriculum at all levels.

Several different approaches to introducing computation beyond the introductory level have been explored. Some[7,8] have increased the flexibility and sophistication of canned programs, an approach that helps develop physical intuition, but does less than is ultimately needed to nurture the skills to create original computer-based solutions. Some[9] have added computational exercises as modules in intermediate and advanced courses, an approach that is complicated by the difficulty of finding the time to provide instruction on the capabilities of the tools. In particular, this approach risks focusing only on those resources needed for the specific exercise so students do not acquire a broad appreciation of the versatility and scope of computational approaches. Others[10] have introduced single, stand-alone upper-level computational physics courses that provide detailed background in computation, but have little impact on the rest of the curriculum and typically are offered late in the undergraduate program, so that students do not encounter computational approaches early enough for these approaches to become second-nature by graduation. Some[11] have introduced majors in computational physics or computational tracks within an otherwise traditional physics major. A few[12] strive to embed computing throughout the curriculum.

At Lawrence University we have tried during the past two decades to design a physics curriculum that, without short-changing important topics from the historical canon, includes opportunities to build skills in the application of computation to numerous problems from several areas of physics. This paper enumerates the underlying convictions that have guided our curricular developments, sets the context by discussing the structure of the Lawrence curriculum and its computational components, and then describes an intermediate course required of all sophomore majors and an advanced elective course available to juniors and seniors.[13] Our hope is that those who seek to incorporate computing into their curricula but are uncertain about strategies will see aspects of the Lawrence approach that can be applied or adapted to their own circumstances.

II. UNDERLYING CONVICTIONS

Practicing physicists routinely do a variety of tasks that support their activities but are peripheral to their main objectives. The more important and frequent of these tasks involve visualizing functions of one, two, and three variables; solving algebraic equations and ordinary and partial differential equations; evaluating integrals; finding roots, eigenvalues, and eigenvectors; acquiring, displaying graphically, and analyzing data; processing images; and writing reports and papers. Increasingly, computational approaches are used to complete these

tasks. To be able to invoke computational approaches when appropriate, physicists must be acquainted with at least one operating system, a versatile text editor, a spreadsheet program, an array/number processor,[14] a computer algebra system,[15] a visualization tool,[16] a programming language that supports standard packages and libraries, a program for circuit simulation,[17] a program for data acquisition,[18] a technical publishing system with the capability for easy inclusion of equations and figures,[19] a drawing program for creating publication quality figures,[20] and a presentation program. The curricula that train these physicists must include an exposure to at least some of these approaches and tools.

Beyond the substance of the approaches introduced, a curriculum that responds to these convictions must ensure that students know about the hazards of finite-precision floating-point arithmetic and other limitations of computation. In brief, graduates should have the ability to recognize when a computational approach has merit, knowledge of an assortment of computational tools, and the skill and confidence to exploit these tools wisely and independently.

III. THE DEPARTMENTAL CONTEXT

The Department of Physics at Lawrence University, a private, coeducational, liberal arts college and music conservatory of 1400 students, has five full-time faculty members and annually graduates an average of ten physics majors, 50% of whom go directly to graduate programs in physics or related areas. Each year, full-time students at Lawrence take three six-unit (3-1/3 semester hour) courses in each of three ten-week terms. Class periods are 70 minutes long.

The typical program of a physics major is shown in Table I. The minimum major is satisfied by the fourteen courses displayed in italics. Courses flagged with an asterisk direct students explicitly to the computer, and in most cases include instruction in at least one computational resource. In all the courses, students are free and are often encouraged to use those resources on their own initiative. Most graduates will have chosen courses in physics, mathematics, and computer science beyond the minimum required for the major, and many will have elected a senior capstone (research) project.

In the two decades since we adopted the goal of designing a curriculum that reflects the convictions laid out in Sec. II, we have equipped (and re-equipped) all laboratories with appropriate hardware and software, revised (and re-revised) our courses, and drafted (and re-drafted) hundreds of pages of instructional materials, some of which have been compiled into a book and an associated solutions manual.[21] Our curriculum now introduces first-year students to on-line data acquisition, the statistical analysis of data, and least squares fitting; introduces fall-term sophomores to the simulation of electronic circuits and computer-based preparation of laboratory reports in *Electronics*, winter-term sophomores to symbolic, numerical, and visualization tools in *Computational Mechanics*, and spring-term sophomores to relaxation methods for Laplace's equation in *Electromagnetic Theory*; reinforces techniques for visualization and the numerical and symbolic solution of ordinary differential equations, evaluation of integrals, and finding eigenvalues and eigenvectors in *Quantum Mechanics*; expands students' exposure to on-line data acquisition, techniques for data analysis and curve fitting, and use of software for generating reports in *Advanced Laboratory*; offers juniors and seniors *Computational Physics*; and incorporates computational approaches alongside traditional approaches in many intermediate and advanced courses.

IV. THE FIRST YEAR

Prospective physics majors first encounter computational approaches in the introductory, calculus-based courses, whose laboratory is equipped with Windows computers, a monochrome laser printer, LabPro[22] interfaces and several Vernier sensors, Spectrum Techniques units[23] for gathering data on radioactive counting rates, and computer-controlled NanoSurf scanning tunneling microscopes.[24] Exercises assigned in the nine three-hour laboratory sessions in each term routinely involve automated data acquisition in several experiments, statistical data analysis in essentially all experiments, least squares fitting of linear and parabolic functions in several experiments, and creation and processing of images with computer-controlled scanning tunneling microscopes. In the lecture portion of the course, students occasionally use the laboratory computers to plot theoretical results or solve ordinary differential equations via Euler and improved Euler methods.

V. THE SOPHOMORE YEAR

A serious introduction to "real" computation occurs in the sophomore year in which majors take a sequence of three required courses supported by the Computational Physics Laboratory, which has nine workstations running LINUX, monochrome and color laser printers, and software in nearly all the categories listed in Sec. II. Majors have 24/7 access to the laboratory.

In *Electronics* fall-term sophomores construct and study assorted analog and digital circuits. Early in the course, students are introduced to MULTISIM,[17] and use this software to complete homework assignments and to predict or interpret experimental results. Toward the end of the term, each student writes a journal-quality article on one of the experiments. Most use WORD for the drafting of this article but some have learned LaTeX.[19]

In *Computational Mechanics*, which is the central course in our effort to include computation into our curriculum, winter-term sophomores are introduced to com-

Year	**Term I**	**Term II**	**Term III**
1	First year studies	First year studies	elective
	elective	**Intro Classical Physics*	**Intro Modern Physics*
	Calculus I	*Calculus II*	*Calculus III*
2	**Electronics*	**Computational Mechanics*	**E & M*
	Diff Eq/Lin Alg	elective	elective
	elective	elective	elective
3	**Quantum*	*Physics elective*	**Advanced Lab*
	elective	elective	*Physics elective*
	elective	elective	elective
4	Capstone	*Physics elective*	elective
	elective	elective	elective
	elective	elective	elective

TABLE I: Typical program of a physics major at Lawrence University. Courses in italics are required for a minimum major in physics. Available physics electives include Thermal Physics, Optics, Advanced Mechanics, Advanced E & M, Mathematical Methods of Physics, Advanced Modern Physics, Plasma Physics, Solid State Physics, Laser Physics, *Computational Physics, Tutorials, Independent Studies, and Capstone Projects. Most of the physics electives are offered in alternate years.

putation in a course that devotes about 60% of the time to traditional topics in intermediate classical mechanics and about 40% of the time to computation. Because this course is required of majors, instructors in subsequent courses can confidently direct students to use computers when appropriate. Students also use the computational laboratory on their own initiative, even if individual faculty members do not make explicit assignments involving those resources. A fuller description of this course is presented in Section VII.

In *Electricity and Magnetism* spring-term sophomores experience an immediate reinforcement of some of the topics addressed in *Computational Mechanics*. Although the details and the extent of computer use depend on the instructor, students use the Computational Physics Laboratory for visualization of problem solutions done by hand, use symbolic and numerical integration to evaluate electrostatic potentials and electric and magnetic fields, solve the Laplace equation using instructor-supplied templates written in IDL, and review numerical approaches to trajectory problems.

VI. THE JUNIOR AND SENIOR YEARS

Subsequent theoretical and experimental courses offer junior and senior majors many opportunities to continue developing their computational skills and, depending on the instructor, these courses direct students to the Computational Physics Laboratory for an occasional exercise. Even in the absence of specific computational assignments, students routinely use visualization techniques on their own initiative in many upper-level courses. Most senior capstone projects exploit the resources of the Computational Physics Laboratory, at least for visualization and preparation of reports (most often with LaTeX). Some recent senior projects, notably those in fluid mechanics, musical acoustics, x-ray diffraction, multiphoton quantum transitions, theoretical explorations of the confinement of non-neutral plasmas, and simulation of planetary formation from dust clouds, have made extensive use of these facilities.

Three upper-level courses make explicit use of the computer. *Quantum Mechanics* is required of fall-term juniors. Although the extent of computer use depends on the instructor, students use the computer for visualization of wave functions and scattering coefficients, symbolic and numerical solutions of eigenvalue problems, symbolic and numerical determination of the evolution of Gaussian wave packets, and symbolic and numerical evaluation of matrix elements in the context of perturbation theory, selection rules, and the Stark effect for higher values of the principal quantum number than are usually addressed by hand.

Juniors enrolled in the required *Advanced Laboratory* find that their computational background supports their efficient learning about new computational techniques applicable to experiment. Most of the experiments exploit visualization software to examine and analyze the acquired data, LaTeX for preparing reports, and presentation software for talks. The laboratory is equipped with an assortment of measuring instruments that can output digital signals. Several experiments involve on-line acquisition of data and ask students to interface the apparatus with a computer. Linear, polynomial, and non-linear least squares fitting is required in many experiments. Occasionally, students use simulations to support the comparison of theoretical predictions with experimental results. LabView[18] is slowly finding its way into some of the experiments.

Computational Physics is an elective course that focuses on the numerical solution of the wave, diffusion, and Laplace equations and on the visualization of these solutions (see Sec. VIII).

VII. COMPUTATIONAL MECHANICS

The main objective of *Computational Mechanics*, which has *Introductory Classical Physics* and *Differential Equations and Linear Algebra* as prerequisites, is the intertwining of a conventional analytical treatment of intermediate mechanics with an introduction to symbolic and numerical computation and to visualization. Computational exercises are drawn mainly from classical mechanics, but also from classical electromagnetism and quantum mechanics. To make time for the introduction of computational topics, Lagrangian mechanics and rigid-body dynamics were moved from what had been a traditional intermediate course in classical mechanics to an already existing alternate-year, junior/senior elective course on *Advanced Mechanics*.

The course begins with a tutorial on the workstations in the Computational Physics Laboratory. In the first week students also review and extend their introductory studies of translational and rotational kinematics and dynamics, impulse, linear and angular momentum, work, kinetic energy, moment of inertia, and forces. They spend the second week in the Computational Physics Laboratory becoming acquainted with the general capabilities of IDL,[14] especially for array processing and for visualization of scalar functions of one, two, and three variables, and with TGIF[20] for generating drawings. In the third and fourth weeks, the course discusses the usual problems in one-dimensional motion via standard analytic techniques, and then extends the definition of potential energy and conservative forces to three dimensions. In the fifth and sixth weeks students are introduced to LaTeX and spend several classes on the standard analytic approaches to the central force problem.

The remainder of the ten-week term includes an orientation to MAPLE[15] for symbolic solution of ordinary differential equations (ODEs) and for evaluation of integrals. Numerical algorithms for solving ODEs and evaluating integrals are introduced, including their use in IDL. Many of the problems already discussed analytically provide examples for this computational component, and examples involving non-linear and chaotic systems are also studied.

VIII. COMPUTATIONAL PHYSICS

Computational Physics, the second computational course, was introduced in the fall of 2004 after several unsuccessful attempts to incorporate its topics as components of other upper-level courses. This junior/senior elective has *Computational Mechanics* as a prerequisite and is an alternate-year offering. It emphasizes problems involving partial differential equations (PDEs) in electromagnetic theory, fluid mechanics, heat transfer, and quantum mechanics, and gives particular attention to techniques for visualization of the solutions.

The course begins by orienting students, most of whom have not taken a formal course in programming, to the elements of programming, briefly in pseudocode and then in FORTRAN (or C) and IDL. After spending one and a half class periods deriving the standard second-order PDEs of mathematical physics (wave equation, diffusion equation, Laplace equation) and discussing the multitude of physical contexts in which these equations appear, the course introduces finite difference methods[25] and applies them to discretize the diffusion and wave equations in the spatial coordinate(s) but not the temporal coordinate, yielding a large set of ODEs to be solved simultaneously subject to appropriate initial and boundary conditions. Students review the ODE solvers studied in *Computational Mechanics* and then move to the FORTRAN solver `lsode`,[26] which gives them their first experience writing driving programs to invoke existing and well-tested subroutines. They then discretize the Laplace equation, yielding a possibly large set of algebraic equations to be solved by an iterative approach, and modify an IDL, FORTRAN, or C template to write programs to implement the standard relaxation algorithm for different boundary conditions. Students next learn about multigrid approaches and write driving programs to invoke existing subroutines `mud2sp` and `mud3sp` from the MUDPACK[27] package of FORTRAN solvers for elliptic PDEs in two and three dimensions. They then discretize the wave and diffusion equations in all coordinates, and modify templates or write their own IDL or FORTRAN (or C) programs to problems with different initial and boundary conditions. About 35% of the course is devoted to these topics.

The final topic of the course, to which about 30% of the time is devoted, addresses finite element methods[28] for solving PDEs. For simplicity, the approach is first applied to ODEs. The development of the technique focuses on a general second-order, self-adjoint, linear, inhomogeneous equation. The overall strategy is given, and templates for coding in FORTRAN, C, and IDL are given for specific equations. Students then modify those templates to adapt them to other examples. The course then spends two class periods describing how to use the commercial programs, MARC (a venerable solver of PDEs by finite element methods) and MENTAT[29] (a more contemporary GUI interface for defining problems, automeshing the geometry, creating the input file for MARC, and examining the output produced by MARC.

Beyond the weekly written assignments (which students document using LaTeX), students complete two medium-length projects, one using finite difference and the other using finite element methods. At the end of the term (35% of the course), each student completes a final, longer project of his or her choice. For each of these projects, students prepare an oral presentation (15 minutes for the first two projects; 25–30 minutes for the end-of-term project) and also a paper. Projects have involved solving Laplace's equation for electrostatic potentials or steady-state temperature distributions in various regular and irregular two-dimensional regions; finding normal modes of oscillation for two-dimensional membranes

of various shapes; solving the one-dimensional time-dependent Schrödinger equation to find reflection and transmission coefficients for a particle directed at a potential barrier; exploring the stability of finite-difference approaches to the wave and diffusion equations; and finding electrostatic potentials in various three-dimensional regions.

IX. SUMMARY AND EVALUATION

The Lawrence approach to incorporating computation in the undergraduate physics curriculum involves instruction and exposure at all levels beginning in the first year. Central to our approach is the inclusion of a focused introduction to numerical and symbolic computation and to visualization as a component of a required sophomore course in intermediate mechanics. Thereafter, students increase their computational skills, sometimes through explicit assignments in later courses, including an elective course in computational physics, and sometimes through their own personal initiative. By graduation, majors have become familiar with many important computational techniques and with a variety of tools for the implementation of those techniques. Further, they have had practice using publishing packages and presentation tools to communicate their work to a variety of audiences.

Evidence of the value of the computational components in our curriculum is largely anecdotal. Students who learn in their sophomore year to use computational resources continue to use those resources confidently and comfortably on their own in later courses and particularly in independent studies and summer research experiences (REUs) at Lawrence and elsewhere. Some of these REUs have been intensively computational, including simulation of planetary formation from dust clouds, and fluid dynamics of upwelling along the Oregon coast; others have used computation to analyze experimental data. All have used presentation tools for required oral reports. A recent student reports that his computational background was a critical factor in the positive decision on his application for a summer REU, especially because he was applying for a position following his sophomore, not his junior year. Students returning from REUs elsewhere and graduates returning from their first few years in graduate programs express gratitude for the introduction we have provided and contend that they are more adept in these areas than many of their peers, who sometimes struggle to live up to the computational expectations of their programs. One student returning from an REU reported that she was the only student in her research group who was able to use LaTeX in preparing reports, which her supervisor required all of his students (undergraduates and graduates alike) to use. Although the platform and software we use at Lawrence are not always replicated at subsequent institutions, students find that they can shift easily, because they need only learn an alternative syntax.

Acknowledgments

The curricular developments described in this paper have been supported by three grants from the National Science Foundation, three from the W. M. Keck Foundation, and one from the Research Corporation and by Lawrence University. The author's departmental colleagues, and especially John R. Brandenberger and the late J. Bruce Brackenridge, have endured many conversations and contributed many creative thoughts as the curricular components have evolved. Approximately 200 students who have survived the courses in the past several decades and two dozen students who have worked as summer research assistants to the author have been significant contributors to the entire enterprise. Finally, thanks go to three anonymous reviewers and the editors of the American Journal of Physics, whose thoughtful criticisms have resulted in a more compact and more valuable exposition.

---

* Electronic address: david.m.cook@lawrence.edu

[1] The Physics Curriculum Workshop Conference on Computers in Undergraduate Science Education sponsored by the Commission on College Physics (CCP) and held at Illinois Institute of Technology in August, 1970, was among the first national gatherings of those who recognized the potential of the new technology. A CCP publication, *Computer-Oriented Physics Problems*, edited by J. W. Robson, and published in August 1971 emerged from that conference as an early effort to provide modules for incorporation into traditional courses. Alhough now dated, pertinent publications include A. M. Bork, *FORTRAN for Physics* (Addison-Wesley, Reading, MA, 1967), H. Peckham, *Computers, BASIC, and Physics* (Addison-Wesley, Reading, MA, 1971), R. Ehrlich, *Physics and Computers* (Houghton-Mifflin, Boston, 1973), and J. Merrill, *Using Computers in Physics* (Houghton-Mifflin, Boston, 1976).

[2] See, for example, W. M. MacDonald, E. F. Redish, and J. M. Wilson, "The M.U.P.P.E.T. manifesto," Comput. Phys. **2** (4), 23–30 (1988); P. Laws, "Workshop physics: Replacing lectures with real experience," in *The Conference on Computers in Physics Instruction: Proceedings*, edited by E. F. Redish and J. Risley (Addison-Wesley, Reading, MA, 1990), pp. 22–32; M. L. DeJong, "Computers in introductory physics," Comput. Phys. **5** (1), 12–15 (1991); W. G. Harter, "Nothing going nowhere fast: Computer graphics in physics courses," Comput. Phys. **5** (5), 466–478 (1991); P. Laws, "The role of computers in introductory physics courses," Comput. Phys. **5** (5), 552–xx (1991); J. M. Wilson, "Computer software has begun to change physics education," Comput. Phys. **5** (6), 580–581 (1991); J. M. Wilson, E. F. Redish, and C. K. McDaniel, "The comprehen-

sive unified physics learning environment (CUPLE): Part I – Background and Operation," Comput. Phys. **6** (2), 202–209 (1992), and "Part II – Materials," Comput. Phys. **6** (3), 282–286 (1992); J. M. Wilson, "The CUPLE physics studio," Phys. Teach. **32** (9), 518–523 (1994); W. Christian and M. Belloni, *Physlets: Teaching Physics with Interactive Curricular Material* (Benjamin Cummings, San Francisco, 2000) and *Physlet Physics: Interactive Illustrations, Explorations, and Problems for Introductory Physics* (Benjamin Cummings, San Francisco, 2003); R. W. Chabay and B. A. Sherwood, *Matter & Interactions I: Modern Mechanics* (John Wiley & Sons, New York, 2007), 2nd ed., and *Matter & Interactions II: Electric and Magnetic Interactions*, (John Wiley & Sons, New York, 2007), 2nd ed.

[3] See, for example, the American Institute of Physics report on the 1998–99 Bachelor's Plus Five Study (search "Bachelor's Plus Five" at `<www.aip.org>`), which documents that five to eight years after graduation, about 25% of those with a physics undergraduate degree and no higher degree declare that they are employed in "software." Surely, the 30% who declare they are employed in "science and lab" or "engineering" also use computational resources to some extent. In the same study, about 45% of the graduates rate "computer programming" as "very important," and only about 37% rate "physics principles" and 33% rate "knowledge of physics" as "very important" job skills. For a review of several studies and citations to the studies, see O. Yaşar and R. H. Landau, "Elements of computational science and engineering education," SIAM Rev. **45** (4), 787–805 (2003), which is available at `<epubs.siam.org/SIREV/sirev_toc.html>`. The importance of incorporating computation in the undergraduate physics curriculum is also discussed in N. Chonacky and D. Winch, "Integrating computation into undergraduate curricula: A vision and guidelines for future development," Am. J. Phys. **76** (***), ***–*** (2008).

[4] (In this listing, when two publication dates appear, the first is the date of publication of the first edition.) See, for example, W. J. Thompson, *Computing in Applied Science* (John Wiley & Sons, New York, 1984); H. Gould, J. Tobochnik, and W. Christian, *An Introduction to Computer Simulation Methods* (Addison-Wesley, Reading, MA, 1987; 2006), 3rd ed.; M. L. DeJong, *Introduction to Computational Physics* (Addison-Wesley, Reading, MA, 1991); A. Garcia, *Numerical Methods for Physics* (Prentice-Hall, Upper Saddle River, NJ, 1994; 2000), 2nd ed.; P. L. DeVries, *A First Course in Computational Physics* (John Wiley & Sons, New York, 1994); Consortium for Upper-Level Physics Software (CUPS), edited by R. Ehrlich, W. MacDonald, and M. Dworzecka (John Wiley & Sons, New York, 1995) [This project yielded nine volumes written by 29 authors for use in standard intermediate and advanced courses on Electricity and Magnetism, Astrophysics, Quantum Mechanics, Classical Mechanics, Nuclear and Particle Physics, Waves and Optics, Thermal Physics, Modern Physics, Solid State Physics.]; N. Giordano and H. Nakanishi, *Computational Physics* (Benjamin Cummings, San Francisco, 1997; 2005), 2nd ed.; R. H. Landau and M. Páez, *Computational Physics: Problem Solving with Computers* (John Wiley & Sons, New York, 1997); R. H. Landau *et al.*, *A First Course in Scientific Computing: Symbolic, Graphic and Numeric Modeling Using Maple, Java, Mathematica and Fortran 90* (Princeton University Press, Princeton, NJ, 2005); A. Shiflet and G. Shiflet, *Introduction to Computational Science: Modeling and Simulation for the Sciences* (Princeton University Press, Princeton, NJ, 2006).

[5] See, for example, P. B. Visscher, *Fields and Electrodynamics: A Computer-Compatible Approach* (John Wiley & Sons, New York, 1988); J. Feagin, *Quantum Methods with Mathematica* (Springer-Verlag, Berlin, 1994); R. Greene, *Classical Mechanics with MAPLE* (Springer-Verlag, Berlin, 1995; 2000), 2nd ed.; J. Hasbun, *Classical Mechanics with MATLAB Applications* (Jones and Bartlett, Boston, 2008).

[6] This project is described in detail in Ref. 3, Chonacky and Winch.

[7] The specific citations in this paragraph identify representative activities. The author does not claim to have cited or to be aware of all contributors and apologizes to those omitted. A more comprehensive listing can be found in R. H. Landau. "Resource Letter CP-2: Computational physics," Am. J. Phys. **76** (***), ***–*** (2008). The September/October 2006 issue of *Computing in Science and Engineering* is on "Computation in Physics Courses." It includes the results of a national survey of uses of computers in undergraduate physics, the texts of five invited papers delivered at the Syracuse meeting of the AAPT in the summer of 2006, and abstracts of the seventeen invited posters at the same meeting.

[8] See, for example, M. Belloni and W. Christian, "Physlets for quantum mechanics," Comput. Sci. Eng. **5** (1), 90–97 (2003), and M. Belloni, W. Christian, and A. J. Cox, *Physlet Quantum Physics: An Interactive Introduction* (Benjamin Cummings, San Francisco, 2005).

[9] See, for example, D. M. Cook, "Introducing computational tools in the upper-division undergraduate physics curriculum," Comput. Phys. **4** (2), 197–201 (1990); D. M. Cook, "Computational exercises for the upper-division undergraduate physics curriculum," Comput. Phys. **4** (3), 308–313 (1990); K. R. Roos, "An incremental approach to computational physics education," Comput. Sci. Eng. **8** (5), 44–50 (2006).

[10] An early course in computational physics was described by W. J. Thompson in "Introducing computation to physics students," Comput. Phys. **2** (4), 14–20 (1988) and an unusual approach is described by R. H. Landau, H. Kowalik, and M. J. Páez in "Web-enhanced undergraduate course and book for computational physics," Comput. Phys. **12** (3), 240–247 (1998), but such courses now exist at numerous institutions. Some have been described in presentations at professional meetings (Ref. 7), but few have been described in detail in the literature. In many cases, these courses also play a role in full-blown computational majors or computational tracks (Ref. 11).

[11] Such an approach is in place at Oregon State University and Austin Peay State University. See R. H. Landau, "Computational physics for undergraduates: The CPUG degree program at Oregon State University," Comput. Sci. Eng. **6** (2), 68–75 (2004); R. H. Landau, "Computational physics: A better model for physics education," Comput. Sci. Eng. **8** (5), 22–30 (2006); and J. R. Taylor and B. A. King III, "Using computational methods to reinvigorate an undergraduate physics curriculum," Comput. Sci. Eng. **8** (5), 38–43 (2006). A survey of several such programs is incorporated in Ref. 3, O. Yaşar and R. H. Landau.

[12] See, for example, R. F. Martin Jr., G. Skadron, and R. D.

Young, "Computers, physics and the undergraduate experience," Comput. Phys. **5** (3), 302–310 (1991); D. M. Cook, "Computers in the Lawrence physics curriculum: Part I," Comput. Phys. **11** (3), 240–245 (1997), and "Part II," Comput. Phys. **11** (4), 331–335 (1997); W. Christian, "Developing a computer-rich physics curriculum at a liberal arts college," Comput. Phys. **11** (5), 436–441 (1997); D. M. Cook, "Computation in undergraduate physics: The Lawrence approach," in *Computational Science – ICCS*, edited by V. N. Alexandrov et al. (Springer Verlag, Berlin, 2001), Part 1, pp. 1074–1083; M. Johnston, "Implementing curricular change," Comput. Sci. Eng. **8** (5), 32–37 (2006); J. R. Taylor and B. A. King III, Ref. 11.

[13] An 82-page report titled "Computation in the Lawrence physics curriculum," which includes detailed syllabi for the central courses, sample assignments and examinations, and a description of the text used in these courses, is deposited at EPAPS Document No. ***. This document may be retrieved via the EPAPS homepage `<www.aip.org/pubservs/epaps.html>` or from `<ftp.aip.org>` in the directory `/epaps/`. See the EPAPS homepage for more information.

[14] For example, IDL, ITT Industries, `<www.ittvis.com>` and MATLAB, The MathWorks, `<www.mathworks.com>`. OCTAVE is available under a GNU General Public License, `<www.octave.org>`.

[15] For example, MAPLE, Waterloo Software, `<www.maplesoft.com>` and MATHEMATICA, Wolfram Research, `<www.wolfram.com>`. MAXIMA, is available under a GNU General Public License, `<maxima.sourceforge.net>`.

[16] For example, KALEIDAGRAPH, Synergy Software, `www.synergy.com`, and/or IDL, MATLAB, and OCTAVE (Ref. 14).

[17] For example, MULTISIM, Electronics Workbench Corporation, `<www.electronicsworkbench.com>`; SPICE is available at nominal cost (start at `<www.berkeley.edu>` and search for SPICE).

[18] For example, LABVIEW, National Instruments Corporation, `<www.NI.com/labview>`.

[19] For example, LaTeX is freely available for many platforms via `<www.tug.org>`.

[20] For example, TGIF, `<bourbon.usc.edu/tgif>`.

[21] D. M. Cook, *Computation and Problem Solving in Undergraduate Physics* (*CPSUP*) (Lawrence University Press, Appleton, WI, 2004); D. M. Cook, *Solutions to Selected Exercises* to accompany *CPSUP* (Lawrence University Press, Appleton, WI, 2004). Contact the author for detailed information.

[22] The *LabPro* hardware and assorted sensors connect externally to a laboratory computer and, in conjunction with the associated software, LOGGERPRO, provide facilities for on-line data acquisition. Vernier Software and Technology, `<www.vernier.com>`.

[23] Spectrum Techniques, LLC, `<www.spectrumtechniques.com>`.

[24] NanoScience Instruments, `<www.nanoscience.com>`.

[25] Finite difference methods involve overlaying a grid of uniformly spaced nodes on the domain of the problem and discretizing the PDE by using finite differences to approximate the first and second partial derivatives of the solution, for example, $(\partial u/\partial x)_{i,j} \approx [u(x_{i+1}, y_j) - u(x_{i-1}, y_j)]/(2\,\Delta x)$ for the first derivative at node (i, j). If only the space variables are discretized, the PDE is replaced with a set of coupled ODEs to be solved for the approximate temporal behavior of the solution at each node. If the time variable is also discretized (or if there is no time variable), the PDE is replaced by a set of algebraic equations for the solution at each node. See, for example, G. E. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations* (John Wiley & Sons, New York, 1960), which is available in a Dover reprint.

[26] The subroutine `lsode` (the Livermore Solver for ODEs) is a component of ODEPACK, which is a large package containing numerous FORTRAN solvers for ODEs. This package is in the public domain. See `<www.netlib.org/odepack>`.

[27] MUDPACK is a package containing numerous FORTRAN solvers for elliptic partial differential equations in two and three dimensions. This package is in the public domain. See `<www.scd.ucar.edu/css/software/mudpack>`.

[28] Finite element methods involve overlaying a network of arbitrarily positioned and not necessarily regularly spaced nodes on the domain of the problem, connecting those nodes to cover the domain with elements (lines in one dimension, triangles or quadrilaterals in two dimensions, tetrahedrons or bricks or other geometries in three dimensions), selecting an approximating function for each element, and determining the constants in those approximating functions so that (1) the difference between the approximating function and the actual solution throughout each element is minimized by one of several criteria and (2) continuity of the function and its first derivatives at the boundaries between elements is assured. The method, which replaces the PDE with a set of algebraic equations for the solution at the nodes, is more complicated than finite difference methods, but is much more easily applied to problems with irregular geometries. For more details see, for example, J. E. Akin, *Finite Element Analysis for Undergraduates* (Academic Press, London, 1986); D. S. Burnett, *Finite Element Analysis* (Addison-Wesley, Reading, MA, 1988); L. R. Ram-Mohan, S. Saigal, D. Dossa, and J. Shertzer, "The finite-element method for energy eigenvalues of quantum mechanical systems," Comput. Phys. **4**(1), 50–59 (1990), and references therein.

[29] MARC/MENTAT is a pair of programs for setting up and solving partial differential equations by finite element techniques and can be leased from MSC Software, `<www.mscsoftware.com>`. Another finite element package is FEMLAB, `<ecs.rutgers.edu/eitlab>`, which is an add-on to MATLAB.