SIAM REVIEW
Vol. 45, No. 4, pp. 787–805

Elements of Computational Science and Engineering Education*

Osman Yaşar†
Rubin H. Landau‡

Abstract. The multidisciplinary nature of computational science and engineering (CSE) and its relation to other disciplines is described. The stages through which CSE education is evolving, from initial recognition in the 1980s to present growth, are discussed. The challenges and benefits of different approaches to CSE education are discussed, as is the emergence of a set of core elements common to different approaches. The content of courses, curricula, and degrees offered in CSE are reviewed, and a survey is made of all undergraduate degree programs. The curricula of different programs are examined for the common "tool set" they define and analyzed for their relative weighting of computing, application, and mathematics. A trend toward a standard curriculum is noted.

Key words. computational science, education, bachelor's degrees, curriculum, courses, tool set

AMS subject classifications. Primary, 97U70; Secondary, 68U01

DOI. 10.1137/S0036144502408075

1. Nature of Computational Science and Engineering.

The past decade has witnessed extraordinary advances in science and engineering that were fueled, in part, by dramatic increases in the power and pervasiveness of computers and communications. We have capitalized on those advances by developing techniques for modern computers that let us better understand systems with ever-increasing complexity and realism. Examples include climate modeling [1], quark structure of elementary particles [2], engine and vehicle design [3], materials development [4], drug development, astronomy [5, 6], nonlinear dynamics and chaotic behavior [7], biodiversity, finance, and the mining of huge data sets such as the virtual humans and the digital sky observatories [8].

1.1. Definition of CSE.

We define computational science and engineering (CSE), or computational science for short, in several ways. Sometimes it denotes the multidisciplinary combination of computational techniques, tools, and knowledge needed

*Received by the editors May 17, 2002; accepted for publication (in revised form) May 7, 2003; published electronically October 31, 2003.
http://www.siam.org/journals/sirev/45-4/40807.html

†Department of Computational Science, State University of New York College at Brockport, Brockport, NY 14420 (oyasar@brockport.edu, http://www.cps.brockport.edu). This author was supported in part by National Science Foundation grants 998094 (National Partnership for Advanced Computational Infrastructure-EOT) and 0226962 (Math and Science Partnership-MSP).

‡Computational Physics Undergraduate Program, Oregon State University, Corvallis, OR 97331 (rubin@physics.orst.edu, http://www.physics.orst.edu/CPUG). This author was supported in part by National Science Foundation grant 998094 (Course, Curriculum and Laboratory Improvement) and the National Partnership for Advanced Computational Infrastructure-EOT.

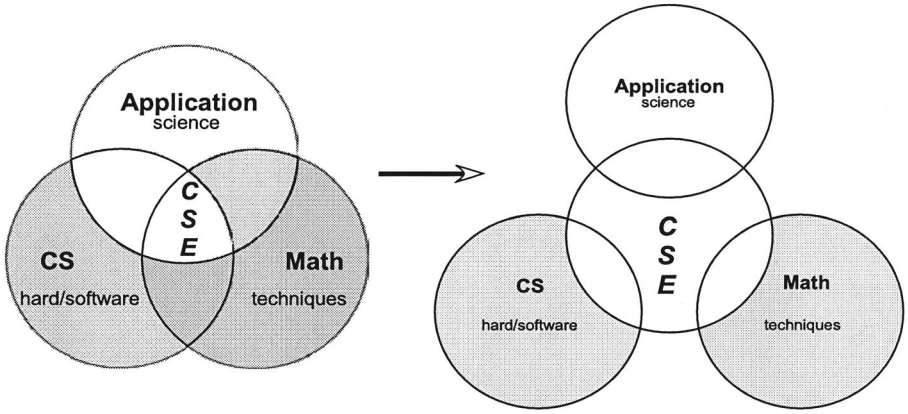


Fig. 1.1 *CSE as a multidisciplinary endeavor connecting computation with mathematics and science. Left: Early view in which CSE is merely the overlap of other disciplines. Right: Current view in which CSE shares common concerns with these disciplines, as well as having content of its own.*

to solve modern scientific and engineering problems. At other times CSE denotes science or engineering that uses computer simulations as its basis, and sometimes it denotes the research and development of computational skills and tools needed for applications.

The original view of CSE, shown on the left side of Figure 1.1, was as the intersection of applied mathematics, computer science, and application sciences. This early view is now being replaced by the one represented by the right side of Figure 1.1 [9, 10, 11], in which an additional, central circle reflects the recognition that, in addition to having common concerns with other disciplines, CSE also contains core elements of its own that draw together and bridge the original three disciplines. (This new view was first communicated in a 2000 article by Yaşar [10] in the *IEEE Journal of Computing in Science and Engineering*; Yaşar later presented it at the SIAM CSE Conference [12].) The core of CSE may be thought of as its collection of computational tools and methods and its problem-solving mindset, which uses knowledge in one discipline to solve problems in another. This CSE core is now being incorporated into computational science courses and books that combine scientific problem solving with computation [13, 14, 15, 16, 17, 18], and into curricula at various levels of education.

1.2. Value of CSE in Research. Most scientific disciplines appear to be benefiting from computer modeling, analysis, and visualization. In fact, the newfound and widespread importance of computation has shifted the paradigm of scientific research to include simulation, along with experiment and theory, as a fundamental technique of science [19]. Simulation and visualization allow us to acquire insights into real-life problems that are too complex or difficult to study analytically, or too expensive, big, small, or dangerous to access experimentally. For example, simulations permit us to study the fuel density, ignition energy, and heat waves in a combustion chamber at temperatures above 3000 Kelvin [3]; the configuration of gluon flux tubes between quarks within the proton [2]; and the possible orbits for earth–asteroid collisions [6].

As more scientists incorporate computation into their work, the value of CSE to the scientific community increases. In particular, CSE working groups permit other scientists to pursue their interests in science without having to spend time developing algorithms and codes. These working groups have focused on identifying, studying, and improving computational methods and software components that are common to many applications, and on testing the performance of the related software on various hardware platforms.

One accomplishment of CSE working groups is the development of mathematical libraries of subroutines for linear algebra, special functions, and other mathematical techniques. These subroutines, whose use is emphasized in the various CSE education programs, form the core of many large-scale computer calculations. Accordingly, it is in the interests of the CSE community to look after the accuracy, performance, robustness, portability, and scalability of these libraries as they are ported to different high-performance computing architectures. These accomplishments have aided the progress made with supercomputers and, with the phenomenal increases in desktop computing power, are now finding wider use.

In addition to the mathematical subroutine libraries, CSE also develops and promotes simulation techniques that have widespread applicability. Consider, for example, the computational technique known as "particle dynamics" in which individual particles are created and their paths are followed in time using the appropriate equations of motion [20]. This technique is used to simulate processes as diverse as the settling of sand particles in a tank, the creation of stars and planets, the motion of molecules in gases and liquids, the growth of thin films, the dynamics of droplets in engines, and the motion of ash particles in industrial burners. Clearly, improving the technique, or educating its practitioners on how to use it better, benefits many areas of science.

The growing volume of CSE techniques and associated software libraries represents a priceless wealth of tools and knowledge. There is already a heavy reliance upon them in the CSE community, and we expect that reliance to grow in the future. For these reasons, we need education in CSE to pass this wealth on to future generations of scientists and engineers.

1.3. Nature of the Job Market. The President's Information Technology Advisory Committee (PITAC) [21] has recognized information and computational technology as one of the engines of economic growth during the last decade. The committee indicates a need for approximately one million people in information and computational technology, a need that cannot be met solely by all computer science departments working at full capacity. The National Science and Technology Council has repeatedly reported the concerns of industry and the national laboratories that they cannot satisfy their growing needs for people trained in information and computational technology. In addition, federal projects such as the Department of Energy's Accelerated Strategic Computing Initiative (ASCI) and the Information Technology Presidential Initiative rely on people with scientific as well as computing knowledge. These needs for more people with a computational education are a consequence of the exponential growth in the power of computers occurring in this decade simultaneously with an extraordinary decrease in the cost of computers. Few other things in life change by factors of 1000 in such a short time. Children now play games on computers of a class that 10 years ago were considered supercomputers and were available only at government laboratories.

Just as computers now permeate many aspects of our daily lives both at work and at home, they also profoundly affect the technical job market. One such way is by placing people in positions that require knowledge in areas beyond their education and (obsolete) job descriptions. Consequently, having multiple skills and majors is viewed as a way of improving one's marketability and employment survival time. Employers seem to prefer people with education in multiple disciplines since they can then hire fewer of them and can retain them for longer periods of time. Yet, attaining multiple degrees is expensive and time-consuming, as is majoring in one department and minoring in several others. In contrast, a multidisciplinary CSE educational program saves time and money for those students who might otherwise follow multiple courses of study. It also provides a coherent and consistent education with minimum duplication and equips students with an interdisciplinary *tool set* (to be described soon) that experience has shown to be useful in a wide range of fields.

2. Nature of CSE Education. Due in part to reports and grants from the federal government, education in science and engineering has responded to the advances in computational science. In 1989, the Office of Science and Technology Policy challenged the educational system to (1) increase the supply of students prepared for careers in science, technology, engineering, and mathematics; and (2) improve the scientific, mathematical, technological, and computational literacy of all students. Recently, the National Science Foundation (NSF) has recognized and advocated computation as an appropriate pedagogy for science and mathematics education [22]. Wright and Chorin's [23] NSF report urged the creation of mathematical modeling courses (the basic approach of CSE) even for high school students.

The traditional teaching of science tends to focus on theory. In contrast, CSE education offers an understanding of science through the computer applications of mathematical models. It teaches science via the method of inquiry in which the computer serves as a virtual laboratory that simulates nature. To aid in the inquiry, facts are presented as needed rather than as individual objects to memorize [24, 25, 26, 27, 28]. As a consequence of its problem-solving nature, the CSE view complements the traditional teaching of science and mathematics. It also makes many science and mathematics concepts more easily accessible to students who may otherwise not be reached; for example, those students not interested in computer hardware, software, and algorithms for their own sake [29]. In addition, CSE enriches the science curriculum by extending the examples used in education to include problems that may not have analytic solutions, thereby extending the range of problems open to study.

The National Foundation for Improving Education has examined the types of learning supported by CSE [30]. These include the learner-centered or constructivist view which suggests that students learn better when they are actively interacting with, rather than just receiving, knowledge. The CSE approach is often both project- and team-based, as well as learner-based and supportive of authentic learning. When successful, this combination of elements can transform uninvolved, at-risk students into active and invested learners.

A CSE education is also well suited to a deductive pedagogy, that is, one that works from the general to the specific. Students start with an awareness that nature and its processes are governed by a small number of basic scientific laws; details and mathematical analyses are added as needed. In cases where the students' mathematics skills are limited, the simulations can be understood via visualizations without delving into mathematical and scientific details. Accordingly, CSE education fosters a view that natural phenomena are basically simple, in contrast to the common student

perception of science as complex. It also provides a basic framework upon which students can build as their skills increase. This ability to stimulate the curiosity of students with interesting and realistic examples and to provide layered learning is a principal motivation for educators to apply and master technology tools.

2.1. Stages of CSE Education. Education in CSE has been evolving in stages. Originally, scientists and engineers taught themselves the computational techniques they needed in the course of solving problems or received them as hand-me-downs from their advisors. The first formal stage, ca. 1980–1990, was *recognition-conception*, in which practitioners recognized that they were doing something new—but not necessarily well. This stage was epitomized by the 1982 Lax report [19], with its early recognition of the paradigm shift in which computation was joining experiment and theory as a basic technique of science. The report recommended that funding agencies provide increased education in scientific computing in order to facilitate the shift.

The second stage of CSE education, ca. 1990–2000, was *infancy* [31, 32, 33], in which the ideas of CSE started to be taught in a few existing or new courses, often by those who were familiar with the ideas through their research. The third stage, ca. 2000–2010, is *early growth* [10, 34]. It is the stage we are in now and is characterized by a number of courses and curricula being designed and implemented at both the graduate and undergraduate levels.

While we expect the future *adult stage* of CSE education to resemble what we have now, it is hard to make reliable predictions in the midst of rapid change. Nevertheless, we would predict that the number of CSE programs and courses will continue to increase either as a deliberate thrust into computation or indirectly as science, technology, engineering, and mathematics disciplines hire faculty with CSE specialties.

We foresee a growing number of courses in "computational X," where X is an established discipline such as mathematics, biology, finance, physics, chemistry, or social science. The pressure for these courses may come from multiple sources. The national laboratories and industries have regularly indicated their continuing needs for scientists and mathematicians who can compute [21, 35]. In addition, there are undergraduate students in existing CSE programs who are interested in applying their newfound skills to discipline X. There are also faculty members in the X departments who want their research students to know how to compute, or who want to modernize the courses they teach. And, finally, there are the CSE educators (the present authors included) who believe as a matter of principle that computational courses are needed to serve society.

It might be that a single "computational X" program will make it easier to start a "computational Y" program sharing some of the same courses. After a number of computational programs are established at any one school, they may choose to organize themselves under a CSE unit for the sake of efficiency and communication. These units may appear as interdisciplinary programs within discipline X, or as a stand-alone department of CSE. Such stand-alone departments may have a core faculty of their own and a number of associated faculty from computer science, mathematics, and other disciplines.

The thrust for new CSE units may also arise from the funding agencies, who are beginning to support CSE more directly. Likewise, there are also internal thrusts from colleges and universities as they follow the movement afoot to restructure themselves into multidisciplinary units that emphasize problem solving in a societal sense [36, 37]. It is logical that these units will benefit from an association with CSE and its scientific problem-solving orientation.

A stand-alone CSE unit can also offer a CSE minor that would benefit other departments. A minor in CSE should include more of the core knowledge in computational tools and methods than is normally expected of a science or engineering student, but not the full spectrum of subjects expected of a CSE major.

2.2. Demands of CSE Education. Because CSE has developed around the need to incorporate computation into modern scientific problem solving, it is a research-driven field that connects faculty interested in computation to those interested in applications (the major disciplines symbolized in Figure 1.1). Faculty interested in the performance of computer hardware and software are to be found in both CSE and computer science; faculty interested in the performance of numerical procedures are to be found in both CSE and applied mathematics; faculty interested in finding a computer-based solution to extend theoretical and experimental efforts are common to both CSE and application areas such as physics, chemistry, biology, earth sciences, business, and finance.

Faculty tend to be attracted to CSE because it provides a means for them to incorporate their research tools and developments into modern courses. It also saves faculty time by bringing students to a level where they can assist faculty in their research. The students, in turn, learn the tools and techniques needed by modern professionals and obtain valuable job skills.

Although integrating your research tools into the classes you teach sounds ideal, it must be noted that the instructional, research, and administrative demands of a CSE program can be a heavy load for faculty as well as for students. Since new tools are constantly arriving in industrial and scientific work environments, we cannot maintain a static curriculum in a field that teaches the use of technology. Likewise, the development of new courses, especially ones in which there is little in the way of established texts, while being exciting, is also challenging and time-consuming.

In addition to curriculum development, CSE programs often house computational laboratories with high-end PCs, workstations, supercomputers, or Beowulf clusters. Maintaining these facilities with ever-aging hardware and software can be a major chore. While a dedicated system administrator can be of great help, as can the resources of a supercomputer center, the direction, management, and continued budgeting of a computational laboratory often falls on the shoulders of a faculty member.

Other demands on CSE faculty arise from students' (quite legitimate) expectation that faculty members are knowledgable in all the CSEs courses offered, even those taught by experts from other departments. Related to this are concerns about promotion and tenure decisions when faculty (1) hold joint appointments in several departments, (2) have extra demands on their time, or (3) are undertaking multidisciplinary research or development that is hard to judge by the norms of a traditional discipline.

As may be true for all multidisciplinary education, a CSE student may face more demands than a student with only one specialization. To start, CSE students need classes, assistance, and advice in more than one field. In addition, students may feel the extra pressure of having to master three fields, while their single-discipline peers need master only one. Furthermore, there is often stress involved in having to deal with an untested and rapidly changing curriculum, especially when there is no one department to call "home."

Yet interdisciplinary education can also open up new and rewarding worlds for students. If their intellectual interests are in both science and computing, then they are given the freedom to follow these interests. This should appeal to students who

would otherwise have been willing to face the rigors of multiple majors and degrees and to those students who realize that this is the modern approach to problem solving.

2.3. Intellectual Content of CSE Education. While there is widespread recognition that computational science is not the same as computer science, at present there is no nationally accredited CSE curriculum. The first CSE courses were taught at research institutions that recognized their growing dependency on advanced computation as a research tool. Some of these courses were adjuncts to the regular curriculum taught at local supercomputer centers and tended more toward training for the latest machine than basic education. Other courses were taught within departments as part of graduate-student research preparation, and some led to a degree containing the suffix "with computing" (the "Michigan Model" [38]). Undergraduate classes soon followed, and today there are even K–12 classes with CSE content [35, 39, 40].

Particularly at the undergraduate level, some understanding of the intellectual content of a CSE education is needed in order to appreciate the different approaches to that education. Yet there are very few undergraduate programs in computational science to generalize about, and, accordingly, the specific subjects needed to provide this intellectual content are still under active discussion (to wit, this paper). Consequently, the description to follow draws from published survey results, our own surveys, and our experiences in developing two independent degree programs.

We start with a survey on the general types of skills that appear to be needed in the workplace. The data in Figure 2.1 were collected by the American Institute of Physics [41]. They indicate, for physics majors five years after graduation, which aspects of their education are most valuable in their current employment. We see that for graduates whose primary field of employment is engineering, mathematics, and science, the three most important skills are scientific problem solving, synthesizing information, and mathematical skills. These skills remain highly important for graduates who find employment related to software, with the latter group also having a high need for computer programming and software development. The importance of mathematics and computer skills was also examined in a National Science Board report [42]. It indicated that 74% of mathematics and computer-science doctorates work in the same field as their degree, in contrast to only 52% of degree holders in life and physical sciences. A similar trend is seen at the bachelor's level (35% versus 22%).

These surveys document the demand for mathematics, computer, and problem-solving skills in the workplace as well as the importance of a science education that can be applied to a variety of jobs. It appears manifest that the integration of mathematics and computer skills with a science and engineering education is a valuable investment, particularly for undergraduates, who, as a group, are more likely than graduate students to find employment outside their discipline. Deciding upon the right balance of mathematics, computing, and sciences is a question we shall discuss soon.

The determination of student learning outcomes (SLOs) is an important prerequisite for the establishment of CSE programs. This is especially true for CSE, where it is (all too) easy to expect an individual course to teach students everything they need to know about the subject. Historically, SLOs have been guided by research needs, that is, in a "top-down" fashion. We are aware of only one published work [43] that suggests a set of SLOs for all levels of CSE education. From a student perspective, typical SLOs for a CSE education might include the following:

- learning high-level computer languages and high-performance computing;
- obtaining knowledge of applied mathematics and computational methods;

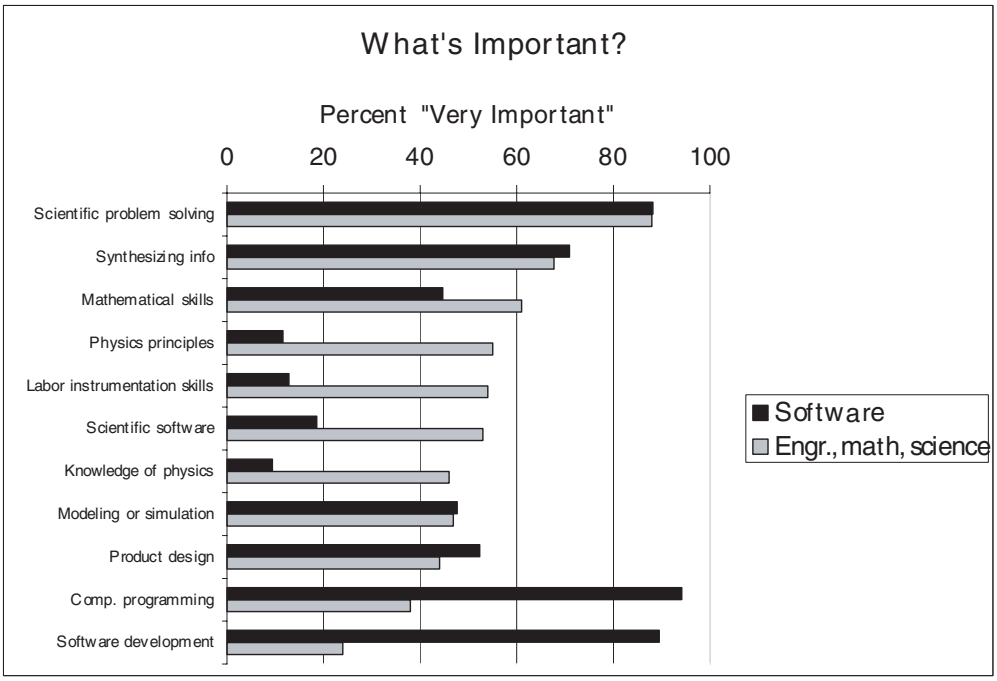


Fig. 2.1 *Importance of knowledge and skills for graduates, 5–7 years after receiving bachelor's degree. The categories are listed in order of importance for physics majors whose primary field of employment is engineering, mathematics, and science (light bars); the dark bars show level of importance for graduates employed in software. (Data courtesy of the American Institute of Physics.)*

- learning the basics of simulation and modeling;
- interpreting and analyzing data visually, both during and after computation;
- applying acquired computing skills to at least one application area;
- learning to communicate solution methods and results effectively.

These learning outcomes are achieved by students studying various subjects and working on explicit computational problems. Because courses in different departments may well cover similar materials, rather than list individual courses, we give instead the six general areas of knowledge (details to follow) that are needed to achieve the SLOs:

- computational tools;
- high-performance computing;
- applied mathematics and computational methods;
- simulation and modeling;
- visualization tools;
- applications in science or engineering.

Although not all of equal weight, these components are all essential and should be covered either in dedicated courses or within the context of other courses.

1. *Computational Tools.* Inasmuch as the computer itself is the main tool of CSE, it is essential to provide environments and situations in which students become comfortable using computers. Typical courses in which this occurs are often called "An Introduction to" or "Fundamentals in" one of the following: computational science,

modeling, problem solving, computational tools, or computer science. Additional knowledge can be gained through courses such as data structures, advanced software tools, and numerical linear algebra. Specific learning outcomes include

- programming in a compiled language such as Fortran90, C, C++, or Java;
- ability to work with UNIX and WINDOWS operating systems;
- familiarity with problem-solving environments such as MAPLE, MATLAB, MACSYMA, and *Mathematica* for both numeric and symbolic computations;
- familiarity with floating-point computations and numerical methods such as integration, differentiation, solutions of ordinary and partial differential equations, and Monte Carlo techniques;
- use of mathematical subroutine libraries and repositories such as BLAS, ScaLAPACK, NetSolve, and JAMA;
- use of two- and three-dimensional visualization software packages such as AVS, gnuplot, ACE/gr (Xmgr), and VisAD.

2. *High-Performance Computing.* Knowledge of high-performance computer hardware and software is important for programming of computationally intensive applications. Topics include parallel computing, high-level languages, and optimizing and tuning techniques. The related theoretical knowledge would come from courses in computer architecture and the theory of programming languages. Specific learning outcomes include

- programming on supercomputers or PC clusters;
- understanding of program speed, memory hierarchy, performance benchmarks, and precision;
- understanding the relationship among architecture, language, and performance;
- experience with industrial benchmarks such as the Linpack Benchmark;
- familiarity with parallel libraries such as PVM, MPI, and MPJ.

3. *Applied Mathematics and Computational Methods.* Basic knowledge here derives from courses in differential, integral, and vector calculus, preferably with a focus on applications and examples. Numerical analysis and differential equations may provide more focused knowledge, as might courses such as applied and computational mathematics, computational methods in physical sciences, and computational physics. Finally, at least one course in probability and statistics is needed in which actual laboratory data are fit. Specific learning outcomes include

- knowledge of computational methods for the numerical solution of differential and integral equations, such as finite-difference, finite-element, discrete particle, discrete ordinates, mesh generation, adaptive meshes, high-order ODE solvers such as Runge–Kutta, fast Fourier transforms, and Monte Carlo methods;
- familiarity with partial differential equations encountered in science and engineering;
- use of scientific subroutine libraries;
- matrix computations with scientific subroutine libraries.

4. *Simulation and Modeling.* This subject is important in order to achieve CSE's goal of realistic problem solving. The skills may be taught in a variety of courses such as simulation and modeling, dynamical systems, problem solving, and "computational X." Specific learning outcomes include

- Familiarity with exact and approximate equations of motion for natural systems. Examples include the continuity equation, equations for momentum,

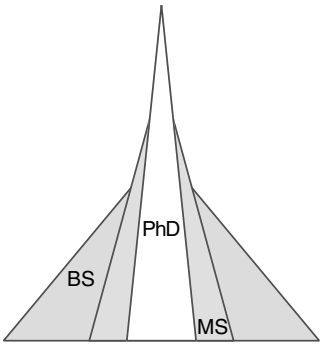


Fig. 2.2 *Perspectives on depth and breadth of computational science education.*

energy, and mass conservation, transport equations, population dynamics, protein folding, Newton's laws, and the Schrödinger equation.
- Familiarity with the steps (problem, theory, model, implementation, assessment) in modeling as applied to a wide range of fields. Examples include engine combustion, electromagnetics, molecular dynamics, genetics, drug design, quantum mechanics, climate modeling, dynamical systems, and finance.
- Ability to carry out statistically meaningful analysis of data from both experiments and simulations.

5. *Visualization Tools.* Familiarity with visualization software for run-time and postsimulation data. Tools should be able to handle large data sets, permit two- and three-dimensional plots, as well as slicing and dicing for higher dimensional data sets, and produce publication-quality figures. The level should be that of AVS, MATLAB, ACE/gr, gnuplot, and OpenGL. Courses to teach these skills include scientific computing, visualization, and computational tools.

6. *Applications in Science or Engineering.* An essential part of CSE education is having students focus on a traditional field of interest such as chemistry, physics, biology, earth sciences, business, criminal justice, art, or engineering. This should help the students in the job market, in applying to graduate school, and in understanding different and realistic ways of looking at issues. Just which field and which courses can accomplish this goal best is probably a function of local politics, expertise, and student interest. Often a "computational X" course serves this purpose well, as it provides exposure to applications while introducing students to the computational aspects of discipline X.

2.4. Types of Programs and Degrees. We now address the (still unsettled) question of transforming the intellectual content of CSE into a curriculum of courses. Because CSE is a new and different field that may draw students away from traditional departments, the academy is naturally slow to certify any CSE program. In addition, CSE is multidisciplinary and there are no set rules that give the proper balance and depth of the four components in Figure 1.1. Finally, there remains the decision as to whether CSE should be taught at the graduate or undergraduate level, and how the content would differ for the two.

Graduate Degree. Figure 2.2 represents the depth and breadth of graduate and undergraduate programs in CSE. Although it may appear that the depth needed for a Ph.D. program makes it formidable to set up, it is probably less of a challenge

than an undergraduate program. This is due to the dual constraints of breadth and limited credit load that are typically imposed on undergraduate degrees. Doctoral CSE programs tend to be of top-down design, with the top-level research needs dictating the courses at the bottom. If there are faculty members carrying on research with high-level computation, then their needs and knowledge tend to determine the elements of a Ph.D. program incorporating CSE at their institution. If there is a supercomputer center on campus, then it tends to naturally provide the computational expertise and resources. While it is important that individual departments offer relevant and modern courses, doctoral students have a great deal of flexibility in meeting the requirements set by their program committees since they do not have a large number of required classes to take in a fixed time period.

A doctoral-level CSE education is also easier to establish than an undergraduate one by nature of the expectation that doctoral students will assume the initiative to learn requisite materials on their own. Typically, much of a CSE doctoral education is assumed to take place when a student performs the simulations that are part of the research for a computationally intensive thesis. If done properly, students should learn about performance of hardware and software, development and optimization of numerical and computational methods, and advanced visualization skills. Yet even a solid design for doctoral education does not guarantee quality. Without adequate background in all the elements of CSE, the thesis will suffer and students may end up with holes in their understanding. We recommend multidisciplinary course work, as well as a graduate committee with members from computer science, mathematics, and application sciences.

Master's Degree. It is hard to generalize about master's degrees. Usually graduate-level courses are required, and often a research project or thesis as well. However, many of the graduate-level classes may cover the same materials as the upper division undergraduate classes in CSE, but perhaps with deeper applications. Accordingly, it may well be that a master's degree in CSE that does not require research or internship provides a similar education to that of an undergraduate degree. If research is required, then a master's program can be quite intensive. In fact, some combinations of programs and advisors may require research that is nearly as significant as the doctoral research of other programs.

Undergraduate Degree. The major obstacle with undergraduate CSE education is that the programs which faculty consider well-rounded combinations of disciplines tend to be so overloaded with classes that they are nearly impossible to complete in four years. With already-full curricula, it is hard to find places for challenging CSE classes, especially with university pressure to broaden the number of core courses, to cap and even decrease the number of credits needed for graduation, and to increase retention (presumably by lowering demands).

Although details vary with local conditions, we conclude that it is best for the undergraduate curriculum to focus on a common *tool set* of subjects that have proven themselves useful in a number of specialty areas. In order to avoid overloading and to provide the needed breadth, we recommend that a B.S. or an M.S. program in CSE have fairly strict curriculum requirements. This takes the place of a high-quality thesis in a Ph.D. program and permits the marketplace and the graduate schools to expect a common preparation.

In spite of the challenges, several schools have experimented with undergraduate CSE programs, and these now provide some field-tested approaches. In Table 2.1 we present a sample curriculum for the B.S. degree in the Computational Science Department at SUNY Brockport [12, 33, 43] This program was started in 1998 by

Table 2.1 *Sample curriculum for B.S. in computational science at SUNY Brockport.*

	Fall	Spring
1	Calculus I Application Sciences I Intro to Computational Sci 2 Electives	Calculus II Application Sciences II Intro to Computer Science 2 Electives
2	Fund. Comp. Sc. I Computational Tools I Discrete Mathematics I 2 Electives	Elementary Statistics Computational Tools II Calculus III 2 Electives
3	Linear Algebra High-Performance Comp Application Sciences III 2 Electives	Simulation and Modeling 3 Electives
4	Applied & Comp. Mathematics Topics in Computational Sci 2 Electives	Scientific Visualization Application Sciences III 2 Electives

Table 2.2 *Sample yearly schedule for B.S. in computational physics (CP) at Oregon State University (*180 *quarter credits, equivalent to* 120 *semester credits). CS refers to computational science.*

	Fall	Winter	Spring
1	Differential Calculus Fitness General Chemistry CP/CS Seminar Perspective	Sci Computing I General Chemistry Integral Calculus Perspective	Computer Sci I Vector Calculus I General Physics Writing I
2	Computer Science II Writing II Vector Calculus II General Physics	Discrete Math Series & Sequences General Physics Perspective	Sci Computing II Linear Algebra App Diff Eqs Modern Physics
3	CP Simulations I CP Seminar Intro Probability Oscillations Static Vector Fields Writing III	CP Simulations II Data Structures Waves in 1D Quantum Mech Central Forces Elective	Periodic Systems Class Mech Energy & Entropy Biology Perspective
4	Num Linear Algbr Electromagnetism Math Methods Phys Electives	Adv CP Lab Soc & Ethic CS Electives Synthesis	Thesis Adv Web Authoring CP Seminar Electives

one of the present authors, Osman Yaşar, and we believe it is the first stand-alone program to have actually issued a B.S. degree in CSE. In Table 2.2 we present a sample curriculum for the B.S. degree in computational physics (the "Perspective" courses in the table integrate fundamental knowledge from science and liberal arts disciplines to develop cultural, historic, and scientific perspectives). This *CPUG* (Computational Physics for Undergraduates) program was started by one of the present authors, Rubin Landau, in 2001 within the Physics Department at Oregon State University [44, 45], and granted its first degree in 2003 (to a transfer student).

3. Survey Results. To develop some quantitative understanding of the intellectual content of CSE programs, we have analyzed the curricula of the two programs detailed in Tables 2.1 and 2.2 as well as the handful of other CSE programs of which we are aware. Our goal is to provide materials for informed and critical discussions on the nature of CSE education. The choices of programs were guided by information we obtained at conferences, the latest version of Swanson's survey of CSE programs [34], a recent computational science Web portal at SUNY Brockport [46], and our own Web searches. For a survey of *graduate* degree programs, we refer the reader to Swanson's survey [34], the SIAM Working Group on CSE Education [11], as well as the SIAM listing of graduate CSE programs [47].

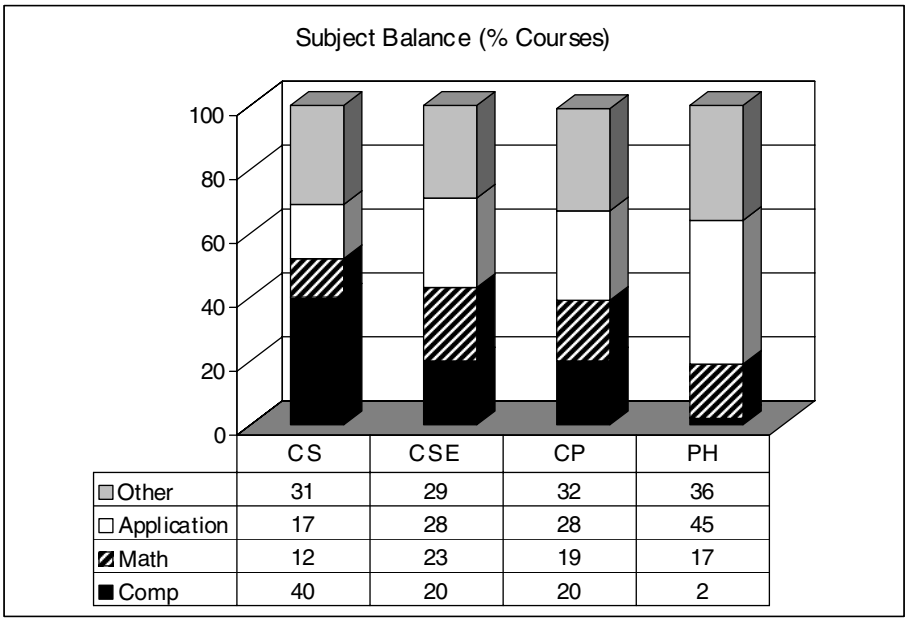
Swanson identifies one B.S. program in computational science, three B.S. programs in computational mathematics, three B.S. programs in computational physics, and several concentrations and minors. The SIAM survey [11] identifies only Brockport as a CSE degree program. We included in our analysis all of Swanson's programs except for the University of Chicago, for which no sample curriculum could be found. Although it did not appear in any Web searches, we include here a degree program at UC Berkeley whose existence was communicated to us personally. Our analysis is comprehensive in its coverage of computational degrees, and representative of computer science and physics.

The Swanson survey is particularly helpful in identifying where programs exist and in providing detailed contents of different computational classes (similar to our earlier description of a standard tool set). The analysis to follow is more general. For each program we look at four years' worth of classes and count the number of courses that fall into the categories of *computing, mathematics, applications*, and *other*. Admittedly, our analysis is crude. We examined neither the materials actually covered in the individual classes nor the amount of "other" that may actually be computing, mathematics, or application. However, we have tried to apply the same rules to all programs (such as including chemistry in the applications category) and have used posted sample schedules as our guide. As we shall see, even though the analysis is crude, there appear to be significant trends that stand out.

As interesting as it may be to see how other countries put together their CSE programs, we have restricted our survey to the United States. This is a consequence of our difficulty in deciding how to count classes and requirements for foreign systems and in extracting up-to-date data from their Web pages. We know there are high-quality CSE programs at schools such as the Australian National University [48], Kanazawa University (Japan) [49], Trinity College, Dublin [50], University of Singapore [51], and the University of Waterloo [52], and we refer the reader to Yaşar's Web portal [46] for links to both national and international CSE programs.

Figure 3.1 shows the average percent of the total curriculum dedicated to computing, mathematics, application, and other for B.S. degree programs in computer science, computational science and engineering, computational physics, and physics. The computer science average is from Carnegie-Mellon University [53], the University of Illinois at Urbana-Champaign [54], and Oregon State University [55]; the CSE average is from SUNY Brockport [56] and the University of California at Berkeley [57]; the computational physics average is from SUNY Buffalo [58], Illinois State University [59], and Oregon State University [44]; the physics average is from the University of Illinois at Urbana-Champaign [60] and Oregon State University [61].

In general we would say that the left column in Figure 3.1 shows the strong computing (black) but weak application (white) components in the computer science degree; the right column shows the strong application but weak computing components



	CS	CSE	CP	PH
Other	31	29	32	36
Application	17	28	28	45
Math	12	23	19	17
Comp	40	20	20	2

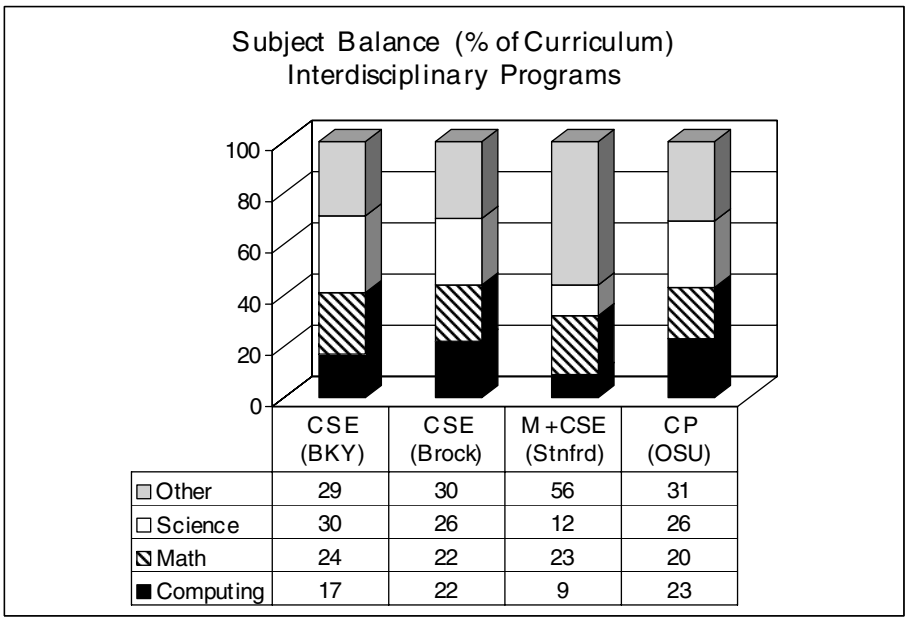
Fig. 3.1 *The average percent of the total curriculum dedicated to computing, mathematics, application, and "other" for B.S. degree programs in (from left to right) computer science, computational science and engineering, computational physics, and physics.*

in the physics degree. The two versions of computational programs in the middle of Figure 3.1 are seen to provide a similar, uniform balance among the components.

In contrast to our analysis, the IEEE Computer Society [62] tells a student interested in a "career in computing" to expect a curriculum in which "35% of your career will be in your major field of science or engineering, 25% in mathematics or science (outside your major field), 25% in arts and humanities, and about 15% in electives." Our analysis indicates that an application-oriented computing education is somewhat stronger in mathematics and has a stronger computing content than the IEEE indicates.

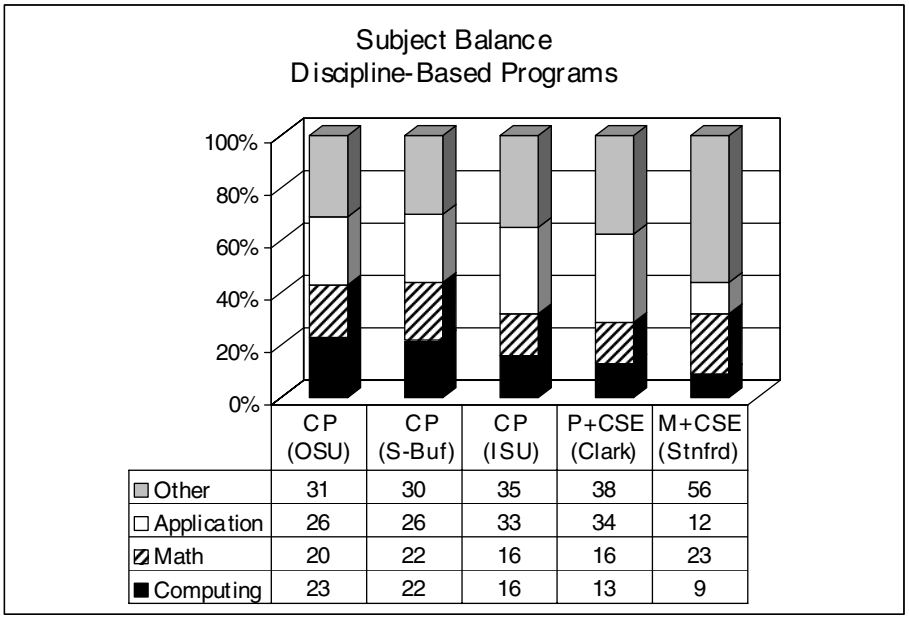
While an undergraduate CSE student may specialize by selecting options in a specific discipline, doing so makes the curriculum similar to one in "computational X" (typically offered within the department of X). Indeed, as indicated in Figure 3.1, an undergraduate degree in computational physics has a similar balance to one in CSE, namely, approximately equal weights for mathematics and computing ($\sim 20\%$), and a higher weight for application ($\sim 28\%$). However, a computational physics degree does contain less physics than a physics degree and less computing than a computer science degree.

Of course, because computational science is still a young and developing field with no set curriculum, some variation in individual programs from these averages is to be expected. Accordingly, in Figures 3.2 and 3.3 we show the subject balance for several of the individual programs that were averaged into Figure 3.1. We see in Figure 3.2 that the CSE programs at Berkeley and Brockport have a similar balance, which is also similar to that in computational physics program at Oregon State University. In contrast, the computational mathematics program at Stanford appears much weaker in computing and application, but since it has a significantly larger fraction of other,



	CSE (BKY)	CSE (Brock)	M+CSE (Stnfrd)	CP (OSU)
Other	29	30	56	31
Science	30	26	12	26
Math	24	22	23	20
Computing	17	22	9	23

Fig. 3.2 *Subject balance for individual computational science and engineering B.S. degree programs at (from left to right) UC Berkeley and SUNY Brockport, and the mathematics plus computational science program at Stanford. The computational physics program at Oregon State is shown for comparison.*



	CP (OSU)	CP (S-Buf)	CP (ISU)	P+CSE (Clark)	M+CSE (Stnfrd)
Other	31	30	35	38	56
Application	26	26	33	34	12
Math	20	22	16	16	23
Computing	23	22	16	13	9

Fig. 3.3 *Subject balance for individual computational physics degree programs at (from left to right) Oregon State, SUNY Buffalo, Illinois State, and Clark (physics plus computational science concentration). The mathematics plus computational science at Stanford is shown for comparison.*

it may well be that a similar balance is obtained after students choose their large number of electives.

We see in Figure 3.3 that the computational physics programs at Buffalo and Oregon State have essentially identical balances, while the programs at Illinois State and Clark emphasize application and other more and computing less. (The nearly exact agreement is probably accidental since we estimate a 1–2% ambiguity in our analysis arising from the choice of options and courses listed in multiple departments.) No doubt some of this uniformity arises from what might be called a "sum rule": Because all programs are designed to have students graduate in four years with a similar set of university-wide requirements, the computational degree programs tend to replace some of what would otherwise be electives with computation and mathematics classes, and thus end up with a similar distribution. Future experience may establish a better balance than we have now, although we suspect that there is no one global solution.

While we have focused here on stand-alone programs, there are other approaches that can supply a quality education. Surveys of programs described on the Web [34, 56] reveal cases in which a B.S. degree in "computational X" is close to a CSE degree with a minor in X, and others in which it is close to a degree in X with a minor in CSE or computer science. So, even though the names may differ, the contents of different programs may be much the same.

4. Summary and Conclusions. Although details vary, CSE education tends to focus on a common tool set of subjects that have proven themselves useful in solving problems in a number of disciplines. While many of these subjects may get coverage in courses taught by traditional departments, we have described the need for separate CSE classes that (1) put the tools together, (2) develop an appropriate problem-solving viewpoint, (3) glue the multiple disciplinary classes together, and (4) develop a sense of belonging to a computational community.

Local politics and course offerings appear to determine how a school chooses to integrate CSE into its programs. We conclude that the CSE content of undergraduate education will continue to grow, either by integrating CSE into traditional classes, by offering specific "computational X" classes, by offering "computational X" degrees, by starting stand-alone CSE units, or by some combination of the above.

No doubt, a specific degree program in CSE has the most potential to provide a coherent and well-managed education. It probably also does the most to advance the field and guarantee continuity within the institution. As campuses develop a number of "computational X" programs, it may be more efficient and simpler to unify these "computational X" programs into a CSE department. Such units may then serve even more departments by offering a minor in CSE. However, it appears that a degree in "computational X" is currently providing a very similar education.

Strong links and cooperation now appear to be building up among the people directing the new CSE programs and those who want to start such programs. Indeed, efforts are being supported by the NSF [40, 63, 64] and DOE [35]. We are now sharing course materials for newly developed courses as well as giving and receiving guidance on the development of a well-balanced curriculum. Development of student learning outcomes, building a consensus on a standard CSE curriculum, and the development of some high-quality textbooks will help in the development of the field and in the placement of our graduates.

While there appears to be general agreement that it is appropriate to teach CSE at the graduate level, an undergraduate education in CSE is still a new thing. A

decade ago, a widely held view was that undergraduates should view computational methods as just "black boxes" that should remain closed [65]. However, we believe that the pervasiveness and importance of computation throughout all of science and engineering means that even undergraduates should be able to enroll in programs that teach them what's going on inside the black box. Time will judge the viability of these programs.

Acknowledgments. We wish to thank Angela Shiflet, Charles Swanson, James Corones, Bob Panoff, and Joe Zachary for valuable suggestions and encouragements; Katie Stowe and Rachel Ivie for providing the AIP survey data; and the SIAM referees and editors for their constructive criticisms.

REFERENCES

[1] *Climate Modeling*, Comput. Sci. Engrg., 4 (2002), special issue.
[2] *LATTICE97*, Nuclear Phys. B, 63 (1998), p. 1.
[3] O. Yaşar, *A scalable model for complex flows*, Int. J. Comput. Math., 35 (1998), pp. 117–128.
[4] *Materials Science*, Comput. Sci. Engrg., 3 (2001), special issue.
[5] *Envision*, NPACI & SDSC Quarterly Science Magazine, 18 (2002).
[6] E. S. Hertel et al., *CTH: A software family for multi-dimensional shock physics analysis*, in Proceedings of the 19th International Symposium on Shock Waves, Vol. 1, R. Brun and L. D. Dumitrescu, eds., Marseille, France, 1993, pp. 377–382.
[7] *Chaos*, 12 (2002).
[8] *Data Mining*, Comput. Sci. Engrg., 4 (2002), special issue.
[9] K. Stewart, R. Giles, and I. Zaslavsky, *Super-Partnerships: Computational Science Curricula, High Performance Computing and the Professional Organizations,* EDUCAUSE '99, http://www.educause.edu/ir/library/html/edu9928/edu9928.html.
[10] O. Yaşar, K. S. Rajasethupathy, R. E. Tuzun, R. A. McCoy, and J. Harkin, *A new perspective on computational science education,* Comput. Sci. Engrg., 5 (2000), pp. 74–79.
[11] SIAM Working Group on CSE Education, *Graduate education in computational science and engineering*, SIAM Rev., 43 (2001), pp. 163–177.
[12] O. Yaşar, *Computational Science Program at SUNY Brockport*, in Proceedings of the First SIAM Conference on Computational Science and Engineering, 2000, Washington, D.C., 2000.
[13] L. D. Fosdick, E. R. Jessup, C. J. C. Schauble, and G. Domik, *An Introduction to High-Performance Scientific Computing*, MIT, Cambridge, MA, 1996.
[14] H. J. Gould, J. Tobochnik, and W. Christian, *Simulations in Physics*, 3rd ed., Addison-Wesley, Reading, MA, 2003.
[15] P. Harrison, *Computational Methods in Physics, Chemistry and Biology*, Wiley, New York, 2001.
[16] R. H. Landau and M. J. Paez, *Computational Physics, Problem Solving with Computers*, Wiley-Interscience, New York, 1997.
[17] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, Cambridge University Press, Cambridge, UK, 1994.
[18] J. L. Zachary, *Introduction to Scientific Programming, Computational Problem Solving Using Maple and C*, Springer/Telos, New York, 1996.
[19] P. D. Lax, *Report of the Panel on Large Scale Computing in Science and Engineering*, DOD, NSF, DOE, NASA, 1982.
[20] R. W. Hocknew and J. W. Eastwood, *Computer Simulations Using Particles*, Adam Hilger, New York, 1988.
[21] President's Information Technology Advisory Committee, *Information Technology Research: Investing in Our Future*, http://www.ccic.gov/ac/.
[22] National Science Foundation, *Math and Science Partnership Program*, http://www.ehr.nsf.gov/msp/.
[23] M. Wright and A. Chorin, *Mathematics and Science*, National Science Foundation, Division of Mathematical Sciences, Washington, D.C., 1999.
[24] A. Harrison, *An Exploration of the Nature and Quality of Undergraduate Education in Science, Mathematics, and Engineering*, Sigma Xi, Research Triangle Park, NC, 1989.
[25] K. L. Johnston and B. G. Aldridge, *The crisis in science education: What is it? How can we respond?*, J. Coll. Sci. Teach., 14 (1984), p. 20.

[26] F. J. Rutherford and A. Ahlgren, *Science for All Americans*, Oxford University Press, New York, 1990.
[27] J. A. Dunkhase and J. E. Penick, *Problem solving in the real world*, J. Coll. Sci. Teach., 19 (1990), pp. 367–370.
[28] National Research Council, *National Science Education Standards*, National Academy Press, Washington, D.C., 1996.
[29] LEAD Center, *Learning through Evaluation Adaption Dissemination*, University of Wisconsin, Madison, http://www.cae.wisc.edu/∼lead/.
[30] National Foundation for Improving Education, *Connecting the bits: A reference for using technology in teaching and learning in K–12 schools*, http://www.nfie.org/publications/connecting.htm.
[31] J. R. Rice, *Academic programs in computational science and engineering*, Computational Science and Engineering, 1 (1994), pp. 13–21.
[32] D. E. Stevenson, *Science, computational science and computer science: At a crossroads*, Comm. ACM, 137 (1994), pp. 85–96.
[33] M. L. Ennis, *Update on the Status of Computational Science and Engineering in U.S. Graduate Programs*, AHPCC99-023, Albuquerque High Performance Computing Center, University of New Mexico, Albuquerque, 1999.
[34] C. D. Swanson, *Computational Science Education Survey,* Krell Institute, http://www.krellinst.org/learningcenter/CSE_survey/.
[35] The Krell Institute Learning Center, http://www.krellinst.org/learningcenter.
[36] The Pew Charitable Trusts, *The Responsive University: Restructuring for High Performance*, http://www.pewtrusts.com, 1998.
[37] The Center for Higher Education Policy Analysis, *Projects/Papers*, http://www.usc.edu/dept/chepa.
[38] *Ph.D. in Scientific Computing at The Center for Advanced Computing*, College of Engineering, Univ. of Michigan, http://cac.engin.umich.edu/academics/.
[39] The Education Center on Computational Science and Engineering, http://www.edcenter.sdsu.edu/.
[40] National Computational Science Institute, http://www.computationalscience.net; Shodor Foundation, http://www.shodor.org; The Computational Science Education Project, http://csep1.phy.ornl.gov/csep.html.
[41] R. Ivie and K. Stowe, *What's Important?*, Physics Trends Flyer, American Institute of Physics, College Park, MD, 1999.
[42] National Science Board, *Science and Engineering Indicators*, Chapters 3-2, National Science Board, Washington, D.C., 1996.
[43] O. Yaşar, *Computational science education: Standards, learning outcomes and assessment,* in Computational Science—ICCS 2001 International Conference, V. N. Alexandrov et al., eds., Lecture Notes in Comput. Sci. 2073, Springer-Verlag, Berlin, 2001, pp. 1159–1169.
[44] *Computational Physics for Undergraduates (CPUG)*, Oregon State University, http://www.physics.orst.edu/CPUG/; sample curriculum at http://www.physics.orst.edu/CPUG/BSCP/curricmod3.html.
[45] R. H. Landau, *Developing components and curricula for a research-rich undergraduate degree in computational physics*, in Computational Science—ICCS 2001 International Conference, V. N. Alexandrov et al., eds., Lecture Notes in Comput. Sci. 2073, Springer-Verlag, Berlin, 2001, pp. 1051–1060.
[46] O. Yaşar, *Computational Science Portal*, http://www.cps.brockport.edu/˜yasar/private/cse.html.
[47] *Graduate Programs in Computational Science*, http://www.siam.org/cse/cse_programs.htm.
[48] *Bachelor of Computational Science*, Australian National University, http://room.anu.edu.au/bcomptlsci/.
[49] Department of Computational Science, Kanazawa University, Japan, http://cmpsci.s.kanazawa-u.ac.jp/English/.
[50] *Moderatorship in Computational Physics*, Physics Department, Trinity College, Dublin, http://www.tcd.ie/Physics/Courses/page39.html.
[51] Department of Computational Science, National University of Singapore, http://www.cz3.nus.edu.sg/AY2001-02handbook.html.
[52] *Computational Science Program*, University of Waterloo, http://www.science.uwaterloo.ca/ustudent/compscience.html.
[53] Computer Science Department, Carnegie Mellon University, http://www.csd.cs.cmu.edu/; sample curriculum at http://www.csd.cs.cmu.edu/education/bscs/currsequence.html.

[54] Department of Computer Science, University of Illinois at Urbana-Champaign, http://old-www.cs.uiuc.edu/; sample curriculum at http://old-www.cs.uiuc.edu/education/undergrad/degrees/cs.html.
[55] Department of Computer Science, Oregon State University; http://www.cs.orst.edu; sample curriculum at http://www.cs.orst.edu/info/undergrad/acsopt.html.
[56] Department of Computational Science, State University of New York College at Brockport, http://www.cps.brockport.edu.
[57] *Engineering Science Program*, University of California at Berkeley, http://www.coe.berkeley.edu/engsci/; sample curriculum at http://www.coe.berkeley.edu/CES/ces_curric.html.
[58] *B.S. Degree in Computational Physics*, Departments of Computer Science, Engineering, and Physics, State University of New York, Buffalo, http://www.physics.buffalo.edu/undergrad/cp.html.
[59] *Computational Physics B.S.*, Illinois State University, www.phy.ilstu.edu/CompPhys/CompPhys.html; sample curriculum at http://www.phy.ilstu.edu/CompPhys/courseq.html.
[60] Department of Physics, University of Illinois at Urbana-Champaign, http://www.physics.uiuc.edu/; sample curriculum at http://www.physics.uiuc.edu/Education/undergrad/programs.
[61] Department of Physics, Oregon State University, http://www.physics.orst.edu/; sample curriculum at http://www.physics.orst.edu/Advising.
[62] IEEE Computer Society, *Careers in Computing*, http://www.computer.org/education/careers.htm.
[63] National Partnership for Advanced Computing Infrastructure, http://www.npaci.edu.
[64] Education, Outreach and Training Thrust Area, National Partnership for Advanced Computational Infrastructure, http://www.npaci.edu/Outreach/.
[65] D. L. Greenwell, R. K. Kalia, P. Vashishta, and H. W. Myron, eds., *Workshop Report, Undergraduate and Graduate Education in Computational Sciences*, Louisiana State University and U.S. Department of Energy, 1991.