Editor: Denis Donnelly, donnelly@siena.edu

A NEW PERSPECTIVE ON COMPUTATIONAL SCIENCE EDUCATION

By Osman Yasar, Kulathur S. Rajasethupathy, Robert E. Tuzun, R. Alan McCoy, and Joseph Harkin

THE FIELD OF COMPUTATIONAL SCIENCE AND ENGINEERING WAS FIRST RECOGNIZED AT RESEARCH INSTITUTIONS AND INDUSTRIAL SETTINGS WHERE DEPENDENCY ON COMPUTERS WAS GROWING RAPIDLY. PROMINENT SCHOOLS WERE FIRST TO establish graduate certificate programs to integrate computational research into PhD education. As new initiatives appeared at the government level for high-performance computing and communications and the job market heated up, the need for new CSE professionals at MS and even BS levels became obvious. To help fill this need, the US Government offers support to educate future computational scientists (see www.itr.nsf.gov), including at the undergraduate level. But how do we educate these students?

At the 1999 High Performance Computer Users Group Conference (see www.hpcu.org), organized by the Computational Science faculty at the State University of New York, Brockport, a session on CSE education gathered valuable input from the community. We had an interesting session there on Brockport's undergraduate CSE program.

CSE as a new focus area

CSE overlaps with many other knowledge areas, so an educational program in CSE naturally draws strength from all of them. Nevertheless, in addition to overlapping with computer science, math, and science and engineering application areas, computational science has its own core knowledge area (see Figure 1). Although some computer science and mathematics programs have championed this new field, CSE also finds strong allies in other science departments, particularly physics and biology. In most cases, it was established through a widely interdisciplinary effort.

Computational science and computer science have common concerns when it comes to computer performance and application optimization; computational science and mathematics have common concerns when it comes to applied math techniques. Finally, CSE shares concerns with many application areas (such as physics, chemistry, biology, earth sciences, business, and art) in terms of finding a computer-based solution to complement theoretical and experimental efforts. In some cases, what we can accomplish through computation cannot be done otherwise. We have gained new insights by modeling and visualizing physical systems that are too small (probing atomic systems, for example), too big (studying the earth and the universe), too expensive, too scarce, and inaccessible experimentally (weighing the impact of an asteroid on earth).

Teaching, research, and service in the CSE field differ from those in other disciplines in many ways. Although the number of students might not be high initially, course preparation time is very demanding. New

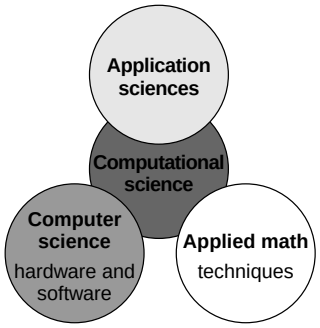

Figure 1. Computational science is a bridge connecting computing and math technology with the sciences, but it is also a discipline of its own.

courses are continually being developed, because there is still no established curriculum, especially at the undergraduate level. Curricula also change frequently as new tools are developed. Faculty in programs housing parallel supercomputers are always on the look-out for new hardware and software. They must also stay in close contact with colleagues in high-performance computing research centers to keep up on supercomputer developments and maintenance. Moreover, faculty research is crucial: a field that involves so much information and draws on knowledge in a wide range of areas needs time and attention to identify and further study common techniques. A good example is particle dynamics, which is encountered in simulations of sand particles in a tank, planetary motion, molecular dynamics, thin-film growth, spray dynamics in engines, ash particles in industrial burners, and many others. Finally, CSE students will require a lot of help and advice in more than one field, and they will expect their CSE faculty members to be knowledgeable in most of the program's courses. All of these responsibilities impinge on time available for other activities. Thus, administrators need to account for faculty time spent on all these unique services to the field when considering the norms for teaching, service, and scholarship workloads.

What should a CSE education include?

A CSE curriculum should include not only the development of computing and mathematical skills in an applied way but also the study of common computational techniques, tools, and high-performance computing systems in the context of applications. This requires study in the following six knowledge areas (undergraduate and graduate), each of which results in specific learning outcomes. Our Web site describes SUNY, Brockport's specific curriculum, including numbered course descriptions (www.cps.brockport.edu).

Computational science tools. Learning to use computers is the most important aspect of CSE, because the computer itself is the tool. Learning outcomes include

- the ability to program in Fortran, F90, high-performance Fortran, C, and C++;
- a knowledge of the Unix operating system and working environment;
- the ability to use software packages such as Maple, Matlab, Macsyma, and Mathematica and the application of calculus techniques within these software environments;
- an understanding of industrial benchmarks such as Linpack;
- the ability to use numerical techniques and applications of integration, differentiation, and partial derivatives;
- the ability to use mathematical libraries such as Blas and ScaLapack and problem-solving environments such as NetSolve; and
- the ability to use visualization software packages such as AVS.

High-performance computing. Learning high-level languages and how to use high-performance computers is essential to tackle computationally demanding real-world problems. Learning outcomes include

- the ability to program on supercomputers,
- an understanding of computer speed and performance benchmarks,
- an understanding of modern computer architecture and language support for performance, and
- the ability to use parallel computers and communication libraries such as PVM and MPI.

Applied and computational methods. Knowledge of applied mathematical methods is essential for CSE majors, no matter how much the field depends on computers. Although a theoretical understanding and a survey of all numerical methods are important, students in CSE must pay closer attention to the use of common methods and their performance. Learning outcomes include

- a knowledge of computational methods (finite difference, finite elements, discrete particle method, discrete ordinates method, random-number generators, mesh generation, adaptive mesh techniques, Runga-Kutta, fast Fourier transforms, Monte Carlo methods, and so on) to numerically approximate solutions to real-world problems;
- a familiarity with the partial differential equations encountered in science and engineering;
- a knowledge of generating simple computational grids;
- a knowledge of discretizing partial differential equations and of numerical schemes to solve them;
- the ability to use these numerical methods and software libraries;
- the ability to use matrix computations and available software libraries (such as Linpack); and
- the ability to apply these methods, through computer programming, to an application area.

Simulation and modeling. This is an important element of CSE, because

it is the ultimate phase of solving a real-world problem, be it a simulation of an engine, an airplane take-off, weather prediction, or design of a new material or drug. Learning outcomes include

- a familiarity with the governing equations of physical systems (continuity, momentum, and Schrödinger equations, to name a few);
- the ability to discretize these equations in space, time, angle, or phase space and to generate a computational grid to solve these equations;
- the ability to apply math and computer programming skills to a variety of applications;
- the ability to analyze simulation input and output data; and
- the ability to use visualization software (such as AVS) to postprocess output data.

Visualization tools. This is another important final phase of problem solving through which students learn how to interpret and analyze data after a simulation is complete. Learning outcomes include

- the ability to use visualization tools such as AVS and
- a knowledge of visualization techniques.

Computer applications. Students' exposure to a field of interest is an essential part of learning in CSE. Advisers help students identify the courses they need (within the CSE program or in related departments) and the application areas they might want to study. It is important to expose undergraduate CSE students to a wide range of applications. The job market they might enter after graduation is dynamic: most active recruitment today is for computational biologists, but the market for graduates in computational chemistry, computational physics, and computational finance is also growing rapidly. Expecting undergraduate students to specialize in just one field before graduation might be unrealistic; even requiring them to take courses in a predetermined application area (such as physics) or in a set of areas would be limiting.

Why aim at undergraduates?

Some CSE educators wonder whether it is practical to have an undergraduate program in this field. There has been an assumption that an undergraduate CSE education would require students to undertake a heavy course load that is difficult to manage. Furthermore, some wonder whether an interdisciplinary education is obtained at the expense of acquiring a deep undergraduate-level knowledge in one particular subject. Although some might be skeptical about the practicality of an undergraduate CSE education, there have not been any experiments with undergraduate programs in this field supporting such views. In fact, to our knowledge, our experiment—the development of SUNY, Brockport's undergraduate program in CSE—was the first, and it was a very fruitful experience. We hope that other newly established undergraduate programs will publish experience reports in the future. It is important to note that long before interdisciplinary programs (such as CSE) were introduced at the undergraduate level, students on many campuses were already pursuing double and even triple majors. This is evidence that as heavy and diverse as a CSE program might be, it is still practical at the undergraduate level. It is important to strike the right balance for a combined knowledge of constituent disciplines.

Introducing CSE to undergraduates creates a coherent and consistent program for students. On the other side of the spectrum is the experience of one of us (Osman Yasar), who earned three MS degrees—in physics, computer science, and engineering—and a PhD in engineering physics to pursue his career goal in CSE. Today, graduate CSE programs offer combined training under one umbrella, which is to students' advantage. Obviously, we cannot expect CSE students to gain the same depth in computer science, math, and application sciences as someone specializing in one of these fields. It is time to publicly admit that CSE is a blend of *practical knowledge* in these constitutive fields, which in no way diminishes the field's importance as an academic and scientific discipline. For students who are interested in the sciences but do not want to go deep into one particular area, there ought to be a program to teach problem solving. Moreover, some of the deep learning that goes on in BS, MS, and PhD programs never gets applied after graduation. Most of us with computer science degrees never had to write an operating system, a compiler, or anything involving TCP/IP protocols after we left school. The same thing applies to math and physics, which require their majors to go deep into the field, yet very few physicists (with just an undergraduate degree) make a living doing quantum mechanics. The realities of the job market motivate students to earn a degree in applied sciences and particularly in CSE—a respected, relatively recognized, and significantly promising field in which to pursue a career.

As mentioned earlier, computational research and education started at the doctorate level and are now working their way down to both masters and

baccalaureate levels. It has been our experience that those of us (computational researchers and educators with doctorate degrees) involved in the design of masters and baccalaureate curricula often expect all students to have a knowledge and ability comparable to our levels upon graduation. Because CSE combines knowledge in computer science, math, and application sciences, mastery of all these requires an extended education; so far, we have seen this combination come to be realized only at the doctorate level. Can this same model be brought down to the undergraduate level? No. We cannot expect doctorate-level depth and breadth nor skills from undergraduates. Nor can we expect undergraduate students to be able to do simulations for us right away. People with doctorates in CSE will most likely find work in their chosen specialty area, but an undergraduate might pursue a career in many directions after graduation. When we extend our scholarly perspectives to a four-year college education, our expectations for specialization run deep and narrow (see Figure 2), often overloading BS and MS students with specialized courses that have many prerequisites.

CSE program at SUNY, Brockport

As a part of integrating new technology into education, SUNY, Brockport, formed discussion groups and interdisciplinary committees in the early 1990s. In 1995, it formed a CSE cluster (among other clusters) to prepare the college for the next century. The institution approved recommendations offering both undergraduate and graduate CSE programs. We created a core curriculum based on existing courses and decided to recruit experienced people from outside who could add new perspectives to the program's future, national appeal, curriculum, and course offerings. The New York State Education Board approved BS and MS degree programs in 1997 and 1998, respectively. Thus, Brockport became the first—and perhaps still the only—degree-granting undergraduate program in CSE.

Creating the program took a great deal of multidisciplinary effort, yet the requirements of sustaining it needed a core faculty body and a unit head to devote full time to its growth and management. Recruiting new CSE faculty proved difficult, because industry, national labs, and computer vendors usually pay qualified computational scientists a higher salary than what academic institutions usually offer. The school then gave precedence to hiring a unit head, who later built a core faculty of qualified people. The tasks of setting future directions, recruitment, and other departmental administration require a great deal of attention from the unit head, so his teaching load has been kept at a minimum. There is much to be investigated about the dynamics of CSE's interdisciplinary blend and about the organization and study of common techniques, overlap, and communication among applications. There must be a delicate balance in terms of what students need to learn about each CSE component and what the university expects of its faculty.

The university established this program for several reasons: to

- prepare for the next century;
- boost enrollment by stimulating renewed interest in the science disciplines;
- harmoniously integrate computing into science and engineering by establishing a coherent and interdisciplinary program;
- avoid duplication of courses in different departments, thus optimizing human and material resources on campus; and
- seize the opportunity of national leadership in curriculum development for CSE.

SUNY, Brockport, provides an excellent environment for undergraduate programs such as CSE to prosper. Entering students need not specify a major for the first two years; in fact, they can switch majors even after that. There is a sense here that the boundaries between separate programs and departments are flexible, perhaps because of the college's small size or because there are many common courses (in general education and in physical education, for example) that bring students with varied interests together. An undergraduate CSE student who later decides to move to another program will probably not lose any credits, be-

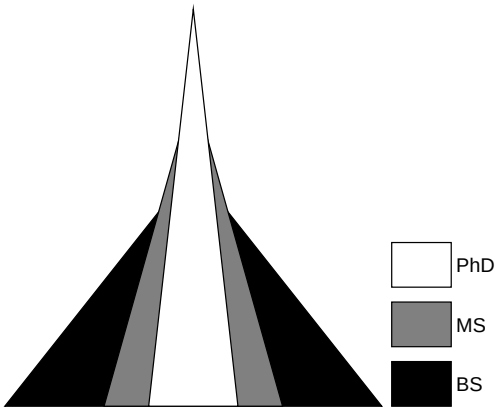

Figure 2. A perspective on the depth and breadth of CSE education.

Table 1. The new undergraduate curriculum.

Course	Number of credits
Prerequisites	
Calculus I	3
Calculus II	3
Discrete Mathematics I	3
Introduction to Computer Science	3
Fundamentals of Computer Science I	4
Total	16
Major Requirements	
Mathematics	9
Calculus III	3
Elementary Statistics	3
Linear Algebra	3
Computer Science	4
Fundamentals of Computer Science II	3
Computational Science	15
Computational Tools I	3
Computational Tools II	3
High-Performance Computing	3
Simulation and Modeling	3
Applied and Computational Math	3
Application Sciences	8
200-level and higher non-CSE courses, to be chosen under advisement	
Electives	6
Selected from upper-division courses, to be chosen under advisement	
Total	42
Minor Requirements	
Computational Science Courses	12
Computational Tools I	3
Computational Tools II	3
High-Performance Computing	3
Simulation and Modeling	3
Electives	8
200-level and higher non-CSE courses, chosen under advisement	
Total	20

cause many of the courses already taken—for instance, in computer science, math, and physics—would count toward any science program. This setting makes it less risky for students to try CSE. The large number of CSE majors in our program gives us a considerable pool to examine the blend and the amount of interdisciplinary course load needed to prepare students for computationally oriented jobs in the work place.

Our program has exceeded its enrollment targets. We have 50 students (30 undergraduate and 20 graduate) af-

Table 2. The new graduate curriculum.

Course	Number of credits
Major Requirements	
Computer science	4
Advanced Data Structures	4
Mathematics	3
Discrete Mathematics II	3
Computational science	16
Scientific Visualization	3
Advanced Comp Software Tools	3
Supercomputing and Applications	3
Graduate Seminar	1
Independent Study	3
Project Paper	3
Total	34
Electives: 500 level and above	11
From the following courses or courses in application sciences as per advisement:	
Computational Techniques	
Computational and Applied Math	3
Computational Methods in Phys. Sciences	3
Deterministic Dynamical Systems	3
Stochastic Dynamical Systems	3
Computer Hardware/Software	
Computer Networks	3
Relational Database Design	3
Object-Oriented Programming	3
Computer Architecture	3
Operating Systems	3
Advanced Computer Architecture	3
Theory of Programming Languages	3
Mathematical Methods	
Numerical Analysis	3
Differential Equations	3
Statistical Methods II	6
Math Models for Decision Making	6

Osman Yasar is a professor and chair of the Computational Science Department at the State University of New York, Brockport. Prior to this, he was a senior scientist at the Oak Ridge National Laboratory's Center for Computational Science, where he started a computational engine modeling group. His research interests include plasma physics, radiation hydrodynamics, engine combustion modeling, and supercomputing. He consults with Lockheed Martin Energy Research Corporation and the Naval Surface Warfare Center at Carderock and chairs the High Performance Computer Users Group. He earned a PhD in engineering physics, an MS in computer science, and an MS in nuclear engineering from the University of Wisconsin–Madison, and an MS in physics and a BS in engineering physics from Hacettepe University in Ankara, Turkey. Contact him at the Dept. of Computational Science, State Univ. of New York, 350 New Campus Dr., Brockport, NY 14420; oyasar@brockport.edu; www.cps.brockport.edu.

Kulathur S. Rajasethupathy is a professor and chair of the Department of Computer Science, SUNY, Brockport. His interests include formal languages, database systems, and computer science education. Contact him at the Dept. of Computer Science, State Univ. of New York, 350 New Campus Dr., Brockport, NY 14420; kraja@cs.brockport.edu; www.cs.brockport.edu.

Robert E. Tuzun is an assistant professor in the Department of Computational Science at SUNY, Brockport. He earned his BS in chemistry and in chemical engineering from the University of Delaware and his PhD in physical chemistry from the University of Illinois at Urbana-Champaign. Contact him at the Dept. of Computational Science, State Univ. of New York, 350 New Campus Dr., Brockport, NY 14420; rtuzun@brockport.edu.

R. Alan McCoy is an assistant professor in the Department of Computational Science at SUNY, Brockport. His focus is scalable math libraries, and he recently has been working on computational finance projects. He received his PhD from the Applied Mathematics and Statistics Dept. at SUNY Stony Brook. Contact him at the Dept. of Computational Science, State Univ. of New York, 350 New Campus Dr., Brockport, NY 14420; amccoy@brockport.edu.

Joseph Harkin is a professor in the Mathematics Department at SUNY, Brockport. Prior to this, he worked in industry. He has been a long-time supporter of computational science. His mathematic research origins are in functional analysis, integration theory, and dynamical elasticity theory. Contact him at the Mathematics Dept., State Univ. of New York, 350 New Campus Dr., Brockport, NY 14420; jharkin@brockport.edu.

ter two years of operation. Besides the majors, the program also serves students and faculty members in other departments through new courses, hardware, and application software. We have recently revised both the BS and MS curricula (see Tables 1 and 2). We have already offered the new courses to both CSE and non-CSE majors, teaching them integrated (computing, mathematical, simulation, and visualization) skills in the sciences. The revised curriculum is also a response to the needs of employees working in Greater Rochester companies such as Kodak and Xerox.

Our efforts have been both interdisciplinary and multi-institutional. The CSE program has acquired a substantial amount of external funds in the form of equipment (including supercomputers) from Intel and SGI and research grants from the US National Science Foundation, the US Departments of Energy and Defense, and industry. We are working with SUNY, Stony Brook; San Diego State University; and Boston University to develop joint courses in high-performance scientific computing and scientific visualization, and we hope to obtain more NSF funds to strengthen these partnerships. Collaboration with the Oak Ridge National Laboratory has been a strong element in establishing our program. We are seeking to disseminate the new curricula through NSF's Education Outreach and Training Partnership for Advanced Computational Infrastructure (www.eot.org) and the High Performance Computing Users Group (www.hpcu.org).

The content of faculty research is tightly connected to curriculum. We have a well-established computational research program in fluid dynamics, engine combustion, plasma physics, parallel computing, visualization, molecular dynamics, drug design, and math software libraries. To introduce students to simulation and modeling in a hands-on way, we must bring the expertise we gain doing research into the classroom; students can learn more effectively by simulating systems of their choice. We must also bring to the classroom our collective experience with common tools such as computing, numerical methods, parallel programming, and visualization. The overall research expertise of our faculty provides a pool of knowledge and tools to support a sound curriculum for students in both CSE and other disciplines.

Helga Rowe Computers allow for the development, adaptation and/or delivery of tools which facilitate more effective thinking, problem solving and learning. These tools are different from norn1al, task-specific tools. We refer to them as cognitive tools because they are knowledge construction and facilitation tools which can be applied in most domains. Cognitive tools are defined as mental and/or technological devices which support, guide and extend the thinking processes of their users.Â This chapter considers the role and effectiveness of computer-based tools for thinking, and how they ean mediate learning. The importance of appropriate human-computer interface designs to ensure that the cognitive tool simplifies, rather than complicates, the user's tasks, is stressed. report - Dmitry Abbakumov "Computational science in education: current trends and practical cases". Reports 2019.Â II Russian-Chinese conference of education researchers "Digital transformation of education and artificial intelligence" September 26-28, 2019, Moscow, Russia report - Daria Kravchenko and Ekaterina Kalyaeva "Educational analytics in the modernization of online courses: experience of 15 universities".Â The International Meeting of the Psychometric Society July 9-13, 2018, New York, USA report - Dmitry Abbakumov "Measuring student's proficiency in MOOCs: Multiple attempts extensions for the Rasch model" (presentation). Computational Science Education: Standards, Learning Outcomes, and Assessment. May 2001. DOI: 10.1007/3-540-45545-0_127.Â We have entered a new phase in the growth process of computational science. The first phase (1990â€“2000), which coincides with the federal high performance computing and communication program, can be named as the recognition phase, at the end of which there was a general agreement to accept computation and computational science as a distinct methodology and discipline. The recognition started at the doctorate level and moved down to at least the baccalaureate level and even to a few high schools. 1. Introduction International collaboration in Computer Science education is becoming more and more widespread. Very often this will simply take the form of student and/or teacher exchanges (e.g. under the EUâ€™s ERASMUS and SOCRATES [2, 3] initiatives). More extensive collaborations most commonly take the form of an â€œunequalâ€ partnership, in which one or more institutions are franchised to shadow courses at the leading institution.Â We now have a shortfall of 15 ECS places, and we need to find a new German-speaking partner to make up this shortage. 3.4. Huddersfield. The University of Huddersfield has existed as an institute since the 1840s. O. Yasar, et al. , â€œA New Perspective on Computational Science Education â€ IEEE Computing in Science and Engineering , Vol. 2, No. 5, 2000. Google Scholar. 2. O. Yasar, â€œComputational Science Program at SUNY Brockport â€ Proceedings of First SIAM Conference on Computational Science and Engineering , September 21-24 2000, Washington, D.C. Google Scholar. 3. Graduate Education for Computational Science and Engineering, SIAM Working Group on CSE Education, http://www.siam.org/cse/report.htm . 4. A. A. Amsden, â€œKIVA-II: A Computer Program for Chemically Reactive Flows with Sprays â€ Technical Report, LA-11560-MS, Los Alamos National Laboratory (1989). Google Scholar. 5.