Computational Science Education: Standards, Learning Outcomes, and Assessment

Osman Yasar
Department of Computational Science
State University of New York
Brockport, NY 14420

Abstract

We have entered a new phase in the growth process of computational science. The first phase (1990-2000), 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. This first decade of growth period saw at least one stand-alone computational department, one stand-alone school at the dean level, and a program to train high school teachers. The second phase (2001-2010), which coincides with the federal information technology program, will witness curriculum standardization at all levels, perhaps accompanied with an accreditation mechanism for future programs. It is important to assess student success in the new programs. In this paper, we will address learning outcomes and assessment techniques, followed by a brief account of research-curriculum integration at our institution. We will also give a brief overview of computational science and engineering.

1. Overview

Professional societies such as SIAM (www.siam.org), IEEE Computer Society (www.ieee.org/computer), ACM (www.acm.org), AMS (www.ams.org) and Society of Computer Simulation (www.scs.org) have all undertaken major initiatives to organize annual conferences on computational science and engineering (CSE). There are also new professional societies putting computational science at the center of their activities. These include Society of Computational Biology and Society of Computational Economics, among others. The number of web sites for CSE programs, research centers, government labs, and industrial settings has grown by an order of a magnitude. A search for "computational science" over the Internet gets several hundreds to thousands of hits and links. A recent resulted in the following number of hits: GoTo (240), LookSmart (130), Lycos (46,045), HotBed (466,500), and AltaVista (65,447). At least the first 500 of such links were reviewed and found to be very relevant to the search topic.

Computational science and engineering has emerged as a new discipline in the past decade. At the core of this development is a dramatic increase in the power and use of computers. Capitalizing on advances in computing technology, new methods and programming tools were developed to solve problems that were not in our reach before. As much of an interdisciplinary program as computational science is, it is also a discipline of its own due to: 1) the amount of knowledge involved in its presentation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1159–1169, 2001.

to colleagues and students, 2) the amount of scientific literature (journals and conferences) devoted to this topic, and 3) the work and service involved for further recognition, establishment, and success of computational culture in educating new generations.

Computational science has two intermingled contexts attached to it: 1) *science of computing,* and 2) *science that is done computationally.* The CSE field investigates computational techniques that are common to many applications; therefore it focuses on the art/science/engineering of computing. The applications that use computing are many, ranging from basic sciences to engineering and industrial problems. All these compute-bound scientific and industrial problems form a bond together with many exchanges and commonalities among each other. Under the umbrella of *computational science,* one can find *computational biology, computational physics, computational chemistry, computational finance, and computational mechanics,* and so on. However, the discipline of computational science does not necessarily cover core knowledge and experience of all these sciences; it only covers their computational aspects. Therefore, computational science serves not a replacement to any of the science disciplines, but as a bridge between sciences, engineering, computing, and mathematics [1].

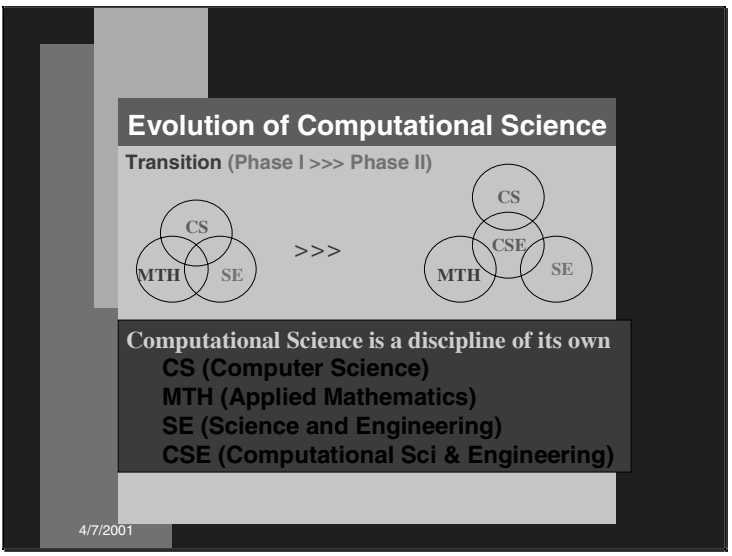

Fig. 1. Evolution of Computational Science and Engineering

There is a natural overlap of CSE with other sciences, however CSE has a core knowledge base of its own. Computational science and computer science have common concerns when it comes to performance of computer hardware/software and anything related to optimizing one's application on computers. Computational science and mathematics have common concerns when it comes to applied math techniques to numerically solve partial differential equations. Finally, computational science shares

concerns of application areas (such as physics, engineering, chemistry, biology, earth sciences, business, art) in terms of finding a computational solution to complement both theoretical and experimental efforts. In some cases, what can be accomplished by computation cannot be done otherwise. Physical systems that are too small, too big, too expensive, too scarce, and not accessible (experimentally) are being modeled on computers with a great deal of success. Examples of probing atomic systems (too small), studying earth and universe (too big), weighing impact of an asteroid on earth and studying internals of an engine piston (not accessible) are just a few. Computer visualization of such systems has also created a new way of gaining insight that otherwise would not have been discovered.

A field that involves so much information and draws upon knowledge in other areas also needs time and attention to identify and study techniques common to many applications. It also needs full-devotion to the study of performance of computer hardware/software as well as of computational methods and tools that otherwise might not have been studied. When one considers all these non-overlapping and overlapping components, the field basically becomes a discipline of its own. At the heart of the field is the study of common computational techniques, and unless there is a full devotion to this study, the field cannot advance very quickly. The transition of CSE from a mere overlap of computer science, math, and applications to a field with its own identity and knowledge base is now taking place, as illustrated in Fig. 1. Although we did not encounter a consensus on this latest development earlier, we now note similar views published recently by our colleagues [3]. Scientists and students in this area have a unique identity as computational scientists and engineers who have gathered a combination of practical knowledge in computing, mathematics and applications.

2. Student Learning Outcomes

At SUNY Brockport, we offer both undergraduate (B.S.) and graduate (M.S.) degrees in computational science. Our program started in the Fall of 1998 and was transformed into a department after two successful years of operation. We have about 50 students enrolled in the program and our first graduates have actually hit the job market. To our knowledge, we were the first undergraduate program in computational science, and now perhaps the only department in this field. Having no precedence before us, we have struggled with many issues including curriculum development, faculty career development, tenure guidelines, recruitment, placement, and documentation. Our curriculum was revised recently and we expect minor revisions in the future. We are still developing new courses, particularly in the area of computational applications.

In an effort to encourage standardization, we published our assessment of the elements of a typical computational science education [1,2]. As mandated by our college policies at SUNY Brockport, we now have identified, in a more concise way, the students learning outcomes (SLO) as listed below. The task before us will enable us to document in detail all measurable aspects of a computational science education, including the seven SLOs we have identified below:

(a) Learning the use of computers and computational tools,
(b) Learning high-level languages and the use of high performance computers,
(c) Obtain a knowledge of applied math and computational science methods,
(d) Learn basics of simulation and modeling,
(e) Learn how to visually interpret and analyze data after a simulation is completed,
(f) Learn about at least one application area to apply acquired computing skills to,
(g) Learn to communicate solution methods and results.

The challenge, of course, is to find ways to measure whether these outcomes have been achieved. In Table 1, we list the first SLO and relevant course objectives used to measure its outcome. Course titles and descriptions in our program can be found at http://www.cps.brockport.edu. The courses referenced in this table, however, are listed here again as:
CPS 101 Introduction to Computational Science
CSC 120 Introduction to Computer Science
CPS 201 Computational Science Tools I
CPS 202 Computational Science Tools II
CSC 203 Fundamentals of Computer Science
CPS 303 High Performance Computing
CPS 433 Scientific Visualization
CPS 602 Advanced Software Tools

Table 1. SLO # 1: Learning the use of computers and computational tools

Course	Course objective
CPS101	(1) To learn about functions, their uses and representations. (2) To learn about behaviors of functions and the rate of change (3) To find a functional relation based on behavioral relation using numerical integration (4) To learn the fundamentals of FORTRAN 77 (5) To learn the UNIX operating system and become comfortable in a UNIX working environment. (6) To solve simple real world problems, using numerical solutions, programming, and: (a) differentiation (rate of change problems), (b) integration (area and volume problems), (c) linear regression. (7) To identify a few industrial and scientific problems and their computational solutions
CSC120	1) To learn the internal workings of computers: (a) hardware components such as CPU, memory, disk storage, peripheral devices, etc., and measures of performance, (b) gates and simple circuits such as flip-flop, (c) data types and their internal representations, (d) operating systems and their components, (e) the need for hardware and software standards, and (f) networks and the Internet. 2) To learn high-level and low-level language concepts 3) To learn program execution in terms of machine instructions 4) To learn elementary concepts and syntax of C/C++ programming languages 5) To learn algorithms: (a) examples of some simple algorithms, (b) designing and testing algorithms, (c) translation from the problem domain to the programming domain.

CPS201	(1)	To review programming languages: C++ & F77
	(2)	To learn about abstract data structures in C++ & F77
	(3)	To learn how to use a symbolic manipulation tool (MATHEMATICA) to model simple problems.
	(4)	Use the graphic capabilities of MATHEMATICA to assist in the analysis of simple models
	(5)	To learn the use of LaTeX and the concise, clear presentation of results from simulations.
	(6)	To learn how to use the UNIX operating system (directories, file editing, program compilation).
CPS202	(1)	To learn basic principles of programming in Fortran 90.
	(2)	To learn the use of the MATHLAB software tool including a) language constructs, b) 2d graphics routines (x-y plots, scatter plots, contour plots) c) 3d graphics (surface and mesh plots, line graphs).
	(3)	To learn the use of the Advanced Visualization System (AVS) software tool. This includes a) generating simple plotting interfaces, b) generating a GUI that can interface with external C and Fortran routines, c) working with 2 and 3d graphics as outlined above.
	(4)	To learn about mathematical algorithms: a) random numbers, b) Gaussian elimination, c) Fast Fourier
	(5)	To learn how to use industry standard computational libraries (LAPACK, ATLAS).
CSC203	(1)	To learn fundamental computer science concepts and programming in C++
	(2)	To learn about sorting and searching techniques
	(3)	To learn about files, trees, recursion, graphs, pointers, and classes
CPS303	(1)	Become proficient in the basic use of the MPI message passing library for solving simple problems in parallel.
	(2)	Become familiar with programming in a batch processing environment (compiling, submitting jobs, checking job status, queues).
	(3)	To learn how to decompose a problem in a manner suitable for efficient parallel implementation including communication structuring.
	(4)	To learn how to evaluate the performance of a parallel algorithm (in terms of speedup, efficiency and scalability) and how to modify a given algorithm for improved performance
CPS433 CPS533	(1)	To learn the value of graphic visualization in the context of large or highly complex data sets.
	(2)	To learn the use of the graphical capabilities of various graphical software packages (MATHLAB, MATHEMATICA, XMGR).
	(3)	Learn to develop graphic applications specific to a given discipline using the Advanced Visualization System (AVS) tool.
	(4)	Learn how to interpret simulation results and detect possible errors in data for model simulations arising from the physical sciences.
	(5)	Develop GUI tools using AVS and MATHLAB.
CPS602	(4)	To learn the use of standard parallel programming libraries (ScaLAPACK, PETSc,MPI).
	(5)	To learn the use of various problem solving environments (Netsolve) and parallel tools.
	(6)	To learn how to use grid generation. Algorithms (including automatic mesh generation) to solve models of partial differential equations.

The course objectives listed above need to be measured in each category and subcategory. The method of measurement is homework assignments and tests. The standard for success has been set to 80 % of the assignments and tests in our program. Further details of assessment techniques is available via the author.

For more information on the remaining SLOs, the reader can contact our department directly at cps@brockport.edu, or visit http://www.cps.brockport.edu.

The field of computational science is very dynamic. It is new and still being defined. Since it is a technology (computer hardware and software) oriented field, the content and the curriculum is often being updated. Our program started only 2 years ago, yet we have already done a major revision to our curriculum. Further, but less radical, revisions are expected in the next 3-5 years. In the next 2 years we will still be in a course development mode at both the graduate and undergraduate levels. The content of these courses and the success of our curriculum will greatly depend on our faculty members' updated knowledge about the computer hardware/software, latest mathematical methods and the computational tools in the market place and the literature. Since the computer technology changes radically every 12-18 months, we must quickly adapt to new technology so that our graduates can adjust more easily to the job market. In one respect, we are more technology dependent than the field of computer science where the basics of computing are taught. In computational science, the computing must be put in the context of applications that are driving the market. This dynamic aspect of computational science requires faculty members to be well connected to the research and the industrial community so they can teach updated material and provide timely advice.

3. Research-Curriculum Integration

The research interests in our department cover engineering and scientific aspects of different areas such as engine combustion, fluid dynamics, molecular dynamics, and weather modeling, yet they all use a set of common tools, namely computation, simulation and visualization. The expertise of faculty members in different industrial and scientific areas is brought into the classroom to introduce students to these topics in a hands-on way where they can learn more effectively by simulating systems of their choice. The collective experience by our team on common tools such as computing, numerical methods, parallel programming, and visualization is also brought to the classroom to advance knowledge of students in engineering and computing sciences.

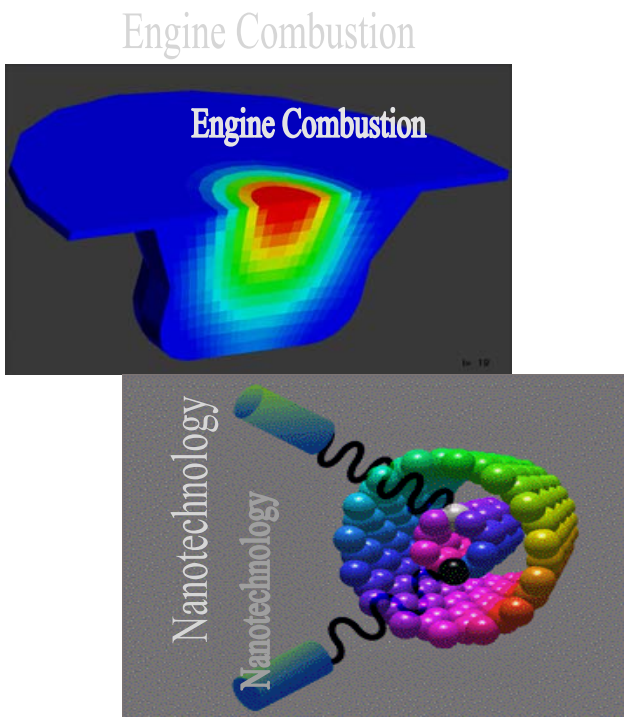

Fig. 2. Two examples of computational science applications at SUNY Brockport

3.1. Engine Combustion

Combustion has been identified as a major study area under the 1999 Presidential Initiative IT-2 (Information Technology Initiative II). Strict regulations on air quality require cleaner engines. Development of full-scale and full-physics (flow, combustion, plasma, spray, radiation) combustion codes is critical for the success of these new programs. Although high-fidelity simulations of internal combustion engines and industrial burners require computers 10,000 times faster than current personal computers, the ability to simulate reasonably representative systems has gone well beyond the circle of a few national labs. For example, a publicly available engine code, KIVA [4], from Los Alamos National Lab can now be run on personal computers. The graphical software to display results in a visual way has also enhanced our ability to understand results and shortened the time to model and analyze engineering systems.

Our combustion team has demonstrated research expertise in engine combustion simulations as well as computational aspects of computer science and mathematics [5-9]. Our version of the engine code KIVA has been referenced by many as the only scalable version that can do multiple engine cycle simulations due to its capacity to simulate multi-million level mesh computations in a reasonable amount of time. This code has been used for joint collaborations with industry. Another aspect of our engine work is the marriage of plasma hydrodynamics [5-6] with engine combustion. We have taken a dramatic approach to fully simulate the interaction between

combustion dynamics and spark ignition. This work resulted in two major CRADAs with industry. Previous approaches to spark ignition dynamics and its effect on combustion dynamics had been limited to crude approximations, yet the amount of spark energy into the combustion chamber is the most critical element of engine operation (for spark-ignited engines). An accurate computation of spark energy deposition into the combustion chamber requires a time-dependent feedback between sparking event and gas dynamics, though this requires solution of both electromagnetic and fluid flow governing equations at a much finer time-scale (nanoseconds) than typical flow simulations.

Integration of scalable engine combustion simulations with advanced visualization techniques has brought our combustion research to a level to be integrated into both engineering and computer science classrooms. Use of AVS (commercial product for visualization) and EIGEN/VR (from Sandia National Labs and Oak Ridge National Lab) to visualize engine simulations have proven a valuable tool for engine designers and future computational scientists and engineers.

The experience gained by our team in computational engine simulations is used in several courses, including Simulation and Modeling, Supercomputing Applications, Deterministic Dynamical Systems, and Scientific Visualization. The engine combustion code to be experimented with in these classes is called KIVA [4]. It has been modified and enhanced by many research groups and these modifications have been presented and examined at the KIVA International Users Group meetings during the Society of Automotive Engineers Convention. Versions of KIVA have been around for years since 1985, but its full potential to teach students about engines has never been utilized due to limited computer resources. Yet, such tools are very common in industry and students both at engineering departments and at computational science programs should be given the opportunity to learn industrial engine design through combustion simulations and engine visualization techniques.

The parallelization of KIVA [7] is also subject of a course within our computational science and engineering curriculum as it teaches about domain decomposition, computation/communication overlap, and effective programming. The availability of the scalable KIVA-3 presents great potential for a computational scientist and engineer to learn about industrial requirements of engines in a class environment by doing different engine simulations for sensitivity analysis. Post-processing is also directly applicable to a course in scientific visualization.

Computational Chemistry

Molecular simulation is an active area of research and scientific application that requires computational techniques used in many areas of study. Since chemical reactions (dynamical processes such as phase separation, crack propagation in brittle materials, and so on; fluid flow; and other phenomena) cannot be directly observed at the experimental level, molecular simulations of these processes and visualization of the results can provide a wealth of valuable information. In addition to dynamical processes, many materials properties can be calculated from molecular simulation data. All of these methods have found uses in basic research in chemistry, biology,

and other fields and in important applications such as rational drug design. In order to increase the range of applicability of molecular modeling techniques, advances in both computational power and algorithms are required and remain an active area of research. Classical and quantum mechanical molecular simulation techniques provide ample opportunity for illustrating computational methods such as 1) numerical solution of deterministic partial differential equations, predictor-correctors, symplectic integrators, 2) ensemble methods, thermodynamic averages, fluctuations, correlation functions, transport coefficients, 3) constraint dynamics, 4) optimization, conjugate gradient and other commonly used methods for molecular mechanics, 5) random number generation and Monte Carlo techniques: random walks, importance sampling, solution of differential equations with diffusion terms, 6) specialized load balancing and domain decomposition strategies.

Our computational chemistry group has extensive background in chemical physics, numerical methods, and molecular simulation methods. In the past five years, they have developed several generalizable, robust, specially portable algorithms for molecular dynamics, molecular mechanics, and quantum Monte Carlo simulations [10-15]. Applications of these new methods include polymer science; nanotechnology; molecular fluid flow; and classical-quantum correspondence in many-body systems. The computational chemistry research interests in our program also include few-body quantum mechanical calculations and the quantum theory of angular momentum.

Molecular simulation is eminently suited for classroom presentation, both at the graduate and undergraduate levels, and it forms an important part of the computational science curriculum. In Simulation and Modeling course students gain the basic knowledge for writing complete simulation programs and for analyzing the results. In Deterministic Dynamical Systems, in Stochastic Dynamical Systems, and in Supercomputing and Applications courses, students learn in more detail topics such as optimization, numerical solution of partial differential equations, Monte Carlo methods, random number generation, use of physical principles for code validation, parallel programming strategies, benchmarking, and specialized load balancing and domain decomposition techniques. These skills are commonly used in climate research, automobile design, environmental research, and a host of other important scientific and engineering applications.

Weather Modeling

Today, the science of weather forecasting relies heavily on numerical weather predictions. Every day, supercomputers at the National Centers for Environmental Prediction (NCEP) run at least five different NWP models; each model with its own unique set of equations representing atmospheric dynamics. In tandem with the increase in computational power, the level of detail and the sophistication of the parameterization of physical processes in these NWP models have increased. However, the translation of these advances into improved weather forecasts has been slow. Weather forecasters, trained professionals who interpret NWP model output, generate public forecasts after comparing an increasingly complex set of models and reconciling discrepancies between them. Without the benefit of a fully integrated

visualization tool, these forecasters cannot realize the full potential of the NWP system.

Virtually all software tools used by weather forecasters to display NWP model outputs generate two- dimensional maps, with some capacity to overlay more than one field. These tools do not fully utilize the wealth of information provided by NWP models; there is tremendous room for improvement. In fact, few, if any, operational forecast offices use three- or four-dimensional visualization tools. The general public find forecast maps difficult to comprehend or of limited use. In both forecasting and public presentation, present day electronic media provide limitless opportunities.

To learn weather forecasting, students majoring in Meteorology at SUNY Brockport use outputs from the NWP models generated at NCEP. A critical step in the forecast process is the ability of forecasters to interpret NWP model outputs in an accurate and timely manner. Roebber and Bosart [16-17] have demonstrated that experience is an essential element in the forecast process, and that human judgement allows skilled forecasters to issue forecasts that are superior to raw NWP model outputs. After proper training, students in two of the Earth Science courses (ESC 312 Weather Forecasting and ESC 490 Weather Briefing) are placed on a rotation to forecast different scenarios with and without the benefit of the four-dimensional data visualization. The overall and individual student forecasting performance is tracked, and the degree to which forecast skills benefited from this new technology is assessed. Every semester, anywhere from 150 to 200 students, mostly college freshmen, register for three of the introductory level courses in the Earth Science department (ESC 102 Elements of Geography, ESC 210 Weather I, and ESC 211 Weather II).

References

1. O. Yasar, *et al.,* “A New Perspective on Computational Science Education,” *IEEE Computing in Science and Engineering,* Vol. 2, No. 5, 2000.

2. O. Yasar, “Computational Science Program at SUNY Brockport,” *Proceedings of First SIAM Conference on Computational Science and Engineering,* September 21-24 2000, Washington, D.C.

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, "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," Technical Report, LA-11560-MS, Los Alamos National Laboratory (1989).

5. O. Yasar, “A New Spark Ignition Model for Engine Combustion Simulations,” *Parallel Computing,* Vol. 27, No. 1-2, 2001.

6. O. Yasar, "A Scalable Algorithm for Chemically Reactive Flows," *Computers and Mathematics*. Vol. 35, No. 7, 1998

7. O. Yasar and C. Rutland, "Parallelization of KIVA-II on the iPSC/860 Supercomputer," in *Parallel Computational Fluid Dynamics*, Editor: R. B. Pelz, A. Ecer, and J. Hauser, North Holland (1993), p. 419-425.

8. O. Yasar and G. A. Moses, "Explicit Adaptive Grid Radiation Magnetohydrodynamics," *J. Computational Physics*, 100, 38 (1992).

9. Y. Deng, R. A. McCoy, R. B. Marr, R. F. Peierls, and O. Yasar, "Molecular Dynamics on Distributed-Memory MIMD Computers with Load Balancing," *Applied Math Letters*, 8 (3), 37-41 (1995)

10. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Automatic differentiation as a tool for molecular dynamics simulations", *Computational Polymer Science* **4**, 75-78 (1994).

11. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Dynamics of a laser driven molecular motor", *Nanotechnology* **6**, 52-63 (1995).

12. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "The dynamics of molecular bearings", *Nanotechnology* **6**, 64-74 (1995).

13. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Molecular dynamics treatment of torsional interactions accompanied by dissociation", *Macromolecular Theory and Simulations* **4**, 909-920 (1995).

14. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Computation of internal coordinates, derivatives, and gradient expressions: torsion and improper torsion," *Journal of Computational Chemistry,* Vol. 21, 553-561 (2000)

15. Kazuhiko Fukui, Bobby G. Sumpter, Donald W. Noid, Chao Yang, and Robert Tuzun, "Analysis of eigenvalues and eigenvectors of polymer particles: random normal modes," *Computational and Theoretical Polymer Science,* Vol. 11, 191-196 (2001).

16. P. J. Roebber and L. F. Bosart, The contributions of education and experience to forecast skill. Weather and Forecasting, 11, 21-40, 1996.

17. P. J. Roebber and L. F. Bosart, The complex relationship between forecast skill and forecast value: A real-world analysis. Weather and Forecasting, 11, 544 – 559, 1996.