Numerical Computation and Computer Science

John Guckenheimer
President, SIAM

October 8, 1999

The continuing explosion of the computer industry raises new challenging problems daily. At the same time, many old challenging problems remain important but unsolved. It is all too easy to pretend that this is not so, in our haste to go on to larger computational tasks. We assume that because a problem is familiar that it is no longer a problem. I hear this attitude today in discussions of many numerical questions. We are conducting ever more complex computations built upon the assumption that underlying numerical methods are mature and reliable. Often, these assumptions are not valid.

Many computer scientists have shifted their attention to new uses of computers whose explosive growth has caught us by surprise during the past few years. Information technologies are seen as a broader arena in which computer science is central. I have a poor understanding of the forces that have led to the current state of affairs, but it seems evident that critical research areas for computing are being neglected or de-emphasized. I discuss this issue in terms of simulation and modeling, relying upon both my research expertise and my perspective as President of SIAM. SIAM is an organization whose mission lies at the heart of computational science.

Few academic departments regard the numerical aspects of computing as part of their core activities. Oversimplifying attitudes, many mathematicians see computation as inferior to proving theorems, many computer scientists focus their attention on linguistic issues or information technologies, and computational scientists from other disciplines believe that Moore's law will solve their remaining problems. We seem to overlook the continuing contributions that mathematics makes to improvements in computing. When we bundle existing algorithms into libraries and wrap them into packages to facilitate easy use, we create de facto standards that make it easy to ignore numerical analysis. We regard the existing base as static and invest in the development of problem solving environments and high level languages. This is needed, but we also need to maintain our investment in continuing research on algorithms themselves.

I maintain that the relative neglect of mathematical aspects of computation weakens the foundations of our growing computational enterprise, foundations that desperately need to be strengthened to support computation at the interface between computers and the physical world. Computers are far more than passive devices for information storage and retrieval. Digital technologies make computers integral components of endeavors such as fly by wire flight control, programmed stock trading and art to part manufacturing. Our success in these endeavors is dependent upon efficient and accurate computations. Algorithmic problems arise repeatedly as we attempt new applications. There are frequently common features in these computational problems, and research directed at finding general solutions to classes of questions has been a critical part of the increased use of digital technologies.

To make this point more concretely, I shall describe some of the current issues that are related to simulation, an area in which I have a personal interest. We seek to implement ambitious simulations of global climate change, entire automobiles, electric power networks, military battles, stock markets and the human brain. Moreover, we want high fidelity in those simulations so that they can be used in planning public policy, industrial design, enhancing reliability of infrastructure systems, personnel training, improving investment earnings and the verification of scientific theories. Current methodologies for these simulation tasks are not up to doing what we want. I describe a few areas that are barriers to further success:

- Interacting subsystems
- Multiple scales
- The effects of uncertainty

Computer simulations are organized in terms of systems constructed from interacting subsystems and components. As the complexity of the modeled object grows, we introduce hierarchical structure to the simulated system for the same reasons that we structure computer code. We have limited capacity to understand how large numbers of interacting components give rise to system behavior, especially when the components differ from one another. We deal with this problem by constraining the interactions, encapsulating subsystems as "input-output" devices whose internal dynamics do not contribute to system behavior. The success of this effort depends upon the extent to which these interactions are (in)sensitive to details within the components. For artificial systems, functional decomposition of systems into subsystems is a design principle. The spectacular success of VLSI technology rests in part on our ability to constrain the interactions among circuit components. We envision enormous design problems when the feature sizes of circuits decrease to the point that additional nonlinear interactions among nearby components become important. In other physical domains, stronger interactions among components is unavoidable but we still strive to produce reliable machinery. Reduced order modeling, aggregation and multiscale analysis are terms that we use to describe simulation technology for simplifying the complexity of subsystems while retaining their fidelity. Computer implementation of these methods is still in a primitive state, but mathematical theories provide a conceptual substrate for further progress.

Decomposition of natural systems into different "scales" is one of the central tasks in producing high fidelity simulations. We seek to understand how macroscopic behavior results from physical laws that operate on smaller scales. For example, we would like to understand fracture in terms of atomistic properties of materials. Reducing all complex phenomena to atomic interactions is clearly a hopeless task. It is preposterous to model the effects of global climate change on natural populations and agriculture on an atomic scale. Doing it at all is problematic. Climate modeling itself is a horrendously difficult task, and natural variability precludes the formulation of precise laws of population dynamics. As we increase the resolution of computer models, the number of model parameters that must be measured or estimated grows. In most circumstances such measurements are too expensive or simply impossible. We don't even know what the most important variables are in determining the response of a population to shifts in climate. To obtain useful information from computer simulation of such systems, the computations must be robust to uncertainty within the model. It would be prudent to have systematic ways of evaluating the effects of uncertainty in model components upon system behavior. Such methods hardly exist at this time.

Tackling the issues of scale, hierarchy and uncertainty in simulations requires the creation of new theories that go well beyond existing ones. We also need significant improvements in more mature areas of numerical analysis. The cornerstone of simulation technology for continuous systems is the numerical integration of systems of ordinary differential equations. These algorithms are very robust, but they do not encompass the full range of systems that we seek to simulate. For example, models of control systems, electrical circuits and mechanical mechanisms are often given as differential-algebraic equations in which constraint equations are added to systems of differential equations. Unlike differential equations, DAE's are not always self consistent or solvable. Symbolic methods are applied to reduce DAE's to differential equations when this can be done explicitly, but that is not always possible. We lack algorithms that recognize when solutions to these systems break down and give diagnostic data in that event. In addition to DAE's, other generalizations of systems of ordinary differential equations are important for models. Digital control systems lead to hybrid models that combine discrete and continuous components. This has been an active research area of several years, but problem solving environments that facilitate the design of hybrid systems are still rudimentary. Establishing an architecture that allows for the efficient description of hybrid systems and supports their simulation remains a research area that requires the joint efforts of computer scientists, mathematicians and engineers.

Beyond simulation, the analysis of models requires additional types of algorithms. Especially with natural systems, only partial information about system parameters can be obtained from measurements. Computational neuroscience attempts to model the nature of learning, memory and computation in the human brain. We view the brain as a complex electrical circuit, but the complicated architecture of synaptic connections among neurons, the spatial distribution of membrane channels within the branched structure of neurons and the highly nonlinear electrical properties of the neurons themselves obscure which components are critical to brain function. Our limited measurement capability makes identification of parameters for these models a task that must be guided by comparisons of model output with experimental data. The amount of computation needed to exhaustively explore parameter spaces grows exponentially with the number of parameters. Thus, we can only vary a small number of parameters independently of each other in simulation studies. This greatly complicates the task of fitting models to data. Indeed, the problems of fitting model parameters inhibit the creation of high fidelity models. As described above, increasing model resolution to include smaller scales in a problem may increase the number of parameters that must be determined to the point that the computations to do so are not feasible. Bifurcation theory provides a framework for direct determination of information about how system behavior changes qualitatively with parameter variations. Implementation of algorithms based on this theory is a step towards computing parameter ranges that produce desired behavior.

¿From my perspective as a dynamicist, limitations of computer simulation technology are evident. Similar shortcomings occur in many other areas of numerical computation. Let me cite an example that illustrates how even algorithms for classical problems need improvements. Roughly a decade ago mathematicians solved a problem that had been popularized by Marc Kac as "can you hear the shape of a drum?" The mathematical problem is to relate the spectrum of eignevalues for the Laplace operator with Dirichlet boundary conditions to the geometry of a planar domain. Classical results give formulas for geometric quantities such as the area of the domain and the length of its boundary in terms of the spectrum. Gordon, Webb and Wolpert gave the first examples of non-congruent planar domains with precisely the same spectrum. Examples include polygonal domains with fewer than a dozen edges. Now the solution of the Laplace equation is one of the most intensely studied problems in numerical analysis. Yet when Toby Driscoll (then a graduate student in applied math) attempted to produce accurate solutions of the eigenfunctions for two such domains, he found that the standard programs for solving the Laplace equation were not up to the task. The concave corners of the domains cause problems for these algorithms. Toby produced new algorithms based on conformal mappings of the domains. Pictures of two of these eigenfunctions are displayed on the home page of the Cornell Center for Applied Mathematics (http://cam.cornell.edu). The point here is that we can hardly take for granted our ability to perform accurate computations for similar, but more complex, industrial problems.

Ironically, as numerical analysis is applied to larger and more complex problems, non-numerical issues play a larger role. Mesh generation is an excellent example of this phenomenon. Solving current problems in structural mechanics or fluid dynamics with finite difference or finite element methods depends upon constructing high quality meshes of surfaces and volumes. Geometric design and constructing these meshes is typically much more time consuming than the simulations that performed with them. Developing data structures that capture the mathematical properties of these complex objects while still supporting efficient computation is important. Another non-numerical issue that has come to be particularly important is the data flow within our computers and algorithms. Moving data within computers has become the limiting factor for the speed of computer arithmetic. Particularly in parallel computers and distributed computing environments, we have complex memory hierarchies whose characteristics determine the efficiency of algorithms. Enormous effort has gone into ordering the arithmetic computations of fundamental linear algebra algorithms so that arithmetic processors do not sit idle most of the time waiting for data to arrive.

The continuing importance of numerical computation as a research area and critical technology is evident. I maintain that these issues go even deeper scientifically. The quantitative predictions of most modern theories can only be derived from extensive computations. In the twenty-first century, I believe that computer models will become a dominant mode of expression of scientific theories. For this to happen, we must be intelligent practioners of numerical computation. This is the driving force behind emerging academic programs in computational science. Computer science has not embraced these efforts as a welcome enlargement of the the domain of computer science. The core of computer science has come to be defined more narrowly with the result that numerical analysis is a shrinking part of computer science. I believe that this will harm computer science in the long run. My attitude towards these issues is shaped by my career as a mathematician. Mathematics has made a tradition of casting off applied endeavors when their practitioners became impatient with the constraints imposed by formal mathematical rigor. The result has been a continuing gap between mathematics curricula and the use of mathematics outside mathematics departments. Mathematics and mathematics education is enormously enriched when strong connections to the rest of science and engineering are maintained. I think that this is true of computer science as well. Numerical computation is an essential component of those connections. We must enhance the role of numerical computation as part of the core of computer science to ensure the vitality of computer science as part of science and engineering.

What do we need to do differently to foster timely advances in numerical computation? The issues cut across mathematics, computer science and other disciplines so thoroughly that extensive cooperation will be needed to achieve our goals. This is simply hard. Government and industry seem to be ahead of universities in this regard. However, most companies do not have the expertise and do not want the responsibility for developing their own software tools. The commercial interests of software vendors are problematic because the development of a better algorithmic infrastructure requires significant investments to update current products. Thus, there is a continuing need for "precompetitive" research in this area. The Federal focus upon multidisciplinary research initiatives the past few years recognizes that we need incentives to work together, but I think that universities deserve mixed marks at best for the successful conduct of such projects. Peer evaluations and rewards remain the purview of departments that frequently take a narrow view of their discipline. Consequently, we make it difficult for individuals to make sustained commitment to projects that require them to work across disciplinary boundaries. We can change this by broadening the scope of departments, giving them overlapping spheres of interest and creating mechanisms for them to work smoothly together. In particular, I think that many departments, including those in computer science and mathematics, need to take responsibility for computational science. They need faculty who are knowledgable and interested in computational technologies. They need curricula that teach students to become skilled and thoughtful users of these technologies in a world that becomes increasingly dependent upon enormously complex computational systems.