Validating the Need to Validate Code

The article "Computational Science Demands a New Paradigm" by Douglass Post and Lawrence Votta (PHYSICS TODAY, January 2005, page 35) makes some good points about the pitfalls of believing computed physics. The authors propose several criteria for verification and validation of large computational models for making public-policy decisions. Among other examples, they mention climate-change calculations.

The article's figure 5 shows several famous bridges, including as the third example the infamous "Galloping Gertie," the Tacoma Narrows Bridge in Washington State, which collapsed in 1940. In that case, previous designs were pushed too far, with just one little thing forgotten: resonance! The film of that collapse is still often used in physics classes to dramatize the importance of resonance. Post and Votta "assert that computational science is currently in the midst of" the stage where computing power makes it possible to outrun good engineering judgment.

The Kyoto Protocol to curtail CO_2 emissions was based on a global circulation model from 1994, now a full decade old. It left out the importance of clouds, because they were just too difficult to model. That model was never even successful in accurately describing the past: It violates the first validation criterion given by Post and Votta. Nevertheless, international policymakers have not attended to validation criteria, have never doubted the truth of the model, and have gathered momentum toward implementing the Kyoto Protocol. Consequently, large economic impact is associated with those computational results.

The Galloping Gertie of environmental science is the Kyoto Protocol. Unless computational scientists learn from its shortcomings, it will discredit future attempts to predict climate change.

Thomas P. Sheahen
(tsheahen@alum.mit.edu)
Deer Park, Maryland

The problems that concern Post and Votta were encountered decades ago in commercial software development and have been solved.

A familiar result is Microsoft's Windows operating system, a product of more than 20 years' work by thousands of people. It now consists of more than 200 million lines of source code. At a smaller but no less impressive scale are numerical analysis programs from such companies as MSC, ABAQUS, Mentor, ANSYS, Dassault, and ALGOR. These firms' codes are commonly used to design bridges, automobiles, networks, buildings, and airplanes. Their development has presented exactly the set of issues Post and Votta describe.

Commercial efforts revolve around solid discipline and management. Perhaps for computational scientists that would qualify as a new paradigm; for software engineers it has become standard practice.

Craig Bolon
(cbolon@verizon.net)
Planwright Systems Corporation
Brookline, Massachusetts

We who work on the system side of high-performance computing development generally think our job is done when the first two challenges that Post and Votta mention, performance and programming, are addressed. I disagree that those two challenges are less urgent than the prediction challenge. I've heard too many complaints about the small percentage of peak that is reached and the dismal state of programming tools. However, from an application viewpoint, I can see that prediction is a formidable challenge.

The article reminds me of a paper I reviewed years ago for *IEEE Computational Science and Engineering*. The author compared several seismic processing packages, and each claimed to find oil in a different spot. Apparently, the results were often wrong. Nevertheless, users of the codes blindly trusted them and spent huge investments drilling for oil.

I wonder if the great importance of verification and validation could explain why some industries have not jumped more quickly into computational engineering. For example, one aircraft manufacturer is reportedly going back to using real-world wind tunnels for part of its development stage. It would be interesting to survey computational engineering companies about their experience with early computational technology.

One issue in successful validation is the availability of data for comparison. For example, to validate that an ocean simulator predicts correctly, one would need to place a huge number of sensors, which is probably impractical. So even if more project time is spent validating, I wonder how far the validation could go.

Rudolf Eigenmann
(eigenman@purdue.edu)
Purdue University
West Lafayette, Indiana

I am also concerned with the issues raised by Douglass Post and Lawrence Votta. A great deal of computational physics involves fitting parameters—for example, some turbulence constants and grid design parameters. Fifty physics models put together, each with a couple of free parameters, could yield 100 parameters that can be used to fit the code to whatever verification and validation tests it needs to pass. Yet we know that an interpolation function fitted to a bunch of points can be wildly wrong between them. This concerns me, before we even start extrapolating the code to regions where its performance is completely untested.

The problem the article highlighted is serious. People who know the issues involved in computational physics are essential. Unfortunately, these days universities turn out users who employ codes as black boxes but do not understand what they do or when their results can be trusted. Moreover, analysis codes are often incorporated into multidisciplinary design optimization algorithms—for example, to design a better aircraft—but the optimization process drives the codes beyond any reasonable applicability. Expert guidance is usually needed to stay within implicit constraints of analysis codes.

Letters and opinions are encouraged and should be sent to Letters, PHYSICS TODAY, American Center for Physics, One Physics Ellipse, College Park, MD 20740-3842 or by e-mail to ptletter@aip.org (using your surname as "Subject"). Please include your affiliation, mailing address, and daytime phone number. We reserve the right to edit submissions.

Rigorous component testing, although necessary, is not sufficient. Software components can be combined, but their combination could be wrong even though the components test well individually. A combination that is insensitive to minor component errors could still give invalid results. Each component has an unstated region of applicability that is often horribly complicated to describe, yet the combination could unexpectedly exceed individual component limits.

Josip Loncaric
(jl-icase@comcast.net)
Los Alamos, New Mexico

Douglass Post and Lawrence Votta's article admirably stresses the need for validation of computation. If I understand their main point correctly, it's that computation is science, not mathematics.

The article motivates me to contrast two fields I work between that use very different paradigms to describe the electrodiffusion of charge. In computational electronics,[1,2] the electric field is traditionally calculated by solving Poisson's equation with far-field boundary conditions but at relatively low resolution. Poisson is solved anew whenever charges move. Computational chemistry, starting more or less with computer simulation of fluids[3] and computational biology,[4] computes the electric field at high resolution and does not deal clearly with far-field boundary conditions.

Electrodiffusion, which has been at the center of electro- and physical chemistry since Michael Faraday's time, is also at the center of electronics, where it describes the movement of charge in semiconductors and most of our digital devices. Electrodiffusion is no less important in biophysics, where it is responsible for the electrical properties of cells and tissues, and controls many biological functions.

It seems to me that these different treatments of similar physics are distinct and unlikely to be equally precise under all conditions. I hope the article helps motivate workers in each tradition to discuss other treatments beyond their own and try to understand the differences. I hope such workers can determine the conditions under which each treatment is accurate. That way we may learn to use each computational tradition of electrodiffusion only in appropriate situations.

References

1. C. Jacoboni, P. Lugli, *The Monte Carlo Method for Semiconductor Device Simulation*, Springer-Verlag, New York (1989).
2. S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer-Verlag, New York (1984).
3. M. P. Allen, D. J. Tildesley, *Computer Simulation of Liquids*, Oxford U. Press, New York (1987).
4. T. Schlick, *Molecular Modeling and Simulation*, Springer-Verlag, New York (2002).

Bob Eisenberg
(beisenbe@rush.edu)
Rush University Medical Center
Chicago, Illinois

Post and Votta correctly point out the need for validation of software programs. Validation is often thought to mean the absolute prediction of measurements. However, new software often replicates measurement trends accurately long before predicting absolute values well. When time is critical, trends are frequently sufficient to guide new designs or predict outcomes. Sometimes the technology is no longer relevant once the model software finally replicates absolute measurements.

R. Casanova Alig
(alig@mailaps.org)
Princeton Junction, New Jersey