

Garbage Out? Verification of Reliable Computations

Wendy Thomas, Associate Professor, Bioengineering, University of Washington.

What is Verification?

When we or our students use a computer to solve a mathematical problem in our class or research, how do we know if the answer we get is correct? If we make a wrong assumption, or type the wrong numbers or equations, we have “Garbage in = Garbage out” regardless of the sophistication of the computer software and hardware we are using. Conversely, if we set up the problem flawlessly, but chose an inappropriate algorithm to solve the equations, we can have a solution that is dominated by numeric artifact. In this essay, I ask how we can rigorously teach students to check the accuracy of their computational work.

I teach computational differential equation modeling to upperclassmen and graduate students at the University of Washington, using MATLAB for ordinary differential equation modeling and COMSOL for partial differential equation modeling. However, most of the points made in this essay can carry over to many other types of modeling and all computational platforms. In my class, I distinguish between two concerns. The first is *validation* that the assumptions we use to write mathematical equations are appropriate for the situation we seek to model, which usually involves a comparison of the computational results to data. The second, and the topic of this essay, is *verification* that the computations are mathematically correct, which usually involves a comparison to a known result for one or more simplified versions of the model.

Motivating Students to Learn Verification

Rigorous verification can be tedious, so I have found it helpful to motivate students to learn and perform verification by explaining and illustrating the need. To explain the need, I compare verification to the need to perform controls in experiments; in both cases, we seek to determine whether our method provides the expected outcome in a closely related situation in which the behavior is already known. Second, I also tell them how they might know if they performed a computation correctly. Most students answer honestly that they would compare their solutions to those of their peers, because we usually teach the basic skills by having all students solve the same homework or lab problems. I then explain that the problem with this method is that employers rarely pay two or more people to solve the same problem, and in addition, knowing that two answers are different does not tell the student which, if either, is correct. Finally, I explain that practicing rigorous verification helps them develop mathematical intuition that will allow them effortlessly to identify many incorrect solutions.

To illustrate the need, I have students practice their verification skills by finding an error in a published journal article that relates to the topic of the class. Depending on the academic experience of the students, I might summarize the problem and erroneous answer presented in the paper, or I might assign the whole article to read. I also assign an open-ended final project that addresses a real world problem unique to each person or team, and requires verification, as explicitly outlined in the grading rubric. The students almost always encounter one or more problems as they build their models, which they are able to identify using the verification tools.

Example of Teaching Verification: Differential Equations

When we use computational solutions of ordinary, partial, or even stochastic differential equations to solve problems, we should verify our work at two stages. First, we need to ask whether the differential equations, or *model*, is a mathematically correct representation of the assumptions used to build the equations. Second, we need to ask if the *solution* we obtained is the correct solution to the differential equations and the assumptions. As noted above, we can't know for sure that either model or solution is correct, but we can determine that it passes a number of tests that would have caught many mistakes, thereby increasing the *level of certainty* that the model and equation are correct.

To test for model correctness, I provide three types of tools or tests. First, I recommend testing for model completeness, by asking if 1) there is one differential equation for each dependent variable, 2) there is one initial or boundary condition for each first-order equation, or two for a second-order equation, 3) all parameters are defined, and 4) all the relevant assumptions were used to build the model. Second, I recommend a dimensional analysis, by asking whether all terms in the model have the same dimensions. Third, I recommend a sign analysis by testing whether the sign of the effect of each variable on the rate of change of the others make intuitive sense given the physics. Finally, of course, one must check that the equations written into the software are identical to those developed on paper. These tests will catch most mistakes, but some mistakes may not be caught until one verifies the solution.

To test for solution correctness, I recommend two simple and general methods. First, one can check for numerical artifact by ensuring that the solution is essentially the same when solved by two different algorithms and/or levels of tolerance. I clarify that "essentially the same" means that any differences between the two solutions are minor and do not affect the conclusions drawn from the solutions. Second, one can simplify the problem to one that can be solved analytically without too much trouble, or for which the solution is intuitively obvious from the physics, and then compare the numeric to the analytic or obvious solution. Examples of simplifications include finding the steady state solution or solving for a subset of the model after effectively removing other parts of the model by setting a parameter to zero. Both of these methods involve comparing solutions obtained in two different ways.

Teaching Verification in Other Settings

In my class, relatively advanced students learn to use verification to ensure that relatively complicated computations are likely correct. However, verification is just as important and useful for freshmen and sophomores first learning to use computation in sciences. For example, a student may simply be asked to solve a differential calculus problem (in a math or a science class) using both analytic and numeric tools. The student can compare the answer obtained by finding the analytic expression for the derivative and then plugging in parameter values, with the answer obtained by directly calculating the computational derivative in MATLAB. As another example, a student using computation to analyze a data set may be provided a practice data set with a known outcome, or may even be guided to create or identify such a data set. These exercises should build student confidence in computational skills, and build good habits, by decreasing student dependence on peer comparison as a verification tool.