Reflections on Computation in Scientific Research: Reproducibility, Lock-in, and Deductive Systems

Victoria Stodden, Yale Law School

The widespread use of computation in science is inevitable, precisely because it is so clearly useful in accelerating research. Even what might be considered some of the least technical edges of the scholarly spectrum are welcoming computational methods. Did Shakespeare really write all the plays attributed to him? The Wordhoard project at Northwestern Universityinvestigates whether word distributions by play are significantly different. Two researchershave used signal processing to "excavate" a Mayan pyramid without physically disturbing it, developing techniques that could preserve artifact features such as the moisture sensitive paint on the Terracotta warriors. Multiscale analysis can reveal fraud by detecting brush stroke and style distinctions between different painters, such as von Gogh, Monet, and others. Prominent and influential new fields, such as bioinformatics and computational statistics, have largely been created as a result of dataset production and the increasingly large range of questions that come within the reach of science, given the computer.

In this note I sketch some preliminary thoughts on potential drawbacks to the pervasive use of the computer in scientific research, expanding on my recent blogpost. Here are three: the increase in opacity of scientific results as traditional publication methods don't incorporate reproducibility; an increased difficulty in assimilating superior solutions to problems; and an epistemological shift as mathematical theorems and properties are established by combinatorial grid search and simulation, rather than deductive connection.

When computational results are communicated it is not standard practice to make the data and code available, rendering it all but impossible to verify the results. moving today's practice of computational research away from the scientific method which since the 1660's has held reproducibility to be a component of scientific research. Generation and re-generation of results often requires a detailed knowledge of parameter settings and software invocation sequences. Without a clear description it can be next to impossible, even for the original scientist, to try the published methodology in a new setting or on a new dataset, or even verify the published results. This is discussed in Reproducible Research in Computational Harmonic Analysis and Enabling Reproducible Research: Licensing for Scientific Innovation, for example. Possibly the most serious consequence

Implementing methodological improvements

With increasing computation more of our scientific ideas are encapsulated in code form. Software is brittle, it breaks before it bends. Computing hardware has grown steadily in capability, speed, reliability, and capacity, but as Jaron Lanier describes in his essay on The Edge, trends in software are "a macabre parody of Moore's Law" and the "moment programs grow beyond smallness, their brittleness becomes the most prominent feature, and software engineering becomes Sisyphean." My concern is that as ideas become increasingly manifest as code, with all the scientific advancement that can imply, it becomes more difficult to adapt, modify, and change the underlying scientific approaches. Perhaps we become, as scientists, more easily locked into particular methods for solving scientific questions and particular ways of thinking.

Examining how ideas change in scientific thinking isn't new. Thomas Kuhn for example caused a revolution in how scientific progress is understood with his well-known 1962 book The Structure of Scientific Revolutions. The notion of technological lock-in isn't new either, see for example Paul David's examination of how we ended up with the non-optimal QWERTY keyboard ("Clio and the Economics of QWERTY," AER, 75(2), 1985) or Brian Arthur's "Competing Technologies and Lock-in by Historical Events: The Dynamics of Allocation Under Increasing Returns" (Economic Journal, 99, 1989). What happens when an approach to solving a problem is encoded in software and becomes a standard tool? Many such tools exist, and are vital to research - Andrej Sali's highly regarded lab at UCSF provides a long list of the codes they use, and the statistical packages in the widely used language R provide another example. David Donoho, professor of statistics at Stanford and my PhD advisor, laments the now widespread use of test cases he released online to illustrate his methods for particular types of data, "I have seen numerous papers and conference presentations referring to "Blocks," "Bumps," "HeaviSine," and "Doppler" as standards of a sort (this is a practice I object to but am powerless to stop; I wish people would develop new test cases which are more appropriate to illustrate the methodology they are developing)." Code and ideas should be reused and built upon, but are we raising the bar to improving solutions? In fact today, perhaps counterintuitively, it's hardware that is routinely upgraded and replaced, not the seemingly ephemeral software.

Math on the computer, not in the mind

By replacing analytical mathematical proofs with computational demonstrations, I wonder whether there is a loss in cognitive tractability and whether it matters. If a proof isn't clear analytically, it's becoming increasingly possible to simulate the properties computationally and establish the result that way, at least over the range of the similation. In 2005 Robert MacPherson of the Institute for Advanced Studiesin Princeton gave a presentation at a meeting the Royal Society discussing 'The Nature of Mathematical Proof'that began,

> In 1609, Kepler made a beautiful conjecture about spheres in space. It was one of the oldest unsolved problems in mathematics. In 1998, Tom Hales produced a brilliant computer-assisted proof of the Kepler conjecture. By now, the theoretical

> part of Hales' proof has been refereed as usual mathematical papers are, but the parts involving the computer have resisted all efforts at checking by humans. Should we think of the Kepler conjecture proved?

My focus is not on the correctness of the proof, which of course is crucially important, but let's assume we know that the proof is in fact correct. The issue is that the computational proof, the deduction, isn't necessarily of the form that a human can understand. As mathematician Thomas Tymoczko wrote in 1979,

> A proof is a construction that can be looked over, reviewed, verified by a rational agent. We often say that a proof must be perspicuous, or capable of being checked by hand. It is an exhibition, a derivation of the conclusion, and it needs nothing outside itself to be convincing. The Mathematician surveys the proof in its entirety and thereby comes to know the conclusion.

A deductive logical system is intended to be tractable to sufficiently skilled minds, and this human surveying can be impossible when proofs are computational. We lose an ability to cognitively understand the deductive system, even if this ability is only possessed by a small number of people with the requisite mathematical skill.

A well-known mathematician complained to me recently that people using computers for proofs ask him why we need an elegant formula, when we can simply use the computer to come up with the numbers. A deductive system has by definition a coherence and is an edifice analytical mathematical proof has been building to be understood. The future benefits from a theorem or formula may be exchanged for easier solutions to problems faced today.

To come back to the point about the correctness of computational proofs it is clear that concerns about cognitive tractability in deductive systems cannot be addressed without an ability to inspect the code. Mathematics has developed strict standards regarding proofs and evidence required to establish a claim as true, and analagous standards must start with open code.