

Robust Simulations*
Ryan Muldoon^{†‡}

[†] rmuldoon@sas.upenn.edu

[‡]The author would like to thank Cristina Bicchieri, Michelle Foa, Paul Humphreys and Michael Weisberg for their helpful comments and suggestions.

Abstract

As scientists begin to study increasingly complex questions, many have turned to computer simulation to assist in their inquiry. This methodology has been challenged both by analytic modelers and experimentalists. A primary objection of analytic modelers is that simulations are simply too complicated to perform model verification. From the experimentalist perspective it is that there is no means to demonstrate the reality of simulation. The aim of this paper is to consider objections from both of these perspectives, and argue that a proper understanding and application of robustness analysis is able to resolve them.

1. Introduction

Analytic models, by which I mean systems of differential or difference equations that serve as representations, are useful precisely because they are simple. Studying them enhances our understanding of the more complicated phenomena they are designed to represent. But as the phenomena scientists study increase in complexity, analytic models may prove to be too simple for some investigations. It is in these situations that computer simulations are being deployed. Simulations are also beginning to be deployed in what has traditionally been experimental work. Besides being utilized to investigate that for which we have no ability to perform experiments, simulations can evaluate the design of possible experiments, assuring that potentially expensive projects will test what they are designed to test. Simulations have a role in bridging the growing divide between our experimental practice and our interest in deriving simple models to aid our understanding.

The increased use of computer simulation in the sciences has raised a number of methodological questions which stem from its differences from analytic modeling and experimentation, respectively. Analytic modeling and experimentation both have well-established standards of rigor, and there is justifiable concern that simulation does not meet either standard. To see why this is the case, let us first consider specific worries from the perspective of analytic modelers and then from the perspective of experimentalists.

Analytic modelers may worry that there is not the same kind of transparency in simulation as there is in analytic modeling, and this manifests itself in numerous ways. Unlike analytic models, simulation work does not generally result in a formal proof of some phenomena, but when it does, the proofs can be of a kind that cannot be processed by a person unaided by a computer. Even more importantly, the implementation details of a particular simulation instance can create new kinds of errors, such as boundary condition bias, that are not present in abstract analysis. The simulation environment itself is a potential source of errors, none of which are in the simulator's control. Though

there are other specific concerns that analytic modelers could raise, I consider the objections relating to a lack of direct inspection – an inability to check each step of the simulation – to be the most damaging claims against the use of computer simulations.

Experimentalists may worry that computer simulation is not grounded in real physical processes. Because of this lack of grounding, any given simulation is very likely not describing the real world. Simulations are being employed to predict future phenomena, such as changes in global climate, but if they are not accurately representing the world – if their results are not in some sense real – then it is not clear that they should have a role in the scientific enterprise. For simulations to find a place in scientific inquiry, there needs to be a mechanism by which we can come to establish the realism of their results. I consider this to be a primary experimentalist objection.

These objections are powerful, but not impossible to overcome. To respond to the objections from analytic modelers, I argue that there are alternatives to direct inspection available that can place simulations on equally solid footing as analytic models. By examining the potential sources of simulation errors from the lowest levels of hardware failure up to the level of software implementation details, we can see how to insulate simulations from unknown errors. The experimentalist's objection is considerable, and here I rely on robustness analysis as a theoretical framework for a notion of independent verification of simulation results. Robustness analysis here refers to the practice of examining a diverse set of models of the same phenomenon, and in the cases where we find invariance across models, attributing it to a core causal feature of the models, if one exists. (Weisberg, 25) I conclude that ultimately, the objections from both the analytic modeling perspective and the experimentalist perspective can be accounted for with robust simulation strategies.

2. The Analytic Modeler's Direct Inspection Critique

Analytic models have several properties that computer simulations lack. Most obviously,

analytic models tend to be simple and permit some mathematical manipulation. A suitably well-trained human can manipulate analytic models and can prove things about them. Hidden in this statement is the crux of the difference between analytic models and simulations: analytic models are designed to be understood by people, whereas simulations are designed to be understood by computers. Where an analytic model is a series of several equations, a simulation is frequently a system of dozens, hundreds, or thousands of equations. This increase in complexity raises a number of concerns regarding how much we can trust simulation results.

To illustrate the concerns of analytic modelers, let us compare the actions taken in response to unexpected results from an analytic model and a simulation. Assuming that the analytic model was correctly designed to capture the appropriate features of the world and that no crucial feature of the phenomena under study was left out of the model, the surprising results would lead one to search for mathematical errors, for which there are a large number of available referees. In the case of a surprising simulation result, this relatively straightforward process quickly becomes unmanageable, because nontrivial simulations are much more complex than nontrivial analytic models. Simulations can and often do model interactions more complex than what analytic models can easily handle, such as non-equilibrium behavior, heterogeneous agents, large numbers of agents, long timescales, and other features.

An example of models that employ such complex interactions are those employed by the National Weather Service, which incorporate thousands of data points and variables, and track multiple interacting dynamics. In the face of unexpected results, attempting to calculate the interactions for even one round of simulation would be a daunting task. If there were an error of some kind, it is not obvious as to where to look for its source. Without the ability to directly inspect the calculations being performed by the simulation, an analytic modeler might wonder how a simulator can differentiate between a genuinely novel but correct result and an error in the simulation itself.

Simulations have numerous potential sources of error. At the lowest level, there is the possibility that computer hardware can malfunction. Next, even if we assume that the hardware is functioning properly, the computer simulation's software environment could be a host to a number of different errors that affect the simulation's operation. Finally, even if the software environment is error-free, there is always the possibility that the simulator programmed the simulation incorrectly. Analytic models, on the other hand, only have the possibility of "programming error." Analytic modelers can make a mathematical or transcription error, but there are no analogues to the other simulation errors. What's worse, these other errors appear to be out of the simulator's control.

We have seen that the analytic modeler has several important concerns, but they all hang on either the possibility of error, or the more general problem of model verification. Because computers can perform far more calculations than people can, we run the risk of having simulation results that cannot be independently verified by a human. The critique rightly points out that we should not trust results that we cannot verify. However, there is a mistake in this line of reasoning: it assumes that the path to verification is always through direct inspection. While this sounds like a reasonable proposition, it fails to recognize the alternative methods of verification that accomplish the same task as direct inspection, but are more suited to computer simulations. With these methods, we can reduce the sources of error down to just programmer error, which is equivalent to the problems analytic modelers face. Just because we want to hold simulation to the same standard as models does not mean that we must use the same method of model verification.

Let us start at the lowest level of error, hardware failures, and work our way up. To give an example of hardware errors, The Pentium "f00f" bug that affected all Intel Pentium processors up through the Pentium Pro line would cause the computer to freeze and possibly corrupt data in any program that called for a certain instruction in the processor. More serious was an earlier design flaw in Pentium chips, the FDIV bug, which was an error in the floating point unit of the processor, causing

constant, but small, numerical mistakes in floating point calculations.ⁱ Though the programmer community noticed these design flaws relatively quickly, and created software workarounds, these flaws show that the possibility of CPU error is a distinct concern.

However, CPU errors can be exposed and eliminated through very low-level robustness checking. To do this, we must consider what means we have to expose the errors. It would be a daunting task to hand-check a processor, or even the design documents for a processor, given the billions of transistors in modern CPUs. What we can do with relative ease is compare the output of identical programs on different processor classes. There are different vendors of computer processors, and many vendors sell more than one line of processors. It would be increasingly unlikely that all of them have the same design flaw, especially given that there are several different basic approaches to processor design. Even given an extremely unrealistic 50% chance of error, with 8 different processor lines the possibility of all of them having the same error is well below one percent. With a more realistic error rate this chance goes quickly towards zero with fewer processor lines. While it is possible that each processor could each have different, independent errors that affect computation, this is a highly unlikely scenario, as it would require every kind of processor, which already undergoes a great deal of testing, to have a noticeable computational disagreement with every other kind of processor. An even more unlikely possibility would be that all the processors have independent errors that somehow manifest themselves in the same way.

Given the extent to which these processors are tested and used, if there were a small drift in numerical calculations, because of the sheer volume of calculations that take place every day, the drift would be quickly exposed. Almost certainly different kinds of processors would drift in different ways, and one would quickly see divergent results. So, given that there are far more computers in use than there are of any other nontrivial scientific tool, and there are a diversity of processors in use, the probability of an undiscovered error in any given processor line quickly drops toward zero. Scientists

are able to benefit from the fact that computers have become commodities, not just because it lowers the price of computing, but more importantly, because it increases their reliability. While any individual computer may have a hardware malfunction, it becomes vanishingly unlikely that a cluster of computers all share the same malfunction. Because of this, malfunctioning computers are readily identifiable.

The next possible source of error in computer simulations is mistakes – or even just an implementation detail that the simulator is not aware of – in the programming language or software library. A common problem of this kind would be rounding implementation details in integer calculations. Software could round up, down, or to the nearest integer. Each rounding method is entirely legitimate, but if the library has a different assumption than the simulator, this may lead to calculation problems as rounding errors are compounded. One can also find errors in the library that affect simulation, particularly if the library is relatively new or is being used in a nonstandard way. A particularly difficult class of errors is memory over-writing. These errors usually manifest themselves as seemingly random failures in which data is suddenly corrupted. More worrisome is the possibility that the corrupt data is not obviously corrupted, and so the simulation continues to run. Whereas large errors tend to be fatal to a computer program, and thus relatively easy to identify, continuous sources of small errors are less noticeable and thus pose the greater challenge.

Just as the diversity of processor vendors and types makes it possible to detect hardware problems, there are also many programming languages and software libraries that enable simulators to locate software problems. If a simulator is concerned that there are errors in the simulation, a good way to isolate the possibility of programming language or library error is to try and replicate the results using a different programming language. This method is no different than the common experimental practice of replication in a different laboratory, using equipment that is similar, but not identical to, the equipment in the original lab. If the phenomenon can be replicated with multiple programming

languages, then there is a decreasing likelihood that there are errors built into the programming languages used, assuming that the programming languages were developed independently.

Finally we arrive at programmer error. This should be reduced automatically if the program is implemented in multiple programming language families, but it is certainly still a problem.ⁱⁱ But note that we have the same options available to simulators that we do to analytic modelers: we can prove the code's correctness. Any language is ultimately represented by a system of recursively enumerable equations, and so is on equivalent ground to an analytic model. Modern techniques in programming also serve to minimize the errors of programming, by introducing minimal units of programs that can then be “unit tested” independently of the rest of the program. This can help ensure that each component of the program is functioning as specified.

Note that this response to programmer error also helps to address the final question of verification. Of course, if we look at the output of what a simulation did at each computational step to verify the result, we would be quickly overwhelmed by the millions of calculations. But, if we understand how the result was generated, and can understand something about the underlying structure of the simulation, then we have all we need for verification of the simulation.

If we have performed the robustness checks suggested above, we have a good deal of confirmation that the underlying computing mechanism is well-functioning. We can safely assume that all the calculations, as specified by the simulator, will be correctly performed. Thus, we have reduced the simulation to its underlying analytic model, one that leverages computers instead of human brains for calculation. This methodology may appear rather burdensome – the programmer must make new programs for multiple processor lines, in multiple programming languages, and then prove the correctness of the code. However, the programmer need only follow this procedure as far as the desired level of confirmation, which is the same as the extent to an experimenter checks their instrumentation depends on the confirmation level they are after. Even so, experimenters have their

own worries about simulations. Let us now turn to their objections.

3. The experimentalist's objection

Experimentalists may contend that models of any kind are suspicious, insofar as it is questionable that they actually imitate real-world processes. Simulations, after all, are not collecting data in the actual world. What is to prevent a simulator from simply making up a set of equations, and then tweaking parameters until the output matches the existing set of observations? The potential danger of computer simulations comes precisely from their low cost and their power. The low cost induces many scientists to choose simulations instead of the possibly very expensive proposition of experimental work. The amazing flexibility of programming languages and simulation environments is such that any data set can be accommodated. Simulations can lead science astray by providing what looks like evidence for a false theory; experiments are much less likely to have this effect, precisely because they are by nature grounded in real physical processes.

Paul Humphreys points to this worry in the context of epicycles. Ptolemaic astronomy could account for any set of observations, given the proper application of epicycles. Programming languages are easier to use, and equally capable of satisfying any observation. (Humphreys, 133) To compound the problem, frequently simulations are employed to investigate dynamics that are too long-term for us to study with standard empirical methods. How are we to confirm the realism of the long-run predictions of our simulations when the methods of calibration can make them conform to any existing data set? This problem was solved for other scientific instruments in a way that we can apply to our situation.

Galileo had two empirical methods for justifying the use of telescopes: first, he used telescopes to examine details of ships that were coming in to port. These details could then be checked with unaided eyes. Similarly, looking at celestial objects that one could barely make out with unaided

vision, he showed that telescopes were able to provide a much clearer picture. This was the step that was open to objection, since the detailed images could have been artifacts of the telescope. However, he had multiple observers across Europe look at the same parts of the night sky at the same time, recording what they saw. They all saw the same thing. (Kitcher, 173-174)

Philip Kitcher has labeled this methodology the Galilean Strategy. It exploits the fact that the limits of our sensory apparatus are fuzzy, and makes an inference from “success to truth.” This inference supposes that those methods that are predictively successful are so because of correct ontological suppositions. Because of their past successes at the fuzzy boundaries of empiricism, we can then extend them beyond the boundaries and remain confident in their future predictions. (Kitcher, 176-177) The trouble with computer simulations appears to be that these options are not available to them to make any model verification claims. However, there is a method of verification available to us that is akin to the Galilean Strategy: robustness analysis.

As we saw in the previous section, a limited notion of robustness analysis, just relying on independence of errors across diverse computational platforms, showed us how to reduce skeptical concerns of simulations to that of models, by means of systematically reducing a simulation's possible sources of error. This use of robustness is not sufficient for the task at hand. One may be able to program a simulation in as many languages as one might please, but that does not help to ground the simulation, or its underlying model, to the real world in a meaningful way. However, there is a stronger notion of robustness that we can leverage, which will get us most of the way to confirmation: robustness across modalities and parameter space.

William Wimsatt has argued that an important feature of robustness is that we no longer treat a single simulation as the appropriate unit of analysis. Instead, single simulations are treated as parts of a larger whole: a class of simulations that are systematic variants of each other. Our proper object of analysis is this class of simulations. To generate this class, we create multiple models that implement

the same basic dynamic by different modalities and across parameter space. In doing so, we create independent pathways of investigation, which allows us to isolate any particular implementation detail by examining its effects on the simulation. In Wimsatt's words, Robustness analysis has "a common theme in distinguishing the real from the illusory, the reliable from the unreliable....and in general... [the] epistemically trustworthy and valuable from that which is unreliable, ungeneralizable, worthless and fleeting." (Wimsatt, 128)

To explore how Wimsatt's insights can be put into practice, consider a phenomenon that calls for explanation, such as the emergence of cooperative behavior in a population of individually rational agents. This has most frequently been modeled with the Prisoner's Dilemma providing the rules of interaction. In modeling social situations, one can take a number of perspectives. An evolutionary model looks at populations as a whole. Learning models are agent-based – that is, they employ methodological individualism. Within either methodological framework, we have multiple methods of implementation, and a range of possible parameters for those implementations. For example, if one were to choose a learning model, the first choice would be to determine which learning rule to use. Next is deciding on the parameters of the learning rule, such as how much memory to give agents. Each of these points of decision can generate several different simulation paths. Approaching the problem of simulation justification from the perspective of many simulations allows us to systematically remove modeling assumptions by varying parameters and modalities. The biases of methodological individualism can in this way be countered with a population approach, while both capture the same phenomenon. So, rather than finding the single simulation that uses the appropriate parameter set to generate the phenomenon, we look for the class of simulations that generate the phenomenon. One simulation can be a fluke. A class of simulations that vary by both parameters and across modalities suggests a robust phenomenon.

Robust phenomena are perhaps less straightforward than they initially seem. In the case of the

emergence of cooperation, even if a wide variety of simulations all result in a cooperative population, they might still differ in important respects. For example, the average time it takes for a population to converge will vary across different simulations. Some simulations will only converge to cooperation on average, rather than every time the simulation is run. So, while a robust phenomenon may vary slightly in character between simulations, this is like how experimental work is done: it is rare that experiments are performed perfectly, or that repetition results in exact replication.ⁱⁱⁱ

Chemistry offers a useful illustration. It is exceedingly rare to find that, for given masses of reactants, that the mass of the products is precisely what theory predicts. Nor is there precise agreement to be found in replication. In fact, one draws suspicion of faking one's data if it looks too correct. Further, changes in experimental setup might prevent replication. The outcomes of many chemical reactions are temperature-sensitive, for example. Finding that a reaction is not robust across all temperatures does not suggest that it is a fluke – instead this informs us of the reaction's sensitivity. Simulators are not limited to merely seeking out invariances: simulators can see, for example, the temperature ranges under which a given reaction will occur. Just as interesting is an investigation into what causes these invariances to fail. After all, scientific inquiry is not only after the regularities themselves, but what the difference-makers, if any, might be. Note how this is very similar to how Hacking confirms the realism of the images taken from microscopes: we come to understand the underlying dynamics by representing and studying them under multiple modalities in a wide parameter space. Through such an examination, we can eliminate concerns over artificial constraints or assumptions. While any individual simulation may be criticizable on the basis of assumptions over particular parameter values, its modality, or other implementation details, a class of simulations that systematically vary these details is not. It is in virtue of this that we must consider classes of simulations as the unit of analysis, rather than individual simulations. If the simulations produce phenomena that are qualitatively similar in the essential respects, then what they see is not an artifact of

the tool, but rather something in the world. But just as experiments do not rely on bare observations, simulations do not have to either.

Simulators can increase a simulation's confirmation by moving beyond just an analysis of the simulations by themselves. It is also possible to introduce an analysis of interventions on these robust simulations. After a robust dynamic is discovered, the simulator can then try and verify the realism of the dynamic by checking to see if interventions on the phenomenon have robust results. If interventions are not robust across modalities, then we have reason to question the underlying dynamic, as it seems like the modalities themselves are playing a role. When the interventions vary in parameter space, we have found useful distinctions that are testable. In the case of robust interventions, we have a large degree of confirmation. In the latter two cases, we have uncovered an operationally definable dynamic that has clear testing implications.

Frequently, these implications can be tested by means of standard empirical methods. It should be noted, however, that simulations themselves have no way of showing a strong model-world relationship beyond the techniques already discussed. While robustness across a class of simulations, and an understanding of what happens under intervention can be suggestive of what the underlying causal structure of the world might be, we are unable to go beyond our tools. If these simulations have gone beyond what our standard tools can show us, then we have confirmed their results more than what was possible before, and as much as can be done now. That is the most that one can hope for from any tool.

4. Conclusion and the Limits of Simulation

Like any other scientific tool, simulations cannot be used in all circumstances with the same levels of reliability or utility. The most fruitful applications of simulations are those cases when there is already a set of empirical data that the simulation can leverage. Simulation by itself, not informed by

any empirical data, is just a more formal version of a priori reasoning. This can be helpful to clarify ideas and make them more precise, but it is on the same epistemological grounding as any rationalist project. As is the case with analytic models, simulations provide a rigorous framework for reasoning, but not a crystal ball: while they can extend our epistemic limits beyond what we can do unaided, they do have their limits. And like any other tool, simulations can be misused, and their conclusions can be over-stated. But if simulators do their work within a strategy of robustness analysis and active intervention, there is a great deal that simulation can teach us about the world.

REFERENCES

- Humphreys, P. Extending Ourselves. 2004. Oxford: Oxford University Press
- Weisberg, M. "Forty Years of 'The Strategy': Levins on Model Building and Idealization," forthcoming in *Biology and Philosophy*, volume 21, number 5.
- Wimsatt, W. C. 1981. Robustness, reliability, and overdetermination. p. 124-163 in M.B. Brewer & B.E. Collins (Eds) *Scientific Inquiry and the Social Sciences*. San Francisco: Jossey Bass.

- i <http://support.intel.com/support/processors/pentium/fdiv/> and <http://support.intel.com/support/processors/pentium/ppiie/> have more details on these processor design flaws.
- ii Different programming language families frequently require different algorithmic approaches to the same task, as they have different fundamental data structures. Because of these different approaches that are needed, programming errors should be independent across implementations.
- iii Simulations have a bit of an advantage in replication: even when random numbers are involved, a pseudorandom number generator with the same initial seed will always produce the same numbers. If exact replications are needed, using the same seed will accomplish this.