Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context

Victoria Stodden, Matthew S. Krafczyk

August 24, 2018

Abstract

We present an experiment using the Enzo simulation code on NCSA's Blue Waters system to highlight the importance of computational and numerical uncertainty in scientific computing on HPC systems. We quantify the (surprising) variability of outputs from 200 identical simulation runs. We make two recommendations to improve the assessment of reproducibility of computational results: the inclusion of computational variability measures in standard reporting practices; and modular software designs that permit localized assessments of the source of computational uncertainty.

1 Introduction

Several recent efforts highlight the importance of reproducibility for computational science [1, 2, 3, 4, 5, 6]. In this article we adopt the simplest definition of reproducibility: using the same software and data to regenerate the same results as in the publication [7]. This definition is, in some sense, a pre-requisite to other notions of reproducibility, such as an independent re-implementation of the experiment, since complete computational transparency is required to reconcile any meaningful differences in results. It should surprise no one re-runnning the same code, even on the same system and with the same random seeds and input data, at a later date can produce different output. Software libraries and versions change presenting application complexity issues that can even extend to the configuration of the network upon which it relies. Hardware can change, and even without changes can be inherently non-deterministic: I/O devices report interrupts at unpredictable times which can affect the scheduling of processes for example. External physical processes such as cosmic rays can flip bits in memory or control logic randomly. Concurrency issues are present in today's systems: applications use multiple processes, threads, and cores, and/or rely on parallel accelerators such as GPUs. With these system-level potential sources of uncertainty, at what point can a researcher know when a result has been satisfactorily regenerated even though it differs from the original due to inherent system uncertainty, or when is it a truly different result? In this article we examine the simplest case - rerunning the same code on the same system - and provide an example of variability in application outcome.

We build on recent recommendations regarding the capture and reporting of computational details in scientific publication, specifically Recommendation 1 from [8]: "Share data, software, workflows, and details of the computational environment that generate published findings in open trusted repositories." We seek to answer a subsequent question: Does satisfying Recommendation 1 permit a researcher in the field to regenerate the same results as in the original article? Or, more precisely, what is the threshold at which we understand the regenerated results to be meaningfully different from the results in the original article, given the same software and input data? We illustrate the nonobviousness of the answer in the next section with an illustration of variability in simulation output in HPC using the Enzo adaptive mesh refinement simulation code [9]. We then suggest two solutions to improve assessments of reproducibility: including bounds based on computational uncertainty with published results (thereby extending Recommendation 1) and, modifying scientific software design to allow for the localization of sources of variability to be identified and traced. We conclude with a discussion of future work.

2 Computational Uncertainty in an HPC System

In order to investigate impact of non-determinism on scientific conclusions, we surveyed several scientific fields searching for codes which either used deliberate non-determinism for stochastic simulation purposes or which could introduce non-determinism through the use of parallelized simulation algorithms with reduction operations. We chose the Enzo Astrophysical simulation system since it satisfies both these criteria and the simulations are carried out over a long physical time scale during which small numerical errors or deliberate stochasticity can accumulate to produce significant shifts in final galaxy properties. We then designed a computational experiment as follows. Using the Blue Water system at NCSA, we repeated a simulation 200 times to produce the same scientific output. We performed a cosmology simulation using an n-body simulation for dark matter and an adaptive mesh to track gas dynamics. We first generated initial conditions: initial grid and particle fields using the `inits` tool from the Enzo software package.

2.1 Experiment 1

These initial conditions were generated for a root grid with a resolution of 32 on a side, a size of 10 Mpc/h per side, 7 levels of adaptive mesh refinement, and periodic boundary conditions. Enzo was then configured to treat these initial conditions at $z = 99$ and to run the simulation to $z = 0$ or the modern day. The `inits` tool was run a single time. The experiment was then repeated 200 times using the initial conditions obtained from the `inits` tool. Once the simulation finished, the rockstar [10] halo finding algorithm and the yt [11] tool suite was used to determine locations and properties of galactic halos which had formed. The scientific output of interest in this case is the mass and location of the various galactic halos and more specifically, the largest galactic halo.

2.2 Experiment 2

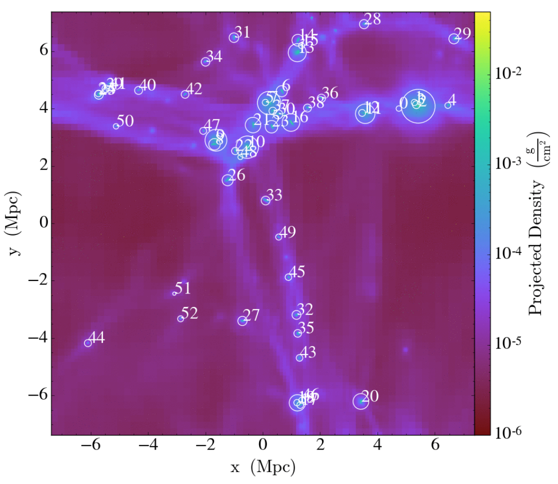
To test whether the computational tool chain had an effect on the eventual non-determinism, we performed experiments using different compiler and optimization levels. For each compiler and optimization level we performed the simulation 200 times in order to extract mean and variance statistics for the largest halo.

2.3 Results

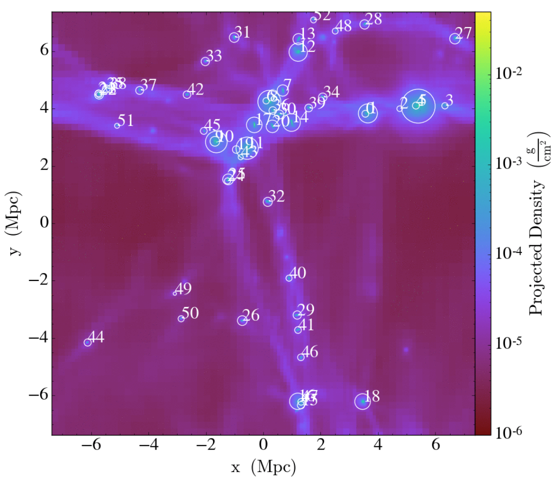
Given the output of the simulation runs, we report the mean of these properties and variance of the 'same' halo across all simulations. This was done using a 'superhalo' algorithm which matched halos across all 200 simulations using a combined metric of mass, position (x, y, z), and virial radius. Only halos which formed a 'mutual match' across all 200 simulations are grouped together into a superhalo. A 'mutual match' is formed between halo $A1$ from simulation A and $B1$ from simulation B when $d(A1, B1)$ is less than $d(A1, Bn)$ for all other n in simulation B and $d(An, B1)$ for all other n in simulation A. Measured properties of the largest halo along with performance metrics are given in Table 1, as well as graphically in Figures 2a and 2b. The figures in 1 showcase some stark differences which can arise, from seemingly identical simulation runs.

2.4 The Toolchain and Workflow

Software for this experiment was managed using the Spack [12] package manager. Once appropriate package definition files were created for Enzo, yt, and rockstar, changing compiler and optimization levels was relatively straightforward. The final dependency tree is also now preserved to be used by a scientist at a later date. The code and data used to produce these results is available at `https://github.com/victoriastodden/ComputationalUncertaintyHPC`.

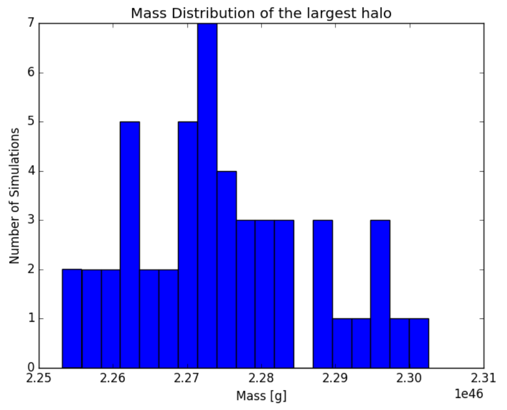


(a)

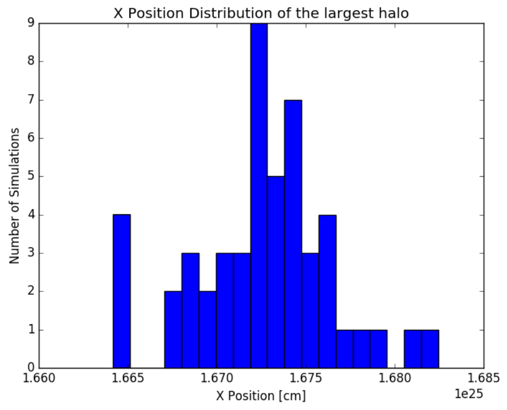


(b)

Figure 1: We show the result of two Enzo cosmological simulations with identical initial conditions and simulation properties in the two panels. The galaxy gas density is projected onto the x-y plane. The white circles indicate detected galactic halos, representing the size (and mass) of each halo. Note for example that halo 49 exists in Figure 1a, and is missing in Figure 1b. This is because minute numerical variations have affected the history of the second simulation to the point that the mass clump where halo 49 should be never became gravitationally bound and hence detectable by the rockstar algorithm. Also note the distances and positions of several of the galaxies are subtly different in the two panels. Close inspection of some of the halos will also reveal different orientations in the two simulations.



(a) Mass distribution of the largest galaxy



(b) X position distribution of the largest galaxy

Figure 2: The two panels show the variability of properties of the largest galaxy from 200 Enzo cosmology simulations. These simulations were performed using the gcc compiler version 6.2.0 with 'high' category optimizations ('-O2'). The mass of this galaxy can vary by as much as 0.5% of the mean value.

Table 1: We show the average mass and standard error of the largest halo for each simulation and for each compiler/optimization combinations. For the cases intel/normal and cce/none note that the average mass is significantly different from $\sim 2.27 \cdot 10^{46}$ g, since the superhalo algorithm was unable to match this large halo across all 200 simulations, indicating that it was either too different to match, the final galactic geometry was different enough to stop a match, or this galaxy simply did not exist in at least one of the simulations.

Compiler	Optimization	Average Walltime	Average Mass of Largest Halo	Std Err of Mass of Largest Halo
`gcc@6.2.0`	None	$\sim$ 22h	$2.27397 \cdot 10^{46}$ g	$1.06934 \cdot 10^{44}$ g
	Normal	$\sim$ 10h	$2.26675 \cdot 10^{46}$ g	$1.21864 \cdot 10^{44}$ g
	High	$\sim$ 9h	$2.27468 \cdot 10^{46}$ g	$1.19946 \cdot 10^{44}$ g
`intel@16.0.3`	None	$\sim$ 39h	$2.27147 \cdot 10^{46}$ g	$1.58783 \cdot 10^{44}$ g
	Normal	$\sim$ 7h	$4.33035 \cdot 10^{45}$ g	$1.24854 \cdot 10^{44}$ g
	High	$\sim$ 6h	$2.26850 \cdot 10^{46}$ g	$1.41447 \cdot 10^{44}$ g
`pgi@16.9.0`	None	$\sim$ 14h	$2.27006 \cdot 10^{46}$ g	$1.21608 \cdot 10^{44}$ g
	Normal	$\sim$ 13h	$2.27201 \cdot 10^{46}$ g	$1.32688 \cdot 10^{44}$ g
	High	$\sim$ 10h	$2.27125 \cdot 10^{46}$ g	$1.1913 \cdot 10^{44}$ g
`cce@8.5.5`	None	$\sim$ 16h	$4.31151 \cdot 10^{45}$ g	$1.35333 \cdot 10^{44}$ g
	Normal	$\sim$ 6h	$2.27129 \cdot 10^{46}$ g	$1.26159 \cdot 10^{44}$ g
	High	$\sim$ 5h	$2.27230 \cdot 10^{46}$ g	$1.34160 \cdot 10^{44}$ g

3 Two Solutions Ideas

3.1 Assess and Report Computational Uncertainty Measures

Ideally, each experiment could be run a large number of times and error bars that assess computational uncertainty.reported alongside results. Of course this is not generally practical in terms of compute-time, so estimates of these metrics could be made by experiment-appropriate modifications that reduce runtimes: downsampling the parameters of the simulation, reducing the number of iterations, or running a subpart of the simulation that is deemed most likely to produce computational uncertainty. Standards have been proffered that encourage the reporting of computational details (such as the ICERM standards [13, 14]) and we suggest including estimates of computational uncertainty as part of standard reporting. The well-known notions of Uncertainty Quantification and Validation & Verification should be augmented to include computational uncertainty.

3.2 Software Design Specifications

The example given in this article could satisfy Recommendation 1 and the ICERM standards and still produce variable output. Software design that permits the localization of variability to particular aspects or parts of the code could allow improved assessment of computational variability. Many applications embed state information in their output, to help the user with debugging and general provenance, however this may not give sufficient information to assess whether results that differ bitwise are scientifically equivalent. Software design (e.g. modularity) that permits the user to source aspects of computational uncertainty by code inspection could allow for improved assessments of expected output variability.

4 Conclusions & Future Work

We present an example of simulation output variability due to computation. We will generalize this example to other codes and scientific fields, and understand the accuracy of computational uncertainty estimates from simulations with reduced runtimes.

References

[1] V. Stodden, F. Leisch, and R. D. Peng, *Implementing Reproducible Research*, ser. Chapman & Hall/CRC The R Series. Chapman and Hall/CRC, April 2014.

[2] L. A. Barba, "Terminologies for reproducible research," *ArXiv e-prints*, Feb 2018.

[3] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden, "Reproducible research in computational harmonic analysis," *Computing in Science & Engineering*, vol. 11, no. 1, pp. 8–18, 2009. [Online]. Available: http://aip.scitation.org/doi/abs/10.1109/MCSE.2009.15

[4] V. Stodden, "Reproducible research: Tools and strategies for scientific computing," *Computing in Science & Engineering*, vol. 14, pp. 11–12, 07 2012. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MCSE.2012.82

[5] P. Ivie and D. Thain, "Reproducibility in scientific computing," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 63:1–63:36, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3186266

[6] V. Stodden, M. S. Krafczyk, and A. Bhaskar, "Enabling the verification of computational results: An empirical evaluation of computational reproducibility," in *Proceedings of the First International Workshop on Practical Reproducible Evaluation of Computer Systems*, ser. P-RECS'18. New York, NY, USA: ACM, 2018, pp. 3:1–3:5. [Online]. Available: http://doi.acm.org/10.1145/3214239.3214242

[7] V. Stodden, "Resolving irreproducibility in empirical and computational research," *IMS Bulletin*, 2013.

[8] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, and M. Taufer, "Enhancing reproducibility for computational methods," *Science*, vol. 354, no. 6317, pp. 1240–1241, 2016. [Online]. Available: http://science.sciencemag.org/content/354/6317/1240

[9] G. L. Bryan *et al.*, "Enzo: An adaptive mesh refinement code for astrophysics," *The Astrophysical Journal Supplement Series*, vol. 211, no. 2, p. 19, 2014. [Online]. Available: http://stacks.iop.org/0067-0049/211/i=2/a=19

[10] P. S. Behroozi, R. H. Wechsler, and H.-Y. Wu, "The rockstar phase-space temporal halo finder and the velocity offsets of cluster cores," *The Astrophysical Journal*, vol. 762, no. 2, p. 109, 2013. [Online]. Available: http://stacks.iop.org/0004-637X/762/i=2/a=109

[11] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, and M. L. Norman, "yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data," ApJS, vol. 192, p. 9, Jan. 2011.

[12] T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski, and S. Futral, "The spack package manager: Bringing order to hpc software chaos," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: ACM, 2015, pp. 40:1–40:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807623

[13] D. Bailey, J. Borwein, and V. Stodden, "Set the default to 'open,'" *Notices of the AMS, Accepted March*, 2013. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.310.4101

[14] V. Stodden, J. M. Borwein, and D. H. Bailey, "'setting the default to reproducible' in computational science research," *SIAM News*, 2013.