A Jupyter Notebook Based Tool for Building Skills in Computational Statistical Mechanics

Brian Smith, Ben Glick, and Jens Mache

***Abstract*— When learning about High-Performance Computing, almost of equal importance to developing computational skills is building an understanding of the applications and implications computational research can have. One field in which high-performance computing is particularly important is computational physics. Many physicists depend on high-performance hardware and software to carry out their research. This project describes an open-source, Jupyter notebook-based web app which helps students both develop skills relating to and appreciation for HPC techniques and builds understanding of complex physical models in the domain of thermodynamics/quantum field theory. This project is designed to help students understand, appreciate, and build skills relating to simulations of statistical systems, which are too complex to describe analytically and instead need to be described by complex monte-carlo simulations.**

I. Introduction

In statistical systems, it is computationally infeasible to compute most physical quantities directly. Using statistical mechanics, the normal way of analyzing a specific system is by observing its partition function. A system's partition function is a vital tool for physicists to be able to calculate the system's observable characteristics. In statistical mechanics, many systems are described by the Boltzmann distribution; the partition function is a proportionality factor describing the specific distribution at hand. From it we can derive almost any observable quantity we want, like magnetization and heat capacity[1].

In any relatively complex model, it is simply infeasible to directly compute the partition function. Instead, we must turn to the tools of statistical mechanics - describing the partition function indirectly based on the output of simulations[2].

II. Our Model

One example of a system like this is the Ising model of a magnet, which is modeled as a 2D lattice with spins s up or down (+/-1) at each lattice point. The magnetization of the system is a sum over the points (i, j) in the lattice. However, the partition function is a sum over every state of the system:

$$H(s) = -J \sum_{<i,j>} s_i s_j - B \sum_i s_i, \quad Z = \sum_s e^{-H(s)/kT}$$

H is the Hamiltonian, or total energy, which is a nearest-neighbor sum over the entire lattice. J and B are interaction and external field constants, respectively[3]. Therefore, even for small lattices, it is impossible to compute the partition function directly. To analyze the model, we need to turn to the tools of HPC and Monte Carlo simulations[4]. With Monte Carlo simulations, we create a virtual instance of the system we would like to study and allow it to evolve in silico. In the case of the Ising model, this means setting up an initialized magnetic lattice, and imposing a temperature and external magnetic field on it; by virtue of statistical mechanics, we can calculate separate probabilities of specific actions taking place within the lattice[5].

Since we can easily characterize the initial state of the system, all we do is run our simulation over the range of conditions we want to understand. Our Monte Carlo simulation uses some physical details of the system to calculate the energy in a given state, and then feed that energy through a probability based on the Boltzmann distribution; normally the partition function would be required at this step, but since our goal is the ratio of probabilities between two states, it becomes unnecessary[2]. We pick a random lattice site and calculate the probability that the specific spin would flip, based on the requisite change in energy:

$$\mathrm{P} = \mathrm{e}^{-H(s)/kT}, \quad P_{flip} = P_1/P_2 = e^{-\Delta H(s)/kT}$$

The system then evolves according to this probability, and will approach the Boltzmann distribution by iterating many times[4]. At each Monte Carlo "step", we measure the energy and store it in an array. At the end of the simulation this array is averaged to get an expectation value of the energy; this process can be used to measure any desired physical quantity, like magnetization.

For the Ising model, this makes for very quick results, which have been put into an interactive app using Jupyter notebooks[6]; at the click of a button in a Jupyter widget, several relevant physical parameters are displayed after only a few seconds.

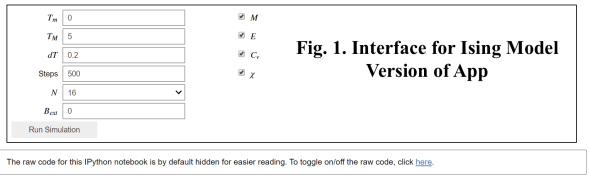

Fig. 1. Interface for Ising Model Version of App

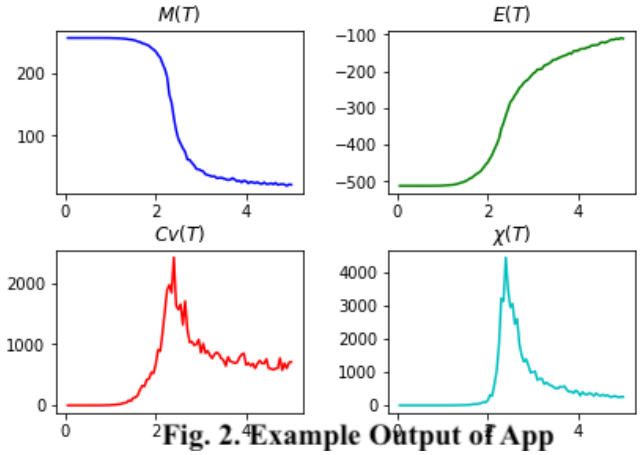

Fig. 2. Example Output of App

For more complex models, the value of HPC is obvious. Take the SU(2) model, which maps onto an XY magnet [7]. Instead of simply mapping spins on the lattice to (+/-1), the XY model allows the spin to be at an angle θ anywhere between 0 and 2π. As a result, its energy is described using several cosine terms, which do not have the benefit of algebraic simplification.

$$H(\theta) = -J \sum_{i \neq j} cos(\theta_i - \theta_j) - B \sum_i cos(\theta_i)$$

This is a far more computationally intensive model and even with Monte Carlo, greater computational power is required to gather results in a timely manner. And for the SU(3) model which we are mapping to the strong force[8], this is even more apparent. It can be shown that certain XY-spin models capture the essential symmetries in the fundamental interactions of quantum chromodynamics, or the strong nuclear force with the various gauge groups SU(2), SU(3), and so on [9]. Without HPC skills and techniques, this work would have been impossible.

This code has been written in such a way as to easily allow us to gather data of related systems. Not only is it easy to update the model we are studying by adjusting the Hamiltionian, which describes the energy of the system, but we can also dramatically alter the kind of data that is output without hassle. For example, say we want a statistical distribution of the energy of the system at a given temperature, rather than a curve describing the energy over a range of temperatures. The code already gathers thousands of raw values to calculate expectation values for the energy. We are able to simply save those raw values and deposit them into histograms.

III. Presentation Outline

In our five-minute presentation, we would cover several of the major aspects of our project. We would briefly cover the relevant physics, including a roughly one minute summary of statistical mechanics and specifically partition functions. We would then discuss how directly computing the partition function of these systems is essentially impossible and therefore explain the necessity of the Monte Carlo simulation. We would then move on to discussing our app, why it is useful, and how it works. Finally, we would cover future directions this work could take, how to try it out yourself, and what new users should expect to gain from it.

IV. Importance, Educational Impact and Conclusion

All told, this work represents useful progress in the field of research computing education as well as in the ability for computational researchers to learn novel HPC techniques and skills. The web app which was created will allow students to quickly grasp both the concepts of this type of statistical mechanics and also the power and necessity of using HPC systems and resources for this type of work. Additionally, the app can be used to help specifically develop skills in both computational sciences and research computing. For physics students, it is useful to help demonstrate the power of HPC and gather high quality, non-trivial results; for CS students, it is instrumental in showing off some of the applications of CS theories and techniques. Additionally, the way the underlying code is designed allows for easy expansion and variation. This allows relatively inexperienced CS and physics learners to quickly develop new models and create novel work while still utilizing and building skills in HPC and research computing.

V. Try It Live

To try a live version of the web app, visit https://jupyter.datasci.watzek.cloud. Please log in with the username "EduHPC19" and the password "ReviewMe!"

Acknowledgment

The authors would like to thank Mohamed M. Anber, Assistant Professor of Physics at Lewis & Clark College for his guidance. His ideas were the origination of the Jupyter notebook work. Much of the code was developed by him and his research students, including Benjamin Kolligs. We would also like to thank Jeremy McWilliams, Digital Services Coordinator at the Watzek Library, Lewis & Clark College for assistance with cyber-infrastructure.

References

[1] J. Hubbard, "Calculation of partition functions," *Physical Review Letters*, vol. 3, no. 2, p. 77, 1959.
[2] M. Newman, *Computational physics*. CreateSpace Independent Publ., 2013.
[3] B. M. McCoy and T. T. Wu, *The two-dimensional Ising model*. Courier Corporation, 2014.
[4] M. Newman and G. Barkema, *Monte carlo methods in statistical physics chapter 1-4*. Oxford University Press: New York, USA, 1999.
[5] D. V. Schroeder, "An introduction to thermal physics," 1999.
[6] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.
[7] M. M. Anber and B. J. Kolligs, "Entanglement entropy, dualities, and deconfinement in gauge theories," *Journal of High Energy Physics*, vol. 2018, no. 8, p. 175, 2018.
[8] M.-Y. Han and Y. Nambu, "Three-triplet model with double su (3) symmetry," *Physical Review*, vol. 139, no. 4B, p. B1006, 1965.
[9] M. M. Anber, E. Poppitz, and M. Ünsal, "2d affine xy-spin model/4d gauge theory duality and deconfinement," *Journal of High Energy Physics*, vol. 2012, no. 4, p. 40, 2012.