A model project for reproducible papers: critical temperature for the Ising model on a square lattice

M. Dolfi, J. Gukelberger, A. Hehn, J. Imriška, K. Pakrouski, T. F. Rønnow, M. Troyer, and I. Zintchenko*
Theoretische Physik, ETH Zurich, 8093 Zurich, Switzerland

F. Chirigati, J. Freire, and D. Shasha*
New York University (NYU), New York, USA

In this paper we present a simple, yet typical simulation in statistical physics, consisting of large scale Monte Carlo simulations followed by an involved statistical analysis of the results. The purpose is to provide an example publication to explore tools for writing reproducible papers. The simulation estimates the critical temperature where the Ising model on the square lattice becomes magnetic to be $T_c/J = 2.26934(6)$ using a finite size scaling analysis of the crossing points of Binder cumulants. In an Appendix and as supplementary material we provide the output of the simulations, the source and binaries of the codes, and Python scripts used to generate all figures and results.

The principle that scientific publications have to be reproducible is a cornerstone of modern science. A theoretical paper typically contains all the steps required to follow the arguments and arrive at the final result. Experimental papers usually go to great lengths in describing the most important details and keep track of workflow in sacrosanct lab notebooks. In computational science the goal of reproducibility is harder to achieve. Reasons are the complexity of computer systems, codes, and analysis procedures and the absence of well established community guidelines and wide-spread tools for reproducibility.

To focus the discussion on comparison of procedures and tools, we picked one simple, yet representative example of a computer simulation in statistical physics, calculation of the critical temperature of the classical Ising model. The current manuscript already contains sufficient details, codes, and scripts to reproduce all the presented numerical results and figures. However, reaching this level of reproducibility required efforts that went far beyond simply obtaining the results. Our goal for future work is to use this paper as an example for a discussion on how reproducibility can best be achieved.

The total computation time for a small system size is in the order of several hours on a single core producing several MB of raw output data. Including large system sizes increases the accuracy of the results, but also the runtime and the amount of data produced and one might need to use large computer clusters. This allows to explore the scalability of tools for reproducibility in computational science.

* Authors are listed in alphabetical order.

Appendix A: Critical temperature for the Ising model on a square lattice

1. Model

The Ising model dates back to 1920 when it was proposed by Wilhelm Lenz as a mathematical model for ferromagnetism and first analytically solved by his student Ernst Ising in one dimension [1]. We will consider the two-dimensional Ising model on a square lattice of size $L \times L$ with periodic boundary conditions. It is described by the Hamiltonian

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j. \tag{A1}$$

where the sum is over nearest neighbours, $\sigma_i \in \{-1, +1\}$ is the spin on site i and J is the coupling strength. In the following we will focus on the ferromagnetic model $J > 0$.

Below a critical temperature T_c the model shows magnetic ordering, whereas above T_c it remains unordered. For the square lattice without an external field T_c is known analytically [2]

$$T_c = \frac{2J}{\ln(1+\sqrt{2})} \approx 2.269185. \tag{A2}$$

and provides a benchmark for our results.

2. Methods

A Monte Carlo simulation using Wolff cluster updates [3] is used to construct new system configurations, employing the MT19937 Mersenne Twister pseudo random number generator [4]. For each parameter set more than 1280000 measurements are performed after discarding 10% additional Wolff updates for thermalization; error estimates are done with binning analysis.

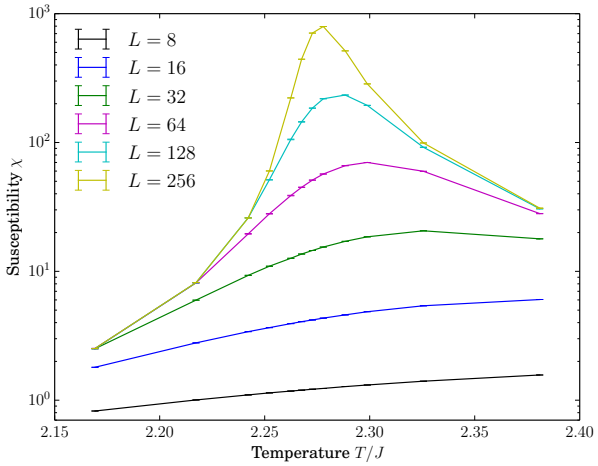

FIG. 1. Temperature dependence of the susceptibility for different system sizes L.

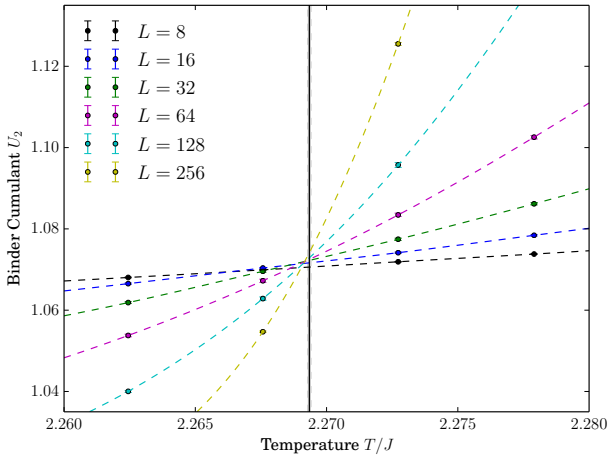

FIG. 2. Temperature dependence of the Binder cumulants for different system sizes. Vertical line and the grey area around it indicate our estimate for critical temperature and for the error respectively. Dashed lines are fits to a cubic polynomial.

The critical temperature can be roughly estimated from the connected susceptibility

$$\langle \chi \rangle_\beta = \beta L^2 \left(\langle m^2 \rangle - \langle |m| \rangle^2 \right), \tag{A3}$$

where $\beta = \frac{1}{k_B T}$ and k_B is the Boltzmann constant. The average is taken over different configurations. $\langle \chi \rangle_\beta$ has a peak around T_c [5], which gives a first rough estimate.

The Binder cumulant

$$U_2 = \langle m^2 \rangle / \langle |m| \rangle^2 \tag{A4}$$

provides a more accurate technique to extract the critical temperature. For different system sizes the temperature dependence of U_2 is expected to cross at different points. The crossing points can be shown to follow

$$T_c^*(L) = T_c^* + AL^{-1/\nu} \tag{A5}$$

where the critical exponent $\nu = 1$ in two dimensions [6]. To extract T_c^* for an infinite system we now fit the positions of the crossing points between system sizes L and $L/2$, respectively, using a least-squares fit weighted with the size of the error bars at each system size, minimizing $\chi^2 = \sum_L (T_L - T_L^l)^2/\zeta_L^2$, where T_L is the crossing point temperature for the systems with sizes L and $L/2$, ζ_L is its standard error, and T_L^l is the value of the linear fitting function. A Jacknife analysis with at least 78 bins is used to estimate the errors of the Binder cumulants and their crossings.

3. Results and Discussion

Figure 1 shows the connected susceptibility defined in Eq. A3 as a function of temperature for system sizes $L = 8, 16, \ldots, 256$. It peaks around $T \approx 2.275J$, which gives a first estimate for the critical temperature.

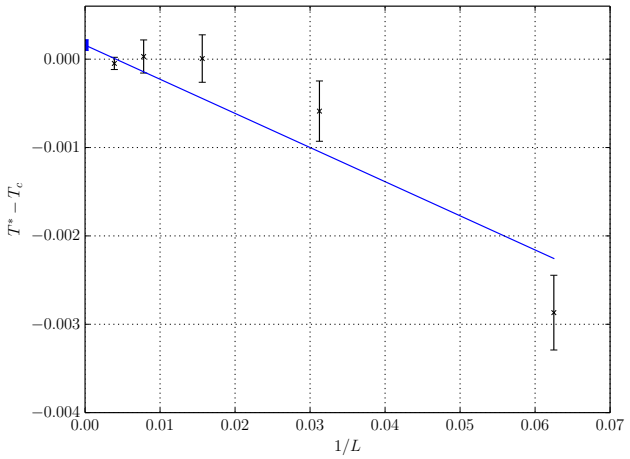

FIG. 3. Finite size scaling of the Binder cumulant crossing points. The dependence can be shown to satisfy $T_c^*(L) = T_c^* + AL^{-1/\nu}$ and $T_c^*/J = 2.26934(6)$ is extracted from a weighted least-squares fit as described in the text.

Figure 2 shows the intersections of Binder cumulants U_2 (A4) with cubic interpolation between all temperatures within each system size. For each pair of consecutive system sizes we identify the temperature at the crossing point. This temperature is plotted in Fig. 3 as a function of the larger system size in each pair. By extrapolating to the limit $1/L \to 0$, a more accurate estimate for the critical temperature $T_c^* = 2.26934(6)J$ is obtained. To check the validity of this value, we plot a data collapse of the Binder cumulant U_2 for different

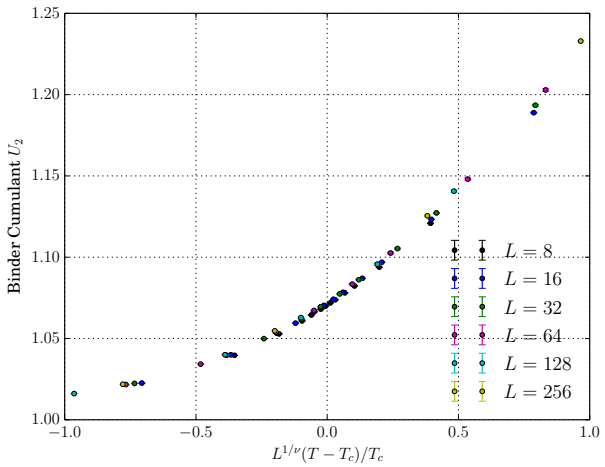

FIG. 4. Data collapse of Binder cumulant. T_c^* is obtained from Fig. 3.

system sizes versus $L^{1/\nu}(T - T_c)/T_c$ (Fig. 4), which according to finite size scaling should satisfy

$$U_2 = \mathcal{F}(L^{1/\nu}(T - T_c)/T_c), \tag{A6}$$

where $\mathcal{F}$ is a universal function. The curves are indeed on top of one another in the vicinity of our estimate for the critical point T_c^*. The reader is invited to use our scripts to test different values of T_c^* and ν in the data collapse (appendix C 3 c).

4. Conclusion

Our final estimate for the critical temperature $T_c^*/J = 2.26934(6)$ is consistent with the analytically known value, which is a good check for the correctness of the analysis in this demonstration paper. Instructions to reproduce all figures and results are provided in appendix C and B. All the analysis scripts, paper and code sources are available as supplementary material to this paper and can be downloaded from **to be inserted**.

As one might expect, the effort involved in making this paper reproducible was much larger than the effort for creating the first results. An important lesson learned from this project is that it will be important to develop best practices and better tools to more easily make simulations reproducible. This prototypical simulation can be a good non-trivial, yet not too complex example of a test project to explore the capabilities of various tools.

Appendix B: Reproducing our setup

This appendix contains instructions for obtaining the raw data and codes necessary to recreate all our results from scratch or run additional simulations. Alternatively, we provide a virtual machine image which can be downloaded from **to be inserted** and imported into the open source software VirtualBox [7]. The machine can be accessed by logging in as user "Mad Scientist" (password: `ilovemath`), opening a terminal and changing into the directory `~/ising_project`.

1. Downloading the raw data

The raw simulation output is available from our archive [8]. It should be extracted into the `data/` directory of the source tree to be accessible to the evaluation scripts. A convenience script `getdata.sh` is provided to download and extract the data on Unix-based platforms. On Windows we recommend to manually download and extract the data.

2. Obtaining the code

There are three versions of the code that can be used interchangeably to create the output data:

- `ising_single`: a single-threaded simulation code.
- `ising_threaded`: a multi-threaded simulation code. It spawns one thread per set of input parameters, up to the maximum number of cores of the machine.
- `ising_mpi`: an MPI version of the code, using distributed memory parallelism.

a. *Precompiled binaries*

For convenience we provide precompiled binaries of `ising_single` and `ising_threaded` for Linux (**version to be inserted**), MacOS X version 10.6 or higher (64-bit), and Windows 8 (32-bit) at **to be inserted**. An MPI version `ising_mpi`, compiled for OpenMPI 1.6.5 is provided only for MacOS X and Linux. Further precompiled versions may be obtained from the authors. These binaries should be placed in the `bin/` directory.

b. *Bulding from source*

The binaries can also be built from source. This requires installation of the ALPS libraries [9] for parallel Monte Carlo simulations. After setting the environment variable `ALPS_ROOT` to the location of the ALPS installation, the simulation codes can be built using the CMake files provided in the `src/` directory. The root of the source tree should be specified as the `CMAKE_INSTALL_PREFIX/` to install the executables into the `bin/` directory. A convenience script `buildall.sh`

is provided for Unix-based platforms. On Windows we recommend the precompiled binaries.

Appendix C: Rerunning the codes and data evaluation

This appendix contains detailed instructions for running the codes and reproducing the numerical results presented in this paper. Additionally we point out parameters the reader may want to change for checking the robustness of our analysis. In this appendix we assume a working setup of our codes and data.

1. Running the code

The simulation is started by executing the run Ising simulation script

```
$ python src/runising.py <sim_executable>   \
      <length> <measurements>               \
      <beta> [<beta_2> ... <beta_n>]
```

The script will run one version of the simulation executables (`ising_single`, `ising_threaded` or `ising_mpi`), which needs to be passed as first argument. Following this argument the script needs to be provided with the side length of the system `<length>`$= L$, the number of Monte-Carlo measurements `<measurements>` and a list of the inverse temperatures `<beta>`$=\beta$ to be simulated. The script will write a log file along with the output of the simulation, which contains important provenance information such as the time when the simulation was started or the hostname of the machine on which the simulation was run.

To recreate the data presented in this paper we provide two convenience scripts which run all necessary simulations automatically. Executing the first script `python simulate_all.py` recalculates all numerical data used in this paper, which will take a few hundred CPU hours. This script can easily be customised to calculate different system sizes, improve statistics by increasing the pnumber of sweeps, or extending the temperature range. If the reader has no computer cluster at hand, he may execute `python simulate_small.py` which will run the simulations for system sizes up to $L = 64$ and will require only a few CPU hours. By default these scripts will use the multi-threaded version of the Ising Monte-Carlo code. The single-threaded or MPI version can be selected by changing the variable `executable` in these scripts.

2. Data evaluation

Analyzing the raw data we calculate the estimate for T_c and produce the figures for the paper and text files containing the information in the figures (for easy access to the numerical values of the data shown). All the data analysis is performed by Python scripts in the `figures` directory, which operate on raw data stored in the `data` directory. The scripts require Python 2.7 together with PyTables [10], numpy, scipy, matplotlib. The scripts `susceptibility.py`, `binder_cumulant.py` and `binder_collapse.py` create all figures needed for the paper in separate subdirectories while `parms.py` contains parameters that control the behaviour of these scripts. In detail the reader needs to execute the following sequence of commands:

```
$ cd figures
$ python susceptibility.py
$ python binder_cumulant.py
$ python binder_collapse.py
```

For each figure a corresponding directory is created, containing a PDF file and accompanying text file.

The estimate of the critical temperature T_c^* is printed to the standard output by `binder_cumulant.py` and has to be manually copied into `parms.py` whenever the raw data or evaluation has been changed because it is used as an input for the data collapse.

3. Suggestions for further analysis

Here we provide some ideas of how one might want to change the evaluation parameters in `parms.py` for checking our evaluation procedure. After editing a parameter the evaluation needs to be rerun as described above.

a. Determination of Binder crossings

The fitting function used to determine the crossing points between the Binder cumulant curves of different system sizes affects our final T_c estimate. Changing the parameter `binder_crossing_fit_kind` from `'cubic'` to, e.g., `'linear'` will change the crossing temperatures plotted in Fig. 3 and hence the final estimate T_c^* calculated by `binder_cumulant.py`. The reader may get an impression of the systematic error connected to this choice by checking how T_c^* and its error estimate change when the fit parameter is changed. A change in the critical temperature estimate T_c^* will also change the data collapse Fig. 4 (when the new value output by `binder_cumulant.py` is copied into `parms.py`).

b. Finite size fitting

Another choice affecting the final result is the range of system sizes that is used for the extrapolation to the thermodynamic limit as shown in Fig. 3. The reader may evaluate the effect of changing the values of the parameters `finite_size_min_L`

and `finite_size_max_L` on T_c^* and its error estimate as printed by `binder_cumulant.py`.

c. *Data collapse*

Last but not least it is instructive to explore the data collapse plot in Fig. 4. For a good estimate of the critical temperature and exponent all the data points will fall onto a single universal curve. By changing the parameters `Tc` and `nu` one can explore the regime of critical temperature T_c^* and critical exponent ν where there is still good data collapse, hence gaining further appreciation of the uncertainty in T_c^*.

[1] E. Ising, Zeitschrift für Physik A Hadrons and Nuclei **31**, 253 (1925), 10.1007/BF02980577.
[2] L. Onsager, Phys. Rev. **65**, 117 (1944).
[3] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).
[4] M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).
[5] K. Binder, Zeitschrift für Physik B Condensed Matter **43**, 119 (1981), 10.1007/BF01293604.
[6] A. Pelissetto and E. Vicari, Physics Reports **368**, 549 (2002).
[7] `http://www.virtualbox.org`.
[8] `http://archive.comp-phys.org/phys.ethz.ch/provenance/data/20131014.tar`.
[9] `http://alps.comp-phys.org`.
[10] `http://www.pytables.org`.

Appendix D: Overview of the source tree