Assignment report – DOS–Ising2D

Gianluca Salvato
gian.salvato@gmail.com – +39 346 870 2549

July 31, 2015

University of Trieste
Inter-University Masters Degree in Physics
(Condensed Matter Physics)
A.A. 2014-2015

Abstract

The aim of this work is to reproduce the results of Wang, Tsai and Exler [1] by using the Wang-Landau algorithm on a $2D$ $n \times n$ lattice with periodic boundary conditions and nearest neighbour interactions. The Wang-Landau approach allows one to obtain the DoS (Density of States) of such a wide class of systems, thus providing the ability to compute many thermodynamic quantities at any temperature.

Keywords: Wang-Landau sampling, Ising2D, Monte Carlo, Computed Density of States

Introduction

The algorithm

The Wang-Landau sampling is a method of the Monte Carlo class (and of the Metropolis subclass) which takes a completely different approach with respect to the traditional Monte Carlo algorithm; it does not focus on the achievement of an equilibrium distribution by means of energy minimization nor does it focus on the immediate building of a probability distribution of the states of a system. Instead, it focuses on an equal exploration of all possible energies of a given system model and builds the corresponding Density of States in energy $g(\epsilon)$ during the process.

This provides several advantages such as a shorter computation time, an improved inspection of the available states and the possibility to obtain the values of many thermodynamic quantities as a function of the temperature with an high accuracy, such as the partition function of the system model:

$$\mathcal{Z}(T) = \sum_{\{x\}} g(\epsilon) e^{-\beta \epsilon(x)}$$

where $\beta = 1/T$ and $\{x\}$ represents the set of all possible states of the system.

The aim of this work is to reproduce the results presented in a paper dated 2004 by Landau, Tsai and Exler [1], and possibly extend the results to larger systems. The system under consideration is a 2D Ising system with nearest neighbour interactions wrapped on a torus (i.e. with periodic boundary conditions). A simple introduction to this system can be found on the lecture notes by Maria Peressi [2].

A further goal is to verify the following relation as presented on the webpage of Gould Tobochnik [3][1]

$$c_v \approx c_v^{(}0) \log(L)$$

[1] The webpage can be found at the following address: http://stp.clarku.edu/simulations/.

where c_v denotes the specific heat of the system, $c_v^{(}0)$ denotes its base value and L denotes the linear length of a single edge in the square 2-dimensional Ising lattice.

Program code

The algorithm for Wang-Landau sampling is clearly and comprehensively described in the paper by Landau, Tsai and Exler [1], so only the most notable features are reported here:

- the initial configuration does NOT play a significant role for the subsequent evolution of the state;
- the probability of visiting a state does NOT (necessarily) increase as the energy of such state decrease from the current value; it is instead inversely proportional to the density of states at the state energy (i.e. the higher the density of states, the lower the probability).

Compiling and running the code

The source code can be compiled using the **cmake** utility but it is recommended to compile it using the provided bash script `build.sh` in order to avoid working directory pollution. Some compile-time configuring options, such as the dimensionality of the system and the pseudo-random number generator (PRNG) to be used, can be modified in the file `CMakeLists.txt` prior to the execution of the utility. The script will output the executable (named `wl`) which can the be run.

The program accepts the following options:

Option	Long form	Meaning	Data type
g	mod-factor	Initial modification factor	double
s	shrink-factor	Shrinking exponent	double
t	mod-factor-threshold	Modification factor threshold	double
f	flatness-treshold	Histogram flatness threshold	double
l	edge-length	Length of a square edge	unsigned int
r	seed	PRNG initial seed	unsigned long

All the tests for this work were performed on a MacBookPro9,1[2] on a single core per simulation. Multiple cores have been used to run different simulations in a parallel and independent environment. The code was compiled using `clang`. The output of `clang --version` is:

```
Apple LLVM version 6.1.0 (clang-602.0.53) (based on LLVM 3.6.0svn)
Target: x86_64-apple-darwin14.4.0
Thread model: posix
```

The code was also successfully compiled under `GNU Linux 3.16.0 (Debian and Ubuntu)` and `FreeBSD 10.1` using `clang` from the default repositories.

Notice: `GCC 4.8` currently only offers limited support for the `C++11` standard and refuses to compile the code due to the use of aliases

Output files

The program outputs three ".txt" files:

[2]The specifics are: Intel Core i7 (2.3Ghz, 4 cores); L2Cache (256KB per core); L3Cache (6MB); RAM (8GB); OSX 10.10.4 (14E46); Kernel(Darwin 14.4.0);

- `input.txt`: contains all information fed to the program along with some other initialization value such as the the seed used by the PRNG. This file is useful both for reproducibility and for parsing by analysis software such as Wolfram Mathematica.

- `log.txt`: contains diagnostic information such as the total number of Monte Carlo steps that were performed and the acceptance ratio of the simulation.

- `dos_formatted.txt`: contains the actual result of the simulation in as a formatted array of values. The file is organized as follows:

Column	Description
i	Simple C-style array index.
ϵ	Energy value of the corresponding histogram channel.
$g(\epsilon)$	Density of states at the corresponding energy.
$log(g(\epsilon))$	Natural logarithm of $g(\epsilon)$.

Data analysis

A bash script (named `collect.sh`) is provided along with the source code in order to run multiple simulations (with different or equal parameters) in independent environments, thus allowing one to collect significant statistics on the results.

Since the 2D Ising model is exactly solvable, the exact results (see [4]) were here used in order to compute the actual relative errors on the dataset.

The analysis of the results has been delegated to another software, Wolfram Mathematica being the obvious candidate. The notebook that was used for this analysis is provided along with the source code and the collection script. The notebook provides simple functions to load the generated data and compare it with the exact result thereby coded.

Code structure

Note: This and the following section contain some technical information about the program and can be quietly skipped. It is only advised to read the following if the reader has the intention of reusing/modifying the code for his/her own purpose.

The source code [3] is organized as follows:

- The `System` template class accepts one empty class template argument and is meant to represent the basic, state-independent properties of the system under consideration, such as the dimensionality and the underlying structure of the lattice representation. The template argument represents a name for the system under consideration. [4] The namespace `names::System` is the place in which all the relevant tags should be placed.

- The `State` template class accepts a `System` class and a `tag` class as template arguments (again, only a dummy class for name resolution). It is meant to provide those properties of the system which are not intrinsic properties of the model but rather depend on the current state of the system itself. This includes a `dof` (degrees of freedom) variable along with an `energy()` function and some utility functions which act on on the `dof`s.

[3] The source code is available and freely downloadable on the GitHub platform at the following address:

[4] Such objects are commonly referred to as "tags" as they provide no data and no instance functions and are actually compiled out of the program at an early stage; they are only used to name a specific template specialization.

- The `Algorithm` template class accepts a `System` class and a `Strategy` tag as template arguments and instantiates the appropriate `State` object for subsequent manipulation. This class is meant to provide all functions and properties which are not intrinsic either to the system itself or to its state but are only relevant to the implementation of the integration mechanics. The `Parameters` nested type contains all the simulation parameters that are used by the algorithm during execution. The namespace `names::Algorithm` is the container for all algorithm names.

- Other utility classes such as `HypercubicGrid` included in the `utils` namespace are meant to provide the representation of some underlying structure of the system under consideration. [5]

The source code follows the **tag dispatching pattern** in order to concretize the actual objects to be constructed. Due to the limited time available for the development, there is certainly room for much improvement; furthermore, due to low relevance for the present work, all unneeded classes have been stripped out from the given source tree.

Implementation

The two tags `HypercubicIsing` and `NearestNeighboursInteraction` are fed to the `System` and `State` templates respectively and the resulting `System` class is fed to the `Algorithm` template class in order to construct the full object.

The details of the algorithm can be found in the paper by Landau, Tsai and Exler [1] as only those implementation-dependent parts are explained below in order for the reader to understand how to use the code.

As a first step it is strongly advised to define aliases to the template specializations that will be used and a logging macro (named `mesg`). The terms `System`, `State` and `Algorithm` will here be used to address such specializations.

The user has to instantiate an object of the `System` class, one of the `Algorithm::Parameters` class and a PRNG object to feed the `Algorithm` constructor.

Nothing more is actually needed and the user can begin the evolution straight away by calling the `step(size_t)` function which accepts the number of steps to be performed as an argument. Due to the nature of the Wang-Landau algorithm it is more convenient to call the `sweep(size_t)` function which repeatedly calls `steps(size_t)` and updates the DoS modification factor when the flatness criterium is satisfied.

Tests

After carefully testing each component of the program a very specific set of convergence tests has been chosen. The only differences between any two tests are the size of the system $N = L^2$ (where L is the length of a single edge of the square simulation box) and the flatness threshold t for the histogram of visited energies.

The following properties are common to all tests:

- system dimensionality $D = 2$;

- initial modification factor $f_i = e^1$;

[5] It's worth noting that some properties (such as the dimensionality of the system, the maximum number of allowed spin sites and the pseudo-random number generator [PRNG]) are hard-coded in the program but can be easily modified by changing the corresponding definitions in the "CMakeLists.txt" file before compiling with the **cmake** utility or via the bash script "build.sh". This choice was made thinking that the advantages of being able to compute many dimensional-dependent quantities at compile time greatly overcome the disadvantage of having to rebuild the software from source; moreover, one can easily build one executable for each needed dimensionality, which not often does anyway exceed 3.

- shrinking exponent $s_f = 1/2$;
- modification factor threshold $f_f = 1 + 10^{-8}$;
- number of MonteCarlo steps[6] per sweep $\tau = 10^4$.

The flatness threshold[7] for the histogram of visited energies has been carefully tuned for each simulation: for small sizes of the system ($L = 6, 8, 10, 12, 16$) the values $t = 0.8$, $t = 0.9$, $t = 0.95$ has been tested to yield little differences in both the simulation time and the accuracy of the computed DoS.

For bigger systems (such as $L = 32$ and $L = 64$) the value $t = 0.8$ has been chosen as higher values would considerably increase the simulation time without noticeably improve the accuracy of the final results. The case $L = 64$ presented some problems as the normalized density of states can quickly overflows the capacity of a `double` variable; we have in fact $\max_\epsilon log(g(\epsilon)) \approx 2835$ and $e^{2835} \approx 10^{1231}$ is clearly not representable as a single or double precision standard floating point values, which should be able to store the value.

For this reason, all simulations were run with a quadruple precision `long double` data type instead. On my machine each `long double` takes 16 bytes of memory space. Furthermore, it is possible to always work with the logarithm of the Density of States in order to avoid overflows (at the cost of losing some precision).

For bigger systems still the value has been kept at $t = 0.8$ but the simulation time results exceedingly increased. Since at the time of writing I'm still waiting for the cases $L = 128$ and $L = 256$ to terminate their parallel execution, I cannot speak about the results of such cases.

The normalization constant is in all cases chosen to be such that $g(\epsilon_{ground\ state}) = 2$.

Main test: reproducing Wang, Tsai & Exler results

The mentioned paper shows the results of the Wang-Landau algorithm (using code wrote by the authors themselves[8]) for the case $L = 16$, $t = 0.8$ and with the same other parameters as described above and shows the error by comparison with the exact result by Ferdinand and Fisher [4]. The same approach will here be used.[9].

Main test results

The average overall computation time for this test has been about $1 : 48$ s per simulation. All results were consistent and a quick statistical analysis showed very standard deviations (as low as 10^{-11}).

The simulation results, along with the relative errors with respect to the exact solutions are plotted from Fig.1 to Fig.8. The errors are calculated for any function $F(T)$ as

$$\varepsilon[F(T)] = \left| \frac{F(T) - F_{exact}(t)}{F_{exact}(T)} \right|$$

As can easily be seen there is no way to distinguish by eye sight between the two curves. Approximate values of the (absolute value of the) maximum relative errors are given in the Tab. 1.

[6] Note the fact that *each* Monte Carlo **step** consist of N *moves*, where $N = L^2$ is the total number of sites in the square lattice.

[7] The flatness threshold is a fractional value in the interval $]0, 1[$ and represents the fraction of the *average* value $\tilde{h}$ of the histogram (which we know *a priori* to be equal to the number of performed steps divided by the number of channels in the histogram) such that if **any** histogram channel i yields a value $h[i] < \tilde{h}$ the histogram is considered NOT flat.

[8] The sample code can be found at the following address: `http://ftp.aip.org/epaps//am_j_phys/E-AJPIAS-72-006406/`

[9] Since this paper cannot be easily be obtained, the exact result is reported in App.A

Table 1: Approximate values of the maximum absolute value of the relative errors for $L = 16$

$F(T)$	$\varepsilon[F(T)]$	$F(T)$	$\varepsilon[F(T)]$
$f(T)$	3×10^{-4}	$u(T)$	3×10^{-3}
$s(T)$	10^{-2}	$c_v(T)$	3×10^{-2} $(T > 1/4)$

Apart from a negligibly small region very close to $T = 0$ all errors are reasonably limited. The specific heat is particularly affected by this problem as below $T \approx 1/4$ the relative error becomes as large as 1.

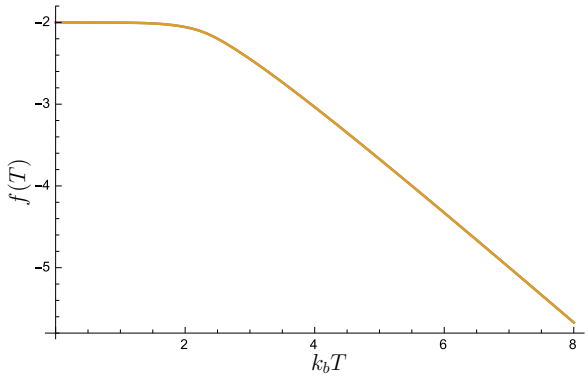

Fig. 1: Computed and exact free energy of the 16×16 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

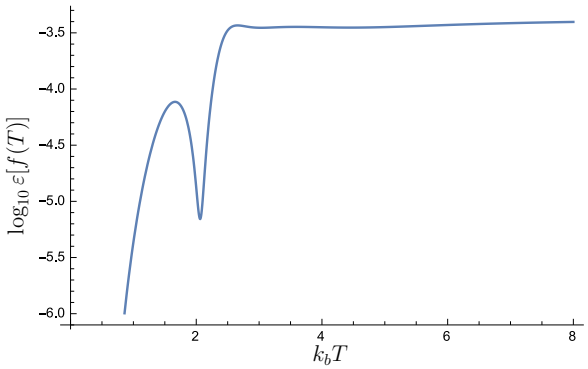

Fig. 2: Base 10 logarithm of the absolute value of the relative error on the free energy, with respect to the exact solution.

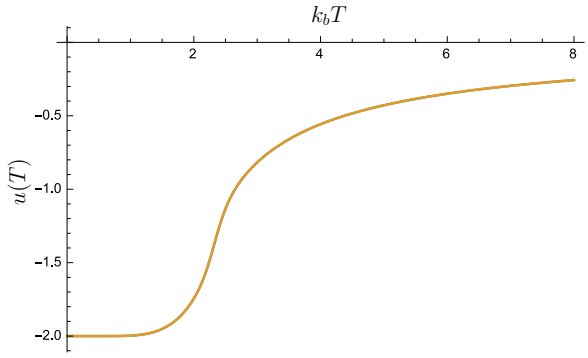

Fig. 3: Computed and exact internal energy of the 16×16 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

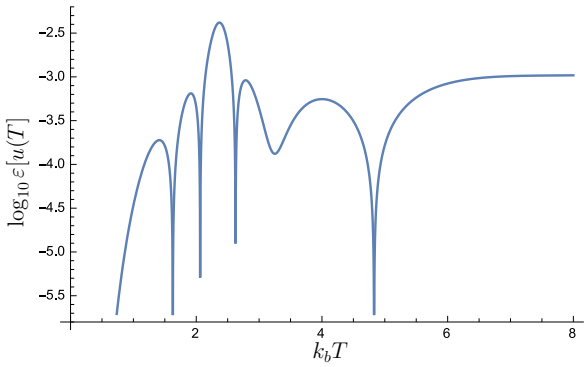

Fig. 4: Base 10 logarithm of the absolute value of the relative error on the internal energy, with respect to the exact solution.

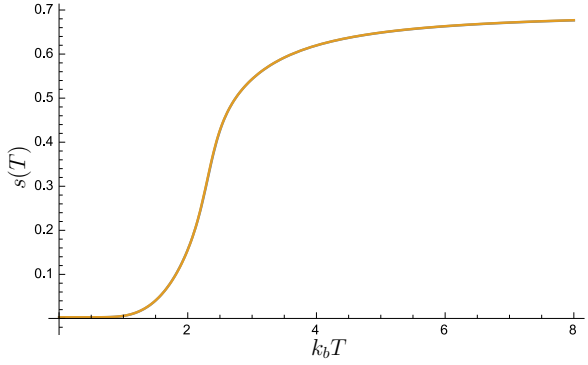

Fig. 5: Computed and exact entropy of the 16×16 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

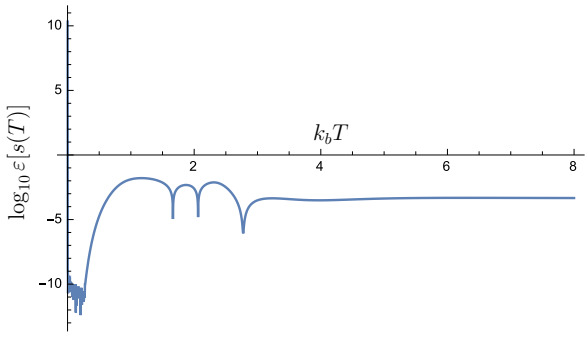

Fig. 6: Base 10 logarithm of the absolute value of the relative error on the entropy, with respect to the exact solution.

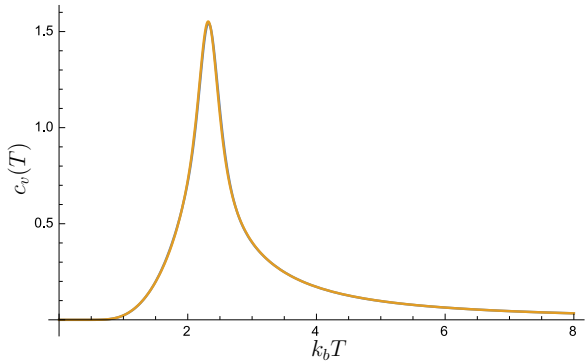

Fig. 7: Computed and exact specific heat of the 16×16 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

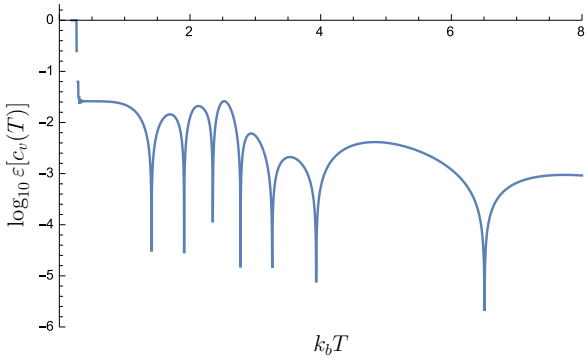

Fig. 8: Base 10 logarithm of the absolute value of the relative error on the specific heat, with respect to the exact solution.

The overall results of this test resemble very closely those obtained by Wang, Tsai and Exler in their paper. The critical temperature (or, better, the maximum of the specific heat) for this case is found at $k_bT_c \approx 2.07$

Other tests: increasing the system size

Other tests were performed by increasing the system size from $L = 16$ to $L = 32, 64, 128, 256$. As mentioned above, the last two tests take a very long time to complete; as a matter of fact some of them are yet not complete at the time of writing (i.e. about 20 hours at this moment). For this reason only the cases $L = 32$ and $L = 64$ are presented below.
For all these tests the flatness threshold has been chosen to be $t = 0.8$ while all the other parameters (except, of course, the seed given to the PRNG) are unchanged.

Other tests results

The average overall computation time for this two tests have been about 23 m and 7 h per simulation for $L = 32$ and $L = 64$, respectively. Again, all results were consistent.
Since the accuracy obtained with the previous test was already satisfying enough, we expect larger systems to yield even better results (at the cost of a much longer simulation time and memory used). We also expect the result to converge uniformly to the exact result for the infinite case, which is given in a paper by Wu, Li and Dai [5] (the final result only is presented in App.A).
As was done for the previous case, the maximum (absolute values of) the relative errors are given in Tab. 2 for $L = 32$ and in Tab. 3 for $L = 64$.

Table 2: Approximate values of the maximum absolute value of the relative errors for $L = 32$

$F(T)$	$\varepsilon[F(T)]$	$F(T)$	$\varepsilon[F(T)]$
$f(T)$	3×10^{-4}	$u(T)$	3×10^{-3}
$s(T)$	10^{-2}	$c_v(T)$	3×10^{-2} $(T > 1/4)$

Table 3: Approximate values of the maximum absolute value of the relative errors for $L = 64$

Table 4: Approximate values of the maximum absolute value of the relative errors for $L = 64$

$F(T)$	$\varepsilon[F(T)]$	$F(T)$	$\varepsilon[F(T)]$
$f(T)$	3×10^{-4}	$u(T)$	3×10^{-3}
$s(T)$	10^{-2}	$c_v(T)$	3×10^{-2} $(T > 1/4)$

The overall results are shown from Fig.9 to Fig.16 for $L = 32$ and from Fig.?? to Fig.?? for $L = 64$.

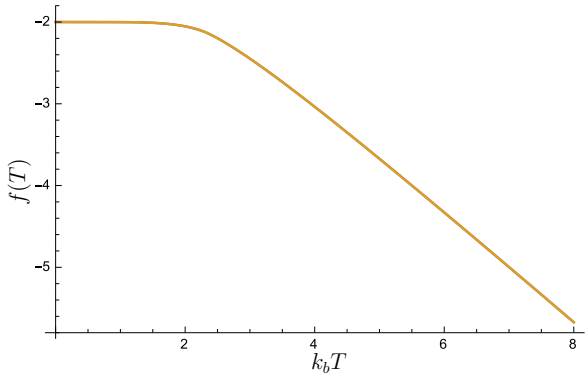

Fig. 9: Computed and exact free energy of the 32×32 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

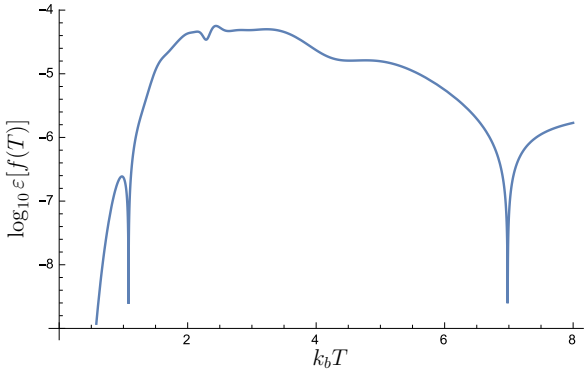

Fig. 10: Base 10 logarithm of the absolute value of the relative error on the free energy, with respect to the exact solution.

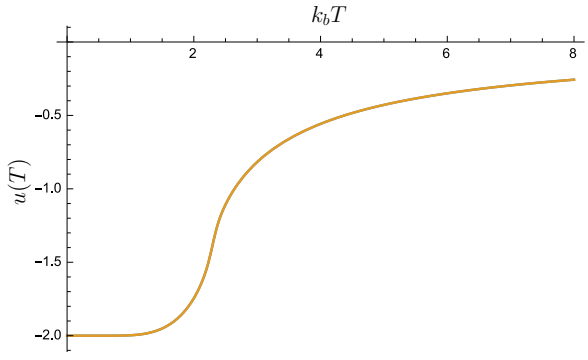

Fig. 11: Computed and exact internal energy of the 32×32 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

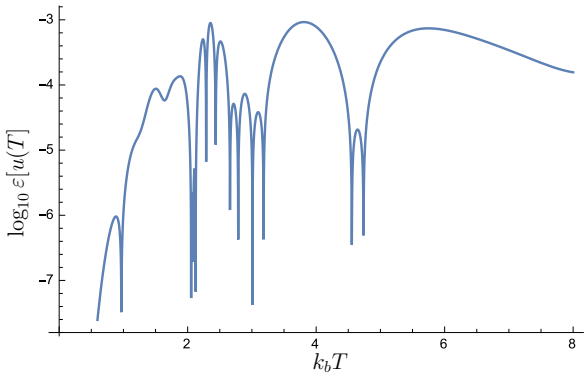

Fig. 12: Base 10 logarithm of the absolute value of the relative error on the internal energy, with respect to the exact solution.

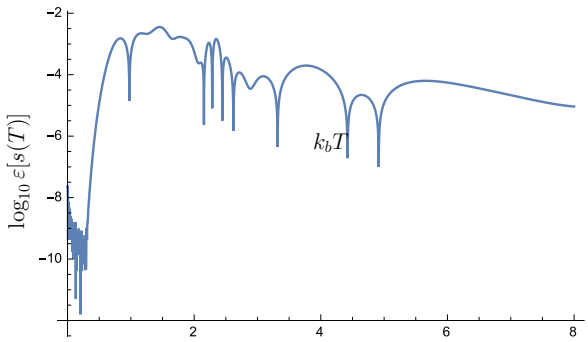

Fig. 14: Base 10 logarithm of the absolute value of the relative error on the entropy, with respect to the exact solution.

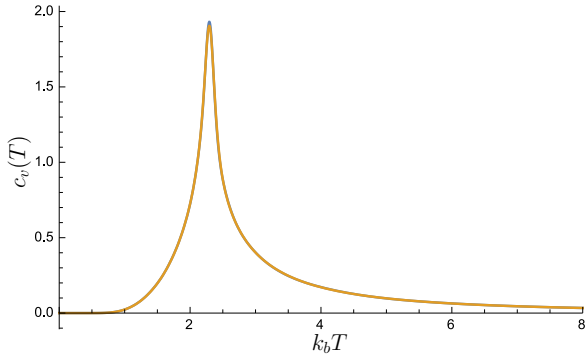

Fig. 15: Computed and exact specific heat of the 32×32 Ising system with nearest neighbours interaction and periodic boundary conditions. The two curves cannot be distinguished by eye sight.

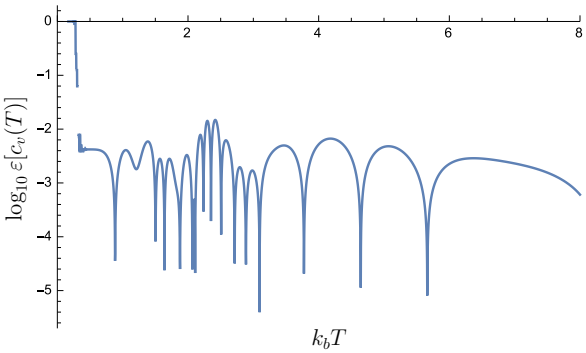

Fig. 16: Base 10 logarithm of the absolute value of the relative error on the specific heat, with respect to the exact solution.

From Fig.15 and Fig.**??** we can see that the peak in the specific heat at the "critical temperature" becomes sharper and sharper as the size of the system increases, as expected. The value of the critical temperature (or, again, the maximum of the specific heat) is found to be $k_bT_c \approx 2.29$ for $L = 32$ and $k_bT_c \approx$ for $L = 64$.

Discussion

A Exact result

The exact result for the partition function of a finite-size 2D $n \times m$ Ising model wrapped on a torus (i.e. with periodic boundary conditions) is given by Ferdinand and Fisher in terms of the

following quantities

$$\begin{cases} \begin{cases} c_l(\beta,n) = \cosh(2\beta J)\coth(2\beta J) - \cos\left(\dfrac{\pi l}{n}\right) \\ \gamma_l(\beta,n) = \log\left(c_l(\beta,n) + \sqrt{c_l^2(\beta,n)-1}\right), \quad l \neq 0 \\ \gamma_0(\beta,n) = (2\beta J) + \log(\tanh(\beta J)) \end{cases} \\ Z_1(\beta,n) = \displaystyle\prod_{r=0}^{n-1} 2\cosh\left(\frac{m}{2}\gamma_{2r+1}(\beta,n)\right) \qquad Z_3(\beta,n) = \displaystyle\prod_{r=0}^{n-1} 2\cosh\left(\frac{m}{2}\gamma_{2r}(\beta,n)\right) \\ Z_2(\beta,n) = \displaystyle\prod_{r=0}^{n-1} 2\sinh\left(\frac{m}{2}\gamma_{2r+1}(\beta,n)\right) \qquad Z_4(\beta,n) = \displaystyle\prod_{r=0}^{n-1} 2\sinh\left(\frac{m}{2}\gamma_{2r}(\beta,n)\right) \end{cases}$$

where J denotes the spin-coupling constant (which is isotropic) and k_b denotes the Boltzmann constant.
The partition function is then

$$\mathcal{Z}(T,n) = \frac{1}{2}\left(2\sinh\left(\frac{2J}{k_b T}\right)\right)^{nm/2} \sum_{i=1}^{4} Z_i(T,n)$$

The exact result for an infinite 2D Ising model is easier to obtain (even tough the computation of the partition function requires a numerical integration). The exact 2D Ising model for the infinite case was solved by Lars Onsager in 1944 and his result is given in a paper by Wu, Li and Dai [5]. We present here only the final result:

$$\begin{cases} \mathcal{Z}(\beta) = \left[\sqrt{2}\cosh(2\beta J)\right]^n e^{I(\beta)} \\ \begin{cases} \kappa(\beta) = \dfrac{2\sinh(2\beta J)}{\cosh^2(2\beta J)} \\ I(\beta) = \dfrac{1}{\pi}\displaystyle\int_0^{\pi/2} \log\left(1+\sqrt{1-\kappa^2(\beta)\sin^2(\phi)}\right) d\phi \end{cases} \end{cases}$$

References

[1] M. Exler D. P. Landau, Shan-Ho Tsai. A new approach to monte carlo simulations in statistical physics: Wang-landau sampling. *Am. J. Phys*, 72, 2004.

[2] Maria Peressi. The ising model in the canonical ensemble. 2014.

[3] J. Tobochnik Gould. Density of states of the 2d ising model, Jun 2010.

[4] Micheal E. Fisher Arthur R. Ferdinand. Bounded and inhomogeneous ising models. i. specific heat anomaly of the finite lattice. *Physical Review*, 185(2):832–846, Sep 1969.

[5] Zheng Dai Xin-Zeng Wu, Di Li. Math 505 project: Ising model - phase transition, May 2014.