FYS3150 – Project 4

Kristian Gregorius Hustad (`krihus`)

November 16, 2016

> Nobody actually creates perfect code the first time around, except me.
>
> *Linus Torvalds*

Abstract

In this report, we study the Ising model for a square lattice. We compare analytical and numerical results, we study the efficiency of our program and the Metropolis algorithm, and we verify numerically Lars Onsager's analytical solution for the critical heat of an infinitely large lattice.

NOTE: All programs and derivations used in them were made in collaboration with Jonas Gahr Sturtzel Lunde (`jonassl`).

GitHub repository at `https://github.com/KGHustad/FYS3150`

1 Introduction

We aim to study the Ising model in for a square $L \times L$ lattice with periodic boundary conditions. We study the case of $L = 2$ and compare analytical and numerical values. We then observe the convergence rate for $L = 20$, and we look at the distribution of energies after reaching the steady state. Finally, we attempt to approximate the critical temperature as L tends to infinity.

Methods are derived in section 2, implementation considerations and results are given in section 3, and finally conclusions are drawn in section 4.

2 Discussion of methods

We have the following expression for the energy of a given state, i,

$$E_i = -J \sum_{<kl>} s_k s_l \tag{1}$$

and its mean magnetization by

$$M_i = \sum_{k}^{L^2} s_k \tag{2}$$

where s_k, s_l are individual spins.

Further, we have the partition function, which is the sum of the energy for all states. We will denote the set of all possible states by $\mathcal{S}$.

$$Z = \sum_{i \in \mathcal{S}} e^{\beta E_i} \tag{3}$$

In general, a $L \times L$ lattice has 2^{L^2} possible states, i.e. $|\mathcal{S}| = 2^{L^2}$

2.1 The case of $L = 2$

2.1.1 Energy and mean magnetization

We will study the case of $L = 2$ and find analycal expressions, which we will later compare to our numerical results.

2.1.2 Expectation values

It is known that the expectation value for the energy, the energy squared, the absolute magnetization [1], the heat capacity and the magnetic susceptibility are respectively

[1] We will not disciminate between positive and negative magnetization

State	Symmetries	Energy (J)	Mean magnetization
↑ ↑ ↑ ↑	1	-8	4
↑ ↑ ↑ ↓	4	0	2
↑ ↑ ↓ ↓	4	0	0
↑ ↓ ↓ ↑	2	8	0
↓ ↓ ↓ ↑	4	0	-2
↓ ↓ ↓ ↓	1	-8	-4

Table 1: All 16 possible states for $L = 2$

$$\langle E \rangle = -\frac{1}{Z}\frac{\partial}{\partial \beta} Z \tag{4}$$

$$\langle E^2 \rangle = \frac{1}{Z}\frac{\partial^2}{\partial \beta^2} Z \tag{5}$$

$$\langle M \rangle = \frac{1}{Z}\sum_{i \in \mathcal{S}} M_i e^{-\beta E_i} \tag{6}$$

$$\langle M^2 \rangle = \frac{1}{Z}\sum_{i \in \mathcal{S}} M_i^2 e^{-\beta E_i} \tag{7}$$

$$\langle C_V \rangle = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2} \tag{8}$$

$$\langle \chi \rangle = \frac{\langle M^2 \rangle - \langle M \rangle^2}{T^2} \tag{9}$$

Using the energy values from table 1, we obtain

$$\begin{aligned} Z = \sum_{i \in \mathcal{S}} e^{\beta E_i} &= 2e^{-8J\beta} + 2e^{8J\beta} + 12 \\ &= 2(e^{-8J\beta} + e^{8J\beta} + 6) \end{aligned} \tag{10}$$

We can insert Z into (4) and (5) to obtain

$$\frac{\partial}{\partial \beta} Z = -16Je^{-8J\beta} + 16Je^{8J\beta} \tag{11}$$

$$\frac{\partial^2}{\partial \beta^2} Z = 128J^2 e^{-8J\beta} + 128J^2 e^{8J\beta} \tag{12}$$

$$\langle E \rangle = -\frac{-16Je^{-8J\beta} + 16Je^{8J\beta}}{2(e^{-8J\beta} + e^{8J\beta} + 6)} = \frac{8Je^{-8J\beta} - 8Je^{8J\beta}}{e^{-8J\beta} + e^{8J\beta} + 6} \tag{13}$$

$$\langle E^2 \rangle = \frac{128J^2 e^{-8J\beta} + 128J^2 e^{8J\beta}}{2(e^{-8J\beta} + e^{8J\beta} + 6)} = \frac{64J^2 e^{-8J\beta} + 64J^2 e^{8J\beta}}{e^{-8J\beta} + e^{8J\beta} + 6} \tag{14}$$

Likewise, we can use the magnetization values from table 1 on the previous page to obtain

$$\langle M \rangle = \frac{1}{Z} \sum_{i \in \mathcal{S}} M_i e^{-\beta E_i} = \frac{1}{Z} \left(8e^{8J\beta} + 16 \right) = \frac{4e^{8J\beta} + 8}{e^{-8J\beta} + e^{8J\beta} + 6} \tag{15}$$

$$\langle M^2 \rangle = \frac{1}{Z} \sum_{i \in \mathcal{S}} M_i^2 e^{-\beta E_i} = \frac{1}{Z} \left(32e^{8J\beta} + 32 \right) = \frac{16e^{8J\beta} + 16}{e^{-8J\beta} + e^{8J\beta} + 6} \tag{16}$$

We now have all we need to find $\langle \chi \rangle$ and $\langle C_V \rangle$, but we will not bother with writing out the expressions here.

3 Implementation and results

3.1 Choice of language

For our implementation, we chose a hybrid Python-C approach – inexpensive operations such as initialization of arrays and extracting the important quantities from the results are done in Python, where we can write short high-level code, while the expensive operations, i.e. the Metropolis algorithm, is carried out in fast C code.

3.2 Random number generation

We use the GNU Scientific Library's implementation of the Mersenne-Twister algorithm for our random number generation. This algorithm satisfies the desired statistical properties and is reasonably efficient[2].

The programs allow the user to fix the seed to a given value. If no seed is provided, the RNG will be seeded from `/dev/urandom`, which is a special file on UNIX-like systems from which random bytes can be read in a thread-safe manner, such that only a single thread can read a byte before a new one is generated.

3.3 Parallelization

The Metropolis algorithm cannot easily (and efficiently) be parallelized without side-effects[3]. One could combine several runs with different seeds to get some kind of average, but such an approach is sub-optimal. However, it is trivial to parallelize multiple runs with differing parameters.

For this purpose, we could have extended our C code with OpenMP or used MPI from Python (which has been shown to achieve similar efficiency to MPI

[2] A minimal benchmark program can be found in `src/c/bench_random.c`

[3] One could be tempted to operate with multiple threads on a single spin matrix, but the issue of maintaining cache coherency would arise, unless, of course, the threads worked on strictly independent parts of the spin matrix, in which case some border spins cannot be flipped. This program suffers from the former issue.

N	μ_E	μ_{E^2}	$\mu_{\lvert M\rvert}$	μ_{M^2}	χ	C_V
1.0E+04	6.1E-04	6.1E-04	5.4E-04	5.8E-04	-5.0E-01	-3.0E-01
1.0E+05	1.2E-04	1.2E-04	4.3E-06	6.2E-05	5.3E-02	-5.9E-02
1.0E+06	3.3E-05	3.3E-05	1.4E-05	2.4E-05	-4.9E-03	-1.6E-02
1.0E+07	2.3E-05	2.3E-05	1.8E-05	2.0E-05	-1.5E-02	-1.2E-02

Table 2: Relative error of numerical results for N sweeps and $T = 1$

from C++ in [1]) but neither OpenMP nor MPI are optimal choices for such a problem. A pool-based parallelization model allows for efficient and automated distribution of tasks at run time. The pool size is usually chosen to be equal to the number of logical processors, and then each worker in the pool will fetch and compute tasks until the pool is empty. We used python's built-in multiprocessing package, which provides a simple API to pool-based parallel processing.

Studying running times, we should not be surprised to find that our parallel program induces no measurable overhead [4] and is hence optimal[5].

3.4 Comparing analytical and numerical results for $L = 2$

From table 2 we see that χ and C_V converges slower than the other quantities. We also see no improvement going from $N = 10^6$ to $N = 10^7$. It appears that 10^6 sweeps over the lattice is sufficient to reach convergence.

3.5 Rate of convergence

After about $2 \cdot 10^5$ sweeps over the lattice, the expectation values seem to have converged. We run our simulations with different input matrices but the same random seed, which mean that the same spins are attempted to be flipped. It appears that sometime before $2 \cdot 10^5$, the spin matrices become equal, and from that point on, they will remain equal. Therefore, we can observe that figure 1a and figure 1b on page 8 are identical.

When the steady state has been reached, we expect the rate of accepted configurations to decline, while the rate is particularly large for a random matrix which is far for the steady state.

For higher values temperatures, one would expect more chaos, and thus more accepted configurations. We see from table 3 on the next page that this is indeed the case.

[4] Strictly speaking, this is only the case for machines that are able to sustain peak clock speed on all cores. Also, hyperthreading is not true parallelism, so in general one cannot expect a speedup greater than the number of cores. For these reasons, a typical laptop will see a higher time usage per job when running in parallel, but that does not imply that our program is suboptimal.

[5] We ran our program with 404 tasks on a machine with 64 logical processors. The computation which completed took 22566 CPU minutes and completed in 357 (wall clock) minutes, meaning 63.12/64 of the time was spent on useful computation (and the rest was spend idling).

T	Homogeneous	Random
1.0	28606	30558
2.4	10864256	10875744

Table 3: Accepted configurations for $L = 2$ after 10^7 sweeps

3.6 Probability distributions

Looking at the distribution of the energy states for $T = 1$, we see that firuge 2a on page 9 is heavily skewed towards the left – almost all the values are all the way to the left. The variance is only about 9.3, which is very reasonable, when we look at the plot.

For $T = 2.4$, the mean is far from the theoretical minimum energy, and as we see in figure 2b on page 9, the distribution of the energies is approximately normal distributed. We find a variance of about 3200, and which is also very reasonable. We could probably get a reasonably good fit by using the mean energy and variance with a normal distribution.

3.7 Critical temperature

From figure 3 on page 10, we see that the peaks seem to converge towards the analytical critical temperature, $\frac{2}{\ln\left(1+\sqrt{2}\right)} \approx 2.269$.

We have

$$T_C(L) - T_C(L = \infty) = aL^{-1/\nu} \tag{17}$$

Inserting for two values L_i, L_j in (17) and subtracting, we obtain

$$\begin{aligned} T_C(L_i) - T_C(L_j) &= a(L_i^{-1/\nu} - L_j^{-1/\nu}) \\ a &= \frac{T_C(L_i) - T_C(L_j)}{L_i^{-1/\nu} - L_j^{-1/\nu}} \end{aligned} \tag{18}$$

Now, selecting our approximations to $T_C(L)$ is somewhat difficult with the significant fluctuations, but a simple solution would be to select the T value corrensponding to the global maximum for each L. Doing this, we find $T_C(100) = 2.277$ and $T_C(140) = 2.275$.

We set $\nu = 1$. Inserting $L_i = 100$ and $L_j = 140$ in (18), we approximate a to

$$\begin{aligned} a &= \frac{T_C(L_i) - T_C(L_j)}{L_i^{-1} - L_j^{-1}} \\ &= \frac{T_C(100) - T_C(140)}{\frac{1}{100} - \frac{1}{140}} \approx 0.7 \end{aligned} \tag{19}$$

Rearranging (17), and inserting $L = 140$ we find

$$\begin{aligned} T_{C\infty} &= T_C(L) - \frac{a}{L} \\ &= 2.275 - \frac{0.7}{140} = 2.270 \end{aligned} \tag{20}$$

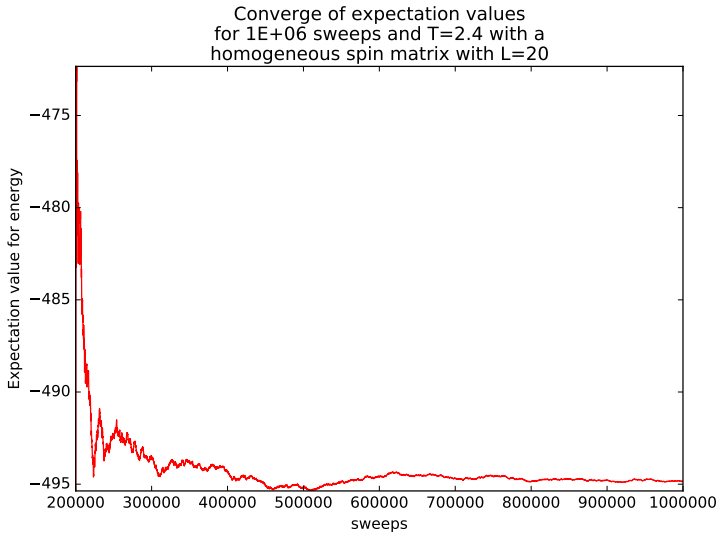
We are quite content with an error of 0.001 from the analytical value.

4 Conclusion

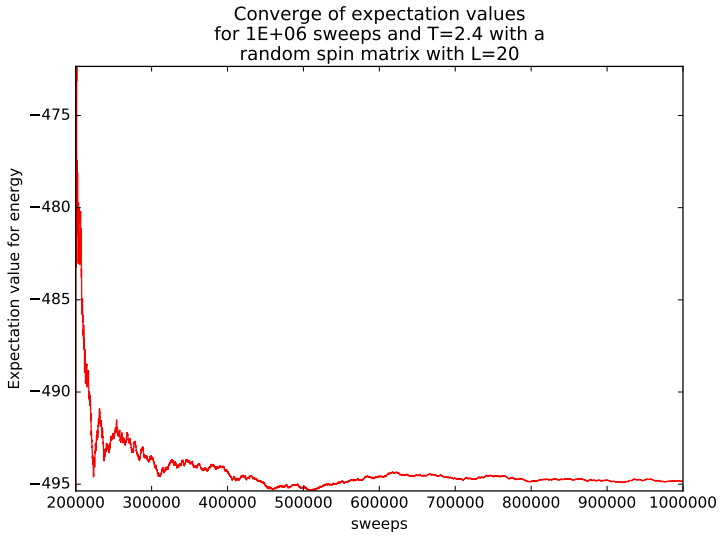
We have found an agreement betweeen the analytical solutions and our program for $L = 2$, we saw that the quite a lot of sweeps are needed to achieve convergence, and we found our approximated critical heat to be in agreement with the analytical solutions of Lars Onsager.

References

[1] M. Mortensen and H. P. Langtangen, "High performance Python for direct numerical simulations of turbulent flows," *Computer Physics Communications*, vol. 203, pp. 53–65, 2016. [Online]. Available: https://www.duo.uio.no/handle/10852/50300

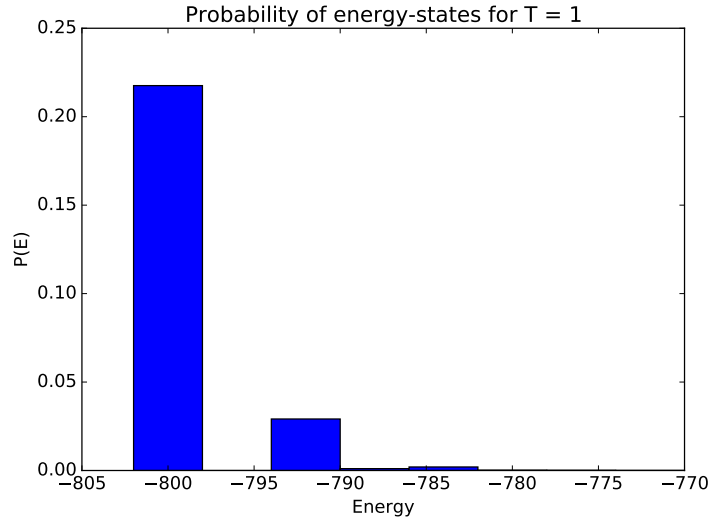


(a) Homogeneous

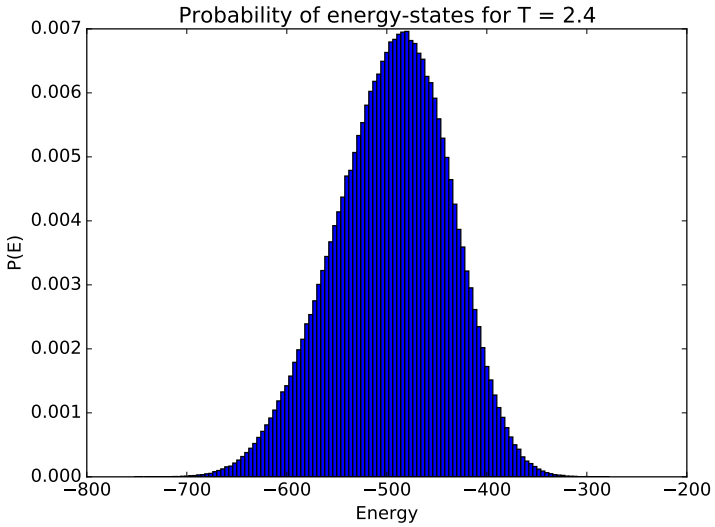


(b) Random

Figure 1: Convergence of the expectation value for the energy



(a) $T = 1$



(b) $T = 2.4$

Figure 2: Distribution of the energies after a steady state has been reached with $L = 20$

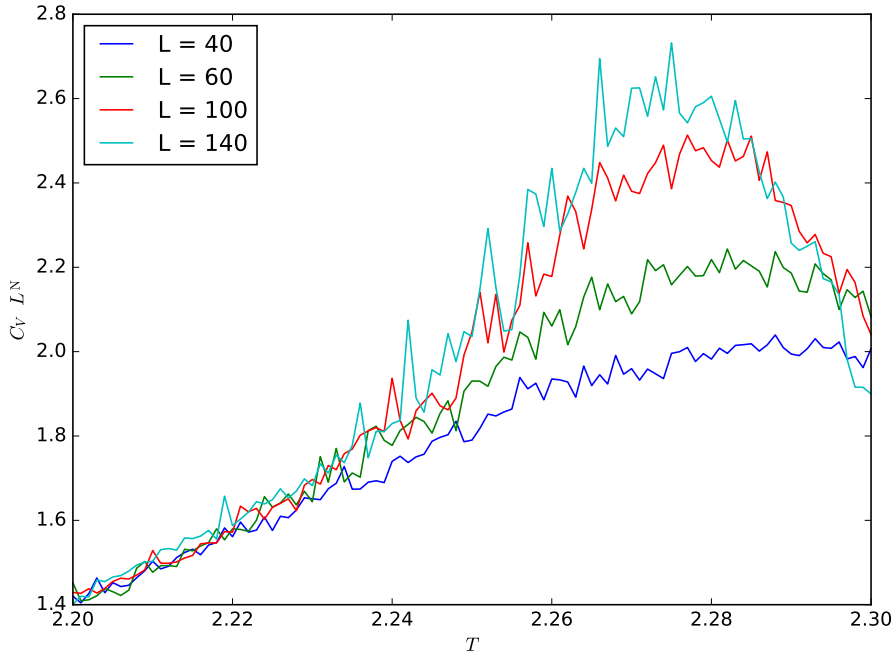


Figure 3: The expectation value for C_V of the sweeps $\in [10^6, 2 \cdot 10^6]$ with $\Delta T = 0.001$

5 Appendix

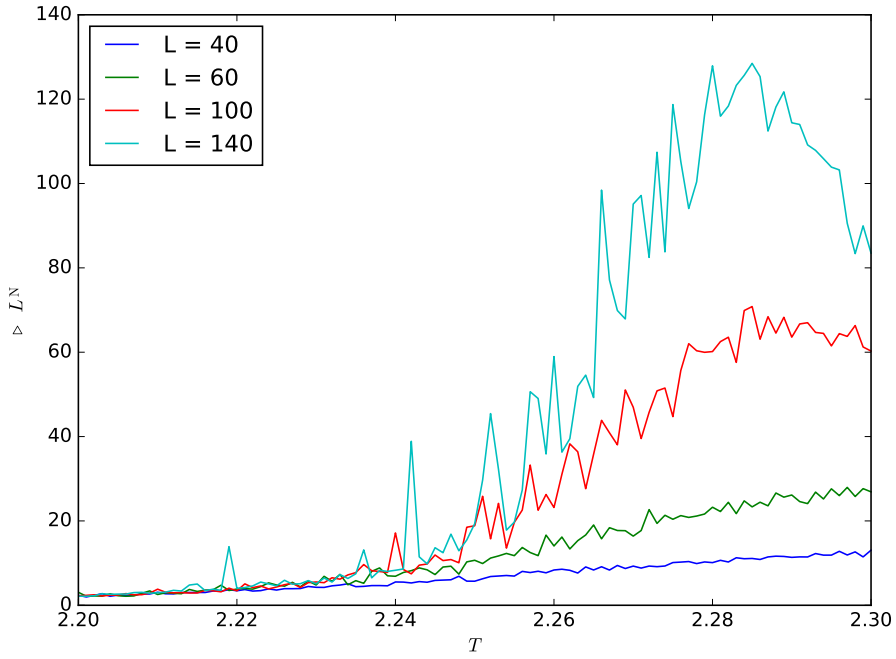


Figure 4: The expectation value for χ of the sweeps $\in [10^6, 2 \cdot 10^6]$ with $\Delta T = 0.001$

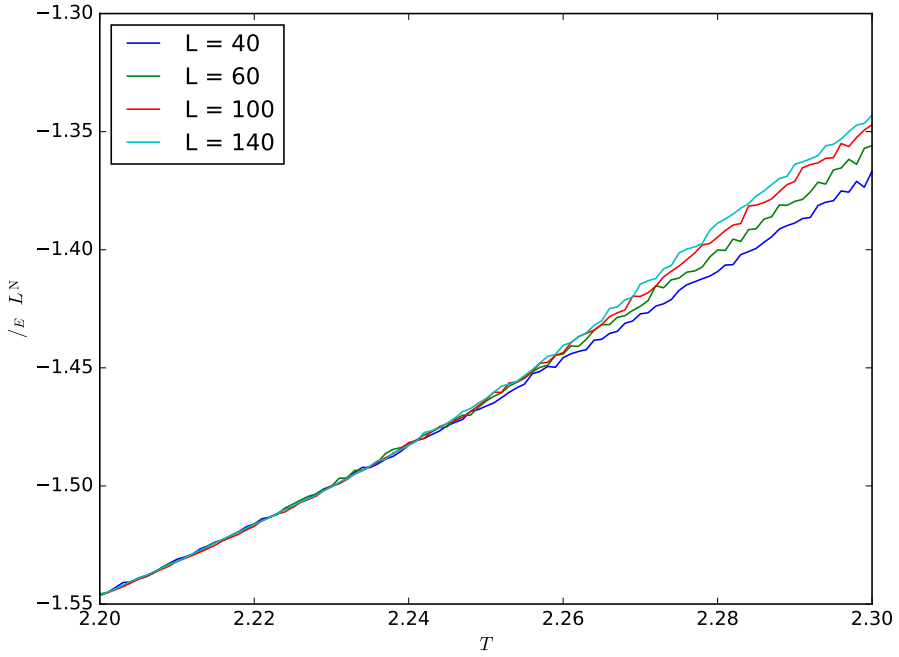


Figure 5: The expectation value for the mean energy of the sweeps $\in [10^6, 2{\cdot}10^6]$ with $\Delta T = 0.001$

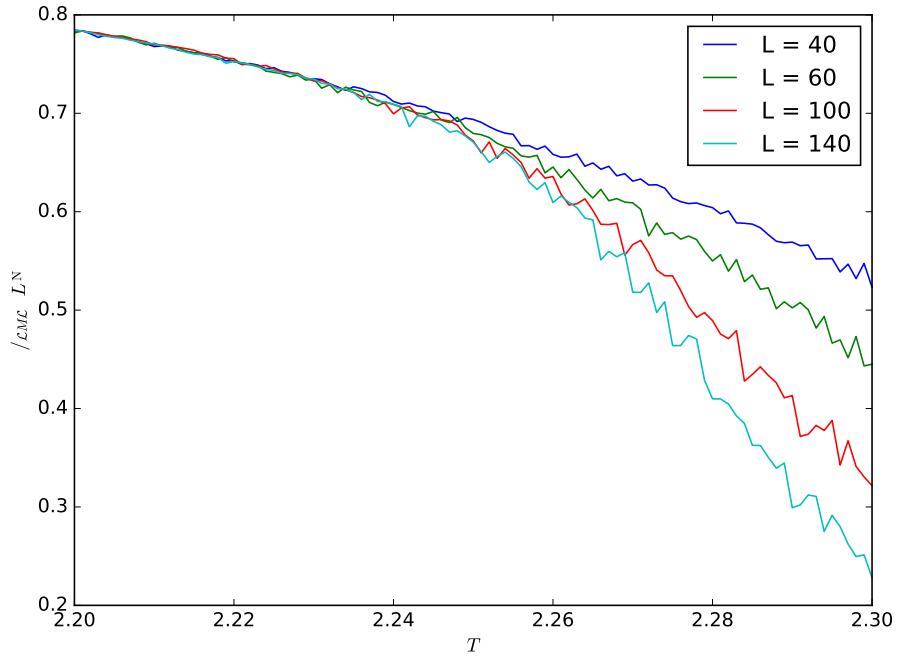


Figure 6: The expectation value for the absolute magnetization of the sweeps $\in [10^6, 2 \cdot 10^6]$ with $\Delta T = 0.001$