Studies of phase transitions in magnetic systems

Vegard Falmår and Sigurd Sørlie Rustad

University of Oslo
Norway

November 23, 2020

CONTENTS

I. INTRODUCTION

For our studies we have used c++ for heavy computation, python for visualization and bash for automation. All the code along with instructions on how to run it, can be cloned from our GitHub repository here[1].

II. THEORY

Statistical terms

In this report we are going to use some basic statistical terms and expressions. For those not familiar with terms like standard deviation and mean value, they are covered below.

The expectation value or mean value, often written as $\langle A\rangle$, is the sum of all values A_i, divided by the total number N of values it can have:

$$\langle A\rangle = \frac{1}{N}\sum_{i}^{N} A_i$$

However, when given a probability distribution P_i, which describes the probability of having outcome A_i, one can also find the expectation value through

$$\langle A\rangle = \sum_{i}^{N} A_i P_i.$$

Variance or standard deviation (σ_A), is a measurement of the variation in a set of data A_i. The mathematical expression is given as

$$\sigma_A = \sqrt{\frac{1}{N-1}\sum_{i}^{N}(A_i - \langle A\rangle)^2} = \sqrt{\langle A^2\rangle - \langle A\rangle^2},$$

where N is the total number of outcomes and $\langle A\rangle$ is the expectation value of A_i.

Canonical ensemble

The probability of finding a system in a given microstate is found through the canonical ensemble, given by equation (1) (see [2] chapter 13.2.2).

$$P_i(\beta) = \frac{\exp(-\beta E_i)}{Z}, \quad \beta = \frac{1}{k_B T} \tag{1}$$

Here $P_i(\beta)$ is the probability of finding the system with energy E_i and temperature T in Kelvin. k_B is Boltzmann constant and Z the partition function given by

$$Z = \sum_{i=1}^{M} \exp(-\beta E_i). \tag{2}$$

Where M is the total number of microstates.

The canonical ensemble and partition function is usually hard to find, however, when obtained we can use them to find many useful relations. Below we list the expressions (without derivation) we need will need in the report. Everything is from [2] chapter 13.2.2.

The mean energy $\langle E\rangle$ given as

$$\langle E\rangle = \frac{1}{Z}\sum_{i=1}^{M} E_i \exp(-\beta E_i), \tag{3}$$

and the mean square energy ($\langle E^2\rangle$) is calculated by

$$\langle E^2\rangle = \frac{1}{Z}\sum_{i=1}^{M} E_i^2 \exp(-\beta E_i). \tag{4}$$

Mean absolute value of the magnetic moment $\langle |M|\rangle$

$$\langle |M|\rangle = \frac{1}{Z}\sum_{i=1}^{M} |M_i| \exp(-\beta E_i), \tag{5}$$

and mean square magnetic moment ($\langle M^2\rangle$) by

$$\langle M^2\rangle = \frac{1}{Z}\sum_{i=1}^{M} M_i^2 \exp(-\beta E_i). \tag{6}$$

With this we can also find the susceptibility χ

$$\chi = \beta\sigma_M, \quad \sigma_M = \sqrt{\langle M^2\rangle - \langle |M|\rangle} \tag{7}$$

Where σ_M is the variance of $|M|$. Specific heat capacity at constant volume C_V is given by

$$C_V = \frac{\beta}{T}\sigma_E, \quad \sigma_E = \sqrt{\langle E^2\rangle - \langle E\rangle} \tag{8}$$

σ_E is the variance of E.

Two dimensional lattice

From [1] the energy in a 2D lattice with no external magnetic field is given by

$$E = -J\sum_{<kl>}^{N} s_k s_l, \tag{9}$$

where $s_k = \pm 1$ (representing the spin direction), N the total number of spins and J a coupling constant indicating the strength of the interaction between neighboring spins. The $< kl >$ means that we sum over the nearest neighbors.

[1] github.com/sigurdru/FYS3150/tree/master/Project3

III. METHODS

In this report we are only going to use periodic boundary conditions, meaning the edges are neighbors. For a square peace of (stretchy) paper, this would look like first folding it into a cylinder and then into a donut-shape.

We are going to simulate a 2D lattice with periodic boundary conditions. This means that the neighbour to the right of s_N takes the value of s_0 and the neighbour to the left of s_0 takes the value of s_N.

A. Testing of algorithm

In order to make sure our algorithm is running correctly, we want to test it. In order to do this, we are going to compare it with analytical results, namely a 2x2 lattice.

By using equation (9) and testing all 16 different combinations for a 2D lattice, we have made a table (see I) that shows all the possible energies and magnetizations, as well as the multiplicity of each configuration (marked as degeneracy). With this we can find the analytical term of the partition function (Z). Reading the values from table I and using equation (2) we get:

Number spins up	Degeneracy	Energy, [J]	Magnetization
4	1	-8	4
3	4	0	2
2	4	0	0
2	2	8	0
1	4	0	-2
0	1	-8	-4

Table I. Table showing the energy, multiplicity and magnetization of different configurations of spins in a 2×2 2D-lattice with periodic boundary conditions.

$$\begin{aligned} Z &= \sum_{i=1}^{16} \exp(-\beta E_i) \\ &= 12 + 2\exp(8\beta) + 2\exp(-8\beta) \\ &= 12 + 4\cosh(8\beta). \end{aligned} \tag{10}$$

With the partition function and the canonical ensemble through equation (1), we can find a lot of useful values. With equations (3-8) we can find expected energy $\langle E \rangle$, mean absolute value of the magnetic moment $\langle |M| \rangle$, susceptibility χ and specific heat capacity at constant volume C_V.

IV. RESULTS

V. DISCUSSION

VI. APPENDIX

A. Algorithm specific optimization

In order to make our algorithm run faster, we do some optimization. We list the optimizations we implemented here, to make our code easier to understand and as a guide for anyone who wants to do something similar.

Through table I, we already know the possible energy differences (ΔE) in the lattice before we start the simulation.

- $\Delta E = \pm 16$ J for the difference between 8 J and -8 J (both ways)
- $\Delta E = \pm 8$ J for the difference between ± 8 J and 0 J (both ways)
- $\Delta E = 0$ J

We can thus compute and store the different values of $e^{-\beta \Delta E}$ beforehand to avoid making these computations every time we update the energy.

As mentioned above, we are going to use periodic boundary conditions. The easiest, but slow way of implementing this, is with if-tests. However we did it with a simple function (in c++),

```
int PeriodicBoundary(int i, int limit, int add) {
    return (i+limit+add) % (limit);
}
```

that takes the current index you are evaluating (`i`), the total size (`limit`) and the amount you want to go forward and backward (`add`). Then we sum all the arguments and take the rest, reassuring us we always arrive at the right index, or if we are at the edge, wrap back around.

[1] Department of Physics, Univeristy of Oslo, Fall semester 2020, Computational Physics I FYS3150/FYS4150, Project 4.

[2] Morten Hjorth-Jensen, Computational Physics, Lecture Notes Fall 2015, August 2015, https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf.