Interpolating Monte Carlo Data using the multiple histogram method

Goethe-Universität Frankfurt am Main
Institute for Theoretical Physics
January 2020

Bachelor Thesis
for gaining the academical grade
Bachelor of Science

David Hyun-Jin Leemüller

1st supervisor: Prof. Dr. Owe Philipsen
2nd supervisor: Dr. Alessandro Sciarra

Selbstständigkeitserklärung

Erklärung nach §30 (12) Ordnung für den Bachelor- und den Masterstudiengang

Hiermit erkläre ich, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderen fremden Texten entnommen wurden, sind von mir als solche kenntlich gemacht worden. Ferner erkläre ich, dass die Arbeit nicht - auch nicht auszugsweise - für eine andere Prüfung verwendet wurde.

Datum:_____ Unterschrift:_____

Abstract

In this work the probability distributions of observables from Monte Carlo simulations will be interpolated into parameters such as temperature by using the multiple histogram method. The main part of the work consists of implementing the algorithm and testing the code extensively. Afterwards, the resulting software will be used to find out the order of a phase transition of a system in $N_f = 2$ *lattice quantum chromo dynamics*. The results are presented graphically from which the conclusions are drawn.

Table of Contents

1 | Introduction

Most problems in physical systems are complex and do not offer analytic solutions. Additionally, theoretical models often cannot be tested experimentally or testing them would be costly, which is why physicists choose to simulate systems on a computer to circumvent this issue.

One well-known method is the Monte Carlo method. It uses random sampling to simulate statistical systems, where brute force approaches cannot be applied. With this method, one can study thermodynamical properties of a system that are not easily predictable. These calculations are usually very costly and require a long computation time. The simulations from which the data will be analyzed in this work took more than 9 months each. For example, locating phase transitions requires to have information about the system at various temperatures in order to be certain about exactly when the transitions occur. Hence, studying phase transitions by using Monte-Carlo only would be quite cumbersome, because it would need a large number of simulations to run in order to draw meaningful conclusions. Understanding the order of the phase transition is even more difficult and it requires larger statistics.

One very efficient way to bypass the problem of having to simulate a large number of simulations at different parameters is to use the results of simulations to deduce the properties of the system in other conditions (e.g. varying pressure, temperature,...). With only one simulation, one can extrapolate the results to nearby values of parameters, but this is often not accurate. Having more than one simulation, it it possible to interpolate the results between the simulations. In the above example this would mean to simulate the system only at a few temperatures over a certain range, where one expects to find a phase transition and then to interpolate for temperatures in between. This method of deduction via extrapolation of the results from one simulation is called the ***single histogram method*** and was first introduced by *Ferrenberg* and *Swendsen* in 1988 [1]. In 1989, they published the method of deduction via interpolation of the results from multiple simulations, which is called the ***multiple histogram method*** [2]. It is called the histogram method since the calculations can be rewritten so that

it uses the histogram $N(O)$, where O is an observable and $N(O)$ the number of times O was measured in the simulation. Normalizing the histogram will yield the probability distribution of that observable in the system. This method allows us after performing a simulation to study the properties of a system within days or even hours, which makes it a very efficient and necessary tool to use in simulation physics. Being able to reduce the number of simulations to be performed from many to just a few is a great help of handling resources on super computers that are often limited. The generality of the method also helps it to be a versatile tool in probing any physical systems.

The method of interpolating the first 4 moments of an observable was already implemented in the "`Monte Carlo Cpp analysis tools`" codebase of the research group of Prof. Dr. Owe Philipsen. The goal of this thesis is to add to the codebase an algorithm that uses the multiple histogram method, which will be explained in chapter 2, to interpolate the normalized histograms, i.e. the probability distributions $P(O)$ in order to have more information about the system.
How to derive that algorithm from the multiple histogram method for observables will then be discussed in chapter 3. In Chapter 4 and 5 the code will be first used to test the correctness of the implementation and then to study the behavior of a system which undergoes a phase transition to find out the type of phase transition. The procedure of how to identify the type of phase transition by looking at probability distributions will also be discussed in Chapter 5.
A short summary of the work and an outlook for possible next steps will be given in the last chapter.

2 | Analysing Monte Carlo Data

There are many ways to analyze data from Monte Carlo simulations. As mentioned in the introduction, this work focuses on the method of interpolating simulated data into a parameter. The multiple histogram method is naturally based on the single histogram method, which therefore will be discussed in the first section. Most of the following equations and derivations are done for the internal energy as the observable and the temperature as the interpolated parameter but can be done for any other observables or parameters. At the end a generalized form of the equation is shown, which is also implemented in the codebase.

2.1 The single histogram method

The single histogram method allows us to take the data from a Monte Carlo simulation at a specific temperature and extrapolate it to nearby temperatures. This technique is used to save CPU time by not having to perform simulations at other temperatures. The following descriptions and notations were taken from Chapter 8.1 of [3].
Let us focus on a statistical spin system without less of generality, since the method is general. For such a system with states μ, one can estimate an observable Q by performing a weighted average with the weight being the Boltzmann probability.

$$\langle Q \rangle = \frac{\sum_\mu Q_\mu e^{-\beta E_\mu}}{\sum_\mu e^{-\beta E_\mu}} \tag{2.1}$$

For a larger system it is not possible to sum over every single state. One can only average over a subset of states, allowing for an inaccuracy to the calculation. Once chosen a random subset from a probability distribution p_μ it follows

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{j=1}^{M} p_{\mu_j}^{-1} e^{-\beta E_{\mu_j}}}. \tag{2.2}$$

Q_M is called the *estimator* of Q and M the number of states in the subset. The greater M becomes, the closer Q_M gets to $\langle Q \rangle$. Most of the times p_{μ_i} are chosen to be the Boltzmann weights for the temperature $\beta = 1/kT$ one is interested in.
Now consider the case where p_{μ_i} is the Boltzmann weight near the inverse tem-

perature β_0 one is interested in:

$$p_{\mu_i} = \frac{e^{-\beta_0 E_{\mu_i}}}{Z_0}, \tag{2.3}$$

where Z_0 is the partition function at the temperature T_0. Using this in Eq.(2.2) one gets

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} e^{-(\beta-\beta_0)E_{\mu_i}}}{\sum_{j=1}^{M} e^{-(\beta-\beta_0)E_{\mu_j}}}. \tag{2.4}$$

With this equation, once performed a Monte Carlo Simulation, it is possible to use the measurements Q_{μ_i} at temperature β_0 and estimate Q_M at another temperature β. It is worth noting that E_{μ_i} is the total energy per state and not the energy per spin. Eq.(2.4) can be rewritten in order to use histograms that store the data from the simulation. Consider the internal energy as the observable.

$$U = \langle E \rangle = \frac{\sum_E E N(E) e^{-(\beta-\beta_0)E_0}}{\sum_E N(E) e^{-(\beta-\beta_0)E_0}} \tag{2.5}$$

Now the sum runs over all possible energy levels of the states where $N(E)$ is the number of times a state with energy E was measured during the simulation. $N(E)$ is called a histogram. The upside of using the histogram is that it is not proportional to the size of the simulation and thus does not take much space in the case of a large system.
That upside, however, comes with a disadvantage that it incurs loss of precision because of approximating nearby energies into one bin. But since disk space is easily accessible these days using Eq.(2.4) would be the better method of estimation.

In Figure 2.1, it is clearly observed that the extrapolation is quite accurate within a close range, marked by two dashed lines but becomes less precise as temperatures in consideration are farther apart.

When the observable is the internal energy, the histogram and Boltzmann weights both are function of E only. Beyond that, the single histogram method can be applied for other observables as well. To that end, Eq.(2.5) is generalized for

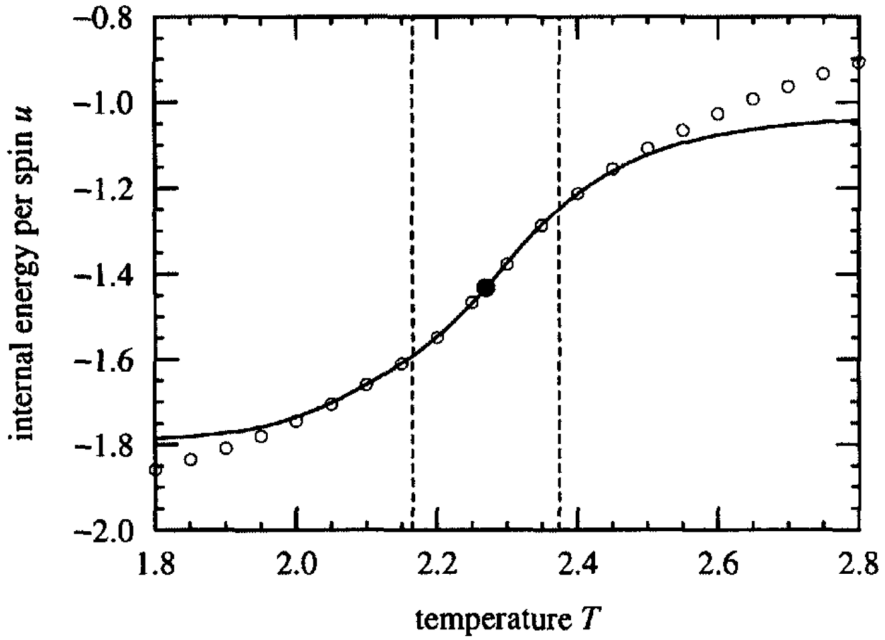

Figure 2.1: This is an example for the accuracy of the single histogram method, which was performed for a 32x32 two-dimensional Ising model with J = 1. The black dot in the middle is the simulation where the data for the extrapolation is from, while solid lines are the extrapolated results and the white dots simulations to compare the results. The vertical lines mark the range, where the single histogram method is accurate. Figure taken from [3].

another observable Q:

$$Q = \langle E \rangle = \frac{\sum_{E,Q} QN(E,Q)e^{-(\beta-\beta_0)E_0}}{\sum_{E,Q} N(E,Q)e^{-(\beta-\beta_0)E_0}}. \tag{2.6}$$

N(E,Q) is now a two-dimensional histogram and gives the number of times a state with energy E and observable value Q was measured. But storing a two-dimensional histogram for a large system has millions of bins and therefore can get very inconvenient to calculate. So just using Eq.(2.4) is the better route.

2.1.1 Extrapolating other variables

So far the histogram method was only introduced with the extrapolation being done for the temperature. As mentioned earlier, the methods can be used for any other variable and also for multiple variables. If the Hamiltonian can be written

in the form

$$H = \sum_k J^{(k)} E^{(k)},$$

one can extrapolate into other variables $J^{(k)}$. $J^{(k)}$ are coupling constants that are scaling the energies in the Hamiltonian and $E^{(k)}$ are dimensionless functions. The generalized form of Eq.(2.2) is:

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} \exp\Big(-\beta \sum_k J^{(k)} E_{\mu_i}^{(k)} \Big)}{\sum_{j=1}^{M} p_{\mu_j}^{-1} \exp\Big(-\beta \sum_k J^{(k)} E_{\mu_j}^{(k)} \Big)} \tag{2.7}$$

Suppose that the Monte Carlo Simulation is run at $\left\{J_0^{(k)}\right\}$, which represents a point in the parameter space while the point of interest is $\left\{J^{(k)}\right\}$. Then the probabilities are

$$p_{\mu_i} = \frac{1}{Z_0} exp\Big(-\beta \sum_k J_0^{(k)} E_{\mu_i}(k) \Big).$$

Using that in Eq.(2.7) one gets:

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} \exp\left[-\beta \sum_k \left(J^{(k)} - J_0^{(k)} \right) E_{\mu_i}^{(k)} \right]}{\sum_{j=1}^{M} \exp\left[-\beta \sum_k \left(J^{(k)} - J_0^{(k)} \right) E_{\mu_j}^{(k)} \right]} \tag{2.8}$$

Consider that now one cannot only measure the total energy of the states but has to measure each $E^{(k)}$ for every sample.

2.1.2 Implementation

Although the single histogram method is easy to implement there is one problem that exists for large systems. The energies in the exponent are the total energies of the states, which means that they grow with the size of the system. Eventually the can produce an over- or underflow on the computer by exceeding the range of numbers it can represent. One way to decrease the probability of that happening is to shift all energies by a constant amount E_0 because the measurements do not depend on the origin of the energies. If the measured energies are all subtracted by the mean energy $E_0 = \langle E \rangle$, the range of energies only increases with $\sqrt{N}$ with N being the size of the system.
However, for a very large system, shifting may not be sufficient. For this case, one

can calculate the logarithms of the terms of Eq.(2.4), which will not overflow the system. Calculating the logarithm of the sum trivially does not work, since one still has to calculate the exponents first, which is why one more trick is needed. Consider two terms e^{x_1} and e^{x_2} with $x_1 > x_2$. The logarithm of the sum would then be:

$$\log(e^{x_1} + e^{x_2}) = \log(e^{x_1}(1 + e^{x_2 - x_1})) = x_1 + \log(1 + e^{x_2 - x_1}) \tag{2.9}$$

The exponent in the logarithm is much smaller now such that there will never be an overflow. For the case of negative observables, however, one cannot take the logarithm of the numerator of Eq.(2.4). This issue is handily resolved by shifting the observables so that every observable is greater or equal to zero or by taking the absolute value. However, taking the absolute requires to store all the signs of the observables, while shifting only requires to store the amount the observables were shifted. Note that when shifting the observables, that they have to be shifted back after the calculations.

If one has performed multiple simulations at different temperatures, one can use an even better method with better results for a wider range. This method is called the multiple histogram method and will be introduced in the next chapter.

2.2 The multiple histogram method

In order to estimate the values of an observable over a parameter range larger than the range one can extrapolate with the single histogram method, one must perform more simulations at different values of the parameter. Then one can use the multiple histogram method to interpolate between the simulations, provided that they are not too far from the to-be-interpolated parameter values. This will be explained in more detail later in section 2.2.2. This method is introduced here in the format of chapter 8.2 of [3] where the parameter is the temperature.

Every simulation can give an estimate of the observable $\langle Q \rangle$ at other temperatures using the single histogram method. The further the temperature gets away from the one from the simulation, the worse the estimate becomes. One can still use every estimate from every simulation and somehow combine them to get a more precise result. A weighted average of single histogram extrapolations of each simulation does not give good results unless the quality of the simulations is very high, i.e. high statistics. That is why another approach is taken involving the density of states. The goal is to get a good estimate for the density of states, from which one can interpolate any observable. This will be shown at the end of this section.

Consider the simplest case, where the observable is the internal energy. Then the probability for measuring a state with energy E for a Boltzmann sampled simulation is

$$p(E) = \rho(E)\frac{e^{-\beta E}}{Z}, \tag{2.10}$$

with $\rho(E)$ being the energy density of states, which gives the number of states of a system with energy E. Z is again the partition function of the system. For a simulation with n measurements and a histogram $N(E)$ one can approximate $p(E)$.

$$p(E) = \frac{N(E)}{n} \tag{2.11}$$

By combining both expressions (2.10) and (2.11) one gets

$$\rho_i(E) = \frac{N_i(E)}{n_i}\frac{Z_i}{e^{-\beta_i E}}. \tag{2.12}$$

The index i stands for the corresponding simulation at β_i. The density of states is a function that only depends on the system and not on the temperature. That means every $\rho_i(E)$ estimates the same function; hence one can use them together to get a more accurate estimate for the density of states. Each simulation gives a good estimate for $\rho(E)$ in the range where $N_i(E)$ is large and a poor estimate where $N_i(E)$ is small. The histograms $N_i(E)$ have a large number of samples close to the temperatures β_i. As the temperature moves further away from β_i, the number of samples decreases and an overlap with the other histograms can be observed. This implies close to β_i the histogram $N_i(E)$ gives a good estimate for the density of states and thus should wield a bigger influence to the calculation than the other histograms. For this reason, one can perform a weighted average, depending on the number of the samples in each region.

Consider a number of measurements x_i of a quantity x with a standard deviation σ_i. x can be then estimated as

$$\overline{x} = \frac{\sum_i x_i/\sigma_i^2}{\sum_j 1/\sigma_j^2}. \tag{2.13}$$

$\overline{x}$ is a weighted average with the inverse of variance as the weights. The measurements are nearly independent, so the error should be Poissonian, i.e. the error scales with the square root of $N_i(E)$:

$$\Delta N_i(E) = \sqrt{\overline{N_i(E)}}$$

$\overline{N_i(E)}$ is the average histogram calculated with the histograms of many simulations at the same temperature. With $\overline{N_i(E)}$ one could also calculate the exact density of states $\rho(E)$:

$$\rho(E) = \frac{\overline{N_i(E)}}{n_i} \frac{Z_i}{e^{-\beta_i E}} \tag{2.14}$$

However, $\rho(E)$ only becomes exact when there are infinite histograms at temperature β_i. This is infeasible but this expression is still useful in the analysis. Since the error of $\rho(E)$ only comes from $N_i(E)$, one can find an expression for the error of $\rho(E)$.

$$\sigma_i = \frac{\Delta N_i(E)}{n_i} \frac{Z_i}{e^{-\beta_i E}} = \frac{\sqrt{\overline{N_i(E)}}}{n_i} \frac{Z_i}{e^{-\beta_i E}}$$

The variance is

$$\sigma_i^2 = \frac{\overline{N_i(E)}}{n_i^2}\left[\frac{Z_i}{e^{-\beta_i E}}\right]^2 = \frac{\rho^2(E)}{\overline{N_i(E)}}.$$

In the second step Eq.(2.14) was substituted in. This shows that the weights for the estimate of the density of states are indeed proportional to the $\overline{N_i(E)}$.
Now, using Eq.(2.13) with x being the density of states, the weighted average of $\rho(E)$ is

$$\rho(E) = \frac{\sum_i \overline{N_i(E)}\,[N_i(E)/n_i]\left[Z_i/e^{-\beta_i E}\right]}{\sum_j \overline{N_j(E)}} = \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{-\beta_j E}}. \tag{2.15}$$

In the last step, Eq.(2.14) was substituted in again to get rid of $\overline{N_i(E)}$, which is unknown.

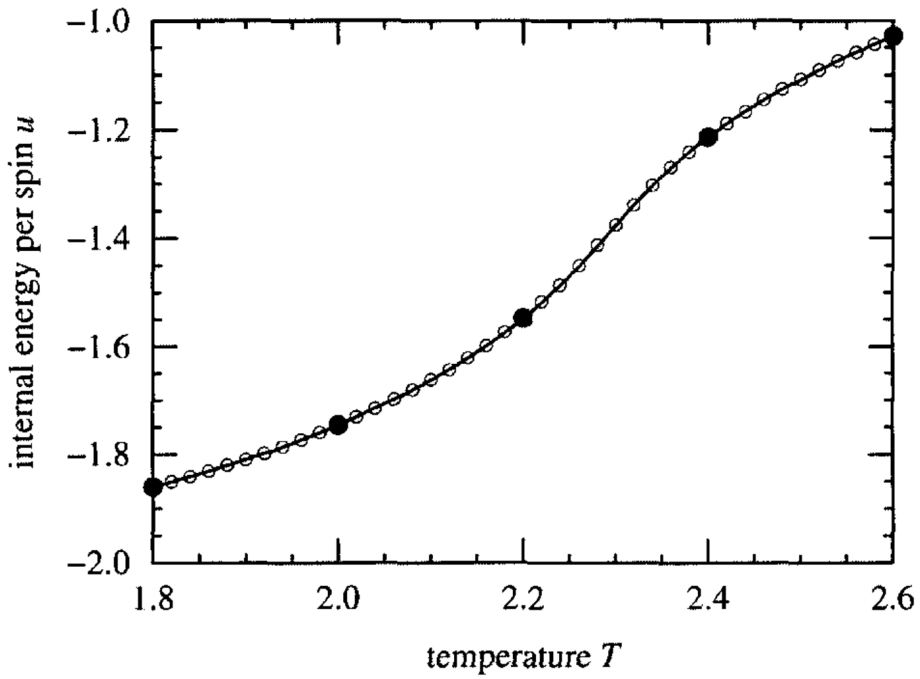

Figure 2.2: This is another example of the 32 x 32 two-dimensional Ising model. The black dots represent the simulations and the solid line the interpolation between the simulations. As a reference, the open circles are also simulations, to see how accurate the multiple histogram method is. Figure taken from [3].

Using the density of states obtained in Eq.(**??**), the partition functions of the

system for $\beta = \beta_k$ can be written as

$$Z_k = \sum_E \rho(E) e^{-\beta_k E} = \sum_E \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta_k - \beta_j)E}}. \tag{2.16}$$

Now Eq.(2.16) can be solved iteratively for Z_k by estimating Z_j, then calculating Z_k for every β_k of the simulations and plugging them back into the equation as the new Z_j. After a certain amount of iteration, the values of the partition function converge to the real values. With those values one can calculate the partition at any temperature,

$$Z(\beta) = \sum_E \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta - \beta_j)E}}. \tag{2.17}$$

Using the equations presented so far, one can calculate the internal energy.

$$U(\beta) = \sum_E E \rho(E) \frac{e^{-\beta E}}{Z(\beta)} = \frac{1}{Z(\beta)} \sum_E E \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta - \beta_j)E}}, \tag{2.18}$$

where E is again the total energy.
As one can see in Figure 2.2, the multiple histogram method gives very accurate results. With only five simulations, the internal energy at every temperature in the interval is known almost exactly.

2.2.1 Interpolating other variables

The generalization of the multiple histogram method is fairly easy. The steps to calculate the partition functions remain the same. The only difference appears in Eq.(2.22):

$$\langle Q \rangle = \frac{1}{Z(\beta)} \sum_{i,s} \frac{Q_{is}}{\sum_j n_j Z_j^{-1} e^{(\beta - \beta_j)E_{is}}} \tag{2.19}$$

Evaluating this equation saves a lot of CPU time and gives excellent results.

2.2.2 Implementation

When the simulations are too far from the parameter value one wants to interpolate, the number of samples is very small, which is why the multiple histogram method will not give good results. A condition for reliable results of the method can be a minimum threshold for the overlap of the distributions. That is why although the method is completely general, one has to check for every given data

if it is reasonable to use the method by looking at the overlap of the simulated distributions. When implementing the multiple histogram method, a problem rises with the iterative calculation of the partition functions Z_k because they can get very large and might produce an over- or underflow. A trick one can use is to normalize the partition functions by a factor A. A possible way might be setting A to the inverse of the geometrical mean of the smallest and largest Z in every iteration,

$$A = \frac{1}{\sqrt{Z_{\text{small}} Z_{\text{large}}}}.$$

However, consider the normalized Z_{large} for a very large system,

$$Z'_{\text{large}} \Rightarrow A \cdot Z_{\text{large}} = \frac{Z_{\text{large}}}{\sqrt{Z_{\text{small}} Z_{\text{large}}}} = \sqrt{\frac{Z_{\text{large}}}{Z_{\text{small}}}}.$$

The fraction $Z_{\text{large}}/Z_{\text{small}}$ can be greater than 10^{600} and thereby produce an overflow. This is why taking the logarithm is a better alternative. Once the logarithm is taken, Eq.(2.9) is applied in calculating the density of states, partition functions and observables. The starting values of Z_k are not important as long as they are greater than zero since they appear in the denominator and in a logarithm.

To know how many iteration are needed, one can set a condition in terms of the change of the partition functions per iteration:

$$\Delta^2 = \sum_k \left[\frac{Z_k^{(m)} - Z_k^{(m-1)}}{Z_k^{(m)}} \right]^2$$

m is the index of the iteration. The criterion for convergence is $\Delta < \epsilon$, where the precision ϵ can be set by the developer and is typically of order 10^{-6}. Another point is that in Eq.(2.16), (2.17) and (2.18) can be rewritten so the histograms $N_i(E)$ do not have to be used:

$$Z_k = \sum_E \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta_k - \beta_j)E}} = \sum_{i,E} \frac{N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta_k - \beta_j)E}} = \sum_{i,s} \frac{1}{\sum_j n_j Z_j^{-1} e^{(\beta_k - \beta_j)E_{is}}} \tag{2.20}$$

The summation index s runs over every state of the i-th simulation, where E_{is} is the total energy of the state s from the i-th simulation. In the same way Eq.(2.17)

becomes

$$Z(\beta) = \sum_{i,s} \frac{1}{\sum_j n_j Z_j^{-1} e^{(\beta-\beta_j)E_{is}}}, \tag{2.21}$$

and Eq.(2.19)

$$U(\beta) = \frac{1}{Z(\beta)} \sum_{i,s} \frac{E_{is}}{\sum_j n_j Z_j^{-1} e^{(\beta-\beta_j)E_{is}}}. \tag{2.22}$$

It is recommended to use the equations above instead of the ones with the histograms in the numerator, since by binning data for the histograms causes information loss.

All of the methods above only interpolate specific values of the observable such as the mean. However, in every simulation one not only measures one value per simulation but a whole histogram $N(O)$. Having the histograms at the new points (temperatures in this case) gives much more information about the system, which, in turn, prove to be useful for more applications.
So now the goal is to find an algorithm that gets that information. In the next chapter it will be discussed how one can interpolate histograms to new temperatures.

3 | Interpolating probability distributions

Recall the expression for the density of states of a simulation,

$$\rho_i(E) = \frac{N_i(E)}{n_i} \frac{Z_i}{e^{-\beta_i E}}. \tag{3.1}$$

As explained in section 2.2, the density of states only depends on the system and not on the temperature. That means for ideal measurements

$$\rho_i(E) = \rho(E) \quad \forall i, \tag{3.2}$$

and therefore rewriting Eq.(3.1) yields

$$N_i(E) = \rho(E) \frac{n_i e^{-\beta_i E}}{Z_i}. \tag{3.3}$$

The measurement is ideal in a sense that one measured infinitely many histograms at the same temperature, took the average and with that calculated the density of states using Eq.(2.14). Since measuring the histogram an infinite number of times is not possible, Eq.(3.3) is only an approximation. $\rho(E)$ can be substituted with Eq.(2.15) to get:

$$N_k(E) = \frac{n_k e^{-\beta_k E}}{Z_k} \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{-\beta_j E}} = \frac{n_k}{Z_k} \frac{\sum_i N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta_k - \beta_j)E}} \tag{3.4}$$

Note that the change of indices from Eq.(3.3) was due for consistency in Eq.(3.4). This equation allows one to extrapolate histograms to new temperatures β_k. However, since the histograms $N_i(E)$ are not stored and therefore unknown, the equation in its current form is not of much use. In order to implement the algorithm, it will be modified in the next section.

3.1 Implementation

The implementation of this method is analogous to the implementation of the multiple histogram method for the observables, since the equations look quite similar. A big difference to the reweighting of observables is that there is no summation over E in the case of interpolating histograms. The fact that the equation is discrete is not a problem since the program also only works with discrete temperatures. Also, n_k is not known for the interpolated histograms, and it makes no sense to deduce the statistics of a simulations that was not performed.

Hence, only the fraction $N_k(E)/n_k$ will be calculated.
Additionally, the histograms get normalized in the code, in order to have the probability distribution with an area $A = 1$.
First, Eq.(3.4) has to be rewritten in order to get use of the multiple histogram method.

$$\frac{N_k(E)}{n_k} = \frac{1}{Z_k}\sum_i \frac{N_i(E)}{\sum_j n_j Z_j^{-1} e^{(\beta_k-\beta_j)E}} = \frac{1}{Z_k}\sum_{i,E'} \frac{\delta_{EE'} N_i(E')}{\sum_j n_j Z_j^{-1} e^{(\beta_k-\beta_j)E'}}$$

$$\Rightarrow \quad \frac{N_k(E)}{n_k} = \frac{1}{Z_k}\sum_{i,s'} \frac{\delta_{EE'_{is'}}}{\sum_j n_j Z_j^{-1} e^{(\beta_k-\beta_j)E'_{is'}}} \tag{3.5}$$

Of course, this equation can also be evaluated for any temperatures:

$$\frac{N(\beta,E)}{n(\beta)} = \frac{1}{Z(\beta)}\sum_{i,s'} \frac{\delta_{EE'_{is'}}}{\sum_j n_j Z_j^{-1} e^{(\beta-\beta_j)E'_{is'}}} \tag{3.6}$$

This now looks like Eq.(2.19) where the observable is a Kronecker delta $\delta_{EE'_{is'}}$.

I will now discuss how a Kronecker delta can be understood as an observable. A histogram stores how many states with a specific energy E of a system were measured during a Monte Carlo simulation. Since a program cannot store a histogram with continuous energies it divides a histogram into bins with a certain bin size. Then the states with energy within the particular range of the bin size are bunched together into the corresponding bin. The Kronecker delta just checks to what bin the term has to be added.
As in the multiple histogram case, high exponents can be dealt with by calculating the logarithm of the histograms and shifting the observables instead of evaluating them directly.
Errors must be accounted for in the process of interpolation. The next section summarizes the error analysis.

3.2 Error Calculation

For every new point and observable one gets a probability distribution that is divided into many bins with certain heights. These heights also have errors to be evaluated, too. There are two main methods to calculate the errors: The **jackknife** and the **bootstrap method**.

The **bootstrap method** is a resampling method where one takes the original set of measurements and replaces entries randomly but only with values from the original measurements [4]. In the resampled set, there can also be duplicates of a measurement. For example if one has a set of numbers $1, 3, 4, 6$, then $1, 4, 4, 3$ or $3, 3, 3, 1$ could be a possible bootstrap set. $1, 3, 6, 8$ would not be allowed because 8 was not in the original measurement. The size of a bootstrap set is the same as the original set. The resampling of the set can be done m times to end up with m bootstrap sets. The bigger m is the more precise the error becomes. The calculations with these resampled sets are the same as for the original set. At the end one has m estimates for a quantity Q_m. In this thesis, the quantity is the height of a bin. These results Q_m are called **bootstrap estimators**. The error can be then calculated in a straightforward way,

$$\sigma = \sqrt{\overline{Q^2} - \overline{Q}^2}, \tag{3.7}$$

where $\overline{Q^2}$ is the average of the square of all estimators and $\overline{Q}^2$ the square of the mean.

The **jackknife method** takes a different approach. Consider a set of measurements with size n. With this set, one calculates the estimate for the quantity Q. Now the first measurement gets left out and the calculation is repeated with a set of size $n-1$. Then one ends up with the first **jackknife estimator** Q_1 [5].
Then the second entry gets left out to get Q_2. This gets repeated n times for every entry of the original set. Then the error is calculated in the following way,

$$\sigma = \sqrt{\sum_{i=1}^{n} (Q_i - Q)^2}. \tag{3.8}$$

For sets with $n \gg 100$, the bootstrap method is more efficient since it saves the user from doing every calculation n times as in the jackknife method. For smaller n, the jackknife method may be used. Both methods are implemented to calculate the error of the probability distributions, but for the test and application of the code the bootstrap method will be used since the data is correlated, which is why bootstrap is preferable [7].

3.3 Structure of the code

The goal was to implement the reweighting of probability distributions in the cleanest way both for the developer and for the user. I started with creating three C++ classes that are the `histogram`, `histogram estimator` and `probability distribution`, which are further discussed down below. These three objects are the main containers for the reweighting of probability distributions which have convenient functionality and are therefore easy to handle. The way the objects are constructed is neither complex nor trivial, which is why a fellow developer would first have to understand the approach in order to modify the class. This is explained in detail in the class. For the user however, the methods of the objects are intuitively named and easy to understand, without knowing much about the internal approach of each class.
The user will have the freedom to configure the reweighting calculation, but will not notice any of the deeper processes of the code, since they are happening in the inner layers of the code. This information may not matter for an average user.

The code is split into many folders handling different aspects of the calculation, e.g. simulation data storing or raw calculations of the reweighting process. It will be next discussed, how and in which order the probability distributions get reweighted from the external point of view, i.e. top-down in the code design.

LqcdParameters: First using command line options the user can set up the reweighting process when running the code, i.e. the `LqcdParameters` executable. These options include whether the probability distribution should be reweighted (which is the default mode) and setting the bin size of the distribution. Suppose here the user wants to reweight probability distributions.

Reweighter: Then that information goes to the `Reweighter` class that manages all of the reweighting procedures[1]. For every reweighting procedure the `Reweighter` creates a "box" called `RawDataAndMetaInfomation`. This box contains all the information for each reweighting procedure like the bin size of the probability distribution and is then handed over to the `MomentsReweighter` class where the explicit calculation is done.

[1] In general, mean, variance, skewness and kurtosis might need different reweighting procedures to be reweighted, since their autocorrelation times (and therefore their bin size in the bootstrap) might differ. Also, the user might wish to reweight several observables at once, which would also result in different reweighting procedures.

MomentsReweighter: In this class the object `Histogram` is getting filled by using Eq.(3.5). This object is then basically the reweighted distribution that consists of many bins with different heights. The method of taking the logarithm, shifting the observables and normalizing the histogram is also performed here. Another object that is filled in this class is the `HistogramEstimator`. This object is similar to the histogram but stores multiple heights per bin. For the error calculation one calculates many estimators of the histogram. The histogram estimator has all of them combined in one container, which is why it contains many different heights in each bin. Depending on the error method used the number of heights in each bin varies.

Back to Reweighter: Now `MomentsReweighter` returns the filled distributions to the `reweighter` class. The goal is that the user ends up with an object that has both the heights and the error. Thus, the final object `ProbabilityDistribution` was created. In the constructor of this object for every bin, the calculation for the error of the heights from the histogram estimator is then performed like discussed in section 3.2. Afterwards, the estimated heights and the calculated error are stored.

Back to LqcdParameters: The last step is to give the final product back to the executable which prints the probability distribution in the following way. For every observable there are as many files as there are points at new betas. In each file the bin with the estimate and error get printed. Therefore, after a successful running of the code, the user ends up with $[N_{Observables}]x[N_{NewPoints}]$ files.

As one can see, it starts in the executable at the top layer, goes deeper into the code until it reaches the deepest layer where the calculation is performed (i.e. `MomentsReweighter`) and then returns the results back through the upper layers to the user. It displays an intuitive structure, hence a developer should be able to follow the flow of the code easily.

4 | Correctness of the code

For a simple methods of a class one might catch bugs in the code. However, many functions are more complex and it becomes more difficult to troubleshoot. Since many files are working together, one cannot run the code just to check one function. That is why we used the Unit Test Framework of BOOST to implement tests for every class and its methods. With that, one can check if the single functionality is working properly. For example, we know that the area of a probability distribution is 1. This can be checked in a unit test after filling and normalizing the histogram. These tests are necessary but not sufficient since also with perfectly working methods the interworking of the files can be wrong. To ascertain correct reweighting, the code was applied on some test data.

4.1 Testrun and Plots

Given were 5 sets of data at different temperatures, which correspond to the histograms from Monte Carlo simulations. Since the code is general, it does not matter what system the test data is from and therefore will not be discussed. The test data sets will be used to reweight the probability distributions, not to other temperatures, but to exactly the same 5 temperatures, where the simulations were performed. Having done that, one can compare the simulated distributions with the reweighted ones.

In the case with our test data, the overlap of the simulated data sets is very large. Therefore, the influence of the other distribution will affect the outcome of the reweighted distribution and it will look similar but not very close to the simulated distribution (Figure 4.6). Since having only one data set in input is sufficient to check if the code works, the 5 data sets are considered separately and not simultaneously.

In the following the results of the test are plotted for every simulated set of data. The purple bars are the simulated probability distribution of the test data. The green arrow dots are the result of the reweighting process which have error bars, estimated with the bootstrap method (Figure 4.1).

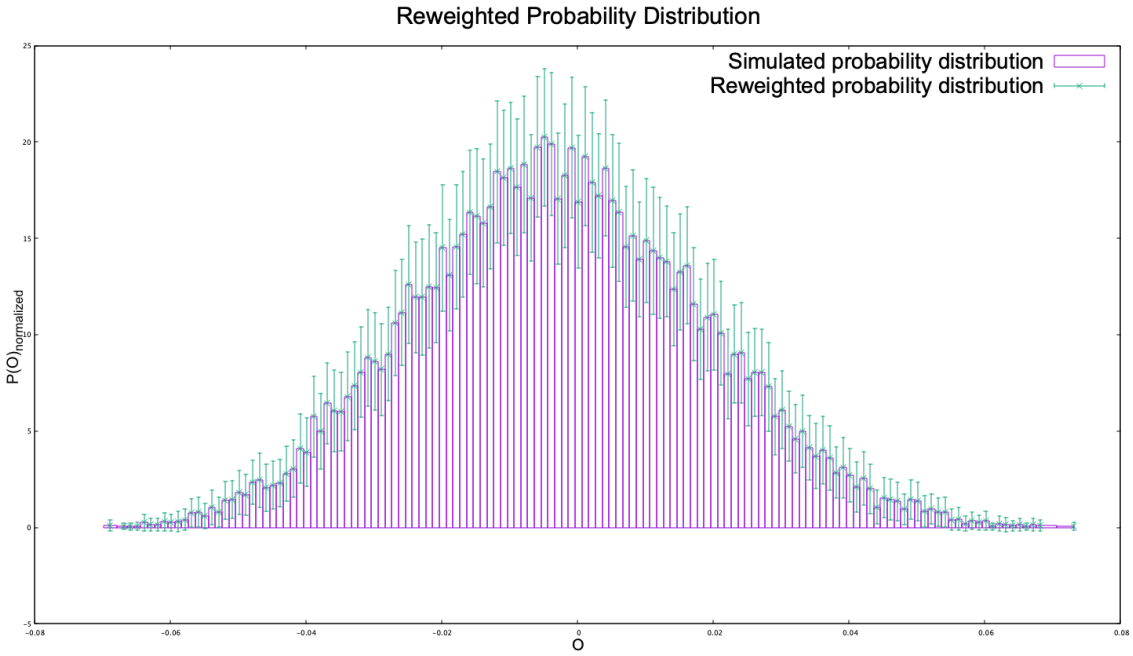

Figure 4.1: Reweighted Probability Distribution with error bars over simulated probability distribution at $\beta = 5.427$.

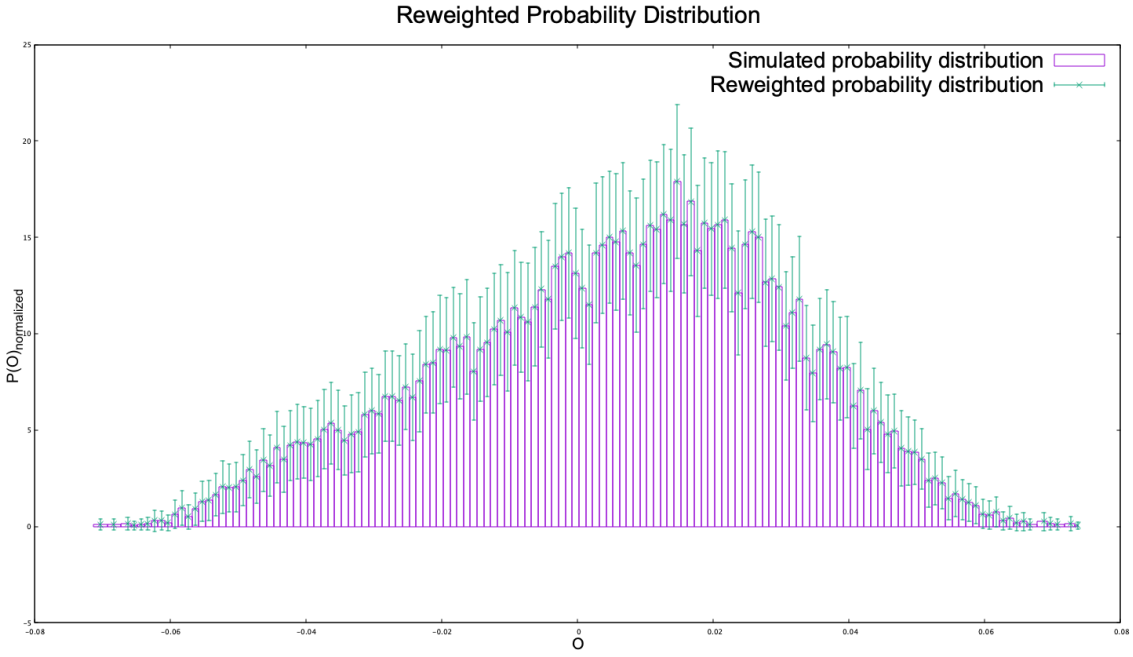

Figure 4.2: Reweighted Probability Distribution with error bars over simulated probability distribution at $\beta = 5.43$.

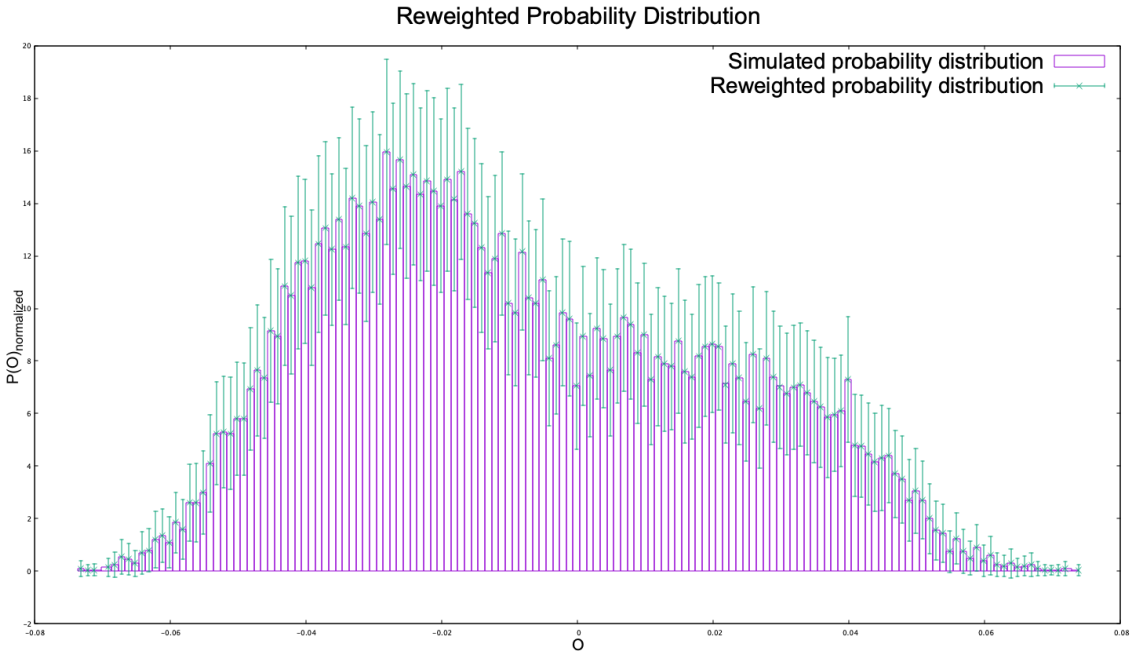

Figure 4.3: Reweighted Probability Distribution with error bars over simulated probability distribution at $\beta = 5.433$.

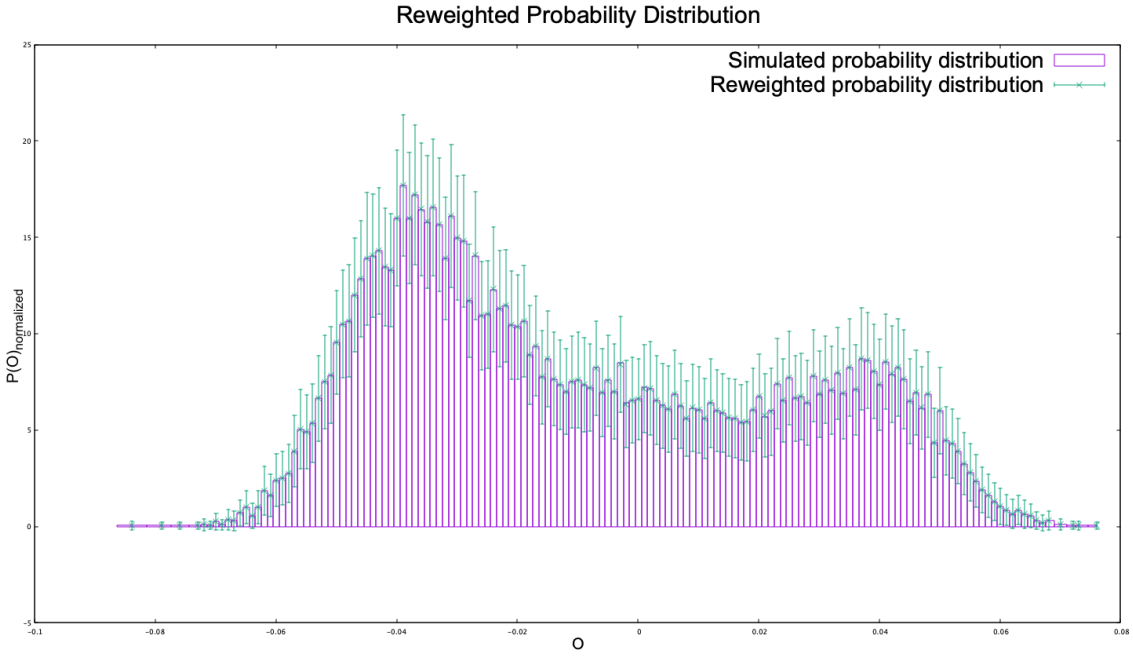

Figure 4.4: Reweighted Probability Distribution with error bars over simulated probability distribution at $\beta = 5.436$.

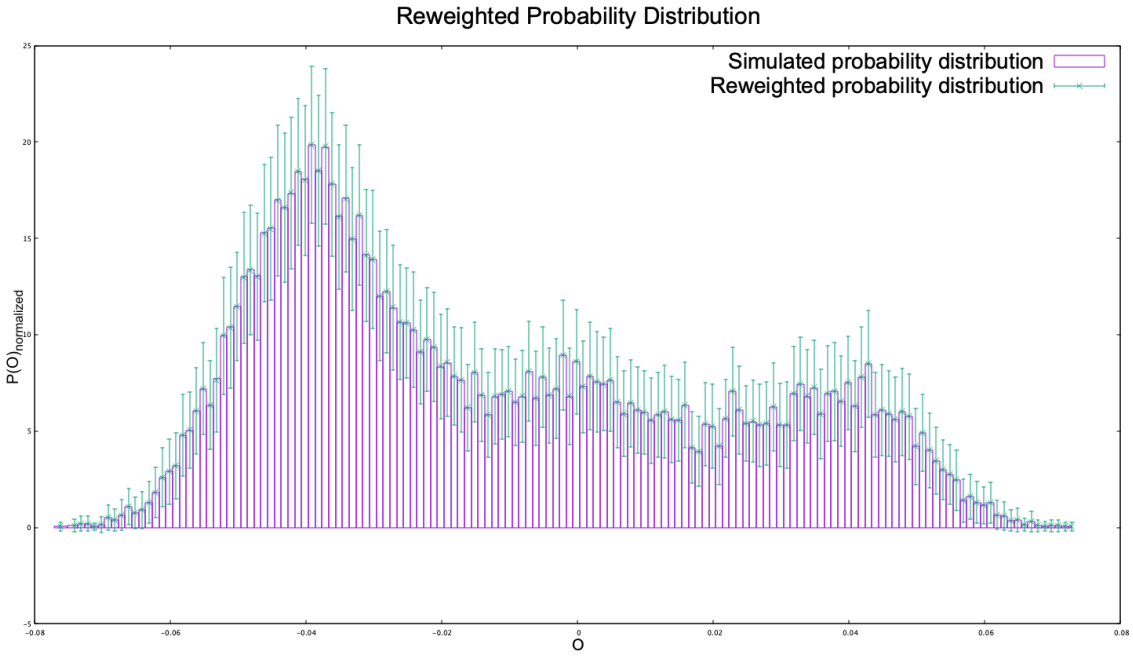

Figure 4.5: Reweighted Probability Distribution with error bars over simulated probability distribution at $\beta = 5.439$.

These plots demonstrate a very positive result. We can see that for every data set the reweighting process produces a probability distribution, which has the exact same height for every bin. The error bars are quite large in this case, because each test file only had around 20.000 data points and only one test file was used per plot in Figures 4.1 to 4.5. Furthermore, the data points get distributed to different bins, so for each bin there are even less data points, giving rise to the big error bars. Certainly, more data points per file as well as multiple files will be used in real life cases to reweight probability distributions.
Now that the code is proven to work correctly, we can look at one important application for reweighting distributions. It is evident that the distributions for $\beta \in \{5.4330, 5.4360, 5.4390\}$ the distributions do not possess a typical form of a Gaussian distribution like in Figure 4.1. In Figure 4.4, there are even two peaks. Apparently, in the range of $5.4270 \leq \beta \leq 5.4390$ something odd is happening because otherwise, a Gaussian spectrum is expected. The effect will be explained in detail in section 5, where the code will be applied to a different system.

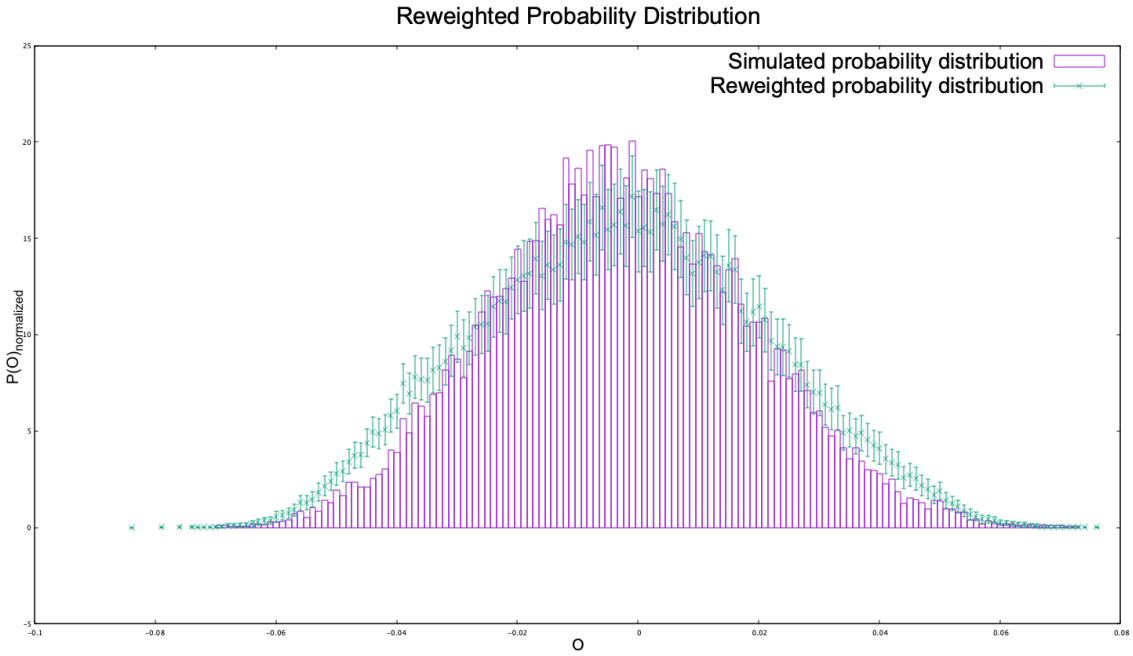

Figure 4.6: Reweighted Probability Distribution at $\beta = 5.427$. Now every test file was used to reweight the distributions. As we can clearly see, the green dots follow the simulated histogram, but are still very far away from the simulated heights of the bins. This might come on one hand from the fact that the simulated data sets overlap a lot, which is why the influence of the data sets at other β's is very big. On the other hand it might as well be caused from the not excellent statistics available.

5 | Probability distributions between two phases

A probability distribution changes its form when the system undergoes a phase transition. One can not only detect a phase transition but also determine its type. To use the reweighting method on a phase transition, we first have to understand how a phase transition is defined.

5.1 Phase transition and Order Parameter

Phase transitions can be described in different ways. An older method is the "Ehrenfest" classifcation which studies the behavior of the free energy [6].
Nowadays, phase transitions are classified by looking at the **order parameter**. An order parameter is a quantity which is zero in one phase and nonzero in the other. An example is a liquid-to-gas transition, where the order parameter is the difference of the densities [8].
The behavior of the order parameter during a phase transition is very helpful to categorize an unknown phase transition. Here are three categories of phase transitions.

First-order phase transition: A first-order phase transition is characterised by a latent heat, where two phases coexist. The order parameter for such a transition is discontinuous (i.e. it jumps from zero to a non-zero value). An example is the phase transition between ice and water [7].
However, the discontinuous jump of the order parameter in a first-order parameter is only true for systems in the thermodynamic limit, where the volume goes to infinity. For finite volumes, the jump gets continuous and smooth. The smaller the volume is, the less rapid the transition of the order parameter between the two values becomes.

Second-order phase transition: A second-order phase transition is a continuous phase transition, which is characterised by a divergent susceptibility. Here, the order parameter is continuous, while the first derivative is discontinuous [7].

Crossover: In a crossover, both phases are continuously connected, so one cannot define a point or line where the phase transition occurs. That is why the crossover is not actually a true phase transition, although the phase changes [7].

In a finite volume, far away from the infinite volume limit, it is almost impossible to distinguish a first-order, second-order transition or a crossover, because everything will be distorted and "crossover like".

5.2 Probability distribution of an order parameter at a first-order phase transition

Here it will be discussed, how one can determine whether a phase transition is first order or not by looking at the probability distributions of the system.
Consider the probability distributions $P(O)$, where O is the order parameter. For an infinite volume, the distribution is simply a Dirac function in both phases and it changes instantaneous when crossing the transition border. Since the simulations work in finite volumes, the distributions become wide. Measuring $P(O)$ at a temperature $T \ll T_c$ or $T \gg T_c$, with T_c being the critical temperature, one will find a symmetric Gaussian distribution[2] since the system is far enough from the transition border. As the temperature comes closer to T_c, the form changes, since the system in a finite volume will explore both phases. This causes the distribution to deform and become asymmetric when the system is close but not at the critical temperature. At the critical temperature $P(O)$ becomes symmetric again, since one half is in one phase while the other half is in the other. Around T_c the distributions are double peak distributions, that correspond to both phases (Figure 5.1).

To study the phase transition, one needs to know the value of the critical temperature. A possibility to find the critical temperature is by simply looking at what temperature the distribution is symmetric with two peaks. Unfortunately, this method is in practice not very accurate, because it is rare to have simulations exactly at $T = T_c$, so another approach that involves the **skewness** is used.
The skewness is defined as

$$B_3(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}^3} = \frac{m_3}{s^3}, \tag{5.1}$$

where $\bar{x}$ is the mean value, m_3 the third moment of x and s the standard deviation of the sample [11].
It describes the asymmetry of a distribution and can be positive or negative

[2] For that, we need to have infinite statistics.

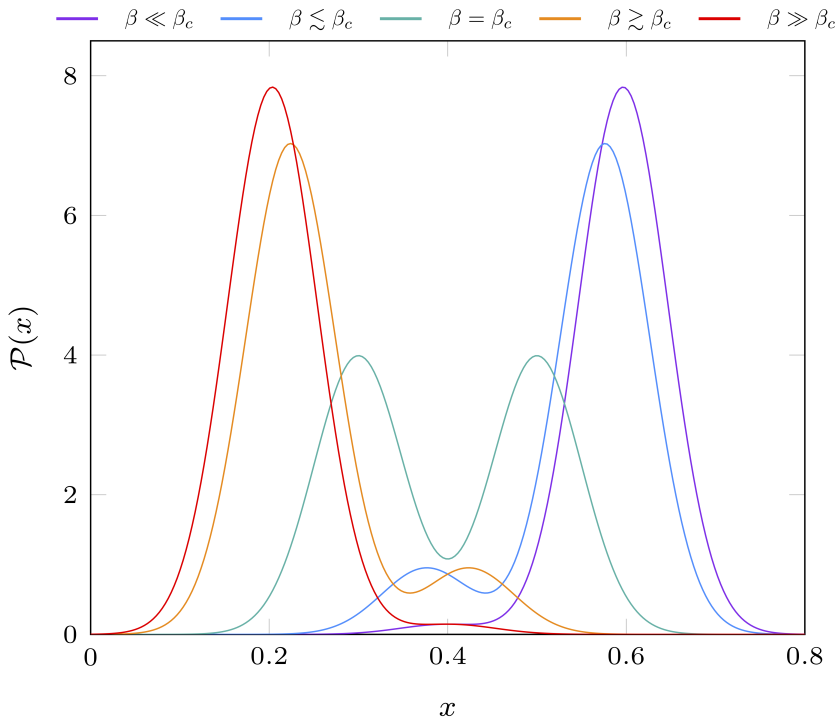

Figure 5.1: Probability distributions of the order parameter x around the critical temperature. The green line represents the symmetric distribution at the critical temperature. Figure taken from [9].

depending on what side the distribution is distorted. When a distribution is skewed to the right (e.g. orange curve in Figure 5.1), the skewness is positive. Right skewed distributions have a negative skewness. For a symmetric distribution, the skewness is zero.

It is perceptible how the skewness will behave when transitioning between two phases. For temperatures far below and above T_c, the distributions are Gaussian and therefore symmetric with a skewness of zero (Figure 5.2).
At T_c, the double peak distribution is also symmetric and therefore also has a vanishing skewness. Only close to T_c, the skewness is nonzero. With that in mind, it is very simple to find the critical temperature. It is a matter of finding the non trivial root, which is at exactly T_c. This is done by reweighting the skewness around the critical temperature.

Next, the probability distributions give clues to determine the type of the phase

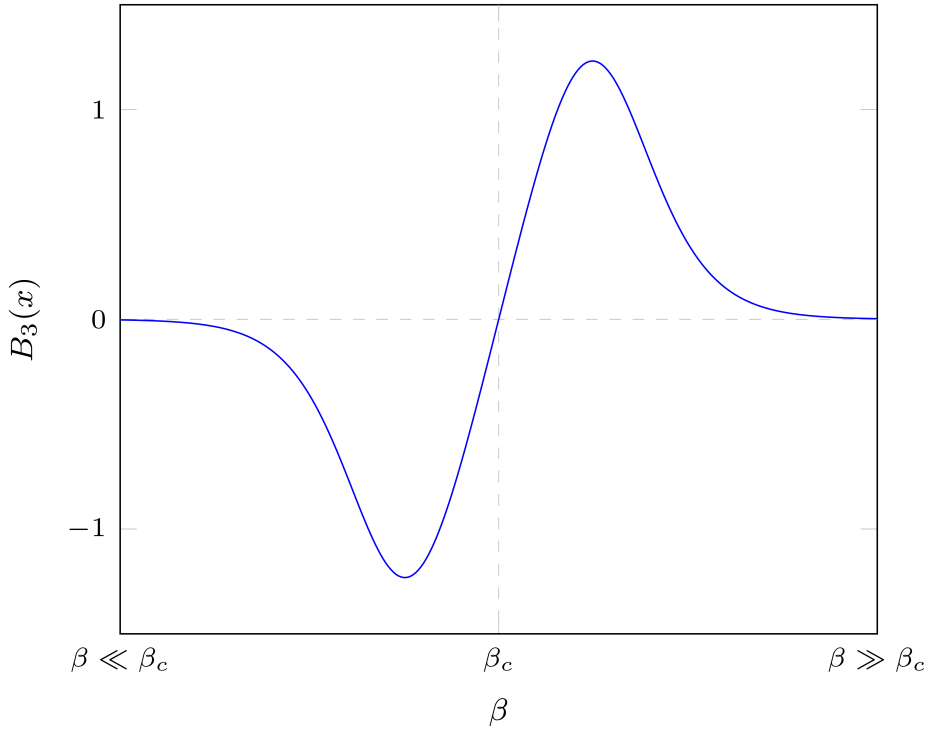

Figure 5.2: Skewness of the distributions shown in Figure 5.1 depending on the temperature at a phase transition. Figure taken from [9].

transition. For that, one has to measure $P(O)$ at T_c for different volumes of the system. Recall that for the thermodynamic limit, where the size of the system becomes infinite, $P(O)$ is a Dirac function. That means there is no double peak distribution but zero width peaks with infinite heights in both phases. For finite volumes, the distributions overlap and one can see a double peak distribution. That means if one measures $P(O)$ at a certain volume and repeats the measurements at bigger volumes, the local minimum between the two peaks should drop with increasing volumes since then the overlap should decrease. So by plotting $P(O)$ for different volumes, one can find out if the system undergoes a first-order phase transition.

5.3 First-order phase transition for heavy up and down quarks

Consider *quantum chromo dynamics (QCD)* at a finite temperature with zero density and chemical potential $\mu = 0$. It was discovered that the phase structure of such a system strongly depends on the masses of the quarks. For more information and an overview, refer to [7]. A useful graphical representation of the phase structure is the *Columbia plot* [10] (Figure 5.3).

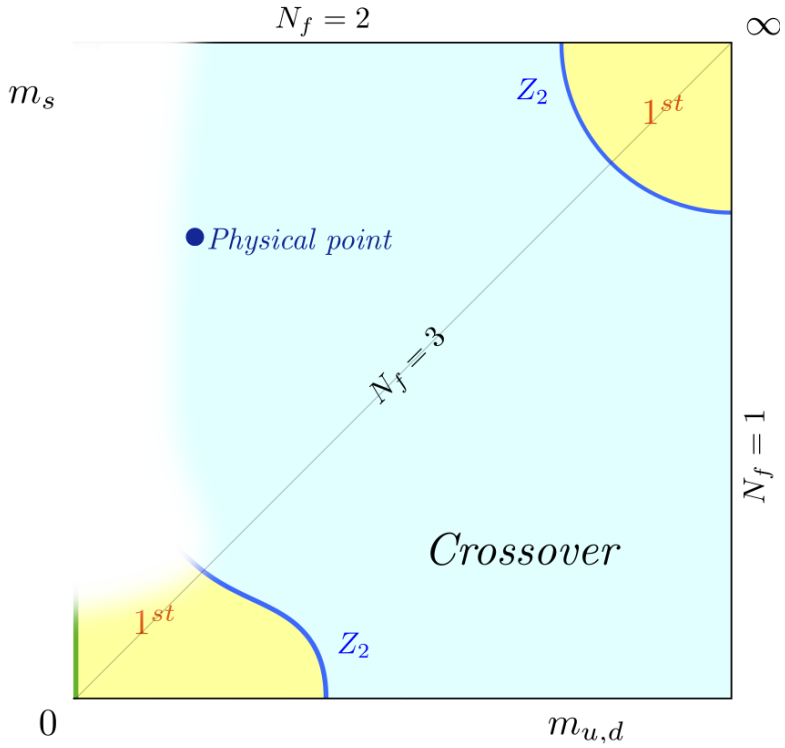

Figure 5.3: The Columbia Plot shows which type of phase transition the system undergoes when changing the temperature. The parameters are the masses of the quarks, where the masses of the up- and down quark are approximated to be equal. The physical point represents the experimentally observed masses of the quarks and therefore the real life scenario. N_f is the number of degenerate flavours. Figure taken from [7].

At the right axis, the masses of the up and down quarks are infinite, therefore, there is only one flavour left, since quarks with infinite masses decouple from the system. On the upper axis, the strange quark has infinite mass, leaving two degenerate flavours. The diagonal shown in Figure 5.3 has three degenerate flavours.
The Columbia plot shows the type of the phase transition at every point when

increasing the temperature. The upper left corner is blurred out because it is not fully known which type of phase transition the system undergoes in that area. In reality, the quarks have specific masses, represented as the *Physical point* in Figure 5.3.

As mentioned previously, a first order transition is discontinuous. In a crossover, however, both phases sort of merge and are continuously connected. That means, between a first order transition and a crossover in the Columbia plot, there has to be a border where there are second order transitions (dark blue line in the upper right and lower left corner). Since for $m_{u,d,s} = \infty$, the transition is known to be fist order, in the upper right corner the first order transition is not only the most upper right point but has a finite area.

Monte Carlo simulations were performed on such a system at the upper axis, where $N_f = 2$ for different values of $m_{u,d}$. There, the order parameter, which describes the phase transition is the norm of the Polyakov loop. The methods discussed in Chapter 3 will now be applied using what we have seen in Chapter 5 to find out the type of phase transition.

5.4 Lattice QCD

In lattice QCD one discretizes spacetime into a lattice of points, where QCD can be solved in a non-perturbative way. The most fundamental property of such a lattice is the lattice spacing a, which describes the distance between to adjacent lattice points. The temperature T and volume V are connected to the lattice spacing by the following expressions,

$$T = \frac{1}{N_t a}, \tag{5.2}$$

$$V = (N_s a)^3, \tag{5.3}$$

where N_t is the number of lattice points in temporal direction and N_s in spatial direction. The coupling constant of QCD is a dimensionless parameter g and describes the strength of the strong interaction. A function that depends on the coupling constant, is the beta function $\beta(g)$, which is therefore also dimensionless. It gives information about the dependence of the coupling constant on a certain energy scale. Furthermore, the beta function is connected to the lattice spacing in a non trivial way. This means, one can change the temperature of

the system by changing $\beta(g)$. We will reweight probability distributions using $\beta(g)$ as the reweighting parameter. In the following, the argument of $\beta(g)$ will be omitted, so it should not be confused with the inverse temperature $\beta = 1/k_B T$.

5.5 Results

Since the data is from a $N_f = 2$ system, we only need to consider the masses of the the up- and down quark. Data were given for $m_{u,d} = 1.15$ and $m_{u,d} = 0.35$. Note that the masses are presented in lattice units, i.e. they are dimensionless. For both masses, there are three data sets for three different volumes. These data sets contain multiple simulation data sets at different temperatures.

For both masses, the chemical potential is, as mentioned above, zero. For all data, N_t is 8. In order have data sets for different 3-dimensional volumes, the number of sites in spatial direction N_s has to be changed, since the lattice spacing is constant.

5.5.1 Phase transition for $m_{u,d} = 1.15$

First the process of identifying the order of the phase transition for the mass $m_{u,d} = 1.15$ will be discussed. We start by looking at the skewness in order to find out the critical beta β_c. Figure 5.4 shows the reweighted skewness as a function of β for $N_s = 40$.

This was done for every volume. Hence, we find for the critical beta's β_c at $m_{u,d} = 1.15$:

$$N_s = 40 \quad \beta_c = 6.04024$$
$$N_s = 48 \quad \beta_c = 6.04009$$
$$N_s = 56 \quad \beta_c = 6.04110$$

We can see that the critical beta for the three volumes is $\beta_c \approx 6.04$. The next step is to reweight the probability distribution at β_c and to compare the distributions by plotting them. Figure 5.5 contains all three probability distributions. A bin size of 0.0004 was used in order to clearly see the profile of the distributions.

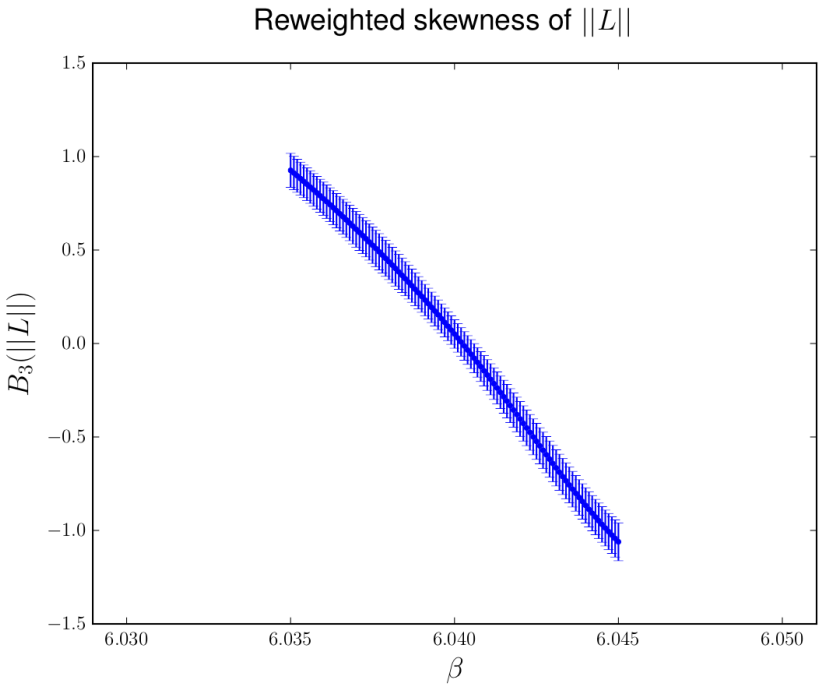

Figure 5.4: Reweighted skewness around a phase transition for $m_{u,d} = 1.15$ and $N_s = 40$.

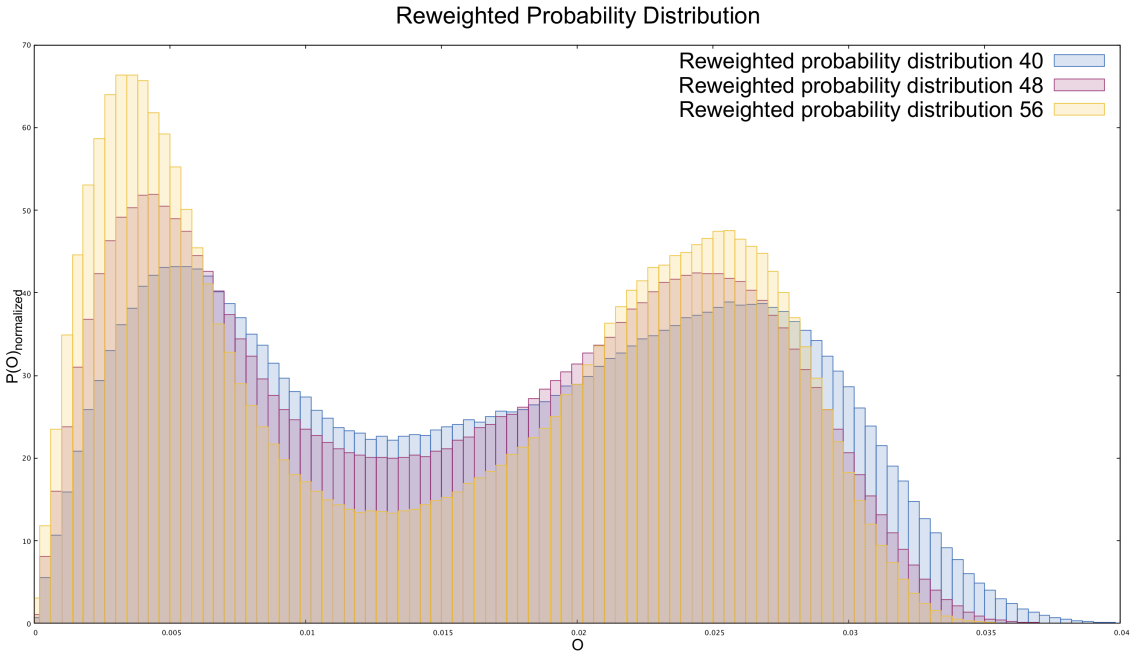

Figure 5.5: Reweighted probability distributions at a phase transition with $\beta_c \approx 6.04$ for three different volumes indicated by the number of lattice points in the three spatial directions (40,48,56) with $m_{u,d} = 1.15$.

However, for such a small bin size the error bars are very big, since less statistics goes in each bin. That is why they are not shown in this plot. One can already tell, that in this plot the local minimum of the distributions decreases with higher volumes but to have a valid argument one has to take the errors $\Delta P(O)$ into account. Therefore, instead of plotting the estimates of the distributions, the values $P(O)+\Delta P(O)$, $P(O)-\Delta P(O)$ are taken and interpolated by using cubic splines. The area between one error above and one error under the estimate is filled to have a good look a middle bend of the distributions between the peaks and also to look at the overlap of each distribution at that bend.

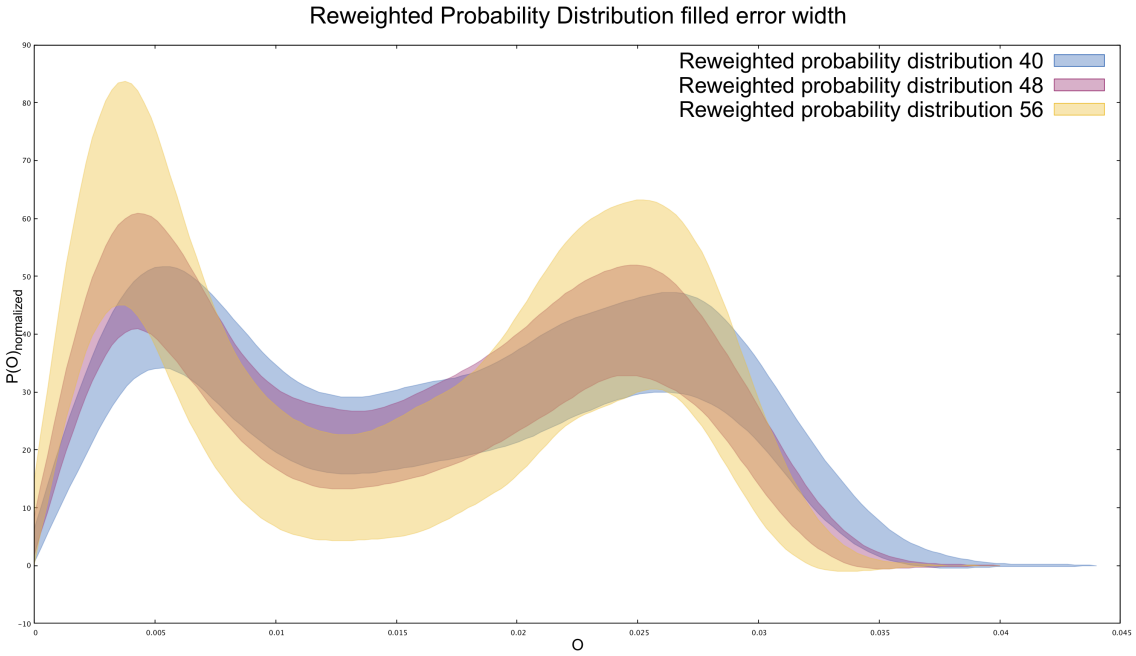

Figure 5.6: Area between the error bars of the reweighted probability distributions at a phase transition with $\beta_c \approx 6.04$ for three different volumes with $m_{u,d} = 1.15$.

In Figure 5.6 we can see the bend very good and also a downward trend of it is visible. Unfortunately, the errors are still large and therefore the overlap is very big. Having more statistic, i.e. more data, would shrink the error bars. The simulations that yielded the data sets took over 9 months to perform. That means, it is not realistic to increase the statistics much because it would take years to produce. The statistics we have here are sufficient to have reasonable results for the estimates. To show the effect of more statistics, suppose to have 4 times more statistics. The error of a sample estimate is inverse proportional to the square root of the number of measurements $\sigma \sim 1/\sqrt{N}$, which means it would

then decrease by a factor of 2. Looking at the error width of the distributions with the smallest and the biggest volume (i.e. $N_s = 40$ and 56), we can see a clear gap between both bends (Figure 5.7). This assures us that the bend is decreasing and that we have a first order phase transition; hence the system is located in the upper right corner of the Columbia plot (Figure 5.3).

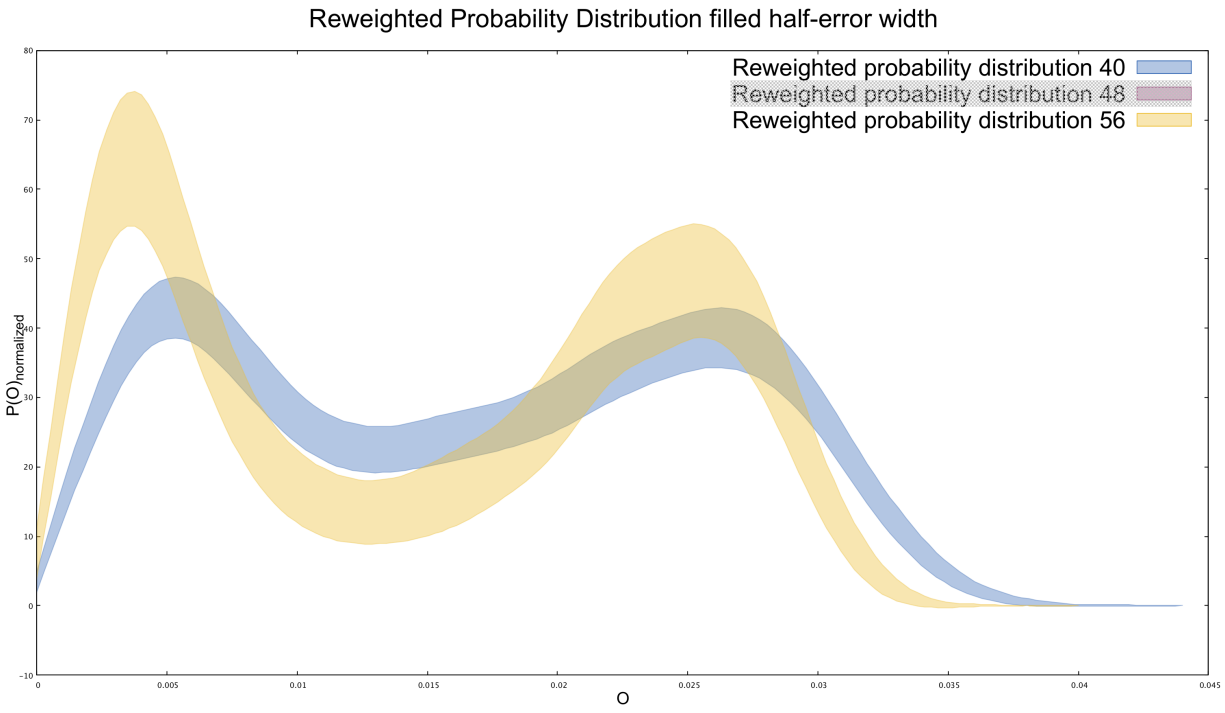

Figure 5.7: Area between the error bars of the reweighted probability distributions at a phase transition with $\beta_c \approx 6.04$ for two different volumes with $N_s \in \{40, 56\}$ with $m_{u,d} = 1.15$. Here, the error bars are shrunk by 1/2 to show the influence of a bigger sample with more statistics.

5.5.2 Phase transition for $m_{u,d} = 0.35$

Now we want to find out the kind of the phase transition for the smaller mass $m_{u,d} = 0.35$. The elaboration is identical to section 5.5.1. One difference is that the number of spatial lattice points N_s goes from 32 to 48 in this instance. The

critical betas at $m_{u,d} = 0.35$ are:

$$N_s = 32 \quad \beta_c = 5.92499$$
$$N_s = 40 \quad \beta_c = 5.92258$$
$$N_s = 48 \quad \beta_c = 5.92154$$

Here, the critical beta is approximately $\beta_c \approx 5.92$. Figure 5.8 shows the result after reweighting the probability distributions at β_c.

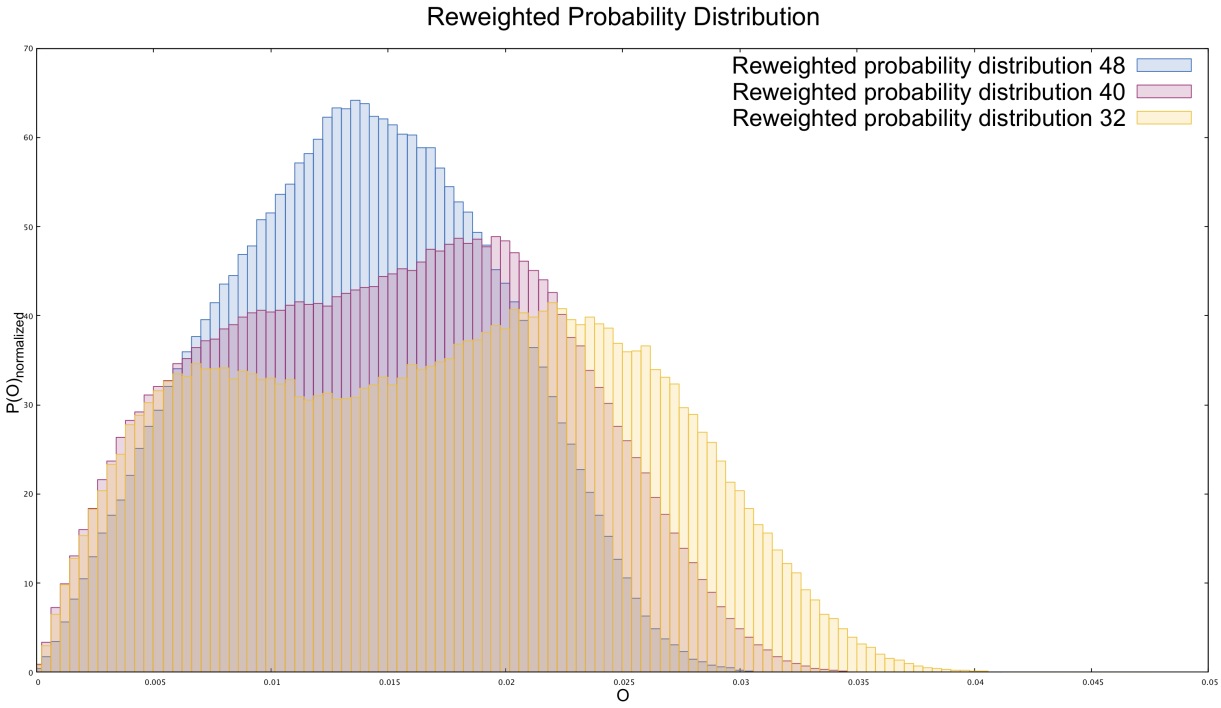

Figure 5.8: Reweighted probability distributions of the estimates at a phase transition with $\beta_c \approx 5.92$ for three different volumes indicated by the number of lattice points in the three spatial directions (32,40,48) with $m_{u,d} = 0.35$.

We can already see that the probability distributions behave differently in this case. For $N_s = 32$ we can see a slight bend in the middle, which is less visible for the bigger volume $N_s = 40$. For $N_s = 48$, we basically have a one peak distribution that tends to look like a normal distribution. That already gives us a hint that for $m_{u,d} = 0.35$ the type of phase transition is different. The overlap of the error widths can be a sufficient argument to tell a distinct trend of the middle bend. As shown in the case above, when there is a region in the middle bend where the error bars do not overlap, it is safe to say that the bend indeed increases or

decreases.

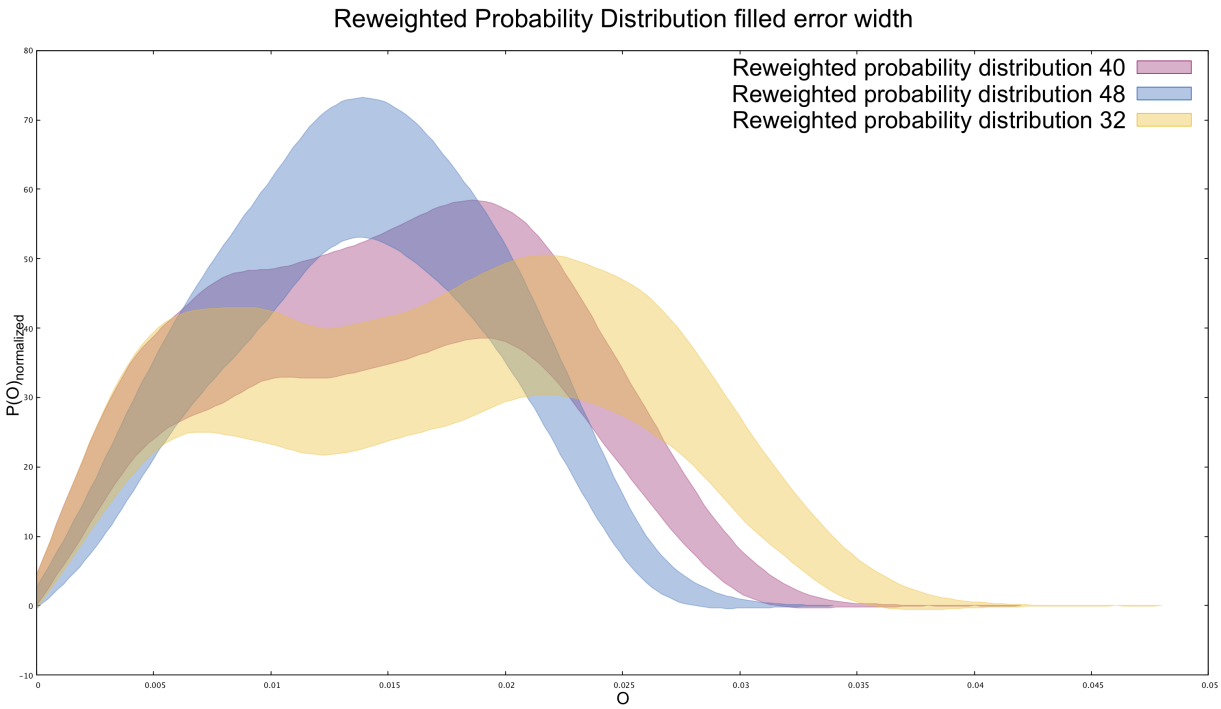

Figure 5.9: Area between the error bars of the reweighted probability distributions at a phase transition with $\beta_c \approx 5.92$ for three different volumes with $m_{u,d} = 0.35$.

In Figure 5.9 we can clearly see that the error width at the bend for $N_s = 32$ and $N_s = 48$ do not overlap. This shows that we do not have the characteristic behavior of a first order phase transition and therefore the system undergoes a different kind of phase transition. We know that in the Columbia plot (Figure 5.3) a first order phase transition only occurs for very small or very big quark masses. Then it either can be a second order phase transition or a crossover.
Both types of phase transitions can be distinguished by looking at the **kurtosis** of the distribution. The kurtosis describes the form of the tails of a distribution [11] and is defined as the following.

$$B_4(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^2} = \frac{m_4}{m_2^2}, \tag{5.4}$$

where m_4 is the fourth moment and m_2 the variance of the sample. For a perfectly normal distribution, as in the case for a crossover in the thermodynamic limit, the kurtosis is exactly 3. For a Z_2 second order phase transition (Figure 5.3) the kurtosis is 1.603 [7].

Therefore, we have to reweight the kurtosis for all three volumes. I found that for $N_s = 32, 40, 48$ at $\beta_c \approx 5.92$, the kurtosis is

$$B_4|_{N_s=32} = 2.012$$
$$B_4|_{N_s=40} = 2.129$$
$$B_4|_{N_s=48} = 2.348$$

The kurtosis at the smallest volume is already bigger than 1.603, which is the kurtosis for a Z_2 second order phase transition for the infinite volume limit. Furthermore, the kurtosis is increasing as the volume was increased. So it is very unlikely that in the thermodynamic limit, the $m = 0.35$ system will turn out to undergo a second order phase transition. The increasing kurtosis could be indicating that it converges to 3. Therefore, it is probable that the system undergoes a crossover. To be very safe, one could have data sets for more simulations at even bigger volumes, but that would require a very long time, especially when increasing the size of the system.

6 | Summary

In this thesis, the algorithm for reweighting probability distributions was introduced and implemented in the codebase "`Monte Carlo Cpp analysis tools`". This has been done in a completely generic way to give it a wide range of applications. In this work the interpolation was only performed in the temperature axis.

After the implementation, the code was tested for the correctness of the interpolated results. This was done by plotting the reweighted probability distributions and the simulated probability distributions at the same temperature on top of each other. As shown in Figures 4.1-4.5, the reweighted and simulated distributions are the same, therefore the tests were successful and showed that the code and the computations work properly.

Then the code was applied to study phase transitions of a system in lattice QCD where the aim was to find the type of phase transition. For that purpose, the critical β values had to be deduced by reweighting and then plotting the skewness which has a non-trivial root at β_c. With that information in hand, the probability distributions were reweighted for three different volumes at β_c to investigate the behavior of the system going towards the thermodynamic limit. By looking how the local minimum between the two peaks behaves it could be checked whether it is a first order phase transition or not. This was applied for two different masses of the up- and down quark, where it was found that for $m_{u,d} = 1.15$ a first order phase transition and for $m_{u,d} = 0.35$ most likely a crossover occurs.

This thesis and the application showed that the reweighting method for probability distributions is very efficient since overall, the code ran for about 2 hours to return reliable results. Since the code works in a completely general fashion, there is not much to be added. However, the executable of the code can at the moment only be used to reweight in one parameter. Hence, one could work to build a more general executable, that can be easily used and allows the reweighting method to be done in multiple parameters. However, this modification is only needed at the top-level of the code, since underneath, all functionalities to reweight in multiple parameters is already provided.

References

[1] Ferrenberg A.M., Swendsen R.H. *New Monte Carlo technique for studying phase transitions*. Physical Rev. Letters 61. 1988.

[2] Ferrenberg A.M., Swendsen R.H. *Optimized Monte Carlo data analysis*. Physical Rev. Letters 63. 1989.

[3] Newman M. E. J., Barkema G. T. *Monte Carlo Methods in Statistical Physics*. Clarendon Press, 1999.

[4] Efron B., Tibshirani R. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[5] Efron B. *The jackknife, the bootstrap, and other resampling plans*. Society for Industrial and Applied Mathematics. 1982.

[6] Jaeger, G. *The Ehrenfest Classification of Phase Transitions: Introduction and Evolution*. Arch Hist Exact Sc. 1998.

[7] Sciarra A. *The QCD phase diagram at purely imaginary chemical potential from the lattice*. PhD Thesis. 2017.

[8] McNaught A. D., Wilkinson A. *Compendium of Chemical Terminology*. IUPAC. 1997.

[9] Cuteri F., Philipsen O., Sciarra A. *The QCD chiral phase transition from non-integer numbers of flavors*. Physical Rev. D97. 2017

[10] Gavin S., Goksch A., Pisarski G. D. *QCD and the chiral critical point*. Physical Rev. D49. 1994

[11] Joanes D.N., Gill C.A. *Comparing measures of sample skewness and kurtosis*. J. Royal Stat. Soc. D47. 1998