Broad Histogram Algorithm Comparison for the Square Well Fluid

By
Tanner Simpson

A THESIS

submitted to

Oregon State University

University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Physics
(Honors Scholar)

Presented June 04, 2018
Commencement June 2018

AN ABSTRACT OF THE THESIS OF

Tanner Simpson for the degree of Honors Baccalaureate of Science in Physics
presented on June 04, 2018.
Title: Broad Histogram Algorithm Comparison for the Square Well Fluid

Abstract approved:

David Roundy

Thermodynamic properties of systems are often investigated computationally. Traditionally, thermal physics simulations are limited by their very small energy ranges and slow convergence. Broad histogram algorithms are a class of Monte Carlo algorithms that can explore an entire energy (and temperature) range in one thermal physics simulation: potentially saving months of compute time. In this paper, we investigate broad histogram methods designed inside (SAD, TMI, and TOE) and outside (Wang-Landau, Transition Matrix Monte Carlo, Wang Landau Transition Matrix Monte Carlo, and Stochastic Approximation Monte Carlo) of the Roundy research group. The square well potential of thermodynamics serves as the algorithm testing platform. This thesis covers the motivation and theory behind each histogram method. We then investigate algorithm performance on two sets of system configurations by analyzing their uncertainty and error when computing each system's entropy over time.

Overall, three algorithms developed in group, SAD, TMI, and TOE consistently converged to low errors. In addition, SAD, TMI, and TOE were straightforward to prepare for simulation, as there aren't any user defined parameters. Wang-Landau Transition Matrix Monte Carlo (WLTMMC) converged to low error and low uncertainty rapidly, but required a predefined energy range. This research demonstrates that the popular Wang-Landau algorithm (while collecting lots of independent samples) may not work well for all systems and could take longer to correct system properties.

Key Words: Monte Carlo, Square well fluid, Broad histogram, Thermodynamics

Corresponding e-mail address: tsimpson1616@gmail.com

Honors Baccalaureate of Science in Physics project of Tanner Simpson presented on June 04, 2018

APPROVED:

David Roundy, Mentor, representing Department of Physics

Bo Sun, Committee Member, representing Department of Physics

Janet Tate, Committee Member, representing Department of Physics

---

Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University Honors College. My signature below authorizes release of my project to any reader upon request.

Tanner Simpson, Author

Contents

List of Figures

1 Introduction

Computational physics serves as a bridge between theory and experiment. Computer simulation allows for investigation of multiple theoretical approaches, without the cost of experimentation. As such, equilibrium properties of thermodynamic systems are often calculated computationally. Thermodynamic problems are usually either investigated through brute force molecular dynamics, or stochastic Monte Carlo approaches.

Monte Carlo methods are often used in this computation, since results at lower energies are statistically difficult to obtain through standard molecular dynamics simulations. However, conventional Monte Carlo algorithms are limited in thermodynamics by their small energy ranges. Broad histogram (sometimes called 'flat histogram' or 'multicanonical') methods are a class of Monte Carlo algorithms which combat this problem. Broad histogram algorithms are able to explore every desired energy of a system. Theoretically, running a single broad histogram simulation should be faster than running multiple canonical Monte Carlo simulations. However, in practice, broad histogram simulations are still very slow to converge to correct results, so an efficient algorithm could save immense compute time [1].

Traditionally, most broad histogram algorithms are first developed to analyze the Ising model. Since the Ising model is one of the most simple systems to simulate, it could hide algorithm inefficiencies. In addition, comparatively little has been done to study broad histogram efficiency on fluids. In this paper, we apply several broad histogram algorithms to the square well fluid of thermodynamics. We test several algorithms on two different system configurations: differing in number of atoms, density, and energy range. The algorithms are compared by computing system entropy and uncertainty as functions of compute time. We then highlight the characteristics of each algorithm that are potentially advantageous in simulation.

1.1 The Square Well Fluid

The square well potential is used in thermodynamics because of its relative simplicity [2] when compared to other potentials (Lennard Jones, Yukawa, etc). Rather than a continuous potential, the square well fluid only has two possible energy states per atom, as in Figure 1. The square well fluid is the most straightforward model to incorporate attraction, which is a requirement for a phase transition between gas and liquid states. Theoretically, the behavior of the square well fluid is well understood.

Figure 1 shows the potential as a function of r, where r is the radial distance between two atoms in the fluid. The potential is infinite within the radius of each atom, σ, ensuring that no two atoms will overlap. The lack of possible overlap between atoms categorizes the square-well as a 'hard-sphere' potential (the other potentials listed above are 'soft-sphere' potentials: allowing overlap). Between σ and $\lambda\sigma$ exists the only attractive region of the potential, of magnitude ε. Beyond $\lambda\sigma$, the potential

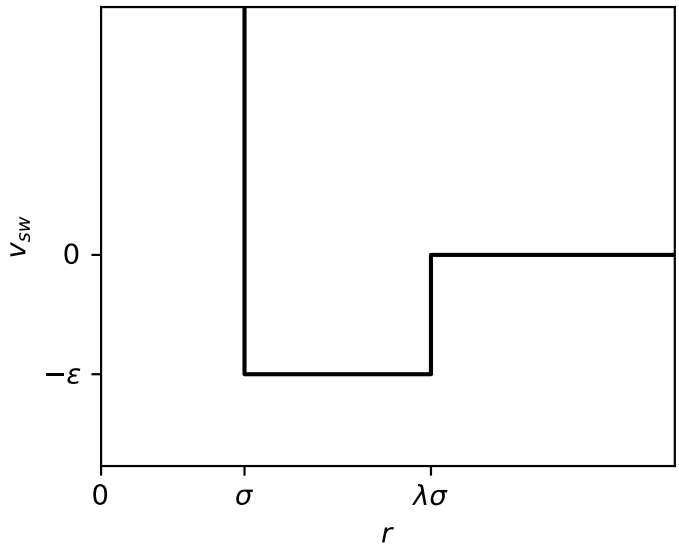

Figure 1: The square well potential as a function of distance between atoms

is zero, which dictates that distant atoms will not feel one another's effects. The square well potential can be mathematically expressed as a piecewise function:

$$V_{sw}(r) = \begin{cases} \infty & r \leq \sigma \\ -\varepsilon & \sigma < r < \lambda\sigma \\ 0 & r \geq \lambda\sigma \end{cases} \tag{1}$$

If the potential is applied to multiple atoms, then the total potential energy (E) of the system is expressed as a summation over each atom (i and j):

$$E = \sum_{i<j} V_{sw}(r_i - r_j) \tag{2}$$

Throughout this paper (and in simulation) the kinetic energy contribution from the ideal gas formulation is ignored, as its contribution is trivial and is easy to solve for analytically if necessary. This paper will also use dimensionless units for simplicity, with energy E/ε, temperature $k_B\mathrm{T}/\varepsilon$, distances r/σ, and density (in terms of a filling fraction, η).

Other models may be more accurate to real-world fluid behavior, but the square well is generally much easier to compute, debug, and comprehend.

1.2 Monte Carlo Simulation

Monte Carlo [3] methods, named after the famed casinos of Monaco, utilize random number generation to solve problems that are either difficult or impossible to solve

through conventional calculation. Monte Carlo methods apply operations repetitively to continuously generate statistics, rather than obtaining direct results. The longer the simulation is run, the more accurate statistics generated. If run for long enough, a well-written Monte Carlo algorithm should *always* compute the correct answer for a problem. In physical computing, Monte Carlo simulation is a complement to molecular dynamics: where the physical trajectories of atoms are tracked and directly computed for every interaction.

In our research, Monte Carlo algorithms are used to determine the movement of atoms. Each atom is either moved or held steady, ultimately depending on the number that is generated. Monte Carlo allows each simulation to pursue arrangements of atoms that are physically unlikely, while equally worth exploring and understanding. For instance, molecular dynamics simulations have difficulty in exploring low energy configurations, as they are statistically unlikely. In the study of fluids, the lower energies are associated with liquid states. This means that a molecular dynamics simulation could take large amounts of time to obtain data on liquids. However, a Monte Carlo simulation can obtain statistics on these liquid states much more often.

A Monte Carlo algorithm's biasing toward certain energies is accomplished by introducing a weighting function, $w(X_s)$, where X is an arbitrary system property that depends on a system state, s. The weight function can be used to determine the probability of accepting a change (referred to in this text as a move) in a simulation:

$$P_m(s_i \to s_f) = \min\left(\frac{w(X_f)}{w(X_i)}, 1\right) \tag{3}$$

In equation (3), s_i and s_f represent the initial and final states, respectively. If the weights (weight function) favor the final configuration, X_f over the initial configuration X_i, then $w(X_f) > w(X_i)$. If this is the case, the move will always be accepted. If the reverse is true, $(w(X_i) > w(X_f))$ then the move will be accepted according to the ratio of the two weights. This type of simulation is called Metropolis-Hastings Monte Carlo [4]. In our simulations, the log of the weights, $\ln(w)$ is stored to reduce floating point data storage and roundoff error.

A conventional weighting function used by physicists in a canonical Monte Carlo simulation relies on a particular temperature, T_0 and the Boltzmann factor:

$$w(x) = e^{-\frac{E}{k_B T_0}} \tag{4}$$

However, since the weights are chosen at a particular temperature, the simulation is limited in the ranges of energies allowed at that temperature. Individual canonical Monte Carlo [5] simulations may leave significant energies of interest unexplored, as shown by the energy histogram in Figure 2. To thoroughly explore an energy range, multiple canonical Monte Carlo simulations must be run, which is inherently inefficient[1] when considering real-world time.

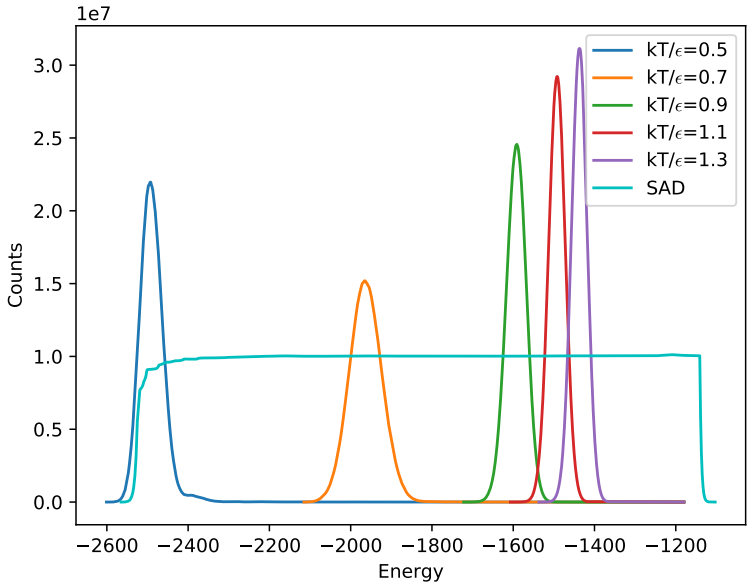

Figure 2: An energy histogram from a simulation calculated with several canonical, fixed temperature simulations (shown in dark blue, orange, green, red, and purple) and one broad histogram simulation (SAD, in cyan). Note the superior energy range of SAD when compared to the canonical simulations.

1.3 Broad Histograms

Broad histogram methods are a class of Monte Carlo simulations used to calculate the thermodynamic properties of a system. At each algorithm's core, Monte Carlo simulation is used. Like standard Metropolis-Hastings Monte Carlo, broad histogram methods use a biasing system to spend compute time at energies with limited statistics. Broad histogram algorithms output data continuously, while gradually converging to the correct result. Broad histogram algorithms generate statistics at *every* energy within a desired range: differing from their canonical Monte Carlo counterparts.

For many situations, gathering data at a wide range of energies for a system is advantageous. Still, results are often remarkably slow in converging to high accuracy. Simulations can take days, weeks, or longer when accurately computing a system property for all energies. Compute time varies with system properties, such as the particle number and density. Individual broad histogram methods balance efficiency and accuracy differently.

2 Theory

In this section, the theory behind each algorithm will be summarized. Each of the tested algorithms can be classified into one of three groups: weighted, transition matrix, or hybrid. In thermal physics simulation, one typical property of interest is an algorithm's density of states (DOS), as other properties are easily calculated from the DOS. The density of states contains much of the useful information about a thermodynamic system. Though each type of algorithm aims to accomplish the same task —finding a correct density of states for a system— they bias and calculate in different ways.

2.1 Weighted Algorithms

Each of the weighted algorithms listed here are generally tied back to the Wang and Landau Algorithm (Wang-Landau) developed in 2001. Rather than using a transition matrix (as discussed in the next section) to store probabilities, Wang-Landau looks at the weights taken from the energy histogram. Each of these algorithms adjust their weights after every move in order to sample every energy with equal number (achieve a flat histogram). Weighted algorithms are often simpler and easier to program than their transition matrix or hybrid counterparts. The three weighted algorithms tested are Wang-Landau, SAMC (Stochastic Approximation Monte Carlo), and the Roundy group's SAD (Dynamic Stochastic Approximation).

The weighted algorithms calculate a system's density of states (D) by taking the inverse of the weight function:

$$D(x) \propto \frac{1}{w(x)} \tag{5}$$

Each of the above are functions of a particular state, x.

2.1.1 Wang-Landau

Wang-Landau, designed as a random-walk in energy space, is an established method that has been in use since the early 2000's [6]. Originally developed for the Ising model, Wang-Landau has been recently applied to fluids. Wang-Landau is known to be exceptionally fast in certain configurations. Wang-Landau modifies the weights, w throughout the simulation. After each move (k), Wang-Landau decreases the weights on a particular energy by a factor, which we call the Wang-Landau factor, e^{γ}. For the original Wang-Landau method, gamma begins as 1 and decreases by 1/2 after each flatness criteria is met (we still refer to it as gamma for reasons that become apparent when looking at SAMC and SAD). The log of the weights are updated according to the equation below:

$$\log(w(E)_{k+1}) = \log(w(E)_k) - \gamma \tag{6}$$

This equation describing the weights is the log of the relationship mentioned in the original paper (this also applies to the update factor, which is referred to in the paper as f_k). The weights decreasing every time an energy is reached ensures that compute time is still spent at energies that are statistically unlikely. Since the weights are continuously modified, Wang-Landau should therefore eventually achieve a flat histogram. Wang-Landau biases the system to spend time at all energies equally rather than just the statistically most likely. Once the histogram reaches a desired level of flatness, the histogram is reset and the Wang-Landau factor is decreased by a set amount. The process is repeated until a cutoff is reached (which I will call the Wang-Landau cutoff, c).

Wang-Landau is designed to explore energy states very quickly and gather information. However, Wang-Landau has some significant drawbacks. Wang-Landau is not guaranteed to converge for all systems. It has been previously demonstrated to gather poor statistics in certain configurations. Perhaps its biggest downside lies in its input parameters. Prior to running a thermodynamic simulation with Wang-Landau, one must input the energy range. If a particular simulation is running for the first time, the energy range is not often known beforehand.

2.1.2 SAMC

Stochastic Approximation Monte Carlo (SAMC) is a recent algorithm developed by Liang [7] that improves Wang-Landau. The creators of SAMC wanted to design an algorithm that behaved similarly to Wang-Landau, but could be guaranteed to converge. SAMC also does not require the energy range as an input. The Wang-Landau factor is replaced by an SA factor, γ^{SA}. The SA factor is continuously adjusted throughout the simulation, rather than a factor that decreases when flatness is achieved. Other than these modifications, SAMC is essentially Wang-Landau. The SA factor depends on two variables, t and t_0:

$$\gamma^{SA} = \frac{t_0}{\max(t_0, t)} \tag{7}$$

The t corresponds to the total number of moves achieved by the simulation. The t_0 is an arbitrary constant selected before the simulation is run. In theory, t_0 should roughly correspond to the total number of moves required to explore every energy. With a correct value of t_0, the simulation should converge to the correct result, with a rate of statistical error proportional to the square root of moves. However, t_0 can be impossible to predict, as shown in Liang's original paper. Moreover, an incorrect value of t_0 can negate convergence of the density of states. This could potentially make SAMC inconvenient to use, as one does not often know beforehand the total number of moves required to explore every energy of the system.

2.1.3 SAD

Comparable to SAMC, SAD utilizes the same formula to adjust the histogram weights. However, guessing a proper t_0 parameter with SAMC has proven to be nontrivial in the past [7]. SAD (Dynamic Stochastic Approximation) is currently under development in the Roundy research group. With SAD, we consider that the gamma factor effectively represents a change in the log of the density of states. Since the log of the density of states is the entropy (ignoring the Boltzmann constant), γ is interpreted as a change of entropy. The total change in entropy that a occurs throughout a simulation can be expressed as:

$$\Delta S = \frac{1}{3}\frac{E_{\max} - E_{\min}}{T_{\min}} \tag{8}$$

where E_{max} is the maximum entropy state and E_{min} is the minimum energy reached at a given compute time in the simulation. The one third factor arises from the geometric ratio in areas above and below a density of states (versus energy) curve. Since we want gamma to represent a small change in the entropy at a point during the simulation (essentially a time derivative of entropy), we can divide by the number of moves (t) and multiply by the total number of energies.

$$\gamma^{SAD} = \frac{1}{3}\frac{N_E}{t}\frac{E_{\max} - E_{\min}}{T_{\min}} \tag{9}$$

The SA factor changes throughout the simulation as new energies are reached. As the simulation progresses, γ^{SAD} tends to decrease by dynamic adjustment. This should guarantee eventual convergence to the correct DOS of the simulation, which is not guaranteed in practice by standard SAMC if an incorrect t_0 is chosen.

2.2 Transition Matrix Algorithms

Transition matrix algorithms store the probabilities of accepting or rejecting moves in a large, normalized matrix. Each of the transition matrix (TM) algorithms have their roots with TMMC, Transition Matrix Monte Carlo. The three TM algorithms we have tested are TMMC, TMI (Transition Matrix Initialization), and TOE (Transitions Optimized Ensemble). From this matrix, different system properties can be calculated. A significant difference between TM algorithms and weighted algorithms is the former's direct calculation of the density of states, rather than relying on the energy histogram. In practice, probabilities are first stored in a collections matrix first and then later normalized into the true transition matrix.

In our research, the columns and rows represent different energy configurations of the square-well fluid that we are investigating. Mathematically, I refer to the transition matrix as T, with the ith and jth entry being T_{ij}. T_{ij} represents the probability that the system will transition from energy E_j to energy E_i after an atom

is moved. The size of T is the greatest possible energy transition, from the maximum entropy state to the minimum important (arbitrary) energy.

Calculating the density of states from this transition matrix is straightforward, as the matrix must have an eigenvalue of one, since the total probability of moving from one energy state to another must be 1. If we represent the transition matrix as an operator, then we can write the following eigenvector-eigenvalue equation, with D again as the density of states

$$TD = D \tag{10}$$

Therefore, solving for the density of states for a transition matrix approach is equivalent to finding the eigenvectors of T. Solving for these eigenvectors is typically done via Gauss-Jordan elimination. Different TM algorithms approach this process in a variety of ways.

2.2.1 TMMC

TMMC (Transition Matrix Monte Carlo) was originally developed by Fitzgerald [8] to guarantee convergence of results rather than the weighted histogram methods.

TMMC consists of two main steps, where the first is identical to the standard Metropolis Monte Carlo technique. First, the probability of accepting a particular move is determined, as shown in equation (3):

$$p_a = \min\left[1, \frac{T_{k\to k+1}}{T_{k+1\to k}}\right] \tag{11}$$

Here, k denotes the current iteration of the transition matrix (before the upcoming move). The final step records each of these probabilities in a transition matrix as described above.

2.2.2 TMI

TMI (Transition Matrix Initialization) is a TM algorithm developed in our group. Though similar to the original TMMC, TMI instead starts at high energies and systematically works its way down to ensure convergence. TMI checks itself throughout a simulation. TMI has two conditions that are met at each energy, before the algorithm works its way down.

1. The density of states must be decreasing at E_i relative to E_{i-1}
2. The relationship between the density of states at E_i and E_{i-1} must be smaller than a set number of round trips (a measure of uncertainty that will be detailed in the methods section and the TOE subsection). This is expressed mathematically as follows, with N representing the pessimistic samples:

$$\frac{D(E_i)}{D(E_{i-1})} < \frac{1}{\sqrt{N_i}} \tag{12}$$

If the two conditions are met at energy E_i, the weight array is set as

$$\ln w(E_i) = \ln D(E_i) \tag{13}$$

If the two conditions are not met at the energy E_i, then the weights at lower energies are set according to a tangent line, with slope β_{tan}.

$$\beta_{tan} = \frac{\ln D(E_{i-2}) - \ln D(E_{i-1})}{E_{i-2} - E_{i-1}} \tag{14}$$

The slope of this line should always be positive, as $\ln D(E_{i-1}) > \ln D(E_{i-2})$. E_{i-1} and E_{i-2} are both converged to their correct values here. Previous iterations of TMI used different slopes for β. TMI3 (as seen in the results below) denotes the third iteration of TMI algorithm. The tangent line is implemented to aggressively push the algorithm to explore the lower energies. Using the tangent line slope, the weights are set at and below E_i as:

$$\ln w(E_i) = \ln w(E_{i-1}) - \beta_{tan}(E_{i-1} - E_{i-2}) \tag{15}$$

To determine when the algorithm has sufficiently sampled the energy range set prior to the simulation, the following check is applied:

$$\ln \frac{D(E_i)}{D(E_{i-1})} > \frac{E_{i-1} - E_i}{k_B T_{min}} \tag{16}$$

If the condition is met, then the weights are set to the canonical weights at all lower energies.

2.2.3 TOE

TOE (Transitions Optimized Ensemble) is another TM algorithm developed in our group. TOE builds off of the optimized ensemble approach developed by Trebst[9]. TOE differs from the other algorithms tested by not necessarily pushing for a flat histogram. Instead, TOE is designed to eliminate uncertainty as fast as possible, by maximizing the number of round trips (defined in section 3.4).

An optimized ensemble approach treats a simulation as a system of 'walkers' moving up and down throughout an energy range. As a walker move up and down through energy space, the walker eventually reaches the maximum energy E_+ (in our case, the max entropy state, as explained in the methods section), as well as the lowest energy known to it, E_-. If the walker more recently encounters the highest energy, the walker is considered 'down-moving'; if the reverse is true, the walker is 'up-moving'. The density of walkers (n_w) at a given energy is a combination of the number of up and down moving walkers:

$$n_w = n_{up} + n_{down} \tag{17}$$

Trebst's optimized ensemble maximizes the number of independent samples by relating n_w to the histogram diffusivity α:

$$n_{opt} \propto \frac{1}{\sqrt{\alpha}} \tag{18}$$

$$\alpha(E) = \langle (E_f - E_i)^2 \rangle - \langle E_f - E_i \rangle^2 \tag{19}$$

E_f and E_i are the final and initial energies when considering an algorithm move, as described in the the background. N_{opt} is the 'optimal' density of walkers to maximize round trips.

TOE is similar to the optimized ensemble approach above, but instead, the histogram diffusivity is calculated from the transition matrix rather than a histogram:

$$\alpha(E) = |T(E_f - E_i)^2 - (T(E_f - E_i))^2| \tag{20}$$

TOE also applies the same checks and method as TMI, working its way from high to low energies.

2.3 Hybrid Algorithms

Hybrid algorithms are a recent development, attempting to merge the benefits of both weighted and transition matrix algorithms [10]. The only hybrid algorithm that I test is Wang-Landau Transition Matrix Monte Carlo, or WLTMMC.

2.3.1 WLTMMC

As its name suggests, WLTMMC is a hybrid between Wang-Landau and TMMC. WLTMMC is designed to take advantage of Wang-Landau's ability to rapidly explore an energy range, while utilizing TMMC's guaranteed convergence. Wang-Landau is run up until a desired cutoff (level of histogram flatness) and then switched to TMMC for convergence[10]. However, the Wang-Landau portion is a bit different, as the data is stored in both a histogram and the transition matrix (like a transition matrix algorithm). The density of states is calculated via the transition matrix, rather than the histogram.

The original (vanilla) Wang-Landau typically relies on a cutoff value of approximately 10^{-10}. For WLTMMC, this cutoff is traditionally much smaller, on the order of 10^{-4} (however, previous papers have been vague on the exact cutoff criteria). This is done to mitigate the possible errors of Wang-Landau, as TMMC is guaranteed to converge to correct results. Wang-Landau eventually reaches a 'statistical plateau' such that running for higher flatness doesn't directly translate to correct results.

3 Methods

This first part of this section will explain the parameters and simulation techniques used to generate comparison data. The second half of the methods section explains how and why the algorithms are compared.

3.1 Simulating and Bookkeeping

The atoms are initialized uniformly in a three dimensional cubic grid, with length specified by the number of atoms, the size of atoms, and the desired filling fraction. Periodic boundary conditions are employed at each wall to better approximate the fluid given only finite atoms. Each atom is assigned the potential as described above in Figure 1. An atom is randomly selected from the grid and is then assigned a translation distance given by a Gaussian distribution. The move is instantly rejected if the spheres overlap (which is theoretically justified, as the potential between two atoms would be infinite (1)). The change in energy is calculated and then the move is either accepted or rejected according to the particular simulation method. The process is repeated until the end conditions are reached.

Throughout the duration of a simulation, the transition matrix and energy histogram data sets are continuously stored, recorded, and altered. In addition, rather than directly focusing on the raw density of states, each broad histogram method utilizes the log of the density of states. This is because the density of states values are often exceptionally large (sometimes as large as e^{300}) and floating point numbers are difficult to store and perform operations on. The data output is stored as text files, which can require large amounts of storage. To combat this problem, the extracted data (in the form of a density of states value) is only recorded in a text file every half hour. The output half hour time scale was somewhat arbitrarily chosen, but shouldn't aid one algorithm over another. The calculations shown in the results section below utilize the data from these text files.

3.2 Simulation Parameters

3.2.1 Energy Range

Each algorithm is confined to an energy range, from the minimum important energy (E_{min}) to the max entropy state, (E_{max}). The majority of our algorithms calculate this range throughout the simulation: with E_{max} being calculated at the beginning and E_{min} calculated consistently. The max entropy state corresponds to the energy limit as $T \to \infty$. As in Equation (4), as $T \to \infty$, $w(x)$ approaches one. For energies above the max entropy state, the weights are also set to one. This means that the energies above the max entropy state are very rarely sampled, which effectively confines the simulation below E_{max}.

The minimum important energy is the cutoff for which lower energies play an insignificant part in computing system properties [11]. The minimum important energy depends on a parameter, T_{min}, or the minimum temperature. Though broad histogram algorithms allow for an infinite range of temperatures explored, we generally choose a T_{min} such that the transition between gaseous and liquid states occurs, but not so low that immense time scales are required for good statistics. The value of the temperature for which the transition occurs is usually found by previous simulation. E_{min} can be expressed as:

$$E_{min} = \frac{1}{k_B T_{min}} \tag{21}$$

Here, the minimum important energy depends directly on the density of states. As the density of states is not known beforehand, calculation of the minimum important energy uses the most recent density of states calculation of the simulation.

In addition to the minimum temperature parameter, WL and WLTMMC require a directly specified energy range (in the form of an E_{max} and E_{min}). For this input, TMI is run for sufficiently long to determine the energy range before running WL or WLTMMC.

3.2.2 End Conditions

To decide if a simulation has finished, we require that a given number of round trips (as discussed in section 4.3) are reached at a minimum important energy (corresponding to the specified minimum temperature). Many Monte Carlo simulations run for a desired number of moves or iterations, rather than a number of round trips. As described below, a specified number of round trips as a cutoff criteria reveals the uncertainty. Often, this is more useful as one does not often know how many iterations to run for, but one can decide on a desired level of uncertainty to attain. For comparison purposes, the simulations run for a near infinite amount of time (and theoretically, obtain better statistics as they progress). Infinite runtime simulations are useful as they can provide information on every algorithm at any desired number of moves.

Each algorithm is compared to a 'golden' standard for a particular filling fraction and number of atoms. The golden standard is an simulation that has run for much longer than the other simulations when compared move by move. In this paper, I use TMI as the golden standard, as it has been previously demonstrated to converge to the correct results in group. For our method of comparison, we have chosen to look at the entropy error and the uncertainty as functions of compute time (or number of moves).

3.3 Entropy and Normalization

The entropy (S) is related to the density of states (Ω) through the Boltzmann interpretation of entropy (neglecting the additive constant rising from using the density of states, rather than direct multiplicity):

$$S = k_B \ln \Omega \tag{22}$$

In this paper, D is used instead of Ω for clarity. Previous research has used the entropy as a means of comparing algorithms [10]. The entropy is useful to calculate as it is a common thermodynamic property of interest. The entropy is also straightforward to infer from the density of states, which in turn is easily found from the histogram or transition matrix (in equations (10) and (5)). In our simulations, the constant k_B is ignored, as it isn't needed for comparison.

For this paper, both the maximum and average entropy error are found for each algorithm, over all simulation moves (k).

$$S_{error}^{max}(k) = \max(S_{golden}(E_i) - S_{method}(E_i)), \text{for all } E_i \tag{23}$$

$$S_{error}^{avg}(k) = \frac{S_{method}(E_i) - S_{golden}(E_i)}{E_{max} - E_{min}}, \text{for all } E_i \tag{24}$$

Since different algorithms may shift the entropy by an arbitrary constant, the error needs to be normalized to allow for common comparison between algorithms. The normalization is done through the simple linear algebra technique of setting the average entropy for different algorithms (across the same system) equal to one another:

$$norm\ factor = \text{mean}(\ln D_{method}(E)) - \text{mean}(\ln D_{golden}(E)) \tag{25}$$

In practice, the errors are then rewritten in terms of the density of states, alongside the normalization factor:

$$S_{error}^{max}(k) = \max(\ln D_{method}(E_i) - \ln D_{golden}(E_i) - norm\ factor), \text{for all } E_i \tag{26}$$

$$S_{error}^{avg}(k) = \frac{\ln D_{method}(E_i) - \ln D_{golden}(E_i)}{E_{max} - E_{min}} - norm\ factor, \text{for all } E_i \tag{27}$$

Here, E_{max} and E_{min} refer to the max entropy state and minimum important energy, respectively.

3.4 Uncertainty and Round Trips

We computationally determine the uncertainty from round trips. One round trip (sometimes referred to as a pessimistic sample [11]) occurs when the simulation

reaches a particular energy, samples all of the energies up until the maximum energy state and then returns to the particular energy. This definition of a measure of uncertainty was first suggested by Trebst in his optimized ensemble paper [9]. One round trip represents one statistically independent sample of an energy.

Round trips have also been used previously in group as a minimum criteria to dictate when a simulation should end. Round trips are computed for every energy in between the minimum energy of interest and the maximum energy state. The round trips are related to the uncertainty through Poisson statistics:

$$\sigma \approx \frac{1}{\sqrt{N}} \tag{28}$$

where N is the number of round trips at that energy. In theory, an efficient algorithm should obtain more round trips at an energy over time than a slow algorithm. However, many round trips may not necessarily imply low error.

4 Results and Discussion

4.1 Entropy Error

The error in entropy is calculated throughout the entirety of the simulation (as measured in moves). Figures 3 and 5 show the average entropy error across all energies for both test systems; Figures 4 and 6 show the maximum error across all energies for both test systems. In both simulations, the golden standard used for comparison is a TMI calculation that has run for approximately 10^{13} moves. Note that each plot has logarithmically scaled axes.

For the 50 atom simulation (Figure 3), there is a definite distinction between methods that do and do not converge to a reasonably correct (average error less than roughly 10%) result within 10^{13} moves. TMI and TOE converge somewhat more slowly than the 10^5 (referring to SAMC's t_0 parameter) SAMC and SAD, but slightly faster than the 10^{-10} cutoff WLTMMC and 10^4 SAMC. The 10^3 SAMC and Wang-Landau seem as if they won't converge to the correct entropy at any number of moves. WLTMMC with a 10^{-4} cutoff and TMMC don't appear to completely converge to a correct result here, but may with more moves.

Between the weighted algorithms, Wang-Landau looks especially poor, as it achieves higher entropy error before slowly coming down. SAD converges similarly to the 10^5 SAMC, agreeing with the theory that SAD 'chooses' the t_0 to match the best version of SAMC. The maximum error plot behaves similarly to the average error plot. There is roughly a factor of ten difference between the average error and the maximum error at any move.

For the 500 atom simulation, there are similar behaviors. There is another distinct difference between methods that converge and those that do not. Here, both varieties

of WLTMMC appear to converge to the correct result, alongside TOE and TMI. SAD and TMMC appear to be gradually converging, though not as rapidly as the other methods. Again, two different SAMC runs (10^3 and 10^4) appear to be unlikely to converge in any reasonable amount of time. Wang-Landau also appears worse than the 50 atom simulation and does not converge.

Figure 9 below shows the weight adjustment factor, γ for each of the three weighted methods. Though not a direct comparison of error or uncertainty, this graph helps visualize the differences between SAMC, SAD, and Wang-Landau. Wang-Landau rapidly lowers the adjustment factor, which yields inadequate statistics when averaged over all energies in a range. SAD and SAMC take a more gradual approach to lowering γ. For the algorithms that converge in the 500 atom simulation (10^6 and 10^5 SAMC and SAD), the weights are more slowly adjusted than their counterparts. The energy range needs to be sufficiently sampled before adjusting γ to yield accurate results.

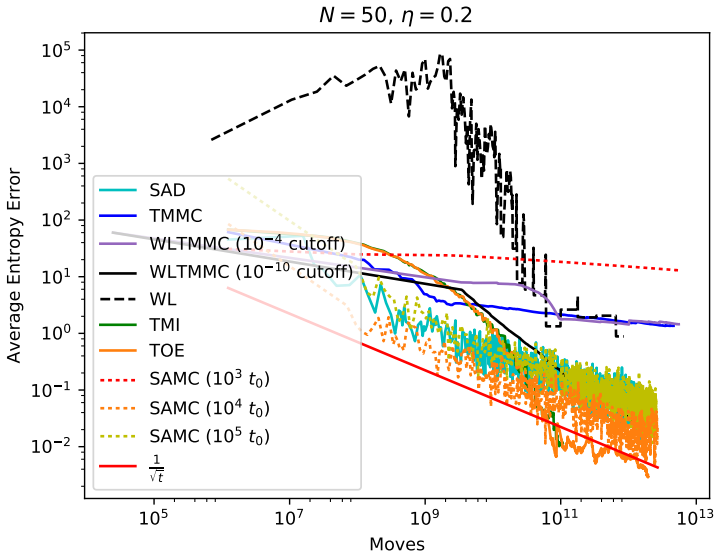

Figure 3: Average entropy error for a 50 atom, 0.2 filling fraction simulation. The red line is provided as a reference, with a slope of $\frac{1}{\sqrt{t_0}}$. TMI is used as the golden standard

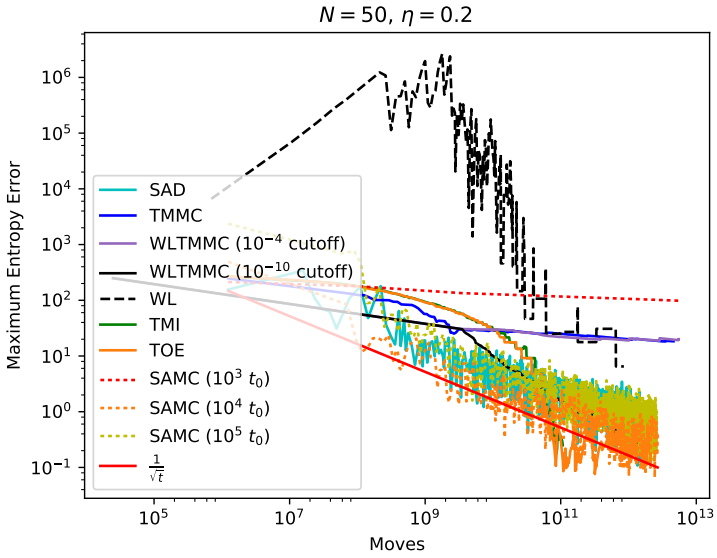

Figure 4: The maximum error attained for a 50 atom, 0.2 filling fraction simulation. The red line is provided as a reference, with a slope of $\frac{1}{\sqrt{t_0}}$. TMI is used as the golden standard.

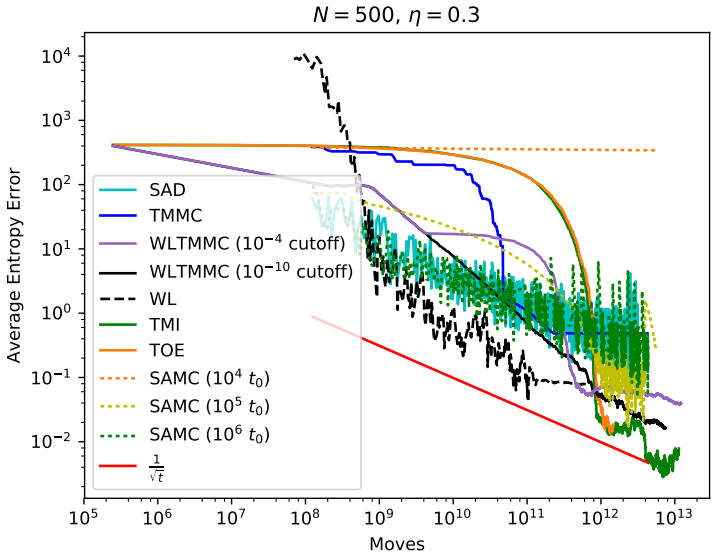

Figure 5: The average error attained for a 500 atom, 0.3 filling fraction simulation. The red line is provided as a reference, with a slope of $\frac{1}{\sqrt{t_0}}$. TMI is used as the golden standard.

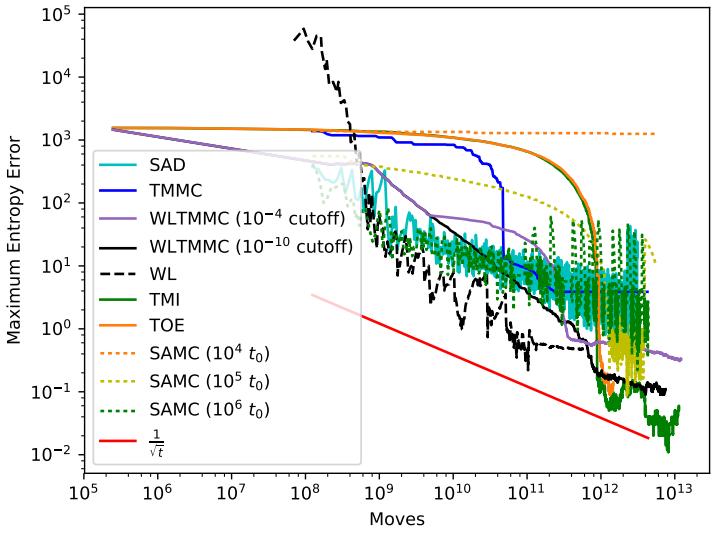

Figure 6: The maximum error attained for a 500 atom, 0.3 filling fraction simulation. The red line is provided as a reference, with a slope of $\frac{1}{\sqrt{t_0}}$. TMI is used as the golden standard.

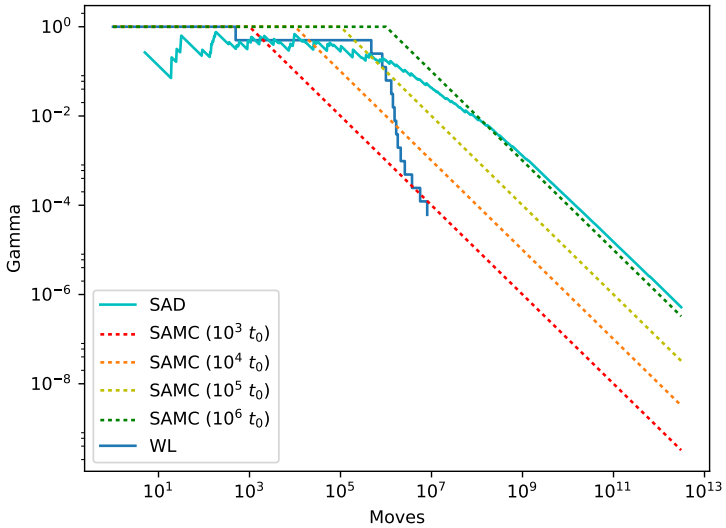

Figure 7: The weight adjustment factor γ, plotted as a function of moves for Wang-Landau, SAD, and four versions of SAMC. The system tested was the 500 atom configuration from above.

4.2 Uncertainty

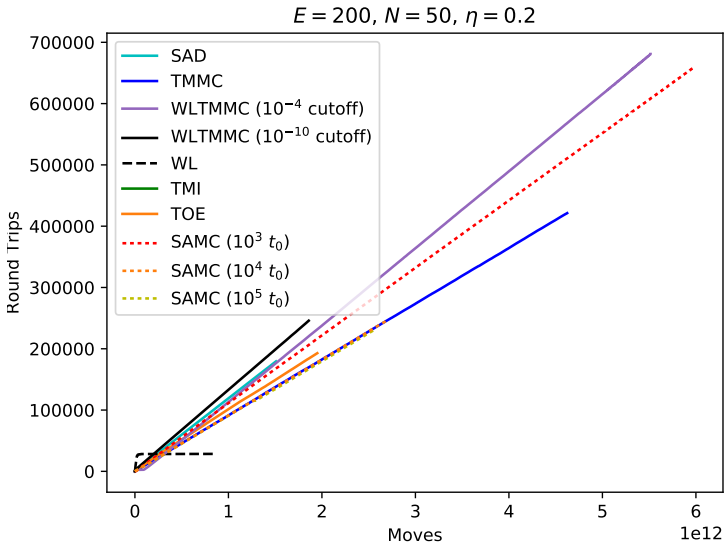

Figure 8: The number of round trips plotted as a function of moves for the energy 200 (E/ε) for the 50 atom simulation. Note that the axes are not logarithmically scaled.

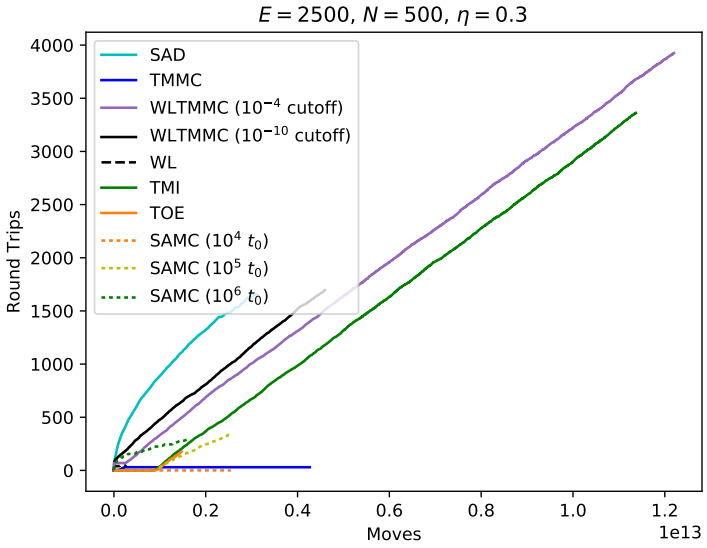

Figure 9: The number of round trips plotted as a function of moves for the energy 2500 (E/ε) for the 500 atom simulation. Note that the axes are not logarithmically scaled.

Figures 8 and 9 show the number of round trips attained by each method for both simulations at a particular energy. The energy chosen for both was somewhat arbitrary, but I chose an energy near the minimum important energy for both cases. This was done because the lower energies are often more difficult for an algorithm to explore and there are generally less round trips at lower energies than higher ones. However, this could also give TOE a slight advantage, as it is designed to generate lots of round trips at the lowest energies.

For Figures 8 and 9, a higher slope represents a more rapid loss in uncertainty at the particular energy in question. Though it may be a bit difficult to see (as it doesn't run for as many moves), Wang-Landau initially has an extremely rapid gain in the number of round trips. Other weighted algorithms tend to share this characteristic (though not as drastic) before gradually lowering the slope. WLTMMC and the transition matrix methods obtain round trips linearly with moves. TMI,TOE, and TMMC gather round trip data linearly, but often after a flat period where the energy is not reached. WLTMMC has the best of both worlds, as the Wang-Landau portion rapidly generates round trips at the start of the simulation, with TMMC taking over shortly after. As expected, the rate at which round trips does not indicate an accurate method for computing system properties.

4.3 Qualitative Assessment and Other Notes

Each of the algorithms that require an input parameter (other than the arbitrary minimum temperature) could present an obstacle for simulating other systems. For instance, Wang-Landau and WLTMMC need a predefined energy range. One does not often know the energy range under investigation: often, a Monte Carlo simulation is used to calculate it. This was a particular nuisance for the square well fluid, as another simulation (TMI) had to run for sufficiently long to calculate the energy range, which was then used to start WL or WLTMMC. Effectively, two simulations had to be run whenever using WL or WLTMMC. Depending on the user's computer setup, this could potentially double the real-world time for simulation, which is not reflected in our moves (compute time) above.

Another input parameter that presented issues was SAMC's t_0. Choosing a wrong value would cause a simulation to obtain drastically wrong data. However, unlike the Wang-Landau and WLTMMC parameter, the ill-effect of this parameter was shown in our data (Figures 3-6). In practice, if one were to investigate a thermodynamic system with SAMC, multiple simulations (of different $t_0 s$) would have to be run to find the correct parameter. Like Wang-Landau, this raises the real-world time for simulation.

Finally, our data doesn't reflect the actual process of implementing the broad histogram algorithms. Often, people investigating thermal physics systems simulate with canonical Monte Carlo, due to its straightforward and intuitive application. If a switch to a broad histogram algorithm were desired, the investigator would have to

rewrite their simulation code accordingly. Between weighted and transition matrix algorithms, there is a significant difference in the level of implementation difficulty. Weighted algorithms are similar to the canonical Monte Carlo, as they directly utilize histogram counts rather than direct probabilities of moves. Writing effective code to bookkeep a transition matrix can be difficult and time consuming. System properties (especially as calculated via the density of states) may be simpler to identify when using a weighted algorithm (compare the process of using Equations 5 and 10). Transition matrices may also be inherently more difficult to understand and implement; therefore, investigators may be more hesitant to utilize them.

4.4 The TMMC Bug and Disclaimer

Very late in the process of writing this paper, a significant bug was found in the Monte Carlo code that could have impacted WLTMMC and TMMC. This bug was likely introduced in between some of the algorithm runs. The runs that could have possibly been affected were both TMMC runs and both WLTMMC runs with a 10^{-4} cutoff. Due to the time required to simulate each configuration (roughly 30 days for the 50 atom, 60+ for the 500 atom), I haven't yet been able to produce new TMMC and WLTMMC runs that prove the bug's impact on the errors, histograms, and round trips data above. Though TMMC and WLTMMC appeared to function fine, their associated data should be interpreted with a side of caution.

5 Conclusion

We compared broad histogram Monte Carlo algorithm efficiency and efficacy when computing the entropy of the square well fluid of thermodynamics. Two systems of 50 and 500 atoms were simulated by several broad histogram methods: Wang-Landau, SAMC, SAD, TMMC, WLTMMC, TMI, and TOE. The error and uncertainty when computing the system's entropy were computed as a function of time.

Regarding the weighted algorithms, only SAD was able to converge properly, regardless of the input parameter. Wang-Landau took significant compute time on the fifty atom simulation to converge, which is slightly alarming given its established use. Out of the transition matrix algorithms, TOE and TMI achieved fairly high accuracy and low uncertainty when run for sufficiently long. TOE and TMI may be faster or slower than SAD, depending on the system. Though the data showed that TMMC could take a long time to converge, a late bug was found, so the results for TMMC (and WLTMMC) should be taken with caution. The sole hybrid algorithm, WLTMMC was able to converge to low error and low uncertainty quickly, albeit with the requirement of specifying the energy range beforehand. Overall, this research shows that the popular Wang-Landau algorithm may not converge to the correct density of states quickly for all systems.

5.1 Future Work

As mentioned in the introduction, broad histogram algorithms have been traditionally tested on the Ising model. A natural step (that is currently a work in progress) is applying each of the algorithms tested to the Ising model for further confirmation of our results.

As the goal of this research was to study broad histogram efficiency on a particular fluid potential, other potentials could be further tested. One such example is the fully repulsive WCA (Weeks-Chandler-Anderson) potential [12]. The WCA potential is a reasonably accurate description for fluids such as liquid argon. As the WCA potential represents a continuous energy distribution as a function of distance, it potentially represents a more difficult problem than the square well potential. As investigating a true, continuous distribution is not computationally feasible, the distribution must be divided into smaller bins (known as 'binning'). It is relatively unknown how each algorithm's performance will scale with the size of these energy bins and thus it provides an area of intrigue. The WCA potential is particularly chosen (rather than other continuous potentials) because of other Roundy group members researching it.

6 Acknowledgements

First and foremost, I would like to thank my advisor, Dr. David Roundy. He has taught me the ins and outs of programming and has committed countless hours helping me throughout my research. I would like to thank Jordan Pommerenck, with whom I frequently collaborated with on programming and understanding the underlying physics. I would like to thank Michael Perlin for writing the first draft of the Square Well Monte Carlo simulation code. Lastly, I would like to thank my family, for encouraging me to pursue my love of science!

References

[1] P Murilo-Castro de Oliveira, TJP Penna, and HJ Herrmann. Broad histogram monte carlo. Technical report, 1997.

[2] Jean-Pierre Hansen and Ian R McDonald. *Theory of simple liquids*. Elsevier, 1990.

[3] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[4] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[5] Alan M Ferrenberg and Robert H Swendsen. New monte carlo technique for studying phase transitions. *Physical review letters*, 61(23):2635, 1988.

[6] Fugao Wang and DP Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86(10):2050, 2001.

[7] Faming Liang, Chuanhai Liu, and Raymond J Carroll. Stochastic approximation in monte carlo computation. *Journal of the American Statistical Association*, 102(477):305–320, 2007.

[8] Robert H Swendsen, Brian Diggs, Jian-Sheng Wang, Shing-Te Li, Christopher Genovese, and Joseph B Kadane. Transition matrix monte carlo. *International Journal of Modern Physics C*, 10(08):1563–1569, 1999.

[9] Simon Trebst, David A Huse, and Matthias Troyer. Optimizing the ensemble for equilibration in broad-histogram monte carlo simulations. *Physical Review E*, 70(4):046701, 2004.

[10] Ruben G Ghulghazaryan, Shura Hayryan, and Chin-Kun Hu. Efficient combination of wang–landau and transition matrix monte carlo methods for protein simulations. *Journal of computational chemistry*, 28(3):715–726, 2007.

[11] Michael A Perlin. Optimizing monte carlo simulation of the square-well fluid, 2015.

[12] John D Weeks, David Chandler, and Hans C Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *The Journal of chemical physics*, 54(12):5237–5247, 1971.