DEVELOPING CONSTANT PRESSURE FUNCTIONALITY IN ORDER PARAMETER WANG-LANGAU SIMULATION

by

Joseph Whittier

A senior thesis submitted to the faculty of

Brigham Young University - Idaho

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics

Brigham Young University - Idaho

December 2025

BRIGHAM YOUNG UNIVERSITY - IDAHO

DEPARTMENT APPROVAL

of a senior thesis submitted by

Joseph Whittier

This thesis has been reviewed by the research committee, senior thesis coordinator, and department chair and has been found to be satisfactory.

Date	Lance Nelson, Advisor
Date	David Oliphant, Senior Thesis Coordinator
Date	Kevin Kelley, Committee Member
Date	Jon Johnson, Committee Member
Date	Evan Hansen, Chair

ABSTRACT

DEVELOPING CONSTANT PRESSURE FUNCTIONALITY IN ORDER PARAMETER WANG-LANGAU SIMULATION

Joseph Whittier

Department of Physics

Bachelor of Science

Crystallization in polymers can greatly change their properties, from a flimsy plastic bag to a cable capable of trading blows with steel. However, the process of crystallization is not yet well understood. Simulation tools that model this process are in continuous development. Making these tools more realistic, and able to mimic real systems, is a major focus of research at this time. This paper covers the transition of the program OPWL (Order Parameter Wang-Landau) from a constant-volume system to a constant-pressure system. This will allow it to more accurately reflect real-world conditions. The basic systems for the constant-pressure functionality has been created, but a number of complications are still present, and additional development is needed.

ACKNOWLEDGMENTS

I would like to thank the members of the Tree Soft Matter Research Group at Brigham Young University. In particular, Dr. Douglas Tree, Aaron Bigelow, and Marti McKendrick, who were also working on the OPWL project with me. Aaron Bigelow in particular was my close associate, and the lead for this project in the Tree Group, during my time developing this functionality. Dr. Pierre Kawak, formerly of the Tree Group, also assisted greatly in understanding his contributions to the project. Funding for my involvement in the internship was provided by the National Science Foundation's REU (Research Experience for Undergraduates) program.

Contents

List of Figures

Chapter 1

Introduction

1.1 OPWL

Order Parameter Wang-Landau (OPWL) is a molecular simulation program developed by the Tree Soft Matter Group at Brigham Young University. It was developed to determine specific properties of atomic and polymer systems, namely "order parameters". Order parameters are measures of how organized a system's configuration is. OPWL was created to study polymer crystallization, and thus primarily tracks two parameters: linear alignment of the polymer chains, corresponding to a nematic phase, and structural alignment, corresponding to a crystalline phase. (See fig. 1.1) These parameters are correlated with temperature and specific heat, determined from the density of energy states. These density of states can be determined quite precisely using the Wang-Landau algorithm.

1.1.1 Wang-Landau

The Wang-Landau algorithm, proposed by Fugao Wang and David P. Landau [1] estimates the density of states in a system through Monte-Carlo methods. It involves

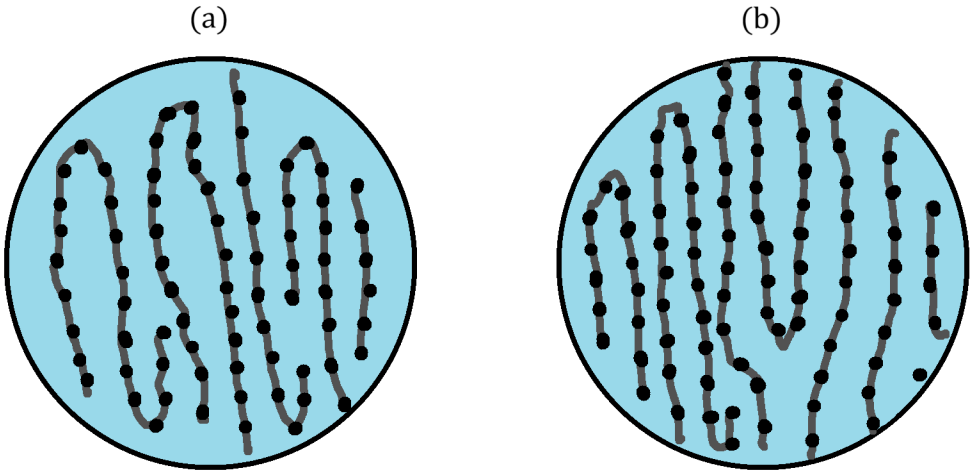

Figure 1.1 Representation of a polymer in a nematic phase (a) and a crystalline phase (b). A nematic phase only has the polymer chains aligned along an axis, while the crystalline phase has individual monomers arranged into some kind of crystalline structure.

a random walk through whatever parameter is in use, typically energy, as is the case in the current research. A histogram of energy states is built as the system's parameters are changed. For discrete sampling in real applications of Wang-Landau, a number of "bins" are designated, and the number of times the system "visits" the bin is recorded for that bin. The great advantage of Wang-Landau over purely random sampling, such as Metropolis-Hastings, is that the entropy is also calculated for that move, and added to a value for that bin. Thus, the simulation is discouraged from entering the same bin many times, and instead tries to make a flat histogram of all bins, regardless of the actual probability of that state. This greatly reduces the number of operations needed to find density of states.

After a flat histogram is achieved with the desired number of samples, it can be reset, and a parameter for the entropy calculation refined. After a sufficently precise estimation has been made, the entropy is used to calculate the density of states. This is then used to calculate most every statistical characteristic of a system, using

the partition function. For the purpose of this research, we used it to find phase transistions. A low density of states, or possible configurations, across a wide spread of energies indicates that the system does not "like" to be near that energy, meaning that it is in between two stable phases. This is critical for research into crystallization, as different states of crystallization will be in different phases.

1.1.2 Monte-Carlo

We will take a moment to review the Monte-Carlo algorithm. Monte-Carlo sampling, so named for its random sampling being similar to gambling at a casino, (only repeated until the results become statistically significant) is a method of determining some characteristic of a system using biased acceptance of random changes to the system. Not all changes are accepted, though. The acceptance criteria is some measurable aspect of the system that can be compared to a desired value to determine if an action on the system is valid or not. This is usually done with a Gaussian distribution, where the probability of acceptance is generated from the difference between the actual value, and the desired value.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{1.1}$$

where μ is the desired value, x is the measured value, and σ is the "width" of the probability curve. As the difference between the two values increases, the probability of the move being accepted drops off exponentially. If the value measured after the action is performed is close to the desired value, the action is accepted.

1.2 Constant-Volume vs. Constant-Pressure

1.2.1 NVT - Constant Volume

For a simulation of molecular behavior, there are a number of constraints that will need to be made on the system. First, a finite number (N) of particles will be necessary, as our computing power is also finite. (A large enough number can approximate an infinite system, which will give us solutions for the behavior of a bulk.) Second, we would like to simulate these particles in close interactions with one another, so they should not be free to escape the system and will be kept inside a closed volume (V) by having them "wrap" from one edge of a box to the opposite.[1] Lastly, we will need to control some aspect related to the energy of the system. Pure molecular-dynamics simulations can keep the energy perfectly conserved, but Monte-Carlo methods do not conserve energy as easily. For Monte-Carlo systems, temperature (T) is much easier to hold steady. The three constant values in such a simulation are thus abbreviated NVT.

These systems are relatively simple to implement, as the number of particles is a fundamental part of the computational design, and the volume is tightly controlled with the wrapping functionality (both particle movement and inter-particle forces are calculated to cross over the boundary). Temperature is controlled with the acceptance criteria of a Monte-Carlo algorithm. This system, however, is not particularly true to real systems. There is no immovable bounding box or magical teleportation gate available to researchers that would keep a set of particles in a fixed volume.

[1] This can cause strange effects in small-N systems, but a sufficently large number of particles will dispell these.

1.2.2 NPT - Constant Pressure

NVT is not the only design for a molecular simulation. Another is the NPT ensemble. In these simulations, the pressure (P) is held constant by regularly adjusting the size of the box to which the simulation is bound. Pressure can be calculated by taking the average of forces across some plane (one of the faces of a cubic box works nicely), and then the volume can be adjusted as another Monte-Carlo move, with a bias towards the target pressure. This is, of course, much more complicated to implement, and as will be discussed later, can cause issues when implemented into a simulation program that assumes a constant volume.

1.2.3 Purpose of NPT Transition

An NPT system aims to reflect reality a little more closely. In many systems, it is much more reasonable for a sample to be held at a constant pressure, as this is a parameter that can be more accurately controlled in a real environment (i.e. a sample at atmospheric pressure), rather than forcing a sample to remain in the same volume, down to the nanometer scale. This is important to work in polymer crystallization, as flexibility in the volume of a system may contribute significantly to the formation of nematic (strand-aligned) and crystalline structures.

1.3 Programming Structure

OPWL is a relatively large program, with the compiled source code totalling 9.5 megabytes, and more than 20,000 lines of written code. Here is given a brief overview of the structure of the program, which will aid discussion of the changes made to facilitate NPT functionality.

1.3.1 Program Files (C++)

The bulk of OPWL is written in C++ to facilitate the high-speed calculations done by compiled code. The chain of files is as follows:

- `main.cpp`
- `driv_base.cpp`
  - `driv_MC.cpp`
  - `driv_WL.cpp`
  - `driv_US.cpp`
  - `driv_md_WL.cpp`
  - `driv_md_US.cpp`
- `checkerboard.cpp`
- `mol_pol.cpp`
- `mover_pol.cpp`
- `interactions.cpp`
  - `pair_base.cpp`
  - `bond_base.cpp`
  - `angle_base.cpp`
- `fly_stats_base.cpp`

To begin, `main.cpp` calls the driver, which is then configured into one of its subtypes, depending on what was chosen in the `params.json` (parameters) file. This decides the general computation loop that OPWL will take — simple Monte-Carlo (MC), Wang-Landau (WL), Umbrella Sampling (US), or pure-molecular-dynamics (MD) versions of WL and US. Then, the checkerboard and molecule objects are constructed. The "checkerboard" is a subdivision of the physical volume of the box into sub-units for computational ease, and the possibility of parallelization of processes (which are not implemented as of writing). The molecule object (managed in `mol_pol.cpp`) manages the positions, bonds, and organization of the "beads" used as representation of molecules.

The `mover_pol.cpp` file is vital and colossal in scale, as it performs all of the moves, or changes, that a molecular system might undergo. The `interactions.cpp`

file is primarily used to calculate the energy of the system, as well as forces. (All of the "interactions" between particles.) It relies on three configurable potential functions for pair interactions, bond interactions, and bond angle interactions, which are specified in the starting parameters. The functions in `fly_stats_base.cpp` are one of the unique portions of OPWL. It can be configured to calculate various statistics, such as nematic (strand-alignment) and crystalline order parameters, energy, radius of gyration, and end-to-end distance of a polymer chain.

The driver file consists of a loop that performs the specified type of sampling, calling `mover_pol.cpp` to perform a move, then checking its probability of success with the parameters generated by `interactions.cpp` and the "on-the-fly statistic" of choice. This is repeated for the number of steps specified in the parameters, or what is decided by the window manager, when operating in Wang-Landau mode.

1.3.2 Processing Files (Python)

While C++ is excellent for efficient code, it is not so conducive to visual analysis and presentation of the data generated by a simulation. Thus, a large number of Python scripts have been written to post-process the data generated into a more useful form. The primary output of the C++ program is a running log of various values of the system throughout the simulation, `Espaced.out`, and the large number of configuration files, which contain the positions of all the molecules in the system at a given snapshot in time.

Typically, OPWL is run in parallel across many temperatures, with a number of duplicates at each temperature to isolate statistical anomalies. The post-processing scripts will collect data from across all of these runs of the simulation, and concatentate them into individual files. These concatentated data files are then analyzed to determine properties like the density of states across the energies covered by the sim-

ulation. Density of states is particularly useful, as this allows us to use the partition function to calculate the specific heat across the temperature range covered, giving us an insight as to the location of phase transitions.

1.3.3 Job Managers and BYU ORC

Given the number of simulations with highly intensive calculations that are needed to get useful information about a particular system, a supercomputing cluster is required. Brigham Young University has a massive compute cluster, the Mary Lou Supercomputer, managed by the Office of Research Computing (ORC). Access to the resources of this compute cluster is managed on a job-based system. OPWL is built with this particular cluster in mind, though adaptation to other CPU-based supercomputers should be quite straightforward.

Each configuration of temperature is its own "job", which is assigned a number of threads, one for each replicate[2] at that temperature. (A single instance of OPWL is a single-threaded task, so it can be used on nearly any computational system.) These are submitted through the supercomputer management system, which uses a bash script (a program that runs console commands independent of a user) to specify the computational resources needed, the time they will be used for, and activates the processes.

A special note should be made here about the "job managers". These are special Python scripts that create several instances of OPWL, and then periodically exchange configurations between the energy windows for a system. This helps to prevent a window "stalling" due to a specific configuration not being able to smoothly move

[2]A replicate is an instance of the program with the same control parameters, though it may have a different intial configuration. The simulations run in this research were done with 6 replicates. Multiple replicates ensure that results are statistically significant.

across energy states. These job managers presented a significant challenge in NPT implementation, as we will see later (sec. 2.3).

Chapter 2

NPT Implementation

2.1 Volume Move

There are a plentitude of "moves" that can be called by OPWL during its operation. These are designed to slightly alter the energy state of the system. These are selected randomly using a generated number. In a given cycle, each bead (representing a monomer or single molecule) in the system is given an opportunity to move; one move is selected, and random parameters (such as distance displacement) are chosen. The change in energy is then calculated, and the probability of keeping the move is determined. The volume move, obviously, need not be potentially called for every bead. Instead, at the beginning of a cycle, before the beads start trying to move, a volume move is attempted.

The "classic" type of volume move is described by Andersen [3] as a "piston" moving in and out of the side of a box. This is difficult for the particular arrangement of OPWL to use, as it is a cubic system composed of cubic cells. Instead, the volume move functions by adjusting the box size, and stretching the coordinates of all beads

correspondingly. The factor of volume move is determined by

$$\Delta x = r \cdot \Delta R \cdot x_{curr} \tag{2.1}$$

where x is the side length of the box, r is some random number $-1 < r < 1$, and ΔR is some fractional factor determined in the parameters. ($\Delta R = 0.01$ was used in most of the simulations discussed here.) The system was originally defined with a fixed maximum value for the volume to change by. However, in simulations of a Lennard-Jones Fluid, when it entered the gaseous phase, the volume did not change fast enough to reach equilibrium within the allowed wall time on the supercomputer. Thus, a factor (ΔR) of the current side length was used instead.

The side length is then adjusted by the calculated amount, and the relative change $f = \frac{x+\Delta x}{x}$ is applied to each of the coordinates of all beads. After this operation, the energy of the whole system is recalculated, and the acceptance criteria (sec. 1.1.2) is compared to determine if the change is kept or reverted.

2.2 Energy Calculation

There were, originally, two different methods of calculating energy: `totalE()` and `totalELongest()`. The `totalELongest()` function uses the most exact, if slowest, method of calculating the energy of the system. The `totalE()` function is called more often, and is faster under most circumstances, though it takes some shortcuts.

2.2.1 Energy Potentials

There are three types of energy calculations performed: pair interaction, bond distance, and bond angle. First, the pair interaction, which determines the forces between two unbonded beads, is calculated. There are several pair interaction options, depending on the type of behavior that we would like to see from the system;

Lennard-Jones, Weeks-Chandler-Anderson and "hard bead" are the most common. Lennard-Jones (LJ) was primarily used in the early stages of these simulations with the eponymous fluid. The LJ potential was calculated as

$$U(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) \tag{2.2}$$

where ϵ is some energy unit (the depth of the "well" of the potential), and σ is the distance where the potential energy becomes positive. (For $r > \sigma$, $U(r) < 0$.) This serves as an approximation of the interactions between the atoms of noble gasses, and is a standard reference model for molecular simulation. Weeks-Chandler-Anderson (WCA) is a modification of the LJ potential that only contains the repulsive part of the formulation, only using the portion for which the potential energy is positive, and remaining zero otherwise. Hard bead simply means that two beads being within the molecular radius of each other is not allowed, and returns an extreme energy for this case, and none otherwise. In `totalELongest`, one of these functions is used to calculate the potential energy of the interaction between each possible pair of beads in the system, and added to the total system energy.

Second, the bond interactions of two adjacent beads on the polymer chains are calculated, almost always using the energy of a harmonic oscillator

$$U(r) = \frac{\epsilon}{2}(r_0 - r)^2 \tag{2.3}$$

and added to the total energy. (r_0 is the equilibrium point for the function.)

Lastly, the *angle* of the bonds is taken into consideration. There are several formulations for this, using linear, quadratic, and step functions. However, the most typically used formula is again a harmonic oscillator, this time using the angle between the two bonds considered.

$$U(\theta) = \epsilon(1 - \cos\theta) \tag{2.4}$$

This is, again added to the total energy.[1]

2.2.2 Fast and Slow Calculation

While the second and third energy calculations (Eq. 2.3, 2.4) for a bead are always relevant in calculating the total energy (when polymers are being modeled), the pair interactions are negligible after distances greater than 4σ. This leads to a large number of unecessary calculations being performed in `totalELongest()`. Thus, an alternative energy calculation was added, `totalE()`. This uses the checkerboarding arrangement mentioned in sec. 1.3. The entire box is broken up into cells, and only the beads in "nearby" cells are considered for pair interactions. Periodically, the energy calculated by this function is compared to `totalELongest()` to ensure that it remains accurate. The system that previously existed was quite opaque about the distance to which this extended, so two modifications were made to the energy calculation process: choosing a more performant calculation algorithm when there are (relatively) few particles in the system, and reconstructing the existing functions.

2.2.3 New Energy Functions

Firstly, if this is a simple simulation, with less than 1000 beads, a streamlined version of `totalELongest()`, `totalELowN()` is selected, as it is more efficient than a cell-based approach. (The overhead for calculating and looping through nearby cells is greater than simply brute-forcing this number of beads.) Secondly, a new energy function was defined, `totalOpt()`. This function is a more transparent and modular cell-based approach. Using the cutoff radius, which is already in use to truncate calculations of pair interactions outside the relevant regions, a "radius" of relevant

[1] It should be noted that the ϵ used in each of these equations is independent, and specific to the type of bond and molecule being modeled.

cells is determined, and then those are looped through for pair interactions. This allowed for more flexibility in the number of cells, and reduced complications created by the size of these cells changing, as the size of cells changes with the changing volume. These small modifications made the energy calculations much more efficient. This is vital to the efficiency of OPWL, as the energy calculations make up the bulk of calculation time.

2.3 Window Manager

The last major hurdle in applying constant pressure (NPT) functionality to OPWL was the "window manager". This is a system that is only active during full Wang-Landau operation. Instead of dividing the simulation into separate submodels by the temperature, the energy of previously generated configurations is used as a reference point, and each simulation is given a range of energies to manage. As an example, say the equilibrium configuration of a range of temperatures appears as it does in table 2.1.

Temperature	Energy	Energy Window
0.5	780	500-900
1.0	1050	850-1500
1.5	1760	1400-1900

Table 2.1 Example set of energy windows. (The values are not reflective of any real simulations.)

A user of OPWL would determine three overlapping energy windows. These are then used by independent simulations that, on their own, behave as described in sec. 1.3. However, in parallel to all these simulations (there will be some number of

replicates for each of these energy windows), there is a Python script that will occasionally pause two simulations in adjacent energy windows, check if both their current energies are in the overlapping zone, and then exchange the configurations being used by these two simulations. This is particularly useful in polymer simulations, where the particular configuration a system is in will not easily move to all the energies in a window. Swapping in a new configuration increases the likelihood that all of the energies can be accessed.

There are some difficulties with this type of system, though. The system of saving and reloading configurations during an exchange is delicate. The exact filenames must be present in both the saving and reading functions, and the system for saving "backups" is also fragile, and any changes made there must be carried upstream to anywhere those configurations might be read at. However, the main problem encountered when implementing NPT to this portion of the simulation was variable control. Unlike previous versions of the code, the volume and side length parameters are not fixed. When exchanging configurations, the new volume must then be loaded.

This caused some issues, as there are multiple parameters based off of the volume that then need to be recalculated. Notably, the `Lx`, `rcell`, and `L_struct` values had to be recalculated at each reload, as well as after a volume move. These variables are the side length of the simulation box, the side length of the subcells, and an additional parameter, respectively. (The function of `L_struct` is hard to determine, but having it not be equal to `Lx` causes crashes.)

Implementing and debugging these modifications took the bulk of the research time devoted to this project. The results were periodically verified, using published research and previous work by members of the Tree Soft Matter Theory Group [2].

Chapter 3

Verification of Results

3.1 Initial Lennard-Jones Fluid

The first goal in developing constant pressure (NPT) functionality was to create a simple system that behaves in a similar way to simulations done by other researchers and in different software. A comparison model in Highly Optimized Object-oriented Molecular Dynamics (HOOMD) was run, with its results shown in fig. 3.1. This representation shows the average final volume of a given set of parameters (after the simulation reaches equilibrium) in a color varying from blue (low volume) to red (high volume). (The volume legend is given in volume per bead.) This will allow us to identify phase transitions — a critical part of the simulations OPWL is designed to do. The phase transitions in a Lennard-Jones fluid are marked by large changes in volume from solid to gas, and moderate changes from solid to liquid.

A computational study of Lennard-Jones fluids in an NPT system was conducted by Yosuke Kataoka and Yuri Yamada [4], and this research was used as a reference, to ensure that our estimated values were within a reasonable range. They compared NPT, NVT, and NVE (constant pressure, volume, and energy, respectively) simula-

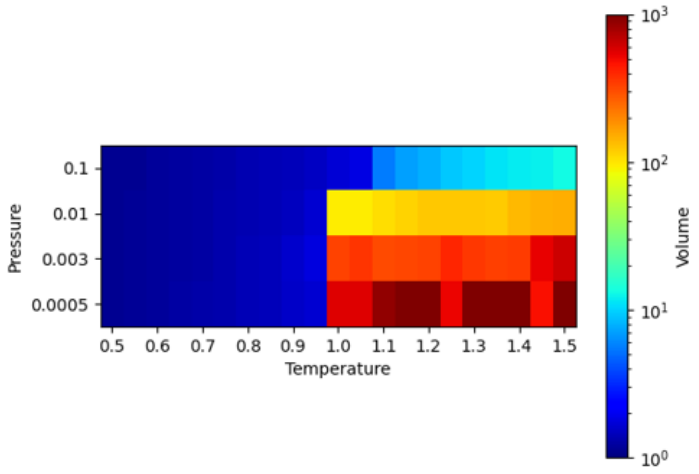

Figure 3.1 Phase diagram for the simulation run in HOOMD.

tions with calculated equations of state to identify phase transitions in the same type of system we were testing.

Our simulations operated at dimensionless pressures of 0.1, 0.01, and 0.003125 ϵ/σ^3. Kataoka and Yamada identified the phase transitions for these pressures at about 1.2, 0.85, and 0.75 ϵ/k_B, respectively. Inital testing was done with all simulations starting from a standard configuration with a cubic box whose side length is $x = 15\sigma$, $N = 864$. (This gives a volume per bead of $3.9\sigma^3$) After the simulation had completed, it was clear that the results did not match well to other research. After consideration, more simulations of the whole space of pressures and temperatures were run with different starting configurations.

Three new configurations were selected, from the final states of previous runs: The smallest was pulled from the final state of the $P = 0.1$, $T = 0.5$ run. This configuration had a box side length of 9.8σ. This same configuration, with the side length overwritten to `Lx` $= 15\sigma$ (so that the beads could freely spread out through the open space at the edges of the box), was used for another simulation. Finally,

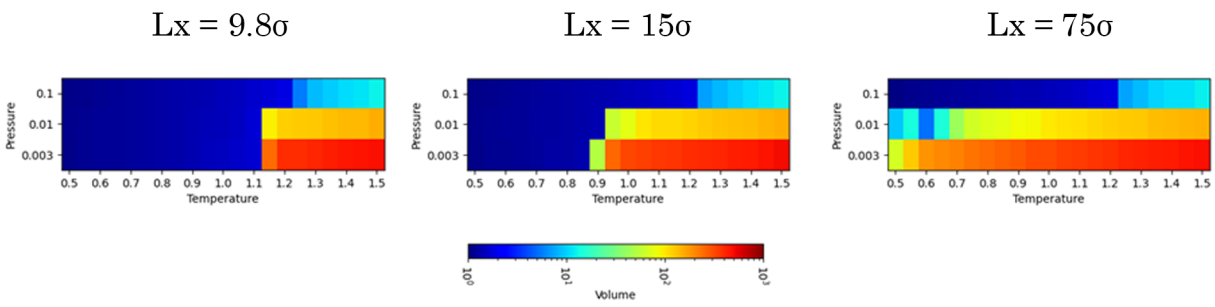

Figure 3.2 The equilibrium phase diagram for simulations run in OPWL.

another configuration was selected from a run with $P = 0.0005$, $T = 1.5$. Its side length was 75σ. The principle in this exercise was to approach the phase transition from either side. The results are shown in fig. 3.2. (Volume is again in volume per bead.)

Some interesting information can be gleaned from these phase diagrams. Firstly, all of the simulations had the same result when the pressure was set to $P = 0.1\ \epsilon/\sigma^3$. This indicates that all of these are firmly solid, or near solid. (The density slowly decreases as $T > 1.2$, matching the expected value.) The simulation started from the densest state (left) is only able to vaporize when the temperature is above $T = 1.1$, well above our expected phase transitions.

Taking a rough average between the medium-size configuration and the large configuration, the phase transition about lines up with expected results. This indicated something important about OPWL that will become an issue later: it does not handle phase transitions with large changes in volume very well.[1]

[1]This is more of an issue specific to a LJ fluid, which becomes gaseous, than with a polymer system undergoing crystallization, the intended use of the simulation.

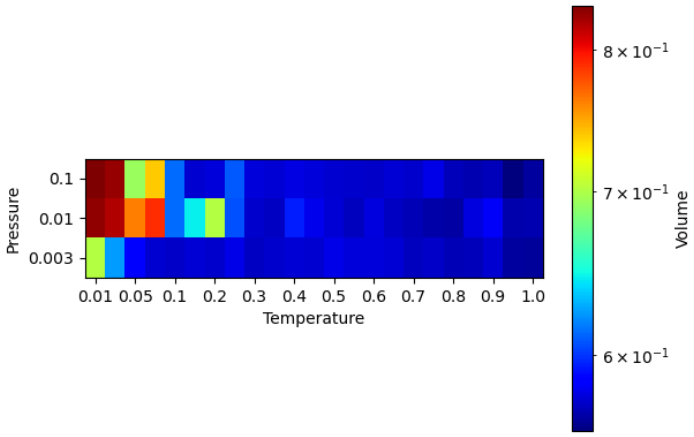

Figure 3.3 Final state for the polymer NPT run.

3.2 Polymer System

Directly translating this type of system for use in polymers generally works, but there are some unique problems. The adjustments to the energy functions had to take into account the difference in energy potentials for bonded and unbonded, as well as the bond angle potential. This leads to a major issue in NPT functionality: the way that the volume move is performed. In the current iteration of the volume move, each coordinate in the positions of all beads in the system are scaled by the expansion factor. This can work for unbonded systems, like the Lennard-Jones fluid. However, the bonded atoms in a polymer generally stay very close to their equilibrium point. Thus separating the molecules in a polymer chain will typically place them in a much higher energy state. This then leads to the volume move being rejected for nearly every attempt. (The acceptance rate for the volume move tended towards $< 0.1\%$.) This leads to the "equilibrium" state at the end of simulation looking very strange. (fig. 3.3)

The scale for this chart is different from the Lennard-Jones fluid, but we can see

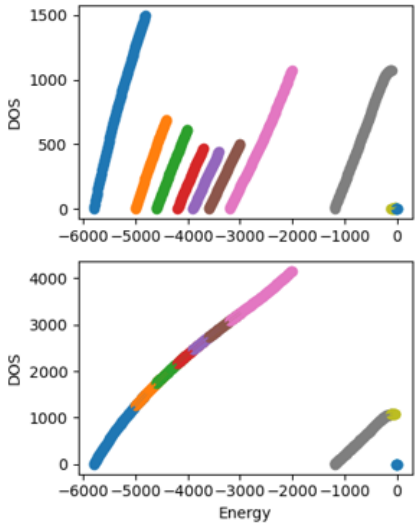

Figure 3.4 Density of States graph for a Lennard-Jones Fluid.

that the low temperatures actually had a *larger* volume than the high temperature systems. This is mostly due to these systems accepting the volume move more often. However, the data is scattered. This is the state of the program as of August 2025. Significant changes will need to be made to the volume move in order for it to sucessfuly model polymers in an NPT system.

3.3 Wang-Landau System

Wang-Landau sampling is an incredibly valuable technique in this type of research. It allows us to have a continuous view of all the energy states a system may be in, and how likely it is to be in those states. It helps us to build energy vs temperature graphs, which then allow us to identify phase transitions. However, its implementation in OPWL has one critical flaw. As noted in sec. 3.1, OPWL struggles to reliably cross large volume changes at a phase transition without over- or under-shooting the temperature significantly. This problem reared its head again when a Lennard-Jones fluid was simulated in a full Wang-Landau system.

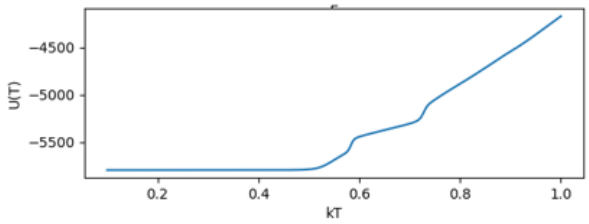

Figure 3.5 Energy vs. Temperature for a Lennard-Jones fluid simulated (unsuccessfully) in OPWL.

The density of states for the Lennard-Jones system simulated in OPWL is shown in fig. 3.4. The top graph contains the density of states for each energy window (see sec. 2.3), while the bottom "stitches" them together to form a continuous line. In this type of graph, an area with relatively low density of states would indicate a phase transition, as it is unlikely for a system to remain in energy states between phases. (The heat of fusion/vaporization will create a "gap" in the energies of steady states.) However, an issue arose in this Wang-Landau simulation (and all others attempted with a LJ fluid). Near the energy of the expected phase transition, the energy window never reached the desired conditions for completion. This occured due to the system not reaching some of the energies desired, causing the final data to never be generated for those windows. This creates a gap in the densities of states, causing the stitching to fail.

Were this data collection successful, the data shown in fig. 3.5 would appear as a simple slope, with a sudden rise at the phase transition, with a second slope continuing after. We do know, however, that outside of these errors in the Wang-Landau portion, OPWL is able to accurately simulate a Lennard-Jones fluid. Fig. 3.6 shows the hexagonal crystallinity of the system across the various energies. We know that a LJ fluid will organize into this type of structure, and it shows a drop-off in crystallinity, indicating melting, at similar energies to those shown in published data. (There is some disruption in the data around the areas noted before, -2000 ϵ.) To

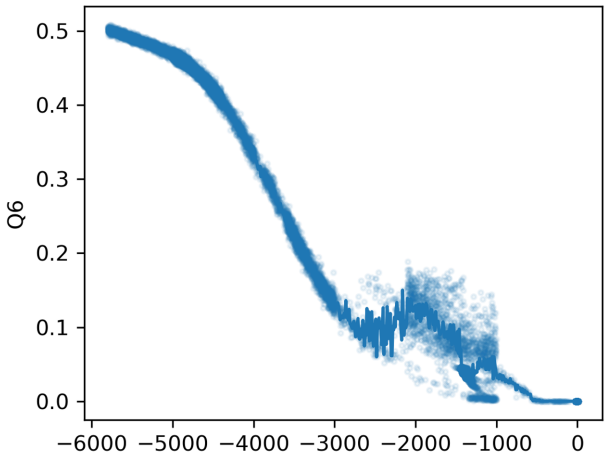

Figure 3.6 Crystallinity of a Lennard-Jones fluid across the energy spectrum.

get a more successful rendering, some change would need to be made to either the way that volume moves are performed, or how OPWL behaves around large energy changes. In its current state, even a simple Lennard-Jones fluid cannot be completely characterized.

Chapter 4

Conclusion and Future Work

Order Parameter Wang-Landau (OPWL) simulation is a powerful tool for modeling molecular systems, but there are some critical issues with its use in constant-pressure systems. There are a few things that can be done to alleviate these issues:

1. Change the way that the volume move works.

2. Make optimizations for the volume move that are polymer-specific.

3. Find solutions for issues in large-volume-change phase transitions.

The current implementation of the volume move is very simple, but causes problems for any type of structured matter. This is less of an issue in a Lennard-Jones (LJ) fluid than for polymers, but can still cause issues in a crystalline phase. For the LJ fluid, simply expanding the bounds of the box without adjusting the positions of the beads could provide a simple solution.

This method would not work so well with a polymer system — a chain that crosses over the boundary and is "scrolled" to the other side would suddenly be snapped (or rather have a massive spike in potential energy) if the spacing between the beads on the chain is suddenly expanded. It would be much more complicated, but a possible

solution would be to identify chains that cross boundaries, and move those with the boundary (keeping the inter-bead spacing the same), while leaving the other chains in their positions. This would appear as those chains that pass through the boundaries being "stuck" in the boundary as it expands, and everything else remaining in place.

The large-volume phase changes present a unique challenge. While it does not cause any catastrophic failures in non-Wang-Landau simulations, it still creates false phase transition points, as noted in sec. 3.1. One possible solution, which may be difficult to control, would be to wildly increase the volume change (of approximately a third of the box length) when volume changes have mostly stabilized. This would check if our system is at a local minimum before the equilibrium point is established.

There is still much work to do, both in the field of polymer crystallization as a whole, but especially with OPWL as a research tool. This research may help build a foundation for the uncovering of the mechanism of polymer crystallization.

Bibliography

[1] Fugao Wang, D.P. Landau, "Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States", Phys. Rev. Lett., American Physical Society, 2001 https://link.aps.org/doi/10.1103/PhysRevLett.86.2050

[2] Pierre Kawak, Dakota S. Banks, Douglas R. Tree; Semiflexible oligomers crystallize via a cooperative phase transition. J. Chem. Phys. 7 December 2021; 155 (21): 214902. https://doi.org/10.1063/5.0067788

[3] Hans C. Andersen; Molecular dynamics simulations at constant pressure and/or temperature. J. Chem. Phys. 15 February 1980; 72 (4): 2384–2393. https://doi.org/10.1063/1.439486

[4] Yosuke Kataoka, Yuri Yamada, "Phase Diagram for a Lennard-Jones System Obtained through Constant-Pressure Molecular Dynamics Simulations", Journal of Computer Chemistry, Japan, 2014 https://www.jstage.jst.go.jp/article/jccj/13/5/13_2014-0016/_article/-char/ja/ (Accessed September 24, 2025)

[5] Pierre Kawak, Christopher Akiki, Douglas R. Tree "Effect of local chain stiffness on oligomer crystallization from a melt", Phys. Rev. Mater., American Physical Society, 2024 https://journals.aps.org/prmaterials/abstract/10.1103/PhysRevMaterials.8.075606 (Accessed September 24, 2025)

[6] Fernández-Pendás, M., Escribano, B., Radivojević, T. et al. Constant pressure hybrid Monte Carlo simulations in GROMACS. J Mol Model 20, 2487 (2014). https://doi.org/10.1007/s00894-014-2487-y