Polymer Chain Monte Carlo

A. Lovell

March 31, 2015

Abstract Write the abstract here - give a brief overview about what you have done and what you're planning on showing.

1 Introduction

Polymer are chains atoms or molecules of the same type that interact directly between sequential atoms through a spring force and through other pairs via some other long- or short-range potential (given, for example, by hydrogen bonds, van der Waals interactions, or the Lennard-Jones interaction). The typical number of atoms in a polymer is between 10^3 and 10^5.

One would often think that a molecular dynamics calculation would be used to simulate a polymer chain. In this case, the movement of the chain would be calculated at each time step, the forces between each molecule calculated, and then used to update the chain's next position. However, because of the different types of motion involved in the problem - ranging from fast motion of the individual units to slow motion of the chain as a whole - this process can be inefficient, or even impossible, for long chains. [2] Instead, Monte Carlo simulations are used to discover properties of the polymer.

This report is broken up into the following sections. In Section 2, we will discuss the theory behind the problem, including the interaction between atoms and the algorithm that was used to run the Monte Carlo simulation. Section 3 gives a brief discussion of the initial conditions of the simulation and the different grids that were explored. The results of the simulation are discussed in Section 4. Finally, in Section 5, we provide a summary of our work along with progress that could be make to this project in the future. The appendix also provides some detail about our error calculations.

2 Theory

In the following section, we will describe some of the concepts that will be needed to simulate a polymer chain.

2.1 Interaction

Polymer chains can be modeled in several ways. One of the most common ways is describing the interaction between sequential pairs of atoms as a stiff spring and putting in a Lennard-Jones potential between every other pair of atoms. **Reference this.** However, in this project we will take a somewhat simplified approach. Instead of stiff springs, we will fix the length between each successive pair of atoms, while keeping the interaction between all other pairs a Lennard-Jones interaction. There are, of course, more complicated that could be implemented to fold the polymer into more complex shapes (such as proteins or even origami ducks), but those will not be explored here.

The Lennard-Jones interaction is

$$V_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r})\right)^{6} \right] \qquad (1)$$

where ϵ is the depth of the interaction, σ describes the particle size, and r is the distance between the interacting pairs. When the two atoms are far enough apart, they can be viewed as non-interacting. For this reason, we can introduced a cut-off radius, r_c, beyond which the potential between two particles is taken to be zero. Thus, we use the interaction

$$V_{LJ}(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] & r \leq r_c \\ 0 & r > r_c \end{cases} \qquad (2)$$

This form takes into account the short-range repulsive force between the two atoms due to Coulomb

repulsion and the longer-range attractive forces from dipole-dipole and dipole-induced dipole forces. An example of the Lennard-Jones potential can be found in Figure 1.
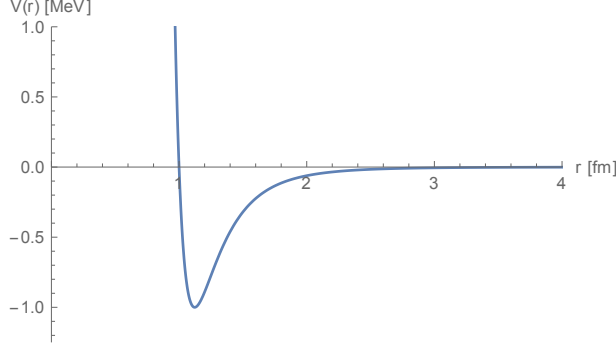

Figure 1: The shape of the Lennard-Jones potential. Here, $\sigma = 1$ fm (the distance where the potential crosses the r-axis) and $\epsilon = 1$ MeV (the maximum depth of the interaction).

2.2 Algorithm

The algorithm that we use to pick the next point for this Monte Carlo simulation is the Rosenbluth algorithm proposed in 1956 by Marshall and Arianna Rosenbluth. As compared to simple sampling - which generates all configurations with equal probability - the Rosenbluth algorithm generates configurations with different probabilities. These probabilities cause the chain to be self-avoiding and make it so that it terminates when it is trapped in a dead end. [3]

On a square (or cubic) lattice, this means that each nearest neighbor is tested as to whether or not it is already occupied, and the the probability that it moves to that cite is based on the energy of the potential new configuration using the statistical factor

$$P_i = \frac{1}{Z} e^{-V_i/T} \qquad (3)$$

where

$$Z = \sum_{i=1}^{N_a} e^{-V_i/T} \qquad (4)$$

Here, V_i is the potential energy between the ith potential configuration and the rest of the chain, N_a is the number of unoccupied nearest neighbors, and T is the temperature of the system. These probabilities are mapped onto the range $[0, 1]$, and a random number is picked within that same range. If this random number falls within the range P_i, then the next atom is added at the position from which V_i was calculated.

However, we can also grow our polymer off of a grid. In this case we have to sample from a circle in two-dimensions or a sphere in three-dimensions. Do do this we sample from $\cos\theta \in [-1, 1]$ and $\phi \in [0, 2\pi]$ (taking $\theta = \pi/2$ for the two-dimensional case, to fix the third dimension). This samples over the circumference of a unit circle in two-dimensions and the surface of a sphere in three-dimensions. Some number, N_a, of the points are randomly chosen, and new positions are calculated based on

$$\begin{aligned}
x_i^j &= x^{j-1} + \sin\theta_i\cos\phi_i \\
y_i^j &= y^{j-1} + \sin\theta_i\sin\phi_i \qquad (5) \\
z_i^j &= z^{j-1} + \cos\theta_i
\end{aligned}$$

The probabilities for each $\mathbf{r}_i$ to be chosen is the same as described above with equations (3) and (4).

2.3 Configuration

The main quantity that we want to calculate is the relationship between the size of the system and the number of atoms in the polymer, which should be related to the dimensionality of the simulation (e.g. two-, three-, or more dimensional space). To do this, we relate the gyromagnetic radius, R_g, to the number of atoms in the chain, N, where

$$R_g^2 = \frac{1}{N} \sum_{i=1}^{N} \langle (r_i - R_{cm})^2 \rangle \qquad (6)$$

and

$$R \sim N^\nu \qquad (7)$$

The temperature, T_Θ, defines the transition between a chain-like polymer ($T < T_\Theta$) and a dense, ball-like polymer ($T > T_\Theta$). Below T_Θ, $\nu = 1/2$, at T_Θ, $\nu = 1/3$, and above T_Θ, ν scales with the dimensionality of the simulation - namely, $\nu = 3/(d + 2)$, where d is the dimensionality of the simulation.

CITE THIS!!!!!!

This is the main observable we will be calculating for this Monte Carlo simulation. From this, we can also define the Θ-temperature, T_Θ.

3 Initial Conditions

The first two atoms of the chain were positioned at (0,0,0) and (1,0,0). Essentially, these positions are arbitrary (only constrained to be 1 unit length apart), but it gives a good starting point for both the polymer on a grid and in free space.

In this calculation, we also chose $\epsilon = 1$ and $\sigma = 1$ in (1). In order to compare to data from the lab, we would just need to change our potentials to include realistic numbers for these two values. Along with this, we also are using units where $k_B = 1$. This gives us temperature in eV, with 1 eV $\approx$ 11600 K. Thus, even seemingly small temperatures become large quickly.

4 Discussion

In this section, we present the main result of our calculation. We were able to vary the temperature of the polymer, from 1.0 eV to 5.0 eV, as well as the number of atoms, $N = 5000, 10000, 15000$, to calculate the radius of gyration and see how well our simulation compares to (7). These results are summarized in Table 1.
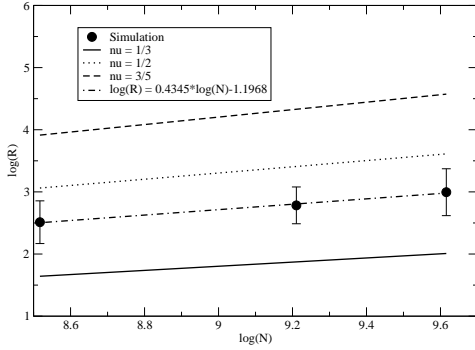
Figure 2: Results for $T = 1.0$ eV. Black dots give the Monte Carlo values calculated for this project, and the dotted-dashed line gives the best fit line through the simulation points. The solid, dotted, and dashed lines show the values of ν from (7). Each of these three lines has the same y-intercept as the best fit line; thus, if one of these lines was the correct description of the simulation, it would go through the black dots and be aligned with the best fit line.

In Figures 2, 3, and 4, we show the results of the simulation plotted along with the best fit line for each set of data. These fits are given in Table 2.
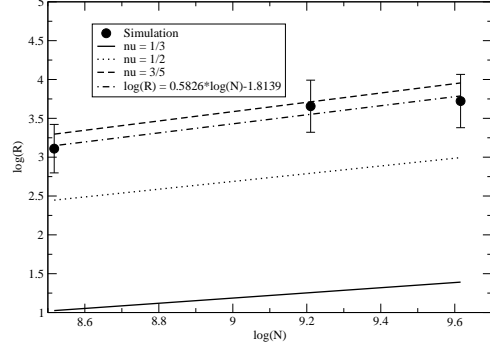
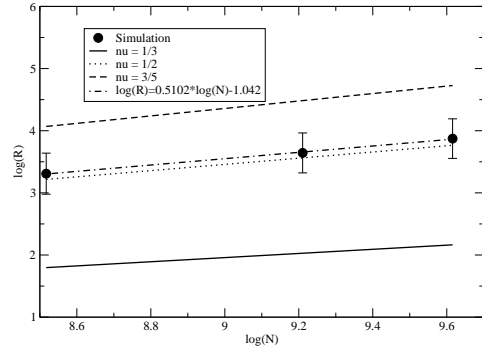Figure 3: Same as Figure 2 but for $T = 2.0$ eV. We can see that $\nu = 3/5$ best describes the simulation.

Figure 4: Same as Figure 2 but for $T = 5.0$ eV. We can see that $\nu = 1/2$ best describes the simulation. This unexpected result is discussed more within this section.

From these fits, we find that $\nu = 0.4345$ for $T = 1.0$ eV, $\nu = 0.5826$ for $T = 2.0$ eV, and $\nu = 0.5102$ for $T = 5.0$ eV. The first two cases match up well with a temperature just below T_Θ for $T = 1.0$ eV and for a temperature above T_Θ that follows the pattern $\nu = 3/(d+2)$ for $T = 2.0$ eV. (Recall, for this simulation $d = 3$ so $\nu = 0.6$.) However, we see that for $T = 5.0$ eV, $\nu \approx 0.5$ which would seem to indicate that this is the Θ-temperature. This cannot be the case. To explain this, we note that 5.0 eV $\approx$ 58000 K, which is an order of magnitude hotter than the

	$T = 1.0$ eV		$T = 2.0$ eV		$T = 5.0$ eV	
	R_g	σ_{R_g}	R_g	σ_{R_g}	R_g	σ_{R_g}
$N = 5000$	2.5119	0.343484	3.109664	0.312228	3.308621	0.330394
$N = 10000$	2.783211	0.295998	3.656107	0.335429	3.642807	0.321192
$N = 15000$	2.994968	0.376915	3.722428	0.343053	3.872835	0.318437

Table 1: Results for a three-dimension polymer chain in free space (not on a grid) for three different temperatures at three different polymer lengths. Both the mean value and the standard deviation for each set of points is given.

T (eV)	**m (fm)**	**b (fm)**	χ^2
1.0	0.4345	-1.1968	0.007327
2.0	0.5826	-1.8139	0.1483
5.0	0.5102	-1.042	0.003000

Table 2: Slopes (m) and y-intercepts (b) for the best fit lines for the simulation data in Table 1. Best fit lines are given as $log(R_g) = m * log(N) + b$. Here log indicates the natural log. In this case, the slope m gives the value for ν from (7).

coolest part of the sun. [1] We assume that at these temperatures, our simple model is no longer valid, and we get nonsensical results from our calculations.

From Table 2, we can also see that the χ^2 values for each of these fits are extremely low. This is a product of the fact that only three data points are being fit with a linear regression. The calculated value for ν also does not take into account the error from the original simulation. **Unless I can ask someone about this and implement it before the end of the day.**

Still, in the region where our simulation is valid, we have calculated reasonable results for our three-dimensional simulation of a polymer chain.

5 Conclusion

Talk briefly about what we calculated. **Varying size, varying dimension, varying temperatures - find the critical temperature Theta.**

There is always future work that could be done regarding this project. While we calculated a polymer chain in three-dimensions, it would also be interesting to calculate higher dimensional chains and examine the scaling of the size of the chain with the number of atoms added to the chain. The same results as **(put reference to either equations or figures)** should hold, but it would be an interesting project nonetheless to generalize the code. It would also be interesting to examine the difference between this free self-avoiding walk calculated here and a self-avoiding walk confined to a grid. In this way, we could see how the size of the polymer is related to the number of atoms and the dimensionality of the problem.

There are also several thermodynamical quantities that could be calculated from this system, including specific heat, thermal energy, and **many more**. Although we used a fixed bond length between each successive atom and a Lennard Jones interaction between all pairs of atoms, there are many other interactions that could describe a more realistic polymer. This includes using a stiff spring to model the interaction between successive atoms. We also could have changed the strength of the interaction between various pairs to see what kinds of shapes we could form with our polymer chain. Still, we successfully modeled a polymer chain of **N** atoms in **two-** and three-dimensions.

Mention PERM, more accurate way of calculating these things.

A Error Analysis

Just as experimentally measured data points should quote errors - whether due to systematics, timing, or unknown quantities - theoretical simulations and calculations should also quote error bars. To do this, we run our simulation several times to find a mean value and a standard deviation. The mean is calculated by

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{8}$$

where $\bar{x}$ is the mean value, x_i are the values of a given quantity from each calculation (for instance the length of the chain), and N is the number of simulations run.

To find the error on each data point, we average the standard deviation for each run.

$$\sigma_x^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{9}$$

Here, σ_x is the error for the variable x from each calculation. We can see that the errors decrease as $1/\sqrt{N}$.

Because we are calculating a fit to data, we should also report a χ^2 value for that fit, to assess its "goodness" - or the quality of the fit when compared to data. [4] In this case, we would calculate the χ^2 of each of the linear fits that are made to the simulation "data".

$$\chi^2 = \sum_{i=1}^{N} \frac{(x^{sim} - x^{fit})^2}{\sigma_x^2} \tag{10}$$

In this case, x^{sim} is the parameter value from the simulation, x^{fit} is the parameter value from the fit through the simulation data, and σ_x^2 is the square of the standard deviation (the variance) from (9).

Although this does not give us an error on the fits that are being made, it does give some indication as to the quality of the fit.

References

[1]

[2] Hendrik Meyer Jörg Baschnagel, Joachim P. Wittmer. Monte carlo simulation of polymers: Coarse-grained models. *John von Neumann Institute for Computing*, 23:83–140, 2004.

[3] Thomas Prellberg. From rosenbluth sampling to perm - rare event sampling with stochastic growth algorithms. *Lecture given at the International Summer School, Modern Computational Science 2012 - Optimization*, 2012.

[4] Ian J. Thompson and Filomena M. Nunes. *Nuclear Reactions for Astrophysics*. Cambridge University Press, 2009.