Polymer Chain Monte Carlo

A. Lovell

March 31, 2015

Abstract Polymer chains are of great interest in biophysics, and while one might think to use a molecular dynamics calculation to simulate their configurations, for practical reasons, Monte Carlo techniques are more widely used. In this project, we will use the Rosenbluth algorithm within a Monte Carlo framework to simulate polymer chains of varying lengths around 10^4 atoms at several temperatures to investigate the correlation between the size of the polymer structure and the number of atoms within the chain. This will be compared to theoretical models to investigate the systems we can create using a fixed distance between sequential atoms and the Lennard-Jones potential between all other pairs.

1 Introduction

Polymers are chains of atoms or molecules of the same type that interact directly between sequential atoms through a spring force and through all 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 or molecules in a polymer is between 10^3 and 10^5 which can all be of the same type or different types.

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 could be followed by calculating the forces between each pair of atoms for each time step 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 atomic 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 a note on the units used in the calculation. The results of the simulation are discussed in Section 4. Finally, in Section 5, we provide a summary of our work, along with further progress that could be made on this project in the future. The appendix then 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 as well as briefly explain the path of the simulation code.

2.1 Interaction

The interactions between the atoms of polymer chains can be modeled in several ways. One of the most common ways is to describe the interaction between sequential pairs of atoms as a stiff spring and put in a Lennard-Jones potential between every other pair of atoms. [2] 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 (infinitely stiff springs), while keeping the interaction between all other pairs a Lennard-Jones interaction. There are, of course, more complicated potentials that could be implemented to fold the polymer into more complex shapes (such as proteins or even origami ducks), but these will not be explored here.

The Lennard-Jones potential is

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

where ϵ is the depth of the interaction, σ is related to 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 would be able to introduce a cut-off radius beyond which the potential between two particles is taken to be zero.

The form of (1) 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. [6] An example of the Lennard-Jones potential can be found in Figure 1.

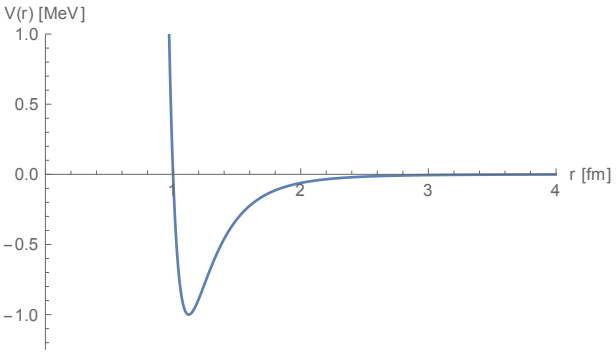

Figure 1: The shape of the Lennard-Jones potential. Here, $\sigma = 1$ fm (the distance where the potential crosses the r-axis) and $\epsilon = 1$ MeV (the maximum depth of the interaction).

2.2 Algorithm

In this Monte Carlo simulation, we build up the polymer chain by adding one atom at a time. The algorithm that we use to pick the next atom's position 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 to terminate when it is trapped in a configuration where every next possible addition would be on top of an already-placed atom. [4] (This is a more straightforward concept for polymers created on a grid, which have a fixed number of nearest neighbors.)

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 the next atom is placed at that site is based on the energy of the potential new configuration using the statistical factor

$$P_i = \frac{1}{Z} e^{-V_i/T} \tag{2}$$

where

$$Z = \sum_{i=1}^{N_a} e^{-V_i/T} \tag{3}$$

Here, V_i is the potential energy between the i^{th} candidate 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 freely, without a grid. In this case we have to sample from a circle in two-dimensions or a sphere in three-dimensions with the radius of the distance between each successive pair of atoms. To 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 points are randomly chosen, and new positions are calculated from

$$\begin{aligned} x_i^j &= x^{j-1} + \sin\theta_i \cos\phi_i \\ y_i^j &= y^{j-1} + \sin\theta_i \sin\phi_i \\ z_i^j &= z^{j-1} + \cos\theta_i \end{aligned} \tag{4}$$

where we are placing the jth atom in the ith candidate configuration. The probabilities for each $\mathbf{r}_i$ to be chosen is the same as described above with equations (2) and (3).

2.3 Configuration

The main quantity that we want to explore 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 gyration radius, R_g, to the number of atoms in the chain, N, where

$$R_g^2 = \frac{1}{N} \sum_{i=1}^{N} \langle (r_i - R_{cm})^2 \rangle \tag{5}$$

and

$$R \sim N^{\nu} \tag{6}$$

The temperature, T_Θ, defines the transition between a chain-like polymer ($T < T_\Theta$) and a dense, ball-like polymer ($T > T_\Theta$). For temperatures 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. [2]

R_g is the main observable we will be calculating for this Monte Carlo simulation. From this, we can also define - or at least put some constraint on - the Θ-temperature, T_Θ.

2.4 Code

In this subsection, we will briefly describe the structure of the code and where the above ideas are implemented.

Initially, two atoms are placed at a distance of one unit length apart from each other. The potential energy of the system is calculated. To prepare to place the next atom, some number of points on the unit sphere are randomly chosen from a distribution of $cos\theta d\theta d\phi$. The potential energy of each of these configurations is calculated, and the Rosenbluth algorithm, described in 2.2, is implemented to choose the position of the next atom. Once the next position has been chosen, the potential energy of the new system is calculated and the method repeats: choose points on the unit sphere, implement the Rosenbluth algorithm, calculate the potential energy of the system.

Once the entire chain (a given number of atoms, N) is constructed, the radius of gyration, from (5), is calculated for this polymer. This chain-building process is repeated again and again to build up enough statistics to calculate a mean and and standard deviation (for the error).

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. Because this calculation is not performed on a grid, for each new position, 90 points on the unit sphere are tested. This more easily prevents the chain from becoming stuck by only having options to place the next atom too close to previously placed atoms.

In this calculation, we also chose $\epsilon = 1$ and $\sigma = 1$ for the Lennard-Jones potential parameters 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 quickly become large - and potentially unphysical.

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 (6) for known values of ν. These results are summarized in Table 1 and the method of error, σ_{R_g}, calculation is discussed in the appendix.

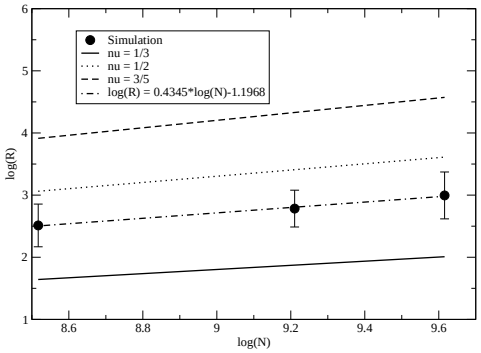

Figure 2: Results for $T = 1.0$ eV. Black dots give the Monte Carlo values calculated for this project, and the dotted-dashed line gives the best fit line through the simulation points. The solid, dotted, and dashed lines show possible, interesting values of ν from (6). Each of these three lines has 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

	$T = 1.0$ eV		$T = 2.0$ eV		$T = 5.0$ eV	
	$log(R_g)$	σ_{R_g}	$log(R_g)$	σ_{R_g}	$log(R_g)$	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 ($log(R_g)$) and the standard deviation (σ_{R_g}) for each set of points is given. Here, and throughout the report "log" means log base e - the natural log.

the simulation plotted along with the best fit line for each set of data. These fits are given in Table 2.

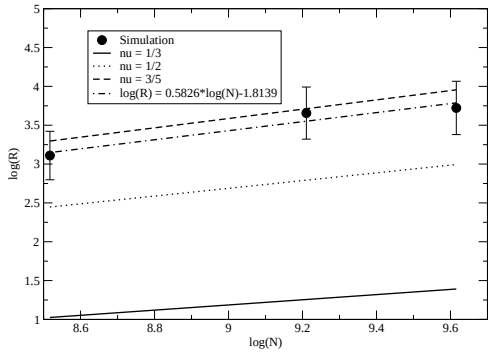

Figure 3: Same as Figure 2 but for $T = 2.0$ eV. We can see that $\nu = 3/5$ best describes the simulation.

From these fits, we find that $\nu = 0.4345 \pm 0.03440$ for $T = 1.0$ eV, $\nu = 0.5826 \pm 0.1642$ for $T = 2.0$ eV, and $\nu = 0.5102 \pm 0.02239$ for $T = 5.0$ eV.[1] 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 postulate an explanation for this, we note that 5.0 eV $\approx$ 58000 K, which is an order of magnitude hotter than the coolest part of the sun. [3] 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 value for each of these fits is extremely low. This is a product of the fact that only three data points are being fit with by linear regression, causing a misleadingly small χ^2 value. Ideally, we want a χ^2 value of 1 for every degree of freedom in the problem (e.g. $\chi^2/dof = 1$) which would indicate a perfect fit. A very low χ^2, such as we have here, indicates that we have under-constrained our model by allowing too many free parameters for the number of data points. Thus, having run the simulation for more N and T values would serve to better constrain our fit, as well as give us more confidence that the parameters we calculated adequately described the results of our simulation.

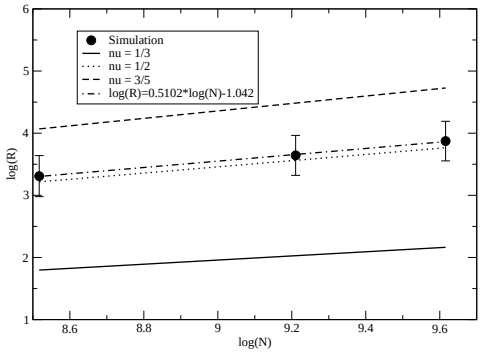

Figure 4: Same as Figure 2 but for $T = 5.0$ eV. We can see that $\nu = 1/2$ best describes the simulation. This unexpected result is discussed more within the text.

We can see from Figures 2, 3, and 4 that the error bars are significant for the range of data displayed. These large errors did not decrease with the number of statistics, a trend which can be seen in Table 3. From these figures, we can see that these large errors contribute to some uncertainty around the slope of

[1] These errors are discussed in more detail in the Appendix.

T (eV)	m	b	s_m	χ^2
1.0	0.4345	-1.1968	0.03440	0.007327
2.0	0.5826	-1.8139	0.1642	0.1483
5.0	0.5102	-1.042	0.02239	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 (6). s_m gives the error on the fit - only describing the difference between the values calculated from the fit and the simulation values, and χ^2 takes into account, somewhat, the error from the simulation.

the best fit lines, as the calculated value for ν does not take into account the error from the original simulation. This uncertainty makes it unclear as to whether or not we can confidently describe the behavior of the chain. Even though the best fit lines have a very low χ^2 value, this value is misleading, as described previously.

Still, in the region where our simulation is valid, we have calculated reasonable results for our three-dimensional simulation of a polymer chain, even if the error analysis leaves somewhat of a question as to the behavior of the chain at our various temperatures of interest.

5 Conclusion

In this work, we were able to construct a Monte Carlo simulation of a three-dimensional polymer chain of approximately 10^4 atoms at temperatures of 1.0 eV, 2.0 eV, and 5.0 eV. The polymer was simulated without being confined to a grid. By comparing the natural logarithm of the gyration radius and number of atoms in the chain, we were able to discover properties of the chain itself and find the exponent, ν. We found that $\nu = 0.4345 \pm 0.03440$ for $T = 1.0$ eV, $\nu = 0.5826 \pm 0.1642$ for $T = 2.0$ eV, and $\nu = 0.5102 \pm 0.02239$ for $T = 5.0$ eV. Through this, we were able to compare to known behavior above and below the Θ-temperature.

However, there is still future work that can be accomplished. While we calculated a polymer chain in three-dimensions, it would also be interesting to calculate higher (or lower) dimensional chains and examine the scaling of the size of the chain with the number of atoms added to the chain, ν. The same scaling ($\nu = 1/3$ below T_Θ, $\nu = 1/2$ at T_Θ, and $\nu = 3/(d+2)$ above T_Θ) should hold, but it would be an interesting project nonetheless to generalize the code. It would also be interesting to examine the differences between this type of self-avoiding walk that is not confined to a grid and a self-avoid walk confined to a cubic grid (or other shape grid). From this, we could see how the dimensionality affects various grids on which the chain is created. For all of this, we could also run the calculation in smaller temperature steps to find the Θ-temperature.

There are also several thermodynamical quantities that could be calculated from this system, including specific heat and thermal energy. We could also change the interaction between the atoms within each chain. Although we used a fixed bond length between each successive atom and a Lennard-Jones interaction between all pairs of atoms, there are other interactions that could describe a more realistic polymer, including using a stiff spring to model the interaction between successive atoms instead of a fixed bond length. We also could have changed the strength (and type) of the interaction between various pairs to see what kinds of shapes we could form with our polymer chain.

There have also been developments within the last two decades of the Pruned-enriched Rosenbluth method which combines the Rosenbluth method with recursive enrichment. This algorithm allows for simulations of up to $N = 10^6$ atoms in a single chain with high statistics, applicable to calculations both on and off of a lattice. [1] In future work, this could be an interesting algorithm to implement, especially if it is more efficient and can give higher statistics in a shorter amount of time than the original Rosenbluth algorithm.

Still, without all of this, we were able to model a polymer chain of 10^4 atoms in three-dimensions without a grid.

	$T = 1.0$ eV	$T = 2.0$ eV	$T = 5.0$ eV
$N = 5000$	64 (0.343484)	500 (0.312228)	1000 (0.330394)
$N = 10000$	43 (0.295998)	500 (0.335429)	810 (0.321192)
$N = 15000$	45 (0.376915)	290 (0.343053)	1147 (0.318437)

Table 3: Number of chains produced for each combination of number of atoms in the chain (first column) and temperature (top row). The number in parentheses gives the error calculated for each of these points, also given in Table 1. It is clear to see that the errors do not decrease as the number of calculations increases, even though, in some cases, the number of simulations increases by orders of magnitude.

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 give 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

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{7}$$

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 that were performed.

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

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{8}$$

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. [5] In this case, we would calculate the χ^2 of each of the linear fits that are made to the simulation data.

$$\chi^2 = \sum_{i=1}^{N} \frac{(x^{sim} - x^{fit})^2}{\sigma_x^2} \tag{9}$$

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 (8).

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.

Using similar ideas, errors for the fits can be calculated (although these errors still do not take into account the error from the simulations). For these errors (such as those in Table 1), we take

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i^{data} - y_i^{fit})^2 \tag{10}$$

where s is the error on the fit, n is the number of data points fit, y^{data} is the result of the simulation, and y^{fit} is the result from the fit (both calculated at the same point). [7]

References

[1] Peter Grassberger. Pruned-enriched rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000. *Phys. Rev. E*, 56:3682–3693, Sep 1997.

[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] University of St. Andrews. The photosphere. *Lecture notes at University of St. Andrews*, 1995.

[4] Marshall N. Rosenbluth and Arianna W. Rosenbluth. Monte carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, 23:356, 1955.

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

[6] Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, Jul 1967.

[7] S.A. Teukolsky W.H. Press, B.P. Flannery and W.T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, 2 edition, 1992.