Monte Carlo simulation of polymers

G.I.L. van Oudenaren (4344650) and H.C.E. van den Brink (4367820)

(Dated: 15 May 2019)

In this report the Rosenbluth algorithm was used to generate polymers. For a given temperature up to a thousand polymers are generated and some statistical entities were deduced. Using this a temperature dependence was found in the mean polymer length. Also the mean square distance was plotted and fitted using the theoretical approximation. The results we found showed similarities with literature which indicates that the simulation can be considered succesfull.

I. INTRODUCTION

The Monte Carlo method is a stochastic simulation method that uses a random sequence of numbers to construct a sample of the population. From this statistical estimates of the parameters can be obtained like the expectation value and/or variance. Using Monte Carlo methods is convenient to simulate polymers due to their random nature.

In the simulation polymers are represented as chains of beads, where the distance between consecutive beads is kept constant. To make this model as realistic as possible we introduce a force that simulates the interaction between the beads. Using the Monte Carlo method on this model we find that the polymer's most likely configuration is the one where the total internal energy of the polymer is the lowest.

II. THEORY

A. Internal energy

If we consider a polymer containing N consecutive beads where $|\mathbf{x}_n - \mathbf{x}_{n+1}| = \sigma$ which holds $\forall n \in \{1, 2, ..N-1\}$. The interactions between the beads can be described by the Lennard-Jones potential. This potential contains an attractive and a repulsive component and can be described by the following equation:

$$U_{LJ} = 4\epsilon\Big(\big(\frac{\sigma}{r}\big)^{12} - \big(\frac{\sigma}{r}\big)^{6}\Big) \tag{1}$$

With inter-atomic distance r and fitting parameters σ for length and ϵ for energy. These were set constant at: $\sigma = 0.8$ Å and $\epsilon/k_B = 0.25$ K. The total potential energy at a bead can be acquired by summing over the potential energy with respect to the position of all precious beads.

The potential energy of the k-th bead at position $\mathbf{x}_k$ is given by:

$$U_{Pot} = 4\epsilon\sigma^6 \sum_{n\neq k} \frac{\sigma^6}{|\mathbf{x}_n - \mathbf{x}_k|^{12}} - \frac{1}{|\mathbf{x}_n - \mathbf{x}_k|^{6}} \tag{2}$$

B. Monte Carlo Method

The Monte Carlo method is used to construct the polymer. To find the location of the next bead we first take an arbitrary angle $\theta_0 \in [0, 2\pi)$. After this we define $k \in \mathbb{N}$ such that we have $\Theta = \{\theta_j \in \mathbb{R} \mid \theta_j = \theta_0 + \frac{2\pi j}{k}, j \in \{1, 2, .., k\}\}$. Now we have k possible angles for the next bead, for each possible location the potential energy $E(\theta)$ is determined using equation 2. Now a random angle is drawn from the set $\Theta = \{\theta_0, \theta_1, .., \theta_k\}$ where θ_j has probability p_j given by:

$$p_j^{(l)} = \frac{w_j^{(l)}}{\sum_i w_i^{(l)}} = \frac{\exp\left[-\beta E(\theta_j)\right]}{\sum_i \exp\left[-\beta E(\theta_i)\right]} \tag{3}$$

Here $\beta = k_B T/\epsilon$. After the beads are added it goes through this process until it has no possible angles left, in other words, all the possible locations intersect a line of previous points.

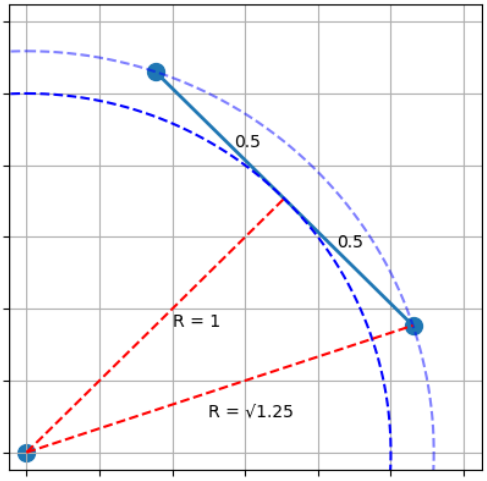

FIG. 1: Geometry of line intersection.

To check for intersection, we have to investigate all points within a radius $\sqrt{1.25}$. The reason for this radius is shown in Figure 1. From a starting position of the blue bead in the bottom left corner, the next bead will be placed somewhere on the circle with radius 1 from the starting position. The edge case of a line between two beads which could intersect the circle of radius 1 (and thus cause the line to the new bead to intersect with it) is the line which is tangent to the circle. From this, we find that the distance to one of the beads which spans the line is $\sqrt{1.25}$. Thus, we have to investigate all points within this radius.

For a starting position $\mathbf{x}_{n-1}$, a potential position $\mathbf{x}_n$ and a neighbouring point $\mathbf{x}_j$ we compare the line segment $\mathbf{x}_{n-1}$ - $\mathbf{x}_n$ with the line segments $\mathbf{x}_{j-1}$ - $\mathbf{x}_j$ and $\mathbf{x}_j$ - $\mathbf{x}_{j+1}$. For each instance we can make 4 combinations of 3 of these points, and calculate the orientation of these 3 points (clockwise, anti-clockwise or co-linear). Comparing these 4 orientations, we can deduce if the line segments intersect or not. If they do intersect, $\mathbf{x}_n$ is removed from the set of potential new positions. The remaining set of points is assigned a weight as described above.

For a complete polymer, the chances of this particular solution is given by the following relation:

$$P \propto \prod_{l=1}^{L} \exp\left[-\beta E(\theta^{(l)})\right] = \exp\left[-\beta \sum_{l=1}^{L} E(\theta^{(l)})\right] \tag{4}$$

C. Mean squared distance distribution

Given that we have generated M polymers where each polymer is a set of coordinates $X_i = \{\mathbf{x_1^{(i)}}, \mathbf{x_2^{(i)}}, .., \mathbf{x_{N_i}^{(i)}}\}$ where N_i is the length of the i-th polymer. The procedure goes as followed, for each $j \in \{1, 2, .., \max\{N_1, N_2, .., N_M\}\}$ we are looking for the polymers that have this element and put this into a set. Then the mean squared distance of these data points is calculated by the following equation:

$$\langle R^2 \rangle_j = \frac{1}{n_j} \sum_{l \in \Omega_j} (\mathbf{x_j^{(l)}} - \mathbf{x_1^{(l)}})^\mathbf{2} \tag{5}$$

Where $\Omega_j \subset M$ is the set that contains the polymers which have an element j.

D. Parameter estimation

Fitting a function $f(N) = a(N-1)^{3/4}$ onto the dataset $\sqrt{\langle R^2 \rangle} = \{y_1, y_2, .., y_N\}$ we find that we have to minimise the following equation with respect to a.

$$J = \sum_{i=1}^{N} \left(y_i - a(N_i - 1)^{\frac{3}{4}}\right)^2 \tag{6}$$

This function can be minimised by taking the first order derivative with respect to a and setting this function equal to zero. This makes us able to find the $a \in \mathbb{R}$ which best fits the data provided.

$$0 = \sum_{i=1}^{N} (y_i - a(N_i - 1)^{\frac{3}{4}})(N_i - 1)^{\frac{3}{4}} \tag{7}$$

This gives the following expression for a:

$$a = \frac{\sum_{i=1}^{N} y_i (N_i - 1)^{\frac{3}{4}}}{\sum_{i=1}^{N} (N_i - 1)^{\frac{6}{4}}} \tag{8}$$

III. RESULTS

To check whether or not the code is sufficient to generate random self-avoiding polymers it is important to check if the lines intersect, since the program should avoid that. Many polymers are checked and we concluded that the algorithm works.

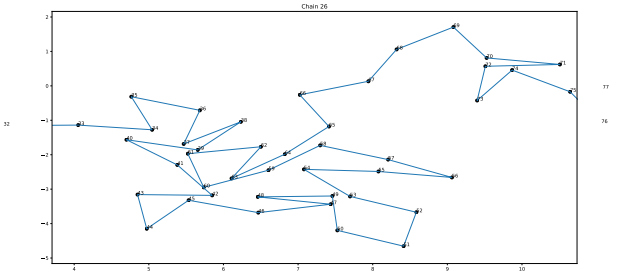

FIG. 2: Example of a chain being able to form complex networks without intersection.

Figure 2 shows an example of a typical chain generated using the algorithms described above. We can observe heavy clustering of beads within a small region without the lines between beads intersecting each other.

Since we are dealing with a stochastic process it is important to remember that more polymers yield a more accurate result. The algorithm is able to generate as many polymers as we want but due to computational limitations we set the limit to thousand. Given a set temperature, thousand polymers are generated and we can plot the lengths of these polymers into a histogram using a function in python. These histograms are then

fitted using the Weibull distribution shown in equation 9.

$$PDF(x; a, c) = ac(1 - e^{-x^c})^{a-1} e^{-x^c} x^{c-1} \tag{9}$$

For different temperatures this is done and the data is shown in Figure 3.

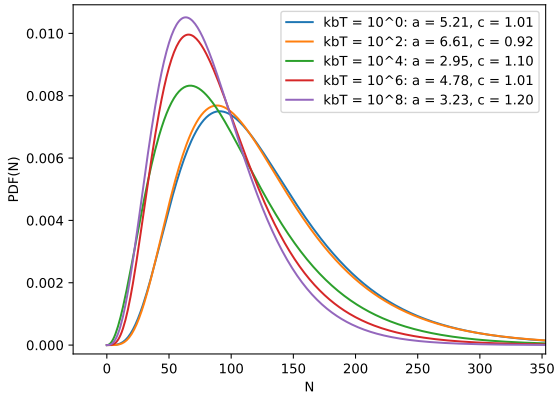

FIG. 3: Distribution of the total amount of beads in a chains for different temperatures.

We can observe a trend in which for higher temperatures the chains tend to be shorter and converge towards the mean amount of beads, while for lower temperatures the distribution is more spread out and allows for the formation of longer chains upwards of 300 beads.

The mean square distance is is plotted for all (sub)-polymers. This gives rise to the curve seen in Figure 4 where the red dotted line is the theoretical line.

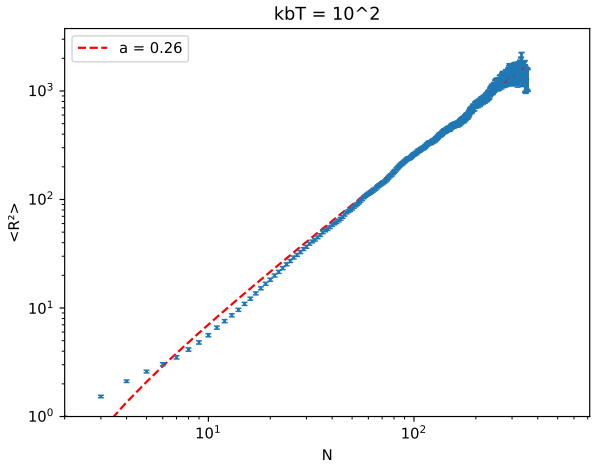

FIG. 4: End to end distance versus number of beads.

IV. DISCUSSION

A. Results

In order to test the validity of the results we have acquired we can compare them to known results in literature. Figure 5 shows results from other simulations.[1]

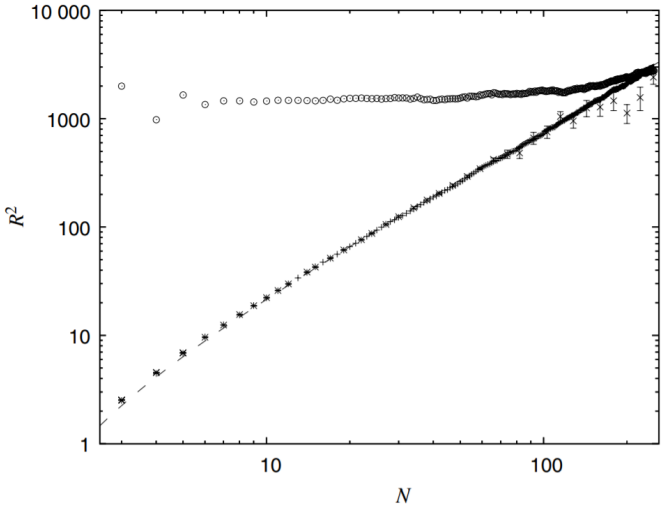

FIG. 5: End to end distance versus number of beads N. Crosses with error bars: Rosenbluth algorithm. Double crosses: PERM algorithm.

From the Weibull distributions in Figure 3 we can see that the median length shifts to the left if the temperature increases. The reason for this has to do with the weights of various positions being linked to their Boltzmann factor. For higher temperatures, the difference between a position with a low or high associated Lennard-Jones potential energy becomes less significant, and thus chains will cluster more, while for lower temperatures they tend to form straighter paths causing lower probability of dead ends, thus generating longer chains.

The mean square end to end distance of the (sub-)chains with a sample size of 1.000 chains is shown in Figure 4. We can observe two things: the fitted function works particularly well from 50 beads on wards and from 200 beads on wards the standard deviation in the mean distance becomes increasing large, which is mainly due to the smaller amount of samples for higher number of beads (see Figure 3).

B. Performance

We aimed to optimise our performance as well as possible by using efficient numerical packages like numpy and scipy. This resulted in relatively fast computational times. For example, it took 59.64 seconds to generate 1.000 chains with a maximal length of 1.000. Based on this information, it is entirely possible to scale up this

simulation to generate millions of polymers within a days time. The simulation were done on a computer with an Intel i7-6500U dual core CPU @ 2.50GHz, meaning much faster results can be obtained using specialised equipment.

V. CONCLUSION

We have successfully created a program that applies the Rosenbluth algorithm to generate sets of polymer chains for a given size, temperature, and maximum amount of beads per chain. Our code is flexible in its parameters, and can accommodate changes in the interactions between beads as well as the amount of allowed positions. The results we have acquired show similarity to those found in literature indicating we have successfully performed our simulation.

VI. COLLABORATION

Our collaboration went smoothly. During this project we aimed at working together at one location more often than the last, as we felt that we could more easily solve problems being in the same room. This also meant that we only committed code from one account which is reflected in the commit history of the repository. We had a meeting at least twice a week where we discussed the progress and future goals.

VII. BILBIOGRAPHY

[1] J. Thijssen, *Computational Physics*, 2nd ed. (Cambridge University Press, 2007).