Computational Physics AP3082 - Project 2 - Monte Carlo simulation of polymers

David Bakker,[1] Klara Chmiel[2] and Nico Kerkhoven[3]

[1] Msc Applied Physics, TU Delft (4675932, D.L.Bakker-1@student.tudelft.nl)
[2] Msc Applied Physics, TU Delft (5344247, K.M.Chmiel@student.tudelft.nl)
[3] Msc Applied Physics, TU Delft (4606841, N.C.Kerkhoven@student.tudelft.nl)

Abstract. In this project we simulate a simplified scenario of polymer folding. The polymers are linear, unbranched, two-dimensional and are placed in a good solvent. Therefore a self-avoiding random walk Monte Carlo simulation can be used to simulate the growth of the polymer and to investigate the dependence of radius of gyration of the polymer on its length. We show that radius of gyration is proportional to the length of the polymer to the power of 3/2, as is theoretically expected. We also show that the Pruned-Enriched Rosenbluth Method (PERM) is more efficient, both in speed and accuracy, than basic Rosenbluth method, when simulating our polymers.

1 Introduction

The goal of this project is to simulate long polymers in a medium and to study how they fold. Polymers are long chain structures that are made up of similar repeating units, such as atoms or groups of molecules, which are called monomers. Simulations of how these polymers are folded are particularly useful in biophysics when studying folding of proteins, which in turn determines and affects their biological functions. Polymer folding properties are also very important in other fields, such as spectroscopy (polymer crystallization kinetics), physical chemistry and material science.

In this project we simulate a simplified folding scenario where linear and unbranched 2D polymers are placed in a good solvent. This results in all polymers being independent of one another, meaning that they don't get in the way of each other. We will use the very popular Monte Carlo method to simulate a large number of polymers at once and statistically analyse the set to extract desired physical properties. In our case we are interested in the radius of gyration and its dependence on the length of the polymer chain. The radius of gyration is the square of the end-to-end distance of the polymer, and is a important quantity in polymer physics.

2 Theory

2.1 Monte Carlo method

The Monte Carlo method is a very popular and extremely powerful tool in statistical physics. Monte Carlo methods rely on randomness. By using random sampling a lot of problems can be solved way quicker and easier than by using the "exact" model. These sort of algorithms are used a lot to evolve statistical ensembles in time. Possible system states are generated randomly based on chosen criteria and one state is then chosen based on probability of its occurrence. In principle these problems can be solved exactly, but can take a lot of computing power. But by the law of large numbers the statistical properties of these randomly generated systems converges quickly to their exact values.

So, once we let a large number of systems evolve and develop using this random sampling, we can extract their statistical properties way faster and the will be very close to the exact value. The expectation values of these physical properties can be determined. We can also predict the errors in our calculations, meaning we can determine how far we are off from the exact values of these expectation values.

2.2 Simulating the polymer structure

To simulate the polymers, we create a lattice composed of equally sized squares of unit length. The atoms or molecule groups are represented as point particles referred to as 'beads' or 'monomers'. In our simulation they are only allowed to be placed on the vertices of the lattice squares. These monomers are connected together to form the whole polymer. We will grow our polymers on this lattice. The polymer is thus described as an ordered set of N points on the lattice. The polymers can not branch of, it is unbranced, meaning that at any monomer there can only be one or two lines connecting to other monomers.

2.3 Polymer growth

We use the Rosenbluth method to build our polymer. The initial bead is placed on the lattice and the polymer grows by adding subsequent beads to it step by step. The addition of the beads is done in a semi-controlled manner:

- The angle between neighboring bonds connecting the beads must always be a multiple of 90°.
- The only allowed distance between neighbouring beads is 1 unit length.
- One lattice vertex can only be occupied by one bead.
- The direction of the next step (up, down, right or left) is decided at random, with equal probabilities of being selected, under the condition that beads are only added to an unoccupied lattice site.

As a result the polymer growth is a self avoiding random walk which can by generated by using the Monte Carlo method. An example of such a polymer can be seen in the Figure below.

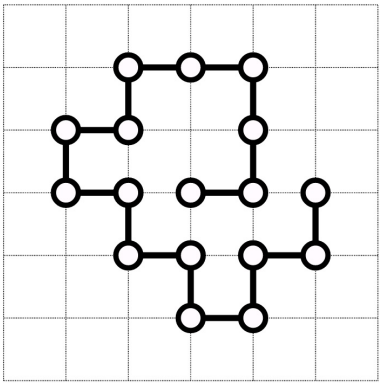

Fig. 1: Schematic of the polymer structure on the square lattice grown by a self-avoiding random walk. Source: Lecture notes[1]

The way we do the simulation in our code is by having a Polymer class, which can grow on its own once you give it a WorldMap. This WorldMap is a map for the square lattice and is used to see in a vertex in the lattice is occupied. Once you have and instance of the Polymer class and you put it in the WorldMap, it grows randomly following the rules listed above until it reaches the maximum length you have given it (or until it gets stuck). Once the polymer has been grown you can extract the radius of gyration from the object. By generating a lot of these polymer objects with different lengths and getting their radius of gyration you can calculate an ensemble average over all these polymers along with the error.

We also want to note that this WorldMap isn't exactly necessary for our purposes, but we thought it would be useful if one would want to extend the simulation to generate polymers close together on

the same lattice. So our code can also generate and grow an ensemble of polymers. It isn't optimized yet, the polymers get stuck in each other, but it is a proof of concept. In the Figure below you can see what such an ensemble can look like.

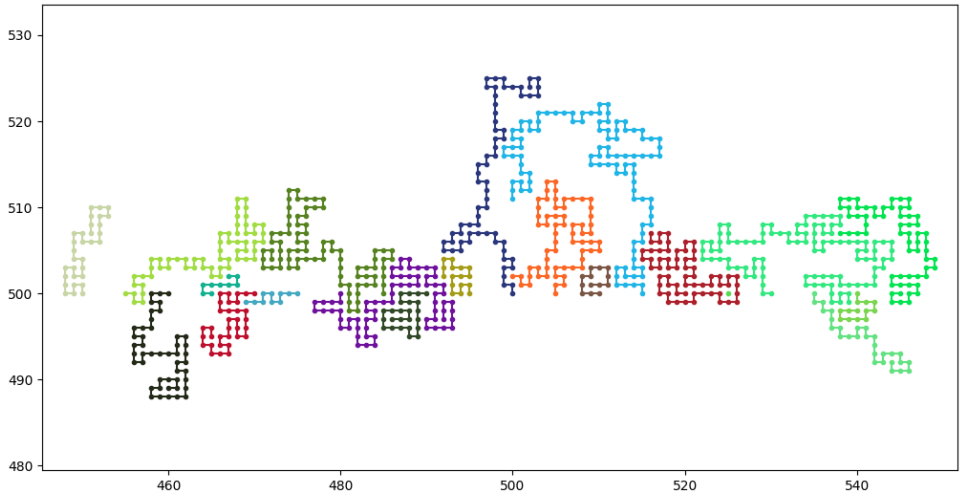

Fig. 2: An example of an ensemble that our code can generate. It is still a proof of concept, the polymers like to get stuck, but if the growing algorithm would be a bit smarter it could be very useful.

2.4 Radius of gyration

The statistical property we want to find from our simulation is the radius of gyration. This is a quantity that is important in polymer physics, and is defined as the end-to-end distance squared of a polymer. This means you calculated the distance from the starting monomer to the last monomer in the polymer chain and then take the square. Doing this for one polymer is not enough, we want to calculate a statistical average of it. That is why we generate a lot of random polymers, calculate their individual radius of gyration and then calculate the average. As one can expect, this quantity will grow for longer polymers. The specific way is scales as a function of length is as follows[1]:

$$\langle r_g^2(L) \rangle \sim L^{2\nu} \tag{1}$$

Where r_g^2 is the radius of gyration, L is the length of the polymer (number of beads in our case) and $\nu = 3/4$ in 2D. This has been theoretically derived, so to check if our simulation is correct we just needs to find the average of the radius of gyration for different lengths of the polymer and fit curve following this scaling law and see how close it is.

Like we explained before, with our code one can create an instance of a polymer with a specific length and extract its radius of gyration. When we can do this for a lot of polymers it is rather trivial to then calculate the average and fit a curve. In practice it is a bit more complicated how we do this (see next section), but this is the general idea.

2.5 The Rosenbluth method

As we have seen, one of the main goals is to find the radius of gyration of a polymer of length L. This can be achieved in many ways, one can simulate all possible polymers of length L and throw away the ones that are not self-avoiding. This method will work but it is not very efficient for larger lengths as you will have to throw away almost all polymers. Thus M. Rosenbluth and A. Rosenbluth [4] developed a Monte Carlo sampling method which approximates the correct distribution of polymers by giving them all weights. The weight of a polymer is dependent on how many possible directions the polymer can choose from for each length-step while remaining self-avoiding. This way we can also eliminate all polymers that get stuck and don't reach the given length, because they get a weight of 0. This method is also known as approximate importance sampling as we estimate the true probability of getting a polymer of length L by looking at the weights of all polymers of length L. To calculate the average radius of gyration, we use the following equation, which has been derived in the Lecture Notes[1]:

$$\langle r_g^2(L) \rangle = \frac{\sum_{k=1}^{N} w_k^{(L)} r_k^2(L)}{\sum_{k=1}^{N} w_k^{(L)}} \tag{2}$$

Here is $w_k^{(L)}$ the weight for polymer k with length L and $r_k^2(L)$ its radius of gyration. As you can see, it is just a weighted average.

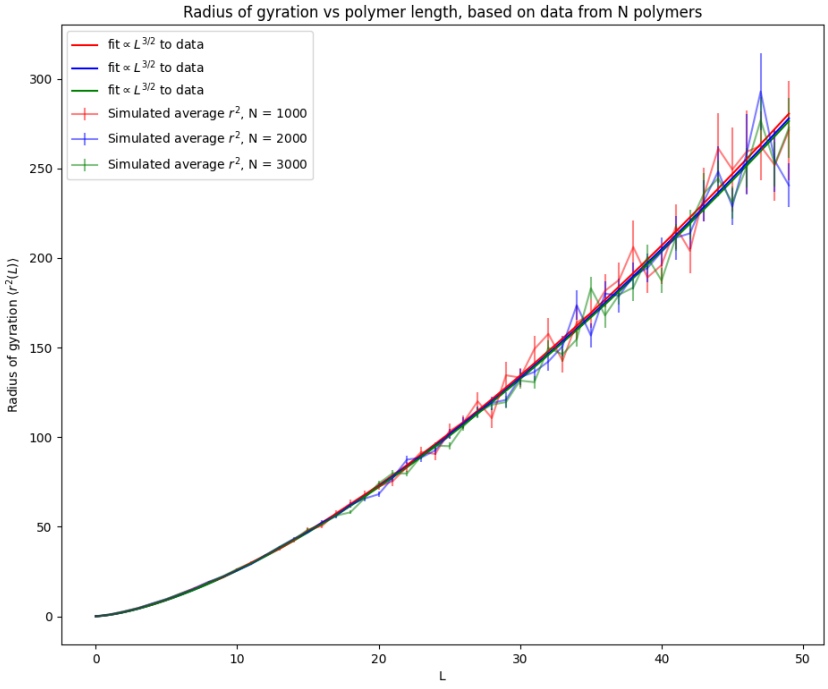

Fig. 3: Radius of gyration (y-axis) as a function of the polymer length (x-axis) for the Rosenbluth algorithm in 2D, together with a fitted curve. The error bars are calculated from Eq. 3

We implemented this algorithm and the results for 2D can be seen in Figure 3. The error bars in the Figure are given by equation 3 below, this equation is an analytical approximation of the standard error of a sample with weights. It is derived from a book on sampling techniques [2] [1].

$$s(\langle r^2(L)\rangle) = \sqrt{\frac{N}{(N-1)} \frac{\sum_{k=1}^{N} \left(w_k^{(L)}\right)^2 \left(r_k^2(L) - \langle r^2(L)\rangle\right)^2}{\left(\sum_{k=1}^{N} w_k^{(L)}\right)^2}} \tag{3}$$

For the simulations in Figure 3, N is very large and thus the fit (see section 2.4) is almost perfect. Still one can see that for larger values of L the error bars become quite large and the radius of gyration fluctuates a lot. Luckily there is a way to reduce this error which shall be discussed in the next subsection.

2.6 Pruned-enriched Rosenbluth method (PERM)

The previous section shows that the Rosenbluth method does indeed work. But there are still some major problems. First of all the Rosenbluth method is quite slow for larger L and fluctuates a lot, in a timed run (Figure 4) for a $L_{max} = 75$, $N = 1000$ it took 66.89 seconds to run. The second major problem is that for large L the weighted average is dominated by only a few polymers with extremely large weights. In this example, at the final length, the largest weight is 1.58×10^{34} which is 16.55% of the total of all the weights. The 50 largest polymers here account for 89.30% of all the weights, considering that there are 1000 polymers in the simulation this is not ideal.

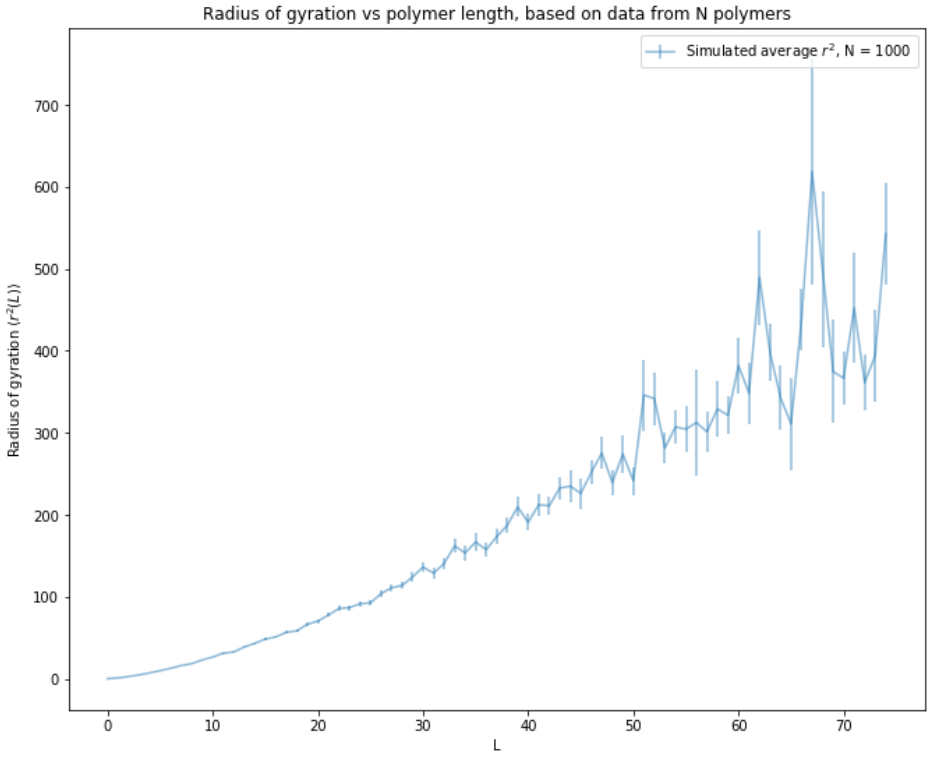

Fig. 4: Radius of gyration as a function of the polymer length for the Rosenbluth algorithm in 2D, $L_{max} = 75$, $N = 1000$. The error bars are calculated from Eq. 3

For these reasons Peter Grassberger [3] developed a new method of simulating self-avoiding polymers, the pruned-enriched Rosenbluth method. The main idea of PERM is to even out the weight distribution of the polymers while keeping the average the same. The algorithm is very similar to the regular Rosenbluth algorithm with a couple of extra steps. In this report the exact math and algorithm behind the code will not be discussed but the general outline is that the weight of each polymer for every length-step is compared to the average weight for that specific length. If the weight is much larger than the average the polymer is doubled (copied), and if the weight is smaller than the average it has a 50 % chance of being deleted and 50 % chance of its weight being doubled.

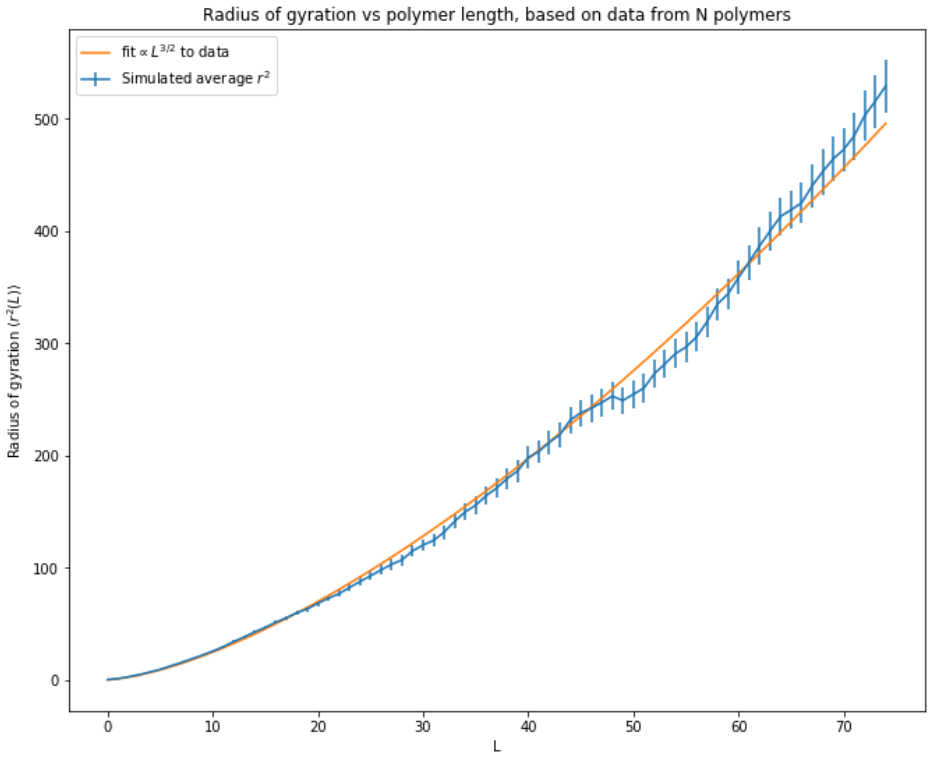

Fig. 5: Radius of gyration as a function of the polymer length for the PERM algorithm in 2D, $L_{max} = 75$, $N = 1000$. The error bars are calculated from Eq. 3

In figure 5 one can see the results of the same calculation ($L_{max} = 75$, $N = 1000$) of the end to end distance of polymers but now using the PERM algorithm. Immediately one can see that the fluctuations are much smaller and that the errors are also way smaller. It also runs much faster, this simulation took 1.89 seconds, which is more than 35 times as fast than the regular Rosenbluth algorithm. However, this is also partly due to an improved and smarter way to calculate the average radius of gyration. Finally, one can observe that the weight distribution is now much more uniform. Of the polymers that are alive at the final length the largest weight is only 0.70 % of the all the weights. The 50 largest polymers here account for 30.60% of all the weights, a major improvement.

3 Conclusion

In this report we have discussed different Monte Carlo algorithms of self-avoiding polymers in 2D where the radius of gyration is calculated. The Rosenbluth method is based on approximate importance

sampling in which one estimates the true distribution of polymers with an approximate distribution based on the weights of the polymers. This method works well, it is already an improvement to random sampling but it breaks down for larger lengths. This is where PERM comes in, a revision of the Rosenbluth algorithm in which one changes the weights of polymers while keeping the average end-to-end distance the same. This method is much faster and fluctuates less and can be seen a better version of the Rosenbluth algorithm as our results show.

References

1. Simulating polymers as self-avoiding random walks, https://compphys.quantumtinkerer.tudelft.nl/proj2-polymers/ , 2nd section, 4th paragraph, date accessed: 04/05/2021
2. Cochran, W.G.: Sampling Techniques, 3rd Edition. John Wiley and sons (1977)
3. Grassberger, P.: Pruned-enriched rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000. Phys. Rev. E **56**, 3682–3693 (Sep 1997). https://doi.org/10.1103/PhysRevE.56.3682, https://link.aps.org/doi/10.1103/PhysRevE.56.3682
4. Rosenbluth, M.N., Rosenbluth, A.W.: Monte carlo calculation of the average extension of molecular chains. The Journal of Chemical Physics **23**(2), 356–359 (1955). https://doi.org/doi:10.1063/1.1741967