A Monte Carlo simulation of polymers with Lennard-Jones interaction

Luka Bavdaz, 4228561*; Fabian Fool, 4224574*; Mathijs Garming, 4223551*; Stefan van der Sar, 4232488*
*Delft University of Technology, Lorentzweg 1, 2628CJ Delft, the Netherlands
Correspondence should be addressed to Mathijs Garming (M.W.H.Garming@tudelft.nl)

***Abstract*—We present a Monte Carlo simulation of 2D chain-like polymers generated by means of the Rosenbluth algorithm and the Pruned-Enriched Rosenbluth Method (PERM). Interaction between polymer beads is described by the Lennard-Jones potential. Additionally, a genetic energy minimization algorithm has been implemented, which reduces potential energy in the polymer exponentially with the number iterations. The PERM algorithm was found to be effective at suppressing high energy configurations and does not suffer from the attrition present in the Rosenbluth algorithm. In agreement with the theory, end-to-end distance and gyradius scale with $N^{1.5}$. Investigation of the effect of temperature suggests that polymers will resemble random walks for high T.**

CONTENTS

I. INTRODUCTION

POLYMERS are lengthy chain-like molecules that are important in biology as well as in materials science. In biology, molecules that are crucial to the inner workings of cells, such as proteins, DNA, and RNA are polymers. These polymers mostly exist in a folded state. Better understanding of the physics of these polymers can help gain insight in how cells work and why they might not in case of disease. For example, misfolding of polymers may result in various diseases including Alzheimer's and Parkinsons's disease [1]. Plastics, very common materials in the modern world, are also polymers. As the demand for different and better plastics is still present, better insight into polymer behaviour at a molecular scale is also relevant for the development of new plastics.

Here, we report on a Monte Carlo simulation of polymer chains in 2 dimensions. Two different algorithms for constructing the polymers are used, namely the Rosenbluth algorithm and the more advanced PERM algorithm. Parameters of interest include the end-to-end distance of the polymer, the radius of gyration/gyradius, and how these quantities depend on the length of the polymer. Furthermore, the persistence length of the polymers and the total energy of the polymers is evaluated. Lastly, the influence of temperature on polymer formation is examined.

In the next sections of the report, relevant theory and methods for this polymer simulation will be explained first, followed by the implementation. Subsequently, results are presented and conclusions are drawn.

II. THEORY AND METHODS

A. Bead interactions and Boltzmann statistics

Polymers are essentially a long chain of monomers or beads, connected by bonds of length l that make an angle θ with the previous bond. We take the bond length to be constant, which is a valid approximation as stretching or compressing bonds is energetically unfavourable [2]. The angle θ is different for every bond, which makes the shape of the final polymer quite irregular. In this simulation, the angle for each bond is chosen by means of random numbers. The likelihood of some angle occurring depends on the interaction energy associated with this angle, which is given by the Lennard-Jones potential:

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

Here, ϵ denotes the minimum potential energy, σ the distance at which the potential equals zero, and r the distance between two interacting beads, which depends on the chosen angle θ. Angles for which the sum of the Lennard-Jones interactions of the bead to be placed with the other beads is high are unlikely to occur as they cost a large amount of energy, whereas angles resulting in a small Lennard-Jones interaction energy are very likely. This is more specifically described by a Boltzmann weight for each angle θ :

$$w(\theta_i) = \exp\left(-\frac{E(\theta_i)}{k_B T}\right) \tag{2}$$

with $E(\theta_i)$ the interaction energy for that angle, T the temperature and k_B the Boltzmann constant. Consequently, the probability of some angle occurring is equal to the Boltzmann weight of that angle divided by the sum of the Boltzmann weights of all possible angles, which is also known as the partition function W [3]:

$$P(\theta_i) = \frac{w(\theta_i)}{\sum\limits_i w(\theta_i)} = \frac{w(\theta_i)}{W} \tag{3}$$

B. Physical quantities

There are a few physical quantities that we will study by the simulations. The most simple one is the end-to-end distance R, which is the distance from the first bead to the last bead, as the name suggests. It is often expressed as the distance squared and given by:

$$R^2 = (\overrightarrow{r}_{last} - \overrightarrow{r}_{first})^2 \tag{4}$$

The theoretical dependence of the ensemble average of R^2 on the number of beads N is according to the relation $\langle R^2 \rangle \propto N^{1.5}$ [3]. A somewhat more interesting quantity is the radius of gyration R_g, which is also known as the gyradius. It is interesting, because it can also be experimentally determined in various ways, which makes this quantity useful in comparing the simulation results with experimental results. The square of the gyradius can be calculated as follows [4]:

$${R_g}^2 = \frac{1}{N_{beads}} \sum_{i=1}^{N_{beads}} (\overrightarrow{r}_i - \overrightarrow{r}_{mean})^2 \tag{5}$$

Hence, R_g^2 represents the averaged squared deviation from the mean bead position. If we assume all beads to have the same mass and the bonds to be massless, multiplying R_g^2 by the mass M of the polymer gives its moment of inertia around the center of mass [4]. Just as in the case of the end-to-end distance, the ensemble average of R_g^2 should be proportional to $N^{1.5}$, which implies that the ensemble average of R^2 and the ensemble average of R_g^2 are proportional to each other [5].

The persistence length is a quantity which provides information on how much the polymer bends. It is the length over which the direction of subsequent bonds is correlated; i.e. the direction of bonds separated by more than the persistence length are uncorrelated. The persistence length is defined as follows [6]:

$$l_p = \frac{1}{l} \langle \vec{l_i} \cdot \sum_{j=i}^{N} \vec{l_j} \rangle. \tag{6}$$

It essentially is the sum of the projections of a bond on every subsequent bond. The average is taken over all possible reference bonds $\vec{l_i}$.

C. Weighted ensemble average and variance

It should be noted that the total Boltzmann weight of a certain polymer is given by:

$$\exp\left(-\frac{E_{total}}{k_B T}\right) = \prod_{L=3}^{N_{beads}} w_L \tag{7}$$

where w_L is the weight of the L^{th} bead given by equation 2 with θ_i replaced by the chosen θ value. Note that the weights of the first and second bead are set to 1 and therefore not included in the product. Since these weights only include interactions with previous beads, we do not count any particular interaction twice. Equation 7 means that the probability to have a polymer configuration with a particular total energy in the generated ensemble is given by the Boltzmann distribution. However, the chance for a particular angle is given by equation 3. Hence, the total probability for a particular polymer configuration actually is given by:

$$P = \prod_{L=3}^{N_{beads}} \frac{w_L}{W_L} \tag{8}$$

This implies that we are off by a factor that is equal to the product of the partition function of each bead, because the polymers should be distributed according to equation 7. As a result, if we want to compute ensemble averages of the physical quantities decribed above, we have to calculate weighted averages and variances with the product of the partition functions of the beads as weights [3].

In general, if we denote the weights by g, the weighted ensemble average of quantity A is given by:

$$\overline{A}_{weighted} = \frac{\sum\limits_{j=1}^{N_{pol}} g_j A_j}{\sum\limits_{j=1}^{N_{pol}} g_j} \tag{9}$$

and the weighted ensemble variance is given by [7]:

$$\sigma^2_{weighted} = \frac{\sum\limits_{j=1}^{N_{pol}} g_j (A_j - \overline{A}_{weighted})^2}{\sum\limits_{j=1}^{N_{pol}} g_j}. \tag{10}$$

To now determine the precision of the calculated quantity A, we use the variance of the mean which is given by the variance of the distribution (equation 10) divided by N' [8], where N' is the number of non-zero weights. Hence:

$$\sigma^2_{A,weighted} = \frac{\sum\limits_{j=1}^{N_{pol}} g_j (A_j - \overline{A}_{weighted})^2}{N' \sum\limits_{j=1}^{N_{pol}} g_j} \tag{11}$$

III. Implementation

In this section we describe the implementation of two methods for generating an ensemble of polymers: The Rosenbluth algorithm and the pruned-enriched Rosenbluth method, which is abbreviated as PERM and is essentially an improvement of the Rosenbluth algorithm. We also describe a genetic algorithm to minimize the potential energy of a polymer as well as the workings of the crossing detector.

A. Rosenbluth algorithm

A polymer is characterized by an array that contains the x and y coordinates of each single bead. The first bead is always placed in the origin, while the second bead is placed 1 bond length away at (0,1). Now, we start adding all the other beads. The coordinates of the next bead depend on the bond length, which is a predefined number, and the angle θ between the new bond and the previous one. For each possible angle we have to calculate the Boltzmann weight (see equation 2). Since this also involves summing the interaction energies of the new bead with each of the previous beads, we have to restrict the number of possible angles in order to finish the simulation within a reasonable amount of time. The angles are equally spaced with the first angle being chosen at random to avoid introducing a lattice. After summing the weights for each angle, we calculate the probability for each angle by means of equation 3.

The so-called 'roulette wheel algorithm' is used to pick an angle from the discrete set of possible angles. A random number between 0 and 1 is generated and we check if this number is smaller than the probability of the first angle. If yes, then we choose the first angle. If not, we add the weight of the second angle to the weight of the first angle and check if the generated random number is smaller. If yes, we choose the second angle. If not, we add the third weight and so on. Once an angle is chosen, the position of the L^{th} bead is given by:

$$\begin{aligned} x[L] &= x[L-1] + l\cos(\theta_{chosen}) \\ y[L] &= y[L-1] + l\sin(\theta_{chosen}) \end{aligned} \tag{12}$$

where l is the predefined bond length. Note that for the calculation of the Boltzmann weights of the new bead we also had to use equation 12 to determine the possible new bead position that determine the interaction energy and therefore the Boltzmann weight of the possible new bead positions.

In this manner, we keep adding beats until the predefined number of beads is reached. However, it may occur that the polymer crosses itself. In this case the energy becomes enormous and all weights become zero because we have finite precision. In this case the polymer is discarded and we start building a new one.

After the polymer is finished we append the array with bead positions to a list so they are available for determining various physical quantities afterwards. Now, we can start generating a new polymer according to the method described above until we have the desired number of polymers.

B. Calculating physical quantities and errors

As explained in section II-C we have to calculate weighted ensemble averages and variances to obtain physical quantities of interest and corresponding errors. To this end, we also save the polymer weight at every step in the polymer generation process. After adding the L^{th} bead we can calculate the polymer weight at that length in the following way:

$$polWeight[L] = polWeight[L-1] \cdot W_L. \tag{13}$$

The weight of the first two beads is set to 1 and W_L is the partition function of the L^{th} bead from equation 3. When a polymer is finished, the array is appended to a list, such that we have all weights available for calculating ensemble averages and variances of physical quantities.

The first physical quantity of interest is the end-to-end distance. Since the first bead of each polymer is always located at the origin equation 4 reduces to simply squaring the x and y coordinates of the new bead and adding them up. For each new bead we store this number into an array and when the polymer is finished, the array is appended to a list, as is done with the positions and weights. Hence, when the whole ensemble is generated, we can calculate the weighted ensemble average of the end-to-end distance and its error as a function of the number of beads by equation 9 and 10.

The calculation of the gyradius is performed using equation 5 and is implemented as follows: First, we loop over all polymers. For each polymer, we look for the last bead. In the second loop ,which is nested inside the first one, we work out equation 5 to calculate the gyradius for each number of beads. Combining these data with the polWeights described above allows us to calculate the weighted ensemble average and variance as a function of the number of beads.

Calculation of the persistence length is done by implementing equation 6. For every polymer, the value of l_p is calculated taking every bond as the reference bond $\vec{l_i}$ once. The average persistence length for the polymer is then obtained by taking the average value of the local persistence lengths. The average persistence length over all polymers is then calculated by taking a weighted average using polWeights as weight.

C. PERM

It is in theory possible to generate polymers of arbitrary length with the Rosenbluth algorithm, but there are multiple problems with this algorithm. First there is a high chance of a polymer crossing itself which results in an exponential attrition as those polymers have to be discarded [9]. In line with this first problem is that this algorithm does not suppress high energy configurations sufficiently [3].

To solve these problems Grassberger [9] has come up with a different algorithm: PERM. The basis of PERM is the same as the Rosenbluth algorithm, but there is some population control which throws away polymers with low weight, the high energy configurations, and adds copies of polymers with large weight. This is implemented by comparing the weight of the polymer after a bead is added with an upper limit and a lower limit. If the weight is below the polymer limit we stop building the

polymer with chance 0.5, or continue building but double the weight. If the weight exceeds the upper limit we continue with two polymers with half the current weight.

The upper and lower limit have to be chosen such that the population remains stationary. A good choice [3] is to take the the limit as multiples of the ratio of the average weight for polymers having reached the current bead and the weight of the polymers with a length of 3:

$$UpLim = \alpha \frac{AvWeight}{Weight3}. \tag{14}$$

The expression for the lower limit is similar. For a constant population an alpha of 2 for the upper limit and 1.2 for the lower limit is suggested in the book by Thijssen.[3] The α does depend on the polymer length, but this can be circumvented by multiplying the weight by $1/(0.75N_\theta)$ at each step.

D. Crossing detector

To quantify the improvements of the PERM algorithm over the regular RM, we compare the average number of crossings in the polymers. In a 2D polymer, a crossing would mean two parts of the polymer occupy the same position, which should be impossible due to the Lennard Jones potential of the configuration. Therefore, we desire the number of crossings to be minimal. However, due to the discrete nature of the beads, the Rosenbluth method does produce crossings occasionally. The crossings can be computed by checking if the lines of two links in the chain intersect in between the bead positions. Cramer's rule can be applied to find the position of the intersection by computing determinants and ratios thereof.

E. Genetic algorithm

The polymers we simulate consist of only one element and only interact via the Lennard-Jones force and are therefore very different than the proteins that consist of many elements and interact in many different ways. Still it is interesting to fold our polymer in the best possible way. The best fold is described in our case by the lowest possible potential energy. Finding this configuration is not easy as there are many configurations possible, among which many have a comparable potential energy.

In our simulation we try to find the minimal potential energy configuration using a genetic algorithm. This algorithm is inspired by the theory of evolution. For this algorithm we start with a certain amount of polymers and do the following steps at each iteration: First we calculate the potential energy of all polymers, then we prune and enrich the population based on the potential energies of the polymers. The third step is to mate chosen pairs and create offspring. The two chosen polymers are cut at a certain bead and then the left part of polymer 1 is connected to the right part of polymer 2 and the other way around. The last step is to mutate some of the polymers. Here we randomize one of the angles in the chain or move one of the beads.

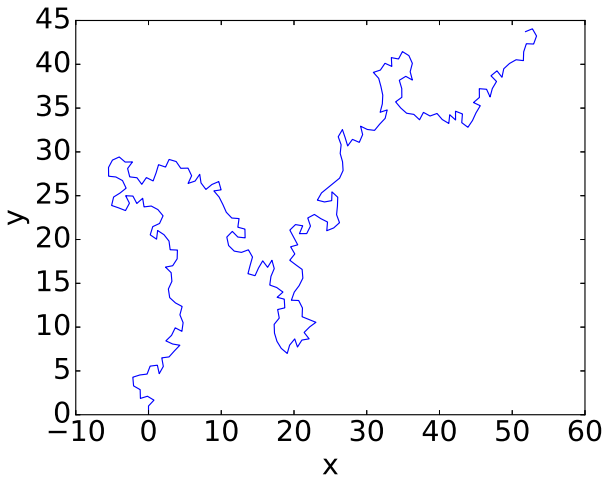

Fig. 1. Typical polymer generated with PERM with a length of 250 beads.

IV. Results

In this section we present and discuss the simulation results. Unless otherwise stated we will restrict the number of angles to 6, use a temperature and bond length of 1 and for the Lennard-Jones potential we will use 0.25 for the potential depth and 0.8 as distance for which the potential is zero. Figure 1 depicts a typical polymer generated with PERM with a length of 250 beads.

A. Attrition

We mentioned in section III-C that there is an exponential attrition when using the Rosenbluth algorithm. This can indeed be seen in figure 2 where we have simulated 10000 polymers up to a length of 400. This figure shows the attrition only due to the polymer weight becoming zero. We see that only 6% of the polymers reaches a length of 400 beads without zero weight. The weight usually becomes zero when the polymer crosses itself as the potential energy then becomes very high, but this is not always the case. There is still a chance that the polymer crosses itself and that the weight does not become zero. In the 6% of the polymer that survived to a length of 400 beads the crossing detector found 4.1 crossings per polymer. This means that it is likely that all remaining polymers contain crossings and have to be discarded, leaving us with no remaining polymers. This implies that is is very hard to generate polymers without crossings for this length and that generating many long polymers will take a very long time. Therefore the Rosenbluth algorithm is totally impractical to generate long polymers.

B. End-to-end distance

In this subsection we present and comparare the results of the end-to-end distance calculation for both the Rosenbluth algorithm and PERM. In both cases the ensemble of polymers consists of 2500 polymers with 250 beads.

We see from figure 3 that the normal Rosenbluth algorithm suffers from fluctuations of the ensemble average R^2 for more than 100 beads. The reason is that in general the algorithm

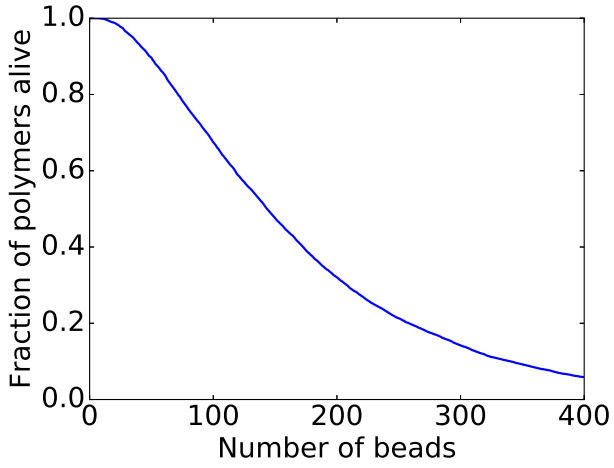

Fig. 2. Attrition due to polymer weight becoming zero in the Rosenbluth algorithm. Only 6% of the polymers reaches a length of 400 beads without zero weight.

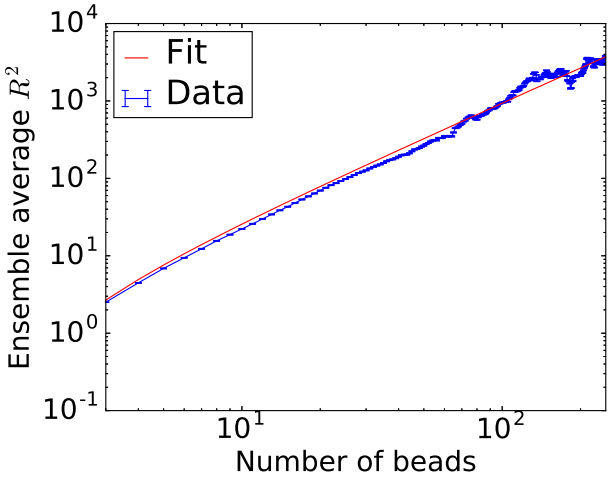

Fig. 3. Ensemble average of the end-to-end distance squared as a function of the number of beads using the Rosenbluth algorithm with fixed population of 2500 polymers.

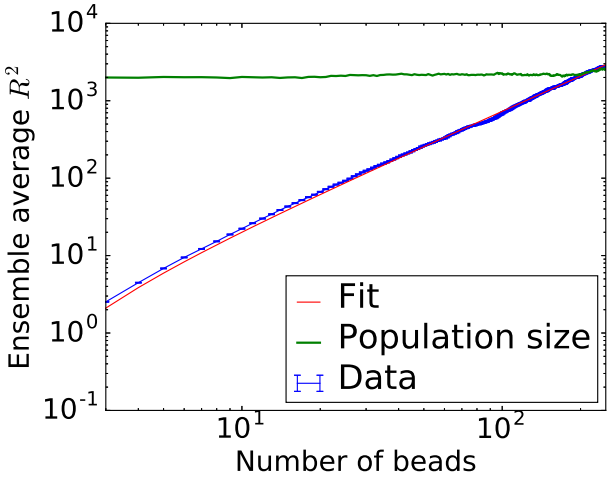

Fig. 4. Ensemble average of the end-to-end distance squared as a function of the number of beads using the PERM algorithm. Note that the population size is not exactly constant for PERM.

does not suppress high energy configuration well enough. High energy configurations are accepted, but they have a small weight. The more beads added, the more opportunities to choose a high energy configuration, so the more polymers have a small weight in the end. Therefore, only a few polymers having large weight dominate the weighted average, which results in bad statistics. If we run the simulation again, the right-hand part of the graph can look completely different. This has also consequence for the red line in figure 3, which represents the fit to $aN^{1.5}$ (see section II-B). It turns out that the parameter a varies significantly between different simulations due to the fluctuations. In general, we can say that a assumes a value between 0.45 and 0.95.

The PERM algortihm should better suppress the high-energy configurations. Figure 4 shows that this is indeed the case, because fluctuations for large numbers of beads are not visibile anymore. The fit is also much better this time, because the fluctuations are gone. The value for a is 0.73 now. However, the green graph in figure 3 shows that the population size is slightly varying in this algorithm. This is a direct consequence of the way this algorithm works (see section III-C).

Since PERM provides better results, all results in subsequent sections are generated using the PERM algorithm.

C. Gyradius

For the computation of the gyradius, a simulation of 2500 polymers with a length of 250 beads has been carried out with PERM enabled. Plotting the gyradius versus the number of beads in a polymer as shown in figure 5 results in a figure that looks quite similar to the end-to-end distance, which was expected. The fit ($R_g^2 = aN^b$) yields coefficient values of $a = 0.09$ and $b = 1.53$, where $b = 1.5$ was predicted by the theory. We may therefore conclude that the data and the theory agree quite well. Furthermore, we can now determine the proportionality relation between the ensemble averages of the squared end-to-end distance and the squared gyradius:

$$\frac{\langle R^2 \rangle}{\langle R_g^2 \rangle} \approx \frac{0.7N^{1.5}}{0.1N^{1.5}} = 7 \rightarrow \langle R^2 \rangle \approx 7 \langle R_g^2 \rangle \qquad (15)$$

D. Persistence length

The persistence length has only been calculated for full length polymers. Using a data set of 3377 polymers with length 250 generated with the PERM algorithm yields an average value of 6.6 with an error of 0.1. Though this error in the mean is quite small, the standard deviation of the distribution is quite large at 4.3. This means that, despite all polymers being generated under the same conditions, a large range of persistence lengths is possible. Comparing this result with the sample polymer in figure 1, we can conclude that the result is realistic.

E. Energy distribution

In figure 6a we show the total potential energy distribution after 10,000 polymers have been grown to 100 beads without

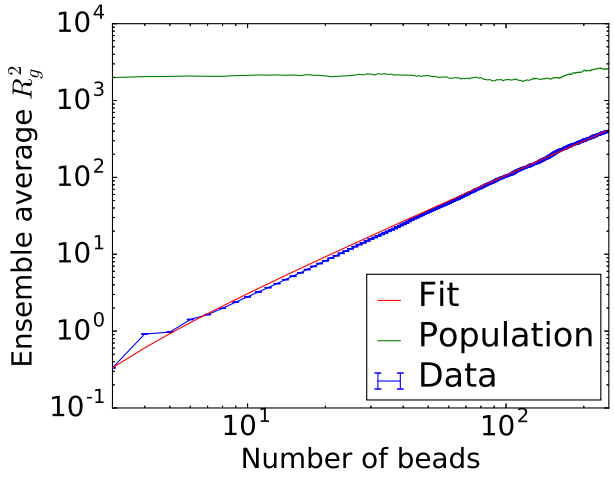

Fig. 5. The square of the gyradius plotted as a function of the number of beads in a polymers with fit. While very small, error bars are also plotted. From N=10 onwards, the fit follows the data very well. Settings: 2500 polymers, 250 beads, PERM enabled.

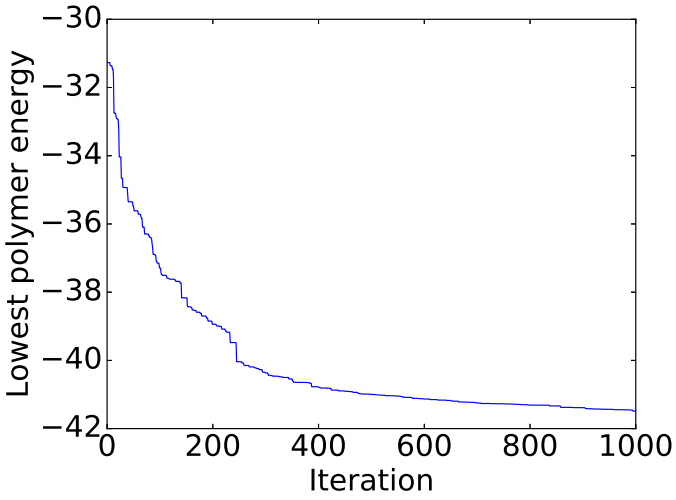

Fig. 7. Energy minimization with 1000 iterations. The energy of the lowest energy polymers seems to decrease exponentially with the number of iterations.

PERM. It is clear that most polymers have a potential energy below zero and very few above that. The distribution looks exponential. This is expected because the probability for a certain polymer configuration depends on the Boltzmann weight. We thus see that for 100 beads the algorithm works as expected. However if we increase the number of beads to 250 the distribution changes as can be seen in figure 6b. The bin below zero still contains the most polymers, but much less compared to before. There are now much more polymers with higher potential energy and the distribution does not look exponential anymore. So we see that for more beads the higher energy configurations are not well supressed anymore using the Rosenbluth method, once again proving that this algorithm is bad for longer polymers. These observations also support the fluctuations observed in figure 3.

If we use PERM to generate the polymers the potential energy distribution turns out to be very different. In figure 6c the energy distribution for polymers with a length of 100 beads is shown. Instead of an exponential distribution the distribution looks closer to a Gaussian and has a sharp cut-off at the right. This is the effect of the pruning and enrichment step in the algorithm. The population is forced within a small band of weights determined by the upper and lower limit and the worst polymers are almost certainly removed. The same kind of distribution is also found for longer polymers with 250 beads as can be seen in figure 6d. The only clear difference is that the spread in energy is larger.

F. Genetic algorithm

To analyse the energy minimization algorithm, we first generate 1000 polymers with a length of 100 beads using the Rosenbluth algorithm. These 1000 polymers are then used in the first iteration of the genetic algorithm. The genetic algorithm then generates a new set of polymers from these polymers in the hope that the potential energy goes down. The minimal potential energy for the first 1000 iterations is shown in figure 7. We see that in the first 250 iterations the potential energy drops fast, but it goes much slower towards the end. It looks like the process of energy minimization does display asymptotic behaviour. Most likely the polymers are close to a local minimum and the algorithm does not succeed to find another, possibly better, minimum. Our algorithm does minimize the potential energy, but there is still room for improvement.

G. Polymer self-crossings

Simulations of 2500 polymers with 250 beads each were used to determine the average amount of crossings with itself. The simulation was repeated a total of 10 times for each method. Using the Rosenbluth method, we find 2.6 ± 0.2 crossings per polymer on average while PERM reduces this number to 0.004 ± 0.002 per polymer. From this, we can confirm that PERM does indeed result in polymers that are closer to the desired configurations with 0 crossings.

H. Effect of temperature

The temperature has an effect on the probabilities of a certain angle. If we look at equation 2 we see that for higher temperatures the term in the exponent becomes smaller, resulting in the probabilities for each angle becoming closer together. For very high temperatures the algorithm will turn from a self avoiding random walk into just a random walk. This should be visible in two things. First the amount of crossing per polymers will increase. We indeed observe this if we run the simulation without PERM as instead of the 4.1 crossings per polymer, which was the case for a temperature of 1 and a length of 400 beads, it increases to 22 crossings per polymer for a temperature of 100 and the same length. This means that most likely all of the 10000 polymers have crossings and that at this temperature the Rosenbluth algorithm is unusable. The second things that should change is the relation between the end-to-end distance squared and the number of beads. For a random walk the power is 1 instead of

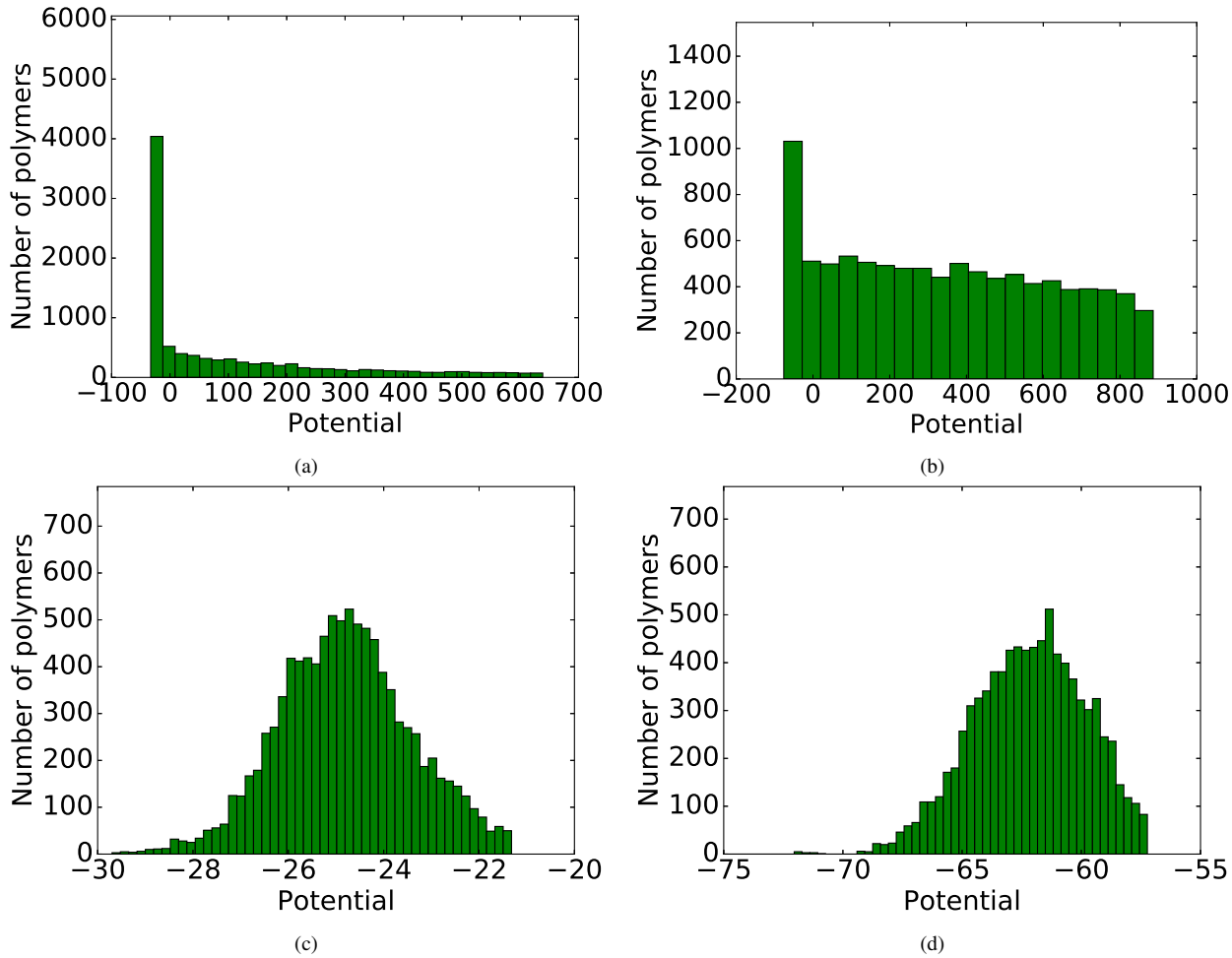

Fig. 6. Energy distribution after a simulation of 10000 polymers. Subfigure (a) shows the result after 100 beads with the Rosenbludth algorithm and (b) after 250 beads. In (c) and (d) we do the same but now with PERM. The population size in (c) is 8074 polymers and 8104 in (d).

1.5.[10] This is not observed if we look at the weighted end-to-end distance. In this case we see more severe fluctuations as compared to figure 3, which is due to very few polymers having a high weight. If we look at the unweighted end-to-end distance the fit returns a power of 1.1 for a temperature of 100. This is still higher than expected, but the temperature is most likely still too low for a completely random walk. So we can conclude that for high temperatures the algorithm will resemble a random walk.

V. Conclusion

The presented simulation yields results that agree with literature as well as physical intuition. The results clearly show that the PERM algorithm is a valuable addition to the Rosenbluth algorithm, particularly in terms of energy distribution and attrition. In agreement to the theory, end-to-end distance and gyradius both scale scale with $N^{1.5}$. Algorithms to calculate the persistence length and to minimise the potential energy also seem to provide the desired result. For elevated temperature, an increased number of crossings has been observed, supporting our hypothesis that the algorithm turns into a random walk for high T.

For future work, the simulation can be enhanced in several ways. One way is to implement polymers containing multiple types of beads with different Lennard-Jones interactions. This would allow for the simulation of polymers built from different types of monomers. Something the simulation presently does not take into account is the stiffness of the polymer. Going beyond the self avoiding random walk model and taking into account the energy cost of bending the polymer by means of, for example, the worm like chain model [11] would also make the simulation applicable in more types of real polymers. Particularly biological polymers such as DNA and proteins could be simulated better this way. Furthermore, the simulation carried out is in 2D, so the step to 3 dimensions could be made. This would give insight into how a polymer would fold when floating around in a solution, such as the cytoplasm of a cell.

References

[1] "Folding@home," https://folding.stanford.edu/.
[2] M. Elnster and T. Kubar, *Biomolecular modeling*. TU Braunschweig, 2011.
[3] J. Thijssen, *Computational physics*. Cambridge University Press, 2007.
[4] I. Teraoka, *Polymer Solutions: An Introduction to Physical Properties*. John Wiley & Sons, 2002.
[5] M. K. Kosmas, "On the mean radius of gyration of a polymer chain," *Journal of Physics A: Mathematical and General*, vol. 14, no. 10, p. 2779, 1981. [Online]. Available: http://stacks.iop.org/0305-4470/14/i=10/a=029

[6] T.Lezon, *Statistical Mechanics of Chain Molecules: An Overview*. University of Pittsburgh.

[7] "Calculation of weighted moments and cumulants of probability distributions and samples," http://www.nematrian.com/WeightedMomentsAndCumulants.

[8] P. Nagele, "Misuse of standard error of the mean (sem) when reporting variability of a sample. a critical evaluation of four anaesthesia journals," *British Journal of Anaesthesia*, vol. 90, no. 4, pp. 514–516, 2003. [Online]. Available: http://bja.oxfordjournals.org/content/90/4/514.abstract

[9] P. Grassberger, "Pruned-enriched rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000," *Phys. Rev. E*, vol. 56, pp. 3682–3693, Sep 1997. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.56.3682

[10] "Random walk–2-dimensional," http://mathworld.wolfram.com/RandomWalk2-Dimensional.html, accessed: 09-04-2016.

[11] J. N. Milstein and J.-C. Meiners, *Encyclopedia of Biophysics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Worm-Like Chain (WLC) Model, pp. 2757–2760.