Monte carlo simulation of polymers using the PERM algorithm

University of Technology, Delft
Daniël Bouman & Kenneth Goodenough
Student number: 4146077 & 4334175

Abstract

Polymers are chemical compounds, whose constituents are multiple repeating units. Monopolymers are polymers that consist of a single repeated unit. To study the behaviour of monopolymers of different lengths at various temperatures, a simulation is written in Python and Fortran. The model for the polymers is based on approximating the interaction between the monomers with a Lennard-Jones potential and a bending energy between adjacent monomers. Simulations were made using the pruned-enriched Rosenbluth method to investigate the scaling behaviour of the end-to-end distance, radius of gyration, and the temperature dependency of the polymers. The influence of the bending energy on the radius of gyration is also discussed.

1. Introduction

Ensembles of polymers are grown using Python. By assuming the polymers are situated in a dilute solution, polymer-polymer self-interactions will prevail. This means that for reliable data of the behaviour of the polymers, creating an ensemble of polymers with only self-interactions will be sufficient. Quantities of interest for polymers are the end to end distance and radius of gyration. The end to end distance is the distance between the beginning and end of a polymer, while the radius of gyration is the root mean square of distance of the components of the polymer from its centre of mass.

The polymers are modelled as consisting of a chain of "beads", where the beads are groups of atoms. The distance between subsequent beads is fixed[1]. The interaction between the beads is modelled by a Lennard-Jones (LJ) potential (Eq. 1), which represents a Van der Waals attraction at larger distances and repulsion at smaller distances, and a bending energy (Eq. 2), which simulates the semi-flexibility of polymers. The LJ potential energy is given by

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

where ϵ is the depth of the potential well, σ is the finite distance where the potential is zero and r is the distance between two particles. The parameters chosen for the LJ potential are $\epsilon/k_b = 0.25$, with k_B the Boltzmann constant, and $\sigma = 0.8$ [1]. The bending energy is given by

$$E_b = \epsilon_b(1-\cos\theta), \tag{2}$$

where $\epsilon_b \geq 0$ is the bending energy parameter and θ is the angle between the new and preceding bead [2].

1.1. Markov chain

For a canonical ensemble the probability of a given state α is proportional to the Boltzmann weight W_α, $P(\alpha) \propto e^{-\beta E(\alpha)}$, where the proportionality constant (the inverse of the partition function) is fixed by the normalization constraint. Here $\beta = 1/(k_B T)$, where T the temperature and $E(\alpha)$ the energy of a state α. For polymers, it is natural to generate configurations bead by bead. The phase space can be sampled randomly, i.e. configurations are generated randomly where the only constraints are the amount of beads and that the distance between subsequent beads is fixed. In the canonical ensemble the thermodynamical average of a

[1]It is possible to set the distance between subsequent beads as variable, where the corresponding energy can be modelled by a harmonic potential. Assuming that the strength of the harmonic potential is sufficiently high, this case reduces to a fixed distance between beads.

quantity A is given by

$$\langle A \rangle = \sum P_\alpha A_\alpha = \frac{\sum W_\alpha A_\alpha}{\sum W_\alpha}, \quad (3)$$

where the sum is over the whole phase space and the weights are

$$W_n = \prod_{l=2}^{L} w_j^{(l)} = \prod_{l=2}^{L} e^{-\beta e_j} = e^{-\beta E_{total}}, \quad (4)$$

where n denotes the polymer, w_j the Boltzmann weight for a bead j and E_{total} is the total energy of the polymer.

If the phase space is sampled K times, then

$$\langle \tilde{A} \rangle_K = \frac{\sum_{n=1}^{K} W_n A_n}{\sum_{n=1}^{K} W_n}, \quad (5)$$

$$\langle A \rangle = \lim_{K \to \infty} \langle \tilde{A} \rangle_K. \quad (6)$$

Each W_n is thus calculated and stored for each polymer. It is easy to see that sampling the phase space for states with low energy will speed up the convergence of $\langle \tilde{A} \rangle_K$ to $\langle A \rangle$. Note that $l = 2$ is the third polymer, since the simulation starts counting at 0.

Since the polymer is not explicitly self-avoiding, the possible new configurations when a new bead is added depends only on the last bead that was placed. This method thus has the Markov property. However, sampling the phase space completely randomly is, as can be expected, incredibly inefficient, mainly due to the high probability of beads being placed close to each other where the hardcore repulsion causes the weight of almost all the generated polymers to become negligible. As outlined below, the Rosenbluth and Rosenbluth (RR) algorithm [3] follows a Markov Chain Monte Carlo method, where a bias is implemented to sample the phase space for lower energy states.

2. Rosenbluth and Rosenbluth algorithm

Setting the distance between subsequent beads to unity, it can be seen that the initial two beads can be placed at placed at positions $(0, 0)$ and $(1, 0)$, without loss of generality. For the third bead, a discrete number of candidate positions θ_j are determined. For the simulations run, $j = 1, \cdots, 6$. Since the distance between neighbouring beads is always one, these positions lie on a circle around the second bead, with a spacing of $2\pi/6$ radians between them. To prevent a bias in the direction of the polymer, these positions are given a new random offset for each new bead.

Since the separation between two adjacent beads is fixed, the energy contribution from adjacent beads is constant. This implies that these interactions have no physical significance. The sum of the weights $w_j^{(l)}$ from the candidate positions is $W^{(l)} = \sum_j w_j^{(l)}$. The final position of bead l is then chosen with probability $w_j^{(l)}/W^{(l)}$, creating a bias towards states with a lower energy. This process is then repeated for the remaining beads.

3. Pruning and enrichment

For long chains the RR algorithm becomes inefficient and inaccurate, as was shown by Batoulis and Kremer [4]. The inefficiency occurs due to the fact that still too much time is spent on generating polymers with a low Boltzmann weight. The inaccuracy is caused due to the fact that the weights of the individual monomers are only weakly correlated, which results in a large spread in the final weights of the polymers. This causes a few configurations to dominate the distribution, causing those configurations to dominate the ensemble average.

To sample the phase space more effectively, Grassberger has suggested pruning and enriching the population. If a partially grown polymer gets a weight below a certain threshold, half of the time the polymer is discarded ('pruned'), and the other half of the time its weight is doubled (to compensate for the pruned polymers) and growing continues. The is done to compensate for changes in the distribution due to removing polymers. When a partially grown polymer gets a weight above a certain threshold, the polymer is split. Both polymers are independently grown from this point on and their weights are halved. The population is now 'enriched'. This combined with the RR algorithm is called the pruned-enriched Rosenbluth method (PERM).

The reasoning behind pruning and enrichment is that it is inefficient to continue growing a partial polymer with a low weight since it will have a low contribution to the weighted average. It is then better to prune this polymer and start over. When a polymer gets a relatively high weight it is important and advantageous to make a copy of this polymer.

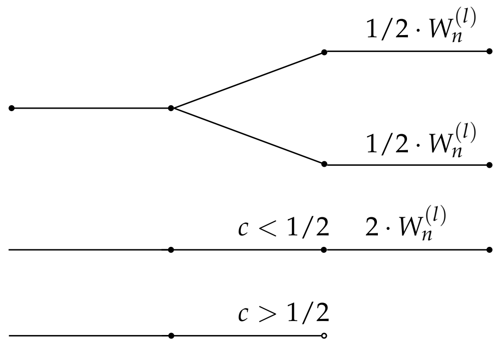

Figure 1: *Pruning and enrichment. The new weights are depicted above each branch. c is a random generated number between 0 and 1. Upper branch: the weight of the partial polymer $W_n^{(l)} > W_+^{(l)}$, so the polymer is split and the weight of both polymers is halved. The middle branch: $W_n^{(l)} < W_-^{(l)}$ and $c < 1/2$ so the weight is doubled and the polymer continues growing. Bottom branch: $W_n^{(l)} < W_-^{(l)}$ and $c > 1/2$ so the polymer is pruned.*

For a finite system the partition sum can be estimated [2] by

$$Z^{(l)} \simeq \hat{Z}^{(l)} = \frac{1}{N_l} \sum_{n=0}^{N_l} W_n^{(l)},$$

where N_l is the number of thus far completed polymers. The partition function is used to determine the upper limit W_n^+ and lower limit W_n^- for the polymer weights, above or below which a polymer is enriched or pruned respectively as described by Hsu and Grassberger [2]. This is done as follows

$$W_+^{(l)} = C_+ \hat{Z}^{(l)},$$
$$W_-^{(l)} = C_- \hat{Z}^{(l)},$$

where C_+ and C_- are constants of order unity and should have a ratio $C_+/C_- \simeq \mathcal{O}(10)$. These constants determine if the population will grow or shrink and are chosen by running the simulation with different values and choosing the best combination.Figure 1 shows an illustration of PERM.

For the first polymer there are no $\hat{Z}^{(l)}$ yet, so for this polymer the upper and lower limits are set to infinity and zero respectively ($W_+^{(l)} = \infty$ and $W_-^{(l)} = 0$), so that the first polymer is effectively created with the RR algorithm.

4. Results and discussion

4.1. Scaling of the end to end distance

The polymer grown with RR and PERM correspond to a self avoiding random walk. It is thus expected that the end to end distance r scales with the number of beads as

$$r \propto (N_l)^{\nu},$$

where ν is the Flory exponent and should be $3/4$ for 2D [1]. In Figure 2 $\langle r^2 \rangle$ is plotted against the number of beads. A trendline is fitted through the data points. The coefficient of determination has the value $R^2 = 0.952$ and the Flory constant was found to be $2\nu = 1.47 \pm 0.0424$ which is in good agreement with the expected value $\nu = 3/4$.

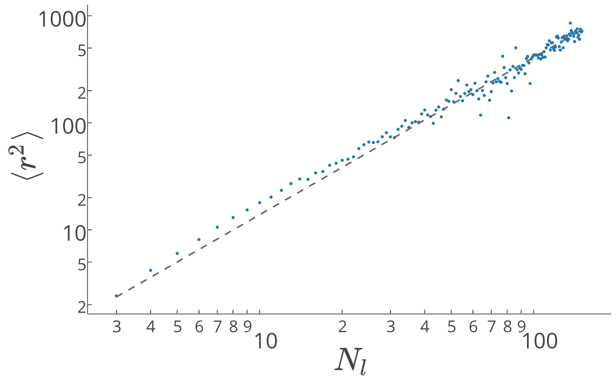

Figure 2: *Average end to end distance $\langle r^2 \rangle$ as a function of polymer beads N on a log scale. The blue dots depict $\langle r^2 \rangle$ and the dashed line is fitted to $\langle r^2 \rangle$ with $a \cdot N^{2\nu}$. The fitted value for* $2\nu = 1.469 \pm 0.04246$.

For low temperatures it is expected that the attractive part of the LJ-potential will dominate, causing the polymer to coil up. For higher temperatures, the repulsive part will always dominate, causing the polymer to expand. Since the LJ-potential is negligible beyond a distance of 2.5σ, the end to end distance will become independent on the temperature for high temperatures. Since a random walk will exhibit an end to end distance r somewhere between these two regimes, it is expected that inbetween the two regimes the temperature T_c can be found where $\langle r^2 \rangle$ equals $\langle r^2_{random} \rangle$, where $\langle r^2_{random} \rangle = N$ is the expectation value of the end to end distance squared for a random walk in 2 dimensions.

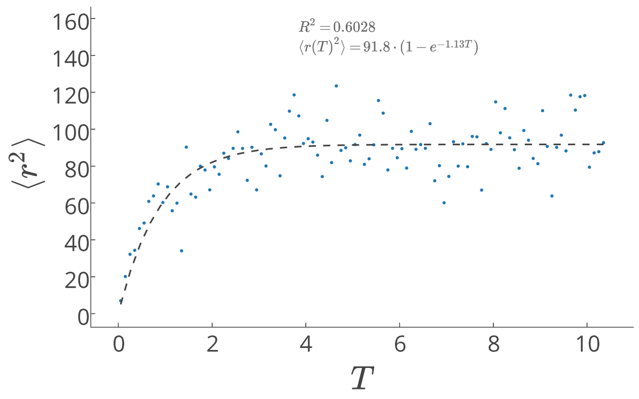

Figure 3: *Expectation value for the end to end distance squared as function of the temperature, for polymers with a length of 30 beads*

Since for high temperatures $\langle r^2 \rangle$ becomes approximately constant, a fit function in the form of $a(1 - e^{-bT})$ was chosen, with a and b the fit parameters. From the fit the value for the critical temperature T_c is found to be approximately 0.35.

4.2. Bending energy

As the bending energy parameter ϵ_b increases, the polymers should straighten out due to straight configurations having a lower energy. At a certain point, the bending energy will overcome the attractive part of the LJ-potential at low temperatures, causing the polymer to uncoil. A simulation was performed where ensembles of 30 beads were generated for ϵ_b / k_b between 0 and 15 at a temperature of $T = 1$.

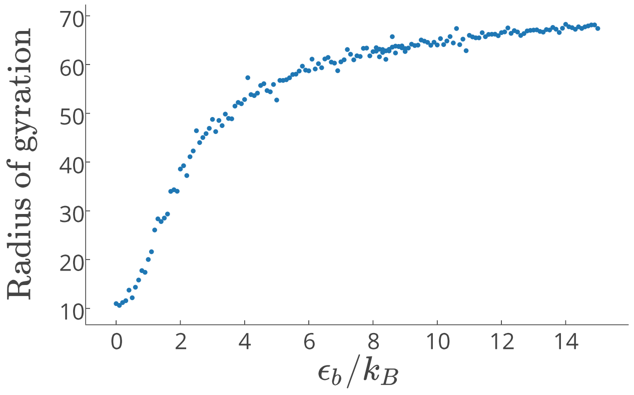

Figure 4: *Expectation value of the radius of gyration as function of the bending energy parameter ϵ / k_b. Values were found by generating ensembles with the PERM algorithm for polymers consisting of 30 beads at a temperature of 1. Notice the several regimes as the bending energy parameter is increased.*

For large values of the bending energy parameter ($\epsilon_b / k_b > 6$), the radius of gyration becomes approximately a linear function of ϵ_b. For low values of the bending energy parameter ($\epsilon_b / k_b < 0.5$) the LJ-potential dominates the effects of the system. Between this regime the bending energy overcomes the LJ-potential. In the appendix a similar figure for $\langle r^2 \rangle$ can be found.

The fluctuations in $\langle r^2 \rangle$ become greater as the polymers get longer. This is most likely due to the fact that the population sizes of the ensembles also fluctuates significantly for larger polymers. This was found to be problem in most of the simulations, and is especially of present for polymers with more than ~ 40 beads, where the population size becomes unpredictable. In Figure 5 the population sizes for the ensembles in Figure 2 are depicted. We got the impression that in literature the population size did not fluctuate by large amounts using PERM. Unfortunately we were unable to find the cause of these large fluctuations.

4.3. Note on the program

The energy calculations, LJ potential and bending energy, are written in Fortran for better performance. To make a connection between the Fortran code and the Python program, an interface is made using *Fortran to Python interface generator* (F2PY). F2PY creates a Python C/API extension module that makes it possible to call the in Fortran written energy subroutine.

5. Conclusion

Most of the simulation results are in good agreement with expectations. The temperature dependency of the end-to-end distance becomes significantly less as the temperature increases, as was expected. The Flory constant was found to be very close to the value found in literature. The scaling behaviour of the end-to-end distance however, shows some fluctuation for longer polymer lengths. This is most likely due to unpredictable population size of the ensembles using PERM. These fluctuations were present in almost all the simulations. Attempts to correct the unpredictability by tuning the coefficients for the upper and lower weight limits, have mostly failed.

References

[1] J. Thijssen. *Computational Physics*. Cambridge University Press, 2007.

[2] Hsiao-Ping Hsu and Peter Grassberger. A review of monte carlo simulations of polymers with perm. *Journal of Statistical Physics*, 144(3):597–637, 2011.

[3] Marshall N Rosenbluth and Arianna W Rosenbluth. Monte carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, 23(2):356–359, 1955.

[4] Jannis Batoulis and Kurt Kremer. Statistical properties of biased sampling methods for long polymer chains. *Journal of Physics A: Mathematical and General*, 21(1):127, 1988.

A. Population sizes

Figure 5 show the same data as was shown in Figure 2 with the population sizes plotted as well. For long polymers the population fluctuates significantly and this is most likely influences the found values for $\langle r^2 \rangle$.

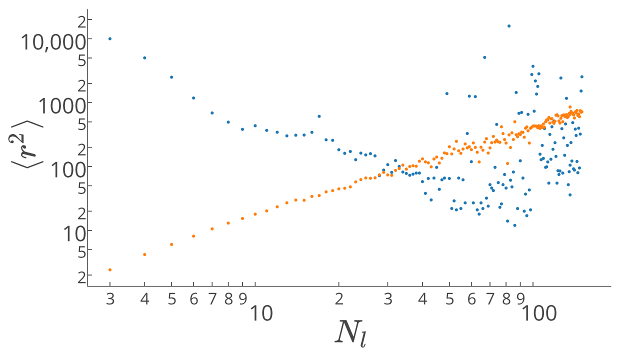

Figure 5: *End to end distance $\langle r^2 \rangle$ as function of polymer length N_l, with corresponding population sizes.*

B. Bending energy

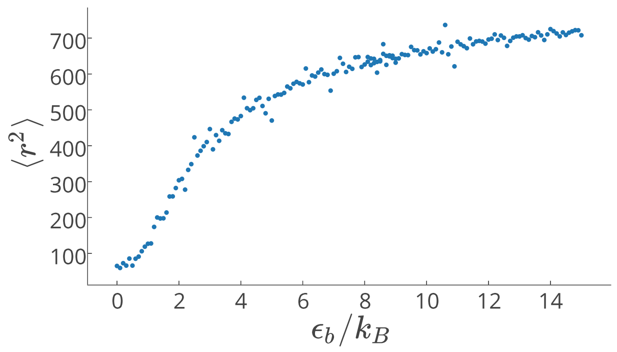

Figure 6