V.T.Koperberg

Monte Carlo simulation of self avoiding random walks

Computational Physics

Teacher: Prof. dr. G. T. Barkema

Date: February 6, 2016

Universiteit Leiden

1. Introduction

Self avoiding random walks (SARWs) have become a fixed topic of study in the discipline of computational physics since Paul Flory first used SARWs to model the behaviour of polymers in 1953 [2]. In this report we explain how we measured the growth exponent ν for SARWs on both a 2-dimensional and a 3-dimensional square lattice. We used a Monte Carlo Markov Chain (MCMC) approach to simulate the SARWs. We implemented this method using three different algorithms: the local move algorithm, the slithering snake algorithm and the pivot algorithm.

2. Self avoiding random walks

A self avoiding walk is a path on a lattice that never returns to the same point and thus does not intersect itself. More formally, a self avoiding random walk S of length N on a lattice L is an element of the path space

$$\Omega_N = \{S = (S_n)_{n=0}^N \in L^{N+1} : S_{n+1} \sim S_n,\ S_n \neq S_m,\ \forall 0 \leq n < m \leq N\},$$

where $S_{n+1} \sim S_n$ denotes that S_{n+1} is a nearest neighbour of S_n. This path space is equipped with the uniform distribution.

These SARWs are often used to model the shapes of polymers in highly diluted solutions, because the configurations of these polymers seem to display random behaviour. The SARW length corresponds to the number of bonds between monomers, thus a SARW of length N models a polymer consisting of $N+1$ monomers. For each $n \in \{0, 1, \ldots, N\}$ the position of the nth monomer is given by S_n. The condition that a SARW cannot intersect itself is imposed, since two monomers cannot occupy the same space.

The two quantities of interest that we are concerned with are the *mean squared end-to-end length* defined as

$$\left\langle r_e^2 \right\rangle = \left\langle \|S_N - S_0\|^2 \right\rangle$$

and the *mean radius of gyration*, given by

$$\left\langle r_g^2 \right\rangle = \left\langle \frac{1}{N+1} \sum_{n=0}^{N} \left\| S_n - \frac{1}{N+1} \sum_{m=0}^{N} S_m \right\|^2 \right\rangle.$$

Here $\|\cdot\|$ is the Euclidian norm on the lattice and $\langle \cdot \rangle$ denotes averaging over all states in Ω_N. In particular we are interested in the limit behaviour of long polymers. The mean squared end-to-end length scales as

$$\left\langle r_e^2 \right\rangle \sim N^{2\nu},$$

where the exponent ν is called the *growth exponent*. The growth exponent has been analytically determined to be $\nu_{d=2} = \frac{3}{4}$ on all regular 2-dimensional lattices. For 3-dimensional lattices the growth exponent is not known exactly, but numerical simulations give the value $\nu_{d=3} \approx 0.588$ [1]. Since the growth exponents are independent of the lattice type in a certain dimension, for this report we only considered the 2 and 3-dimensional square lattices, $L = \mathbb{Z}^2$ and $L = \mathbb{Z}^3$.

3. The algorithms

In this section we will give a short description of the three different algorithms that we used. In the following a basic understanding of MCMC methods is assumed. For additional information about MCMC methods in general, and explanations of the detailed balance and ergodicity condition, we refer to [3].

3.1. **Local move.** The first MCMC method we used to sample the path spaces was the *local move* algorithm. This algorithm constructs a Markov chain with state space Ω_N. Let $S^t \in \Omega_N$ be the state at time t, then the state at time $t+1$ is constructed as follows:

(1) Select a random monomer $i \in \{0, 1, 2, \ldots, N\}$.
(2) Determine the set of lattice sites $\mathcal{N}_i$ such that $\tilde{S} \in L^{N+1}$ given by $\tilde{S}_n = S^t_n$ for all $n \neq i$ and $\tilde{S}_i \in \mathcal{N}_i$ is a SARW.
(3) If $\mathcal{N}_i = \{S^t_i\}$, then take $S^{t+1} = S^t$. Else, choose a random lattice site $\tilde{S}_i \in \mathcal{N}_i \setminus \{S^t_i\}$ and let the state at time $t+1$ be given by $S^{t+1}_n = S^t_n$ for all $n \neq i$ and $S^{t+1}_i = \tilde{S}_i$.

A visualisation of this algorithm is given in figure 1. The Markov chain generated by this algorithm does indeed satisfy the detailed balance condition. However, some states cannot be reached from another state, see for example figure 2. This violates the required ergodicity property. This is solved by limiting the state space of the Markov chain to the communicating class of states that can be reached from the initial state S^0. This might slightly bias the measurements. However, for a reasonably chosen initial state, it is assumed that the states not in this class comprise such a small portion of Ω_N that this bias is negligible.

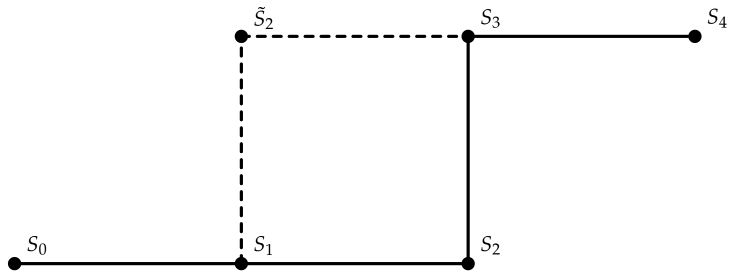


Figure 1. The continuous line depicts the state S^t at time t. During the first algorithm step, site S_2 was randomly chosen. The set of possible sites for this monomer consists of $\mathcal{N}_2 = \{S_2, \tilde{S}_2\}$. During the third step, the only choice for $\tilde{S}^t_2$ is $\tilde{S}_2$. The state S^{t+1} at time $t+1$ will therefore be given by $S^{t+1}_0 = S_0$, $S^{t+1}_1 = S_1$, $S^{t+1}_2 = \tilde{S}_2$, $S^{t+1}_3 = S_3$ and $S^{t+1}_4 = S_4$. Note that if monomer S_1 was chosen during the first step, there are no possible alternative sites and $S^{t+1} = S^t$.

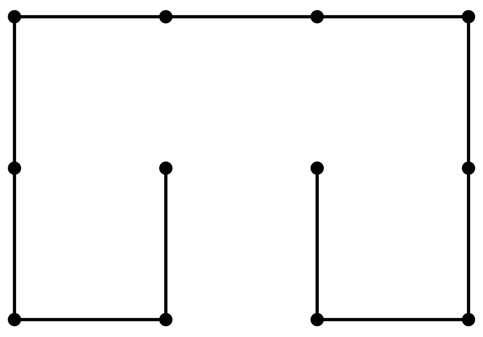


Figure 2. With the local move algorithm it is not possible to reach this state from any other state. [1]

3.2. **Slithering snake.** The second algorithm used was the *slithering snake* algorithm. With this MCMC method the next state S^{t+1} is constructed from the current state as follows:

(1) Remove the monomer S_0.
(2) Choose a random lattice site that is a nearest neighbour of S_N and that is not occupied by S_{N-1}.
(3) If the chosen site is unoccupied, then add a new monomer to that site and connect it to S_N. Relabel all monomers such that the neighbour monomer of the removed monomer is now S_0, the newly added monomer is S_N and for each $1 \leq i \leq N-1$, the ith monomer neighbours the $i-1$th and the $i+1$th monomers.
    Else, if the chosen site was already occupied, place the removed monomer S_0 back in its original position. Then relabel all monomers by interchanging S_0 and S_N and such that for each $1 \leq i \leq N-1$, the ith monomer neighbours the $i-1$th and the $i+1$th monomers.

Like the local move algorithm, this algorithm also violates the ergodicity property. Not all states can be reached from other states. This is again solved by restricting the state space to the states that can be reached. Moreover, this algorithm also doesn't satisfy the detailed balance condition. However, the lack of detailed balance does not cause any bias [4].

3.3. **Pivot.** Our final MCMC algorithm is the pivot algorithm. With the pivot algorithm, not only the lattice sites of each monomer, but also the angles at each monomer site are recorded. Using this we can generate a new state in our Markov chain in the following manner:

(1) Select a random monomer $i \in \{1, 2, \ldots, N-1\}$.
(2) Choose a random lattice site s that neighbours S_i, but is not occupied by S_{i+1}.
(3) Generate a candidate state $\tilde{S}$ by taking $\tilde{S}_j = S_j$ for all $j \leq i$. Take $\tilde{S}_{i+1} = s$ and for each $j > i+1$ take S'_j such that the angle at the jth monomer remains the same.
(4) If this candidate state is self avoiding, take $S^{t+1} = \tilde{S}$. Else, keep the current state $S^{t+1} = S^t$.

An example pivot step is given in figure 3. The pivot algorithm is the only of the three used algorithms that satisfies both the detailed balance and the ergodicity requirements.

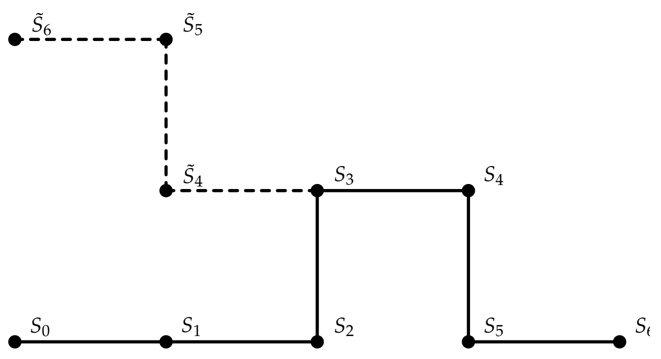


FIGURE 3. First site S_3 is chosen as our pivot point. Then $\tilde{S}_4$ is selected as the new location for S_4. The positions of $\tilde{S}_5$ and $\tilde{S}_6$ are chosen in such a way that the angels are preserved. This candidate state is self avoiding, thus it will be accepted as the new state.

4. Method and results

4.1. **Two-dimensional lattice.** Using these algorithms we sampled Ω_N on the 2- dimensional square lattice for different values of N. Since we were primarily interested in the growth of the end-to-end length for long polymers, we tried to take N as large as possible. However, since the local move algorithm and the slithering snake algorithm used quite a bit more computation time than the pivot algorithm, we were unable to sample the same polymer lengths with all three algorithms. The different polymer lengths that were sampled with each algorithm are given in table 1.

Algorithm	Sampled polymer lengths
Local move	$N = 10, 20, 30, \dots, 70$
Slithering snake	$N = 10, 20, 30, \dots, 150$
Pivot	$N = 10, 20, 30, \dots, 150, 200, 250, \dots, 950, 960, 970, 980, 990, 1000$

Table 1. Different sample spaces that were used with each algorithm

As our sample size we took $M = 100,000$ for each sample space Ω_N. Our initial state S^0 was chosen on the positive x-axis, $S^0_n = (n, 0)$ for all $0 \leq n \leq N$.

4.1.1. *Thermalisation.* Thermalisation was done by measuring the number of time-steps required for two different events to occur: $S^t_N = (x^t_N, y^t_N)$ is located left of the y-axis, i.e. $x^t_N < 0$; and the number of vertical steps equals the number of horizontal steps, i.e. $|\{n : x^t_{n+1} - x^t_n = 1\}| = \frac{N}{2}$. Note that for the second criterion, we assume that N is even. Since we only sampled Ω_N for even N, this is an allowable assumption. As our thermalisation time we took twice the time required for both events to have occurred. This was done to prevent that our sampling began in a state in which one of these events occurred. After thermalisation we took a sample after every sweep, where a sweep denotes N time-steps. For each sample we calculated the squared end-to-end length and the gyration radius.

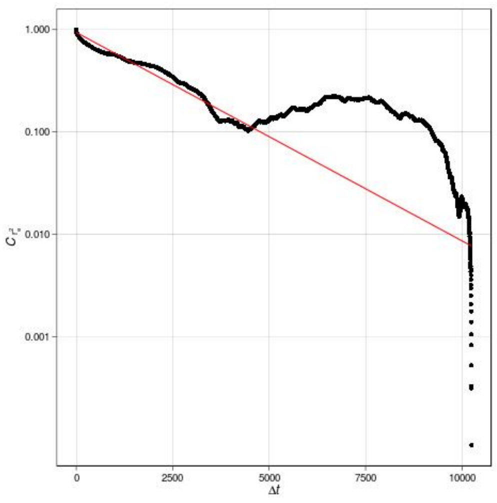
4.1.2. *Correlation time.* To determine how many of our 100,000 measurements were uncorrelated, we calculated the sample autocorrelation function $C_{r_e^2}(\Delta t)$ of the squared end-to-end length. The sample autocorrelation function is given by

$$C_{r_e^2}(\Delta t) = \frac{\left\langle r_e^2(S^{t+\Delta t}) - \overline{r_e^2} \right\rangle \left\langle r_e^2(S^t) - \overline{r_e^2} \right\rangle}{\left\langle \left(r_e^2(S^t) - \overline{r_e^2} \right)^2 \right\rangle}.$$

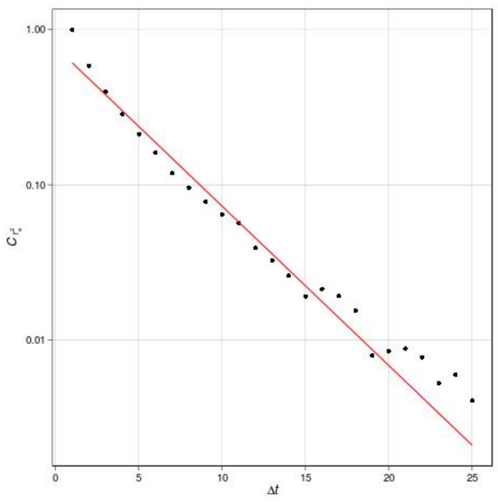
Here $\overline{r_e^2}$ denotes the sample average of the squared end-to-end length

$$\overline{r_e^2} = \frac{1}{M} \sum_{t=0}^{M-1} r_e^2(S^t).$$

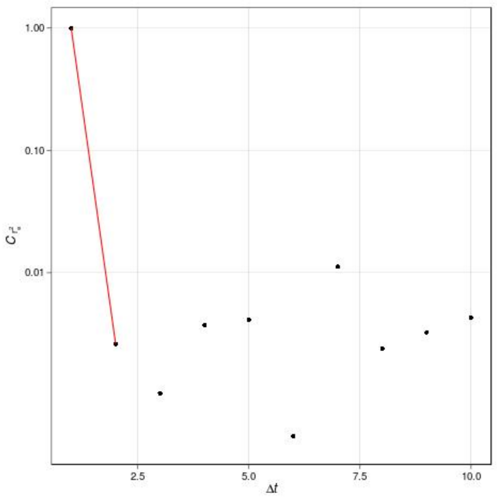
We assumed that the logarithm of the autocorrelation behaves like $\frac{-\Delta t}{\tau}$, where τ is the correlation time [3]. We computed the correlation time τ by fitting a line through the first part of the graph of $\log(C_{r_e^2})$ using the least square method. The correlation is then given by minus the reciprocal of the slope of the fit. For each algorithm we used the maximum measured polymer length. This was done since the correlation time increases as N increases and we were interested in the minimum number of uncorrelated samples for each polymer length and thus the maximal correlation time. The correlation times for each algorithm can be found in table 2. Two samples were said to be uncorrelated if at least 2τ sweeps were performed between each measurement.



(A) *Autocorrelation for the local move algorithm for* $N = 70$, *with fit through the first 4500 measurements.*



(B) *Autocorrelation for the slithering snake algorithm for* $N = 150$, *with fit through the first 19 measurements.*



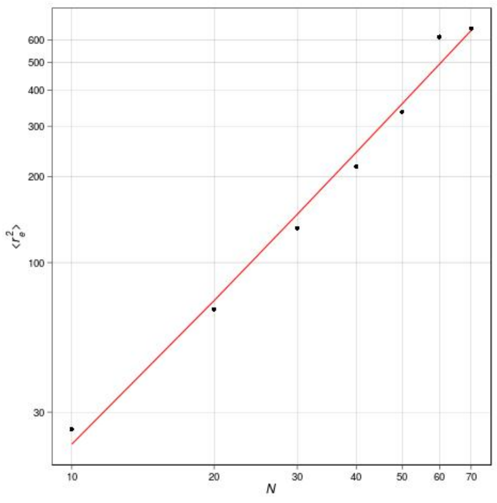
(C) *Autocorrelation for the pivot algorithm for* $N = 1000$, *with fit through the first 2 measurements.*

FIGURE 4. *Autocorrelation function of the squared end-to-end length for each algorithm on a logarithmic scale. The red lines are linear fits through the first linear part of each graph. Time is measured in sweeps.*

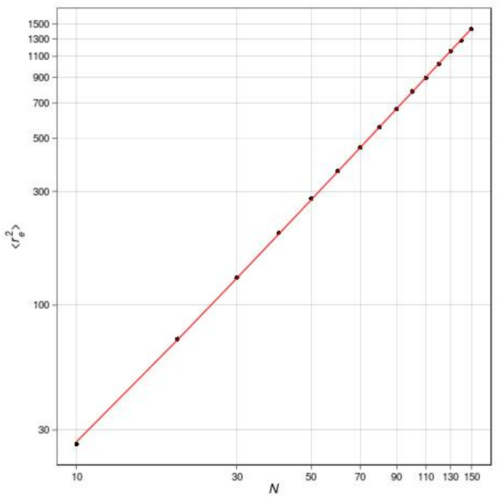
Algorithm	Correlation time	Number of uncorrelated measurements
Local move	2133.7	23
Slithering snake	4.2	10,000
Pivot	0.2	100,000

TABLE 2. Correlation time and number of uncorrelated measurements for each algorithm.

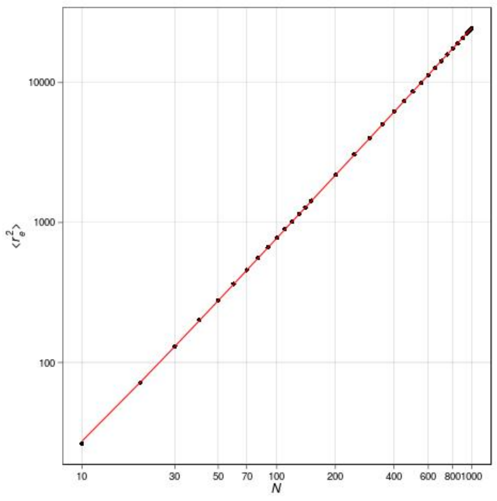
4.1.3. *Growth exponent.* The growth exponent ν was measured in each of the three simulations using the above mentioned algorithms. This was done by plotting the sample mean squared end-to-end-length against the polymer length N on a log-log scale. We could then fit a straight line through the data with the method of least squares. The plots of the sample mean squared end-to-end-length with linear fits are depicted in figure 5. The growth exponent ν could then be determined, since the slope of this fit equals 2ν. Our estimates of the growth exponent can be found in table 3.



(A) *The local move algorithm.*



(B) *The slithering snake algorithm.*



(C) *The pivot algorithm.*

FIGURE 5. *Average squared end-to-end length measured with each algorithm on a log-log scale. The red line is a linear fit.*

We can see that the estimate for ν as given by the local move algorithm, $\nu \approx 0.893$, does not correspond with the known value of $\nu = \frac{3}{4}$. This is to be expected, since we have only measured polymers up to length $N = 70$ with this algorithm and only used 23 uncorrelated measurements. Therefore, a large inaccuracy should come as no surprise. Both of the other two algorithms gave quite accurate estimates, but the pivot algorithm proved superior.

Algorithm	Growth exponent
Local move	$\nu \approx 0.893$
Slithering snake	$\nu \approx 0.760$
Pivot	$\nu \approx 0.753$

TABLE 3. Values for the growth exponent found with each algorithm.

4.1.4. *Gyration radius.* For each sampled polymer we also measured the radius of gyration. As expected the gyration radius turned out to scale similar to the squared end-to-end length [1]. As can be seen in figure 6, the ratio $\frac{\langle r_g^2 \rangle}{\langle r_e^2 \rangle}$ goes to a constant K_{ratio} as N grows large. We found $K_{\text{ratio}} \approx 0.14$.

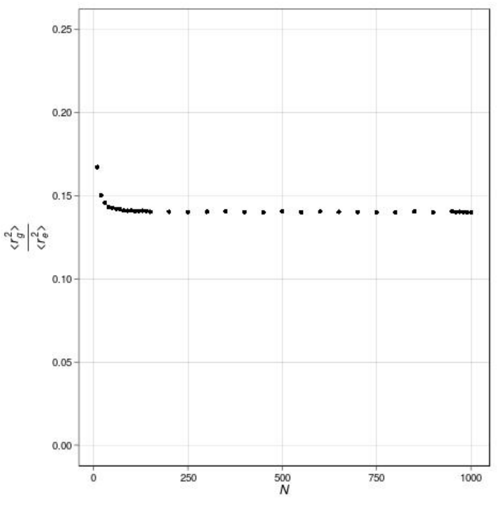


FIGURE 6. *Ratio between the average gyration radius and the average squared end-to-end length, measured with the pivot algorithm. The ratio approaches a constant $K_{ratio} \approx 0.14$.*

4.2. **Three-dimensional lattice.** Since the pivot algorithm turned out to be the most accurate, 3-dimensional polymers were only sampled using the pivot algorithm. The different polymer lengths that we used were $N = 10, 20, 30, \ldots, 120$. Again, we had a sample size of $T = 100.000$. Also our initial state and thermalisation criterion remained identical to the 2-dimensional case. After computing the autocorrelation function for $N = 120$, we found a correlation time of $\tau = 0.24$. This meant that our data comprised 100.000 uncorrelated measurements.

4.2.1. *Three-dimensional growth exponent.* For each length N we computed the average squared end-to-end length. We then plotted these averages on a log-log scale. Fitting a straight line through this graph gave the plot that can be seen in figure 7. Using the slope of this fit to estimate the growth exponent gave the value $\nu \approx 0.619$. This is still quite far from the previously found estimate value that was mentioned earlier. This can be explained by the lack of long polymer samples, since the maximal polymer length that we used was 120.

4.2.2. *Gyration radius.* In the 3-dimensional case, we also measured the gyration radius of each sample. The gyration radius again turned out to be a constant multiple of the squared end-to-end length. However, in the 3-dimensional case this constant was equal to $K_{\text{ratio}} \approx 0.16$, as can be seen in figure 8.

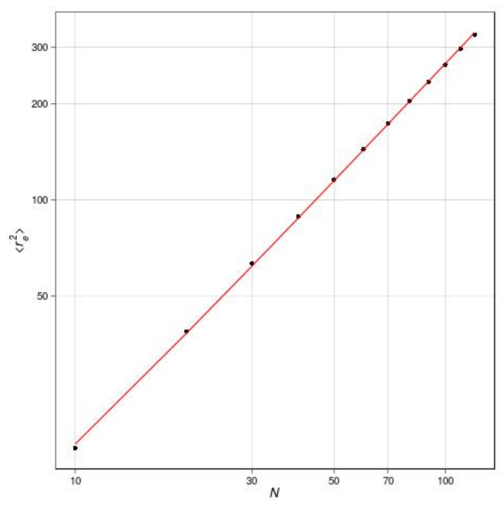


Figure 7. *Average squared end-to-end length on a 3-dimensional lattice on a log-log scale. The measurements were done with the pivot algorithm. The red line is a linear fit.*

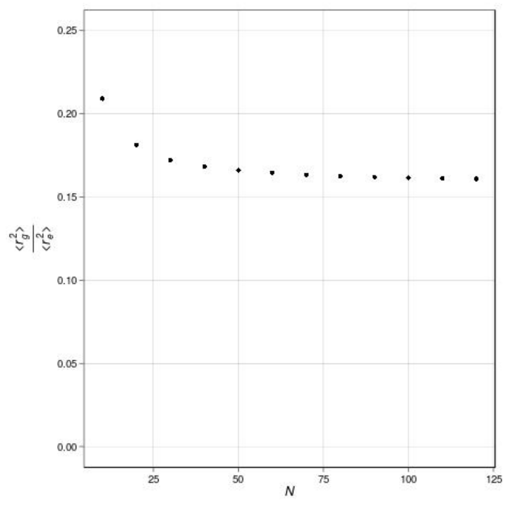


Figure 8. *Ratio between the average gyration radius and the average squared end-to-end length, measured with the pivot algorithm. The ratio approaches a constant $K_{ratio} \approx 0.16$.*

5. Conclusion

Finding accurate estimates for the growth exponent of SARWs proved to be rather difficult. Only the pivot algorithm seemed to give a reasonable estimate in the 2-dimensional case, $\nu_{d=2} \approx 0.753$. For the 3-dimensional case even the pivot algorithm was unable to find an estimate that corresponded with previous results. The pivot algorithm gave the estimate $\nu_{d=3} \approx 0.619$.

6. Discussion

One interesting observation was the change in the ratio between the gyration radius and the squared end-to-end length when going from the two-dimensional to the 3-dimensional square lattice. For the 2-dimensional lattice we found the ratio $K_{\text{ratio},\, d=2} \approx 0.14$, where in the 3-dimensional case we found $K_{\text{ratio},\, d=3} \approx 0.16$.

References

[1] GT Barkema. Monte carlo simulation of lattice polymer models. Computational Physics course handout.

[2] Paul J Flory. Principles of polymer chemistry. 1953.

[3] MEJ Newman and GT Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press: New York, USA, 1999.

[4] Frederick T Wall and Frederic Mandel. Macromolecular dimensions obtained by an efficient monte carlo method without sample attrition. *The Journal of Chemical Physics*, 63(11):4592–4595, 1975.