POLYMERS

César Beleño, Kaven Yau
Computational Physics Project, Universität Bonn, WS 10/11

ABSTRACT

In this report we present the results of the simulation of polymers in the athermal limit; that is, the polymer chain is immersed in a good solvent. For that purpose static and dynamic random walk generation methods were considered. Within the static methods we implemented the simple and biased sampling; while among the dynamic methods, we used reptation, kink-jump and pivot algorithm. For each case, the algorithm efficiency was analyzed, the end-to-end distance was obtained, and a value for the fractal dimension was obtained for each case. A description of each algorithm is also included.

1 Introduction

A polymer is a macromolecule composed of repeating structural units (monomers), typically connected by covalent chemical bonds. It can be modeled as a very large chain of N monomers, with N being of the order of 10^3 - 10^5. Their typical length ranges from 10^{-5} to 10^{-2} cm, although they are not wider than 10^{-7} cm. They can be found in a range of materials, from textile fibers, plastics, rubber, and cellulose, to the proteins in living organisms[1].

Although the chemical properties of the polymers can be described well by chemical formulas, their physical properties are strongly related to their geometry. A characteristic example is the distance between one end and the other end of the polymer, which characterizes the behavior of a polymer submerged in a chemical solution. From the literature, it is known that for a diluted solution of polymer chains the dependence of this quantity to the number of monomers in the chain is given by eq.1:

$$\langle R_N^2 \rangle = aN^{2\nu} \tag{1}$$

where the proportionality constant a depends on the structure and on the solvent used in the chemical solution. For two dimensions (2d) the value for the parameter ν is 3/4, for three dimensions (3d) it is approximately 0.592 and for 4 dimensions (4d) it is 1/2 [2].

The numerical simulation in this study will focus on linear polymers in a liquid solvent. Since only a small fraction of the space containing the solution is actually occupied by the polymer chain, the remaining space is assumed to be filled with the solvent molecules. The interactions present in this kind of system can be classified as between monomers, between solvent molecules, and between solvent molecules and monomers. These interactions can be well described by the Lennard-Jones interaction, which is strongly repulsive at short distances and weakly attractive at longer distances. As the repulsive part dominates, there is a more simplified approach, known as excluded volume interaction, which basically states that two monomers cannot occupy the same spatial position[3].

2 Self-avoiding walks (SAW) and Polymer chains

Let's consider a d-dimensional lattice. A self-avoiding walk (SAW) is a sequence of distinct points in the lattice such that each point is a nearest neighbor of its predecessor. The SAW model is well-known for polymer molecules[7].

In a d-dimensional (hyper)cubic lattice subject to unitary steps constrain, there are $2d$ step choices available for each site. By adding the constrain that each step remembers locally the former step, so it is not allowed to step backwards, one choice is removed. Therefore, at any given site, there are $2d-1$ choices. The only exception is the first step, where all $2d$ choices are available[1].

[1] Usually the first step can be chosen arbitrarily due to symmetry considerations. Therefore, this exception is irrelevant.

Generally, we are interested in the average of certain quantities such as the end-to-end distance. For a given walk size, the average, the standard deviation, and the relative error are computed using standard statistics. In the case of end-to-end distance quantity, the relative error approaches a constant and therefore it can be used as a test for randomness in the algorithm.

The partition function that provides information about the efficiency of generation of chains of length N can be written as:

$$\langle Z(N)\rangle_{ss} = \frac{\text{number of accepted walks}}{\text{number of attempts}} \tag{2}$$

If each walk has a different weight W, the weight has to be taken into account. For the standard deviation, it is more convenient to use the following expression[2]:

$$\sigma_X = \sqrt{\langle X^2\rangle - \langle X\rangle^2} \tag{3}$$

And for the partition function[5]:

$$Z_{bs} = \frac{\sum_{\text{all walks}} W_{walk}}{\text{number of attempts}} = \langle W\rangle \tag{4}$$

In both the simple and biased sampling the partition function present the following form:

$$Z_N = C\exp(-\lambda N)N^{\gamma-1} \tag{5}$$

where C is a normalization constant, λ is a model dependent constant called the attrition constant, and γ is a critical exponent that is independent of the model. Another quantity that accounts for the fluctuation of the end-to-end distance es the relative variance given by:

$$\Delta R^2 = \frac{\sqrt{\langle R^4\rangle - \langle R^2\rangle^2}}{\langle R^2\rangle} \tag{6}$$

This quantity tends to a constant due to the lack of self-averaging of the end-to-end distance [4].

3 Model and Algorithms

Among all the existing polymer models, we will restrict ourselves to d-dimensional (hyper)cubic lattices. From here on, we will use the word lattice to refer to a (hyper)cubic lattice.

In our analysis, a flexible polymer chain in a periodic lattice is replaced by a randomly generated SAW. As there are many methods to randomly generate a SAW, the algorithms used in this study are listed, explained, and discussed in this section.

3.1 Static methods

Static methods generate a sequence of statistically independent samples. In other words, each iteration is independent of former iterations[7]. The static methods used in this study are the simple sampling and the Rosenbluth method.

3.1.1 Simple Sampling

The simple sampling is, as the name indicates, a quite uncomplicated method. The steps are generated randomly, with the only constrain that each step remembers locally the former step, so it is not allowed to step backwards. For a d-dimensional lattice, the algorithm is the following[4]:

[2]The weighted average formula is omitted, but it can be found in most basic statistic books.

i. Set the origin of the polymer. In our study, we set it at the coordinate system's origin.

ii. Generate the first step randomly or choose it arbitrarily.

iii. Choose randomly one of the $2d-1$ possible steps.

iv. If the step leads to self-intersection, reject the generated walk and start over again. This step is needed to ensure randomness.

v. If the step leads to an available site, add the step to the walk.

vi. If the walk reaches the wanted length, accept the walk. Else, go to step iii and repeat until the walk is either accepted or rejected.

vii. Repeat until the wanted walks amount is reached.

Advantages:

- It is easy to program in a computer.
- the method is intuitive.
- The algorithm has no bias and all possible configurations can be reached.

Disadvantages:

- The algorithm efficiency decreases very fast as the walk size increases. For very long polymer chains, the low efficiency becomes prohibitive.

3.2 Biased sampling

The biased sampling [3] is an improved simple sampling method. The main problem of the simple sampling is that at large walk sizes and small dimensions the rejection rate is very high. In order to reduce the rejection rate, the steps that lead to self-intersection are excluded in step iii [5].

To correct for the bias that is introduced by the exclusion of those steps, a weight W is assigned to each walk, based on how many steps were excluded. The weight for a walk size N, in a n-dimensional lattice is given by:

$$W = \prod_{i=1}^{N} w_i \tag{7}$$

where w_i are the partial weights for each step and are defined as:

$$w_0 = 1 \tag{8}$$

$$w_i = \frac{l_i}{2d-1} \tag{9}$$

where l_i is the number of available steps from where the i-th step was chosen. Note that if there are no available steps, the weight becomes 0 and the walk is rejected. Summarizing, for a d-dimensional lattice, the Rosenbluth method algorithm can be implemented as follows [2]:

i. Set the origin of the polymer. In our study, we set it at the coordinate system's origin.

ii. Generate the first step randomly or choose it arbitrarily.

[3] Also called Rosenbluth method

iii. Find all possible steps that do not lead to self-intersection.

iv. If no steps are available, set the weight to 0, reject the generated walk, and start over again. Else, add the step to the walk and compute the partial weight using equation (9).

v. If the walk reaches the wanted length, accept the walk and compute the total weight using equation (7). Else, go to step iii and repeat until the walk is either accepted or rejected.

vi. Repeat until the wanted walks amount is reached.

Advantages:

- It offers a significant improvement to the efficiency compared to the simple sampling.
- The method is still intuitive and relatively easy to program.
- The algorithm has no bias and all possible configurations can be reached.

Disadvantages:

- Although the efficiency is better than the simple sampling efficiency, it still decreases very fast as the walk size increases. For very long polymer chains, the low efficiency becomes prohibitive.

3.3 Dynamic methods

Dynamic methods generate a sequence of correlated samples. In other words, each iteration depends on at least the last iteration[7]. The dynamic methods we used are the reptation, kink-jump and pivot algorithms. All these algorithms are Markov chains, that is, the new configuration depends only on the last configuration[8].

3.3.1 Reptation algorithm

This is one of the most efficient generation techniques and is applicable to any linear polymer chain model. The algorithm can be stated as follows[2, 6] [4]:

i. Generate or choose a n-steps SAW.

ii. Choose an end randomly and remove this site.

iii. Choose randomly one of the $2d - 1$ possible steps on the other end.

iv. If the attempt leads to self-intersection, retain the previous configuration and count it as the new configuration.

v. If the attempt does not lead to self-intersection, count it as the new configuration.

vi. Go to step ii and repeat until the wanted walks amount is reached.

Advantages:

- The method is very efficient.
- Each iteration requires little computational work.

[4] Other algorithms for the reptation method exist

Disadvantages:

- The method is not intuitive.
- The algorithm is somewhat sensitive to the initial condition.
- The results are biased: There are some configurations that we will never obtain, such as double ended "cul-de-sac" configurations (figure). In this way, the reptation method includes a small bias that can be neglected with sufficient statistics.

3.3.2 Kink-Jump algorithm

This model is used to simulate the dynamics of the collisions between the polymer chain and the solvent molecules. For that purpose, the chain is assumed to be composed of beads connected by bonds, whose motion is restricted to a lattice. The algorithm can be described as follows[2]:

i. Generate or choose a n-steps SAW.

ii. Select a bead (occupied site) at random on the walk.

iii. If the bead is not the head or the tail, then the bead can move to only one site without breaking the walk continuity. If this site is not occupied, move the selected bead there. Otherwise, keep the current walk.

iv. If the selected bead is the head or the tail, move it to one of the two (maximum) possible unoccupied sites so that the bond to which it is connected changes its orientation by ± 90. If no sites are available, keep the current walk.

v. Count the walk.

vi. Go to step ii and repeat until the wanted walks amount is reached.

Advantages:

- Each iteration requires little computational work.
- It simulates the dynamics of the collisions between the polymer chain and molecules.

Disadvantages:

- The results are biased: there are some configurations that we will never obtain, such as double ended "cul-de-sac" configurations (figure).
- The algorithm is highly sensitive to the initial condition. This means that it requires many iterations before given a configuration reaches an essentially new configuration.
- the algorithm is biased, favouring longer chains.

3.3.3 Pivot algorithm

The pivot algorithm generates SAWs from an initial SAW. At each iteration, the SAW from the last iteration is modified by randomly choosing a pivot bead and randomly applying one lattice symmetry operation on one of the two sub-walks. The new SAWs will always have the same number of steps as the initial SAW. The algorithm can be sketched as follows[7]:

i. Generate or choose a n-steps SAW.

ii. Randomly choose a pivot from the beads on the walk. The walk can be then divided in two sub-walks joined by the pivot.

iii. Randomly choose one of the two sub-walks.

iv. Randomly choose a lattice symmetry operation and apply it to the chosen sub-walk.

v. If the attempt violates the self-intersection constraint, retain the previous configuration and count it as the new configuration.

vi. If the attempt does not break the self-intersection constrain, count it as the new configuration.

vii. Go to step ii and repeat until the wanted walks amount is reached.

Advantages:

- Fast convergence: after a few iterations, a configuration can be considered to have reached an essentially new configuration.
- The algorithm is unbiased and all possible configurations can be reached.

Disadvantages:

- Each iteration requires more computational work than the other two dynamical methods.
- Its implementation in a computer program is difficult for high dimensions: the number of symmetries increases very fast and the symmetry operations are difficult to compute.
- For long polymer chains, the efficiency is very low. But this is compensated with a fast convergence.

4 Numerical Results and Analysis

4.1 Static Algorithms

4.1.1 Simple Sampling

We begin our analysis by computing the simple sampling walk termination size distribution for different dimensions. The results are depicted in figure(1).

This study gives us an approximate cut-off point for the walk length. For values higher than this point, the termination rate becomes higher than 99% (Arbitrary criteria). In this fashion, we can establish a certain walk size limit depending on the lattice dimension. In 2 dimensions the point is reached for $N > 45$, for $N > 81$ in three dimensions and for $N > 151$ in four dimensions.

This constitutes a disadvantage of the simple sampling algorithm in simulating longer chains, as real polymer chains have around 10^3 - 10^5 monomers.

Note that the chain cannot terminate with one or two steps without breaking the SAW's non-reversibility condition. In the 2d case, the chain has a high probability of breaking with 3 steps [5]. There are 8 ways in which this configuration could appear. The first step has a probability $1/(2d)$, while the second, third, and fourth steps have a probability $1/(2d-1)$. Thus the probability that the chain stops at N=3 is approximately 7.4% which is in perfect agreement with the value shown in the plot.

By studying the convergence of $< R_N^2 >$ as a function of the number of SAWs, we can determine the minimum amount of walks needed to get a stable mean value. This can be seen in figure 2, for fixed walk sizes of 40, 80 and 120 steps in 2d, 3d and 4d respectively.

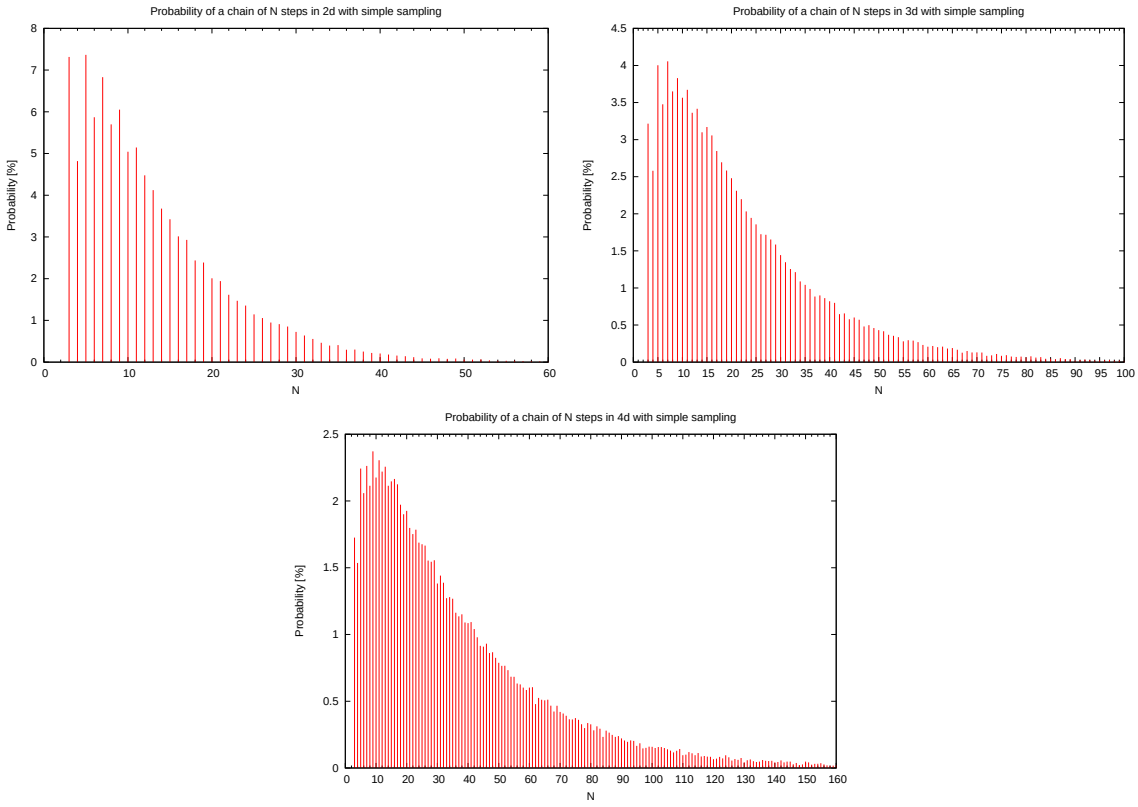

Figure 1: Probability distribution of lengths where the walks terminate for simple sampling: (above left) 2 dimensions, (above right) 3 dimensions and (below) 4 dimensions.

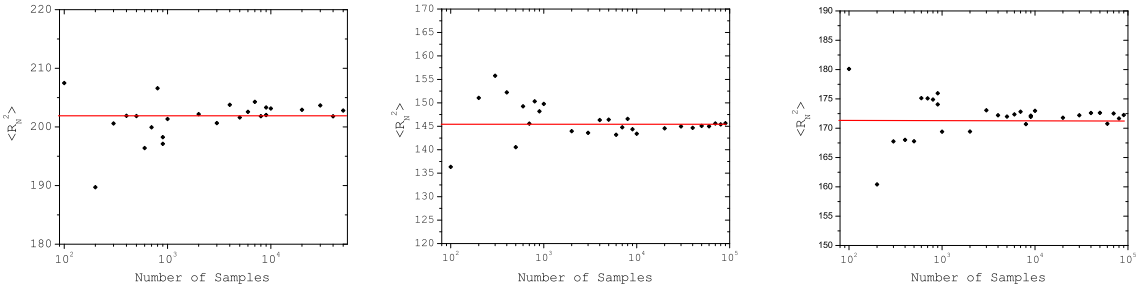

Figure 2: Convergence of $< R^2 >$ as a function of the number of samples for a fixed walk size: (left) two dimensions N=40, (center) three dimensions N=60, (right) four dimensions N= 100.

From these plots we can estimate that more than 2000 chains are needed to gather enough statistics.

The acceptance fraction gives us the ratio of successfully generated SAWs to the number of attempts made. This is a measure of the difficulty of producing long chains in the simple sampling method.

The efficiency decays exponentially ($E \approx \exp(-\lambda N)$) by increasing the number of steps, leading to the so called attrition problem. This can be seen in figure 3.

The value of the fit parameters (λ and γ) for the studied chains (see eq. 5) are summarized in

[5]This can be seen if the chain follows a square path

table1.

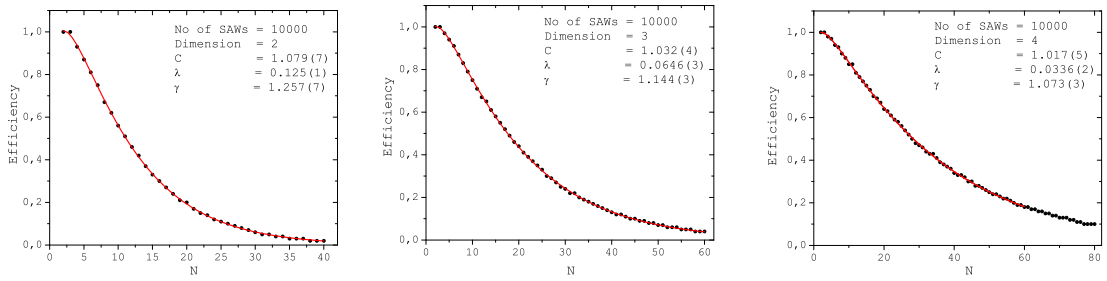

Figure 3: Efficiency of the simple sampling algorithm as a function of the number of steps for simple sampling: (left) two dimensions N=40, (center) three dimensions N=60, (right) four dimensions N= 80.

d	2	3	4
λ	0.125 ± 0.001	0.0646 ± 0.0003	0.0336 ± 0.0002
γ	1.257 ± 0.007	1.144 ± 0.003	1.073 ± 0.003

Table 1: Values of the attrition constant λ and for γ for different dimensions, a sample of 10000 chains was taken. The chains were composed of 40, 60 and 80 steps for 2d, 3d and 4d respectively .

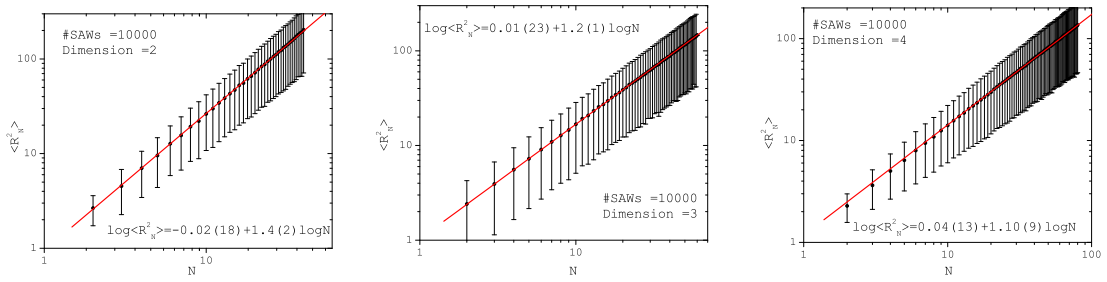

Figure 4: End-to-end distance as a function of the number of steps for simple sampling: (left) two dimensions N=40, (center) three dimensions N=60, (right) four dimensions N= 80.

One standard quantity used in such studies is the fractal dimension ν (see eq.1). It is given by the slope of $< R^2 >$ vs N in double logarithmic scale (Fig.4).

In table 2, we can see the fractal dimension values for different number of samples. They are quite close to the predicted values of 3/4, $0,59$ and 1/2 for 2d, 3d and 4d respectively.

In Fig.5 shows that the relative errors (see eq.6) do not present any dependency on the number of steps. They are rather constant, which is further proof that there is no bias.

4.1.2 Biased Sampling

In the biased sampling, despite the fact that the walk knows locally which steps are forbidden, there are still possibilities that the walk terminates due to a dead end.

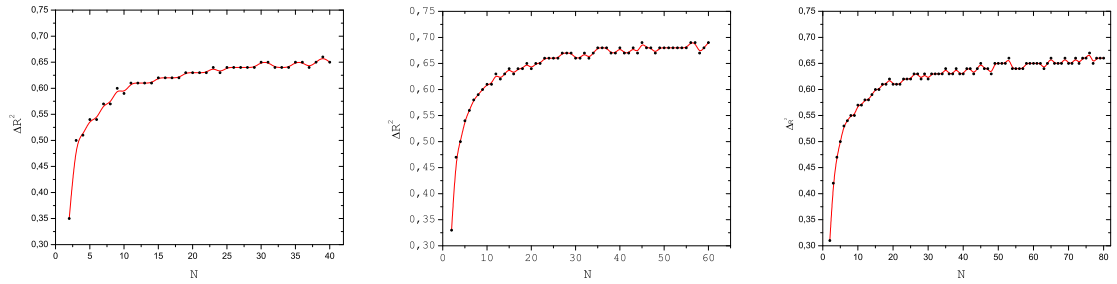

Figure 5: Mean square error of the end-to-end distance as a function of the number of steps for simple sampling: (left) two dimensions N=40, (center) three dimensions N=60, (right) four dimensions N= 80.

Number of Samples	2d	3d	4d
100	0.72 ± 0.08	0.61 ± 0.06	0.55 ± 0.05
2000	0.72 ± 0.08	0.60 ± 0.06	0.55 ± 0.05
5000	0.72 ± 0.08	0.60 ± 0.06	0.55 ± 0.04
10000	0.72 ± 0.07	0.60 ± 0.05	0.55 ± 0.04

Table 2: Values of ν for a different number of samples from the simple sampling algorithm. The walk size was 40, 60 and 80 steps for 2d, 3d and 4d respectively .

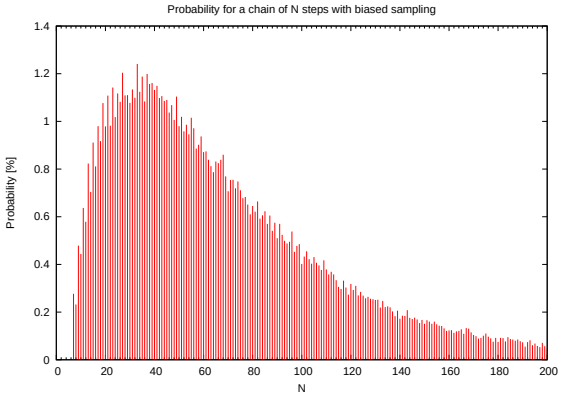

Figure 6: Probability distribution of length where the walks terminate for 2 dimensions in the biased sampling algorithm.

This probability is shown in fig. 6. This is only available for 2d lattices: at higher dimensions, the termination probability drops drastically as more possible steps become available per site. This in turn increases drastically the chain length, and therefore the computing time, making it prohibitive for 3d and 4d lattices. It can be observed from the plot that the range of the number of steps is almost 4 times bigger than its simple sampling counterpart, meaning that we can generate longer chains with this method.

After applying the fit for the efficiency (see Fig.7), we observe that the values for the attrition constant decrease. For instance, in the 2d case $\lambda = 0.0219 \pm 0.0001$, which is almost 6 times smaller than the attrition constant for simple sampling. Hence, the biased sampling method is superior to the

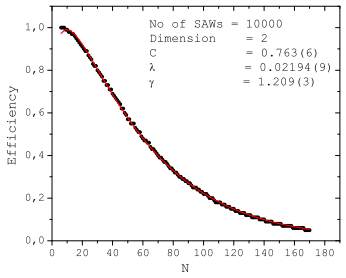

Figure 7: Efficiency as a function of the number of steps for biased sampling only for the 2d case.

simple sampling method when the generation of longer chains is required.

The probability of survival of a chain of N steps is given by $E = 0.763(6)\exp(-0.0219N)N^{0.209}$. For example, there is a probability of 22% to generate successfully a chain with 100 steps. On the other hand, for the 3d and 4d cases the acceptance fraction is very close to 1 for the studied range, and producing a fit was not possible.

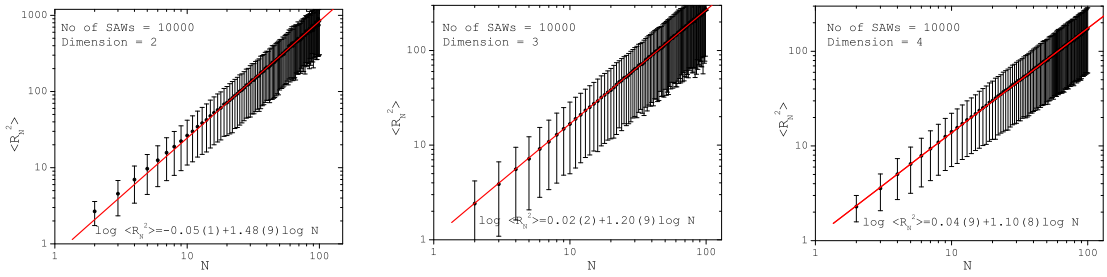

Figure 8: End-to-end distance as a function of the number of steps for biased sampling: (left) two dimensions N=40, (center) three dimensions N=60, (right) four dimensions N= 80.

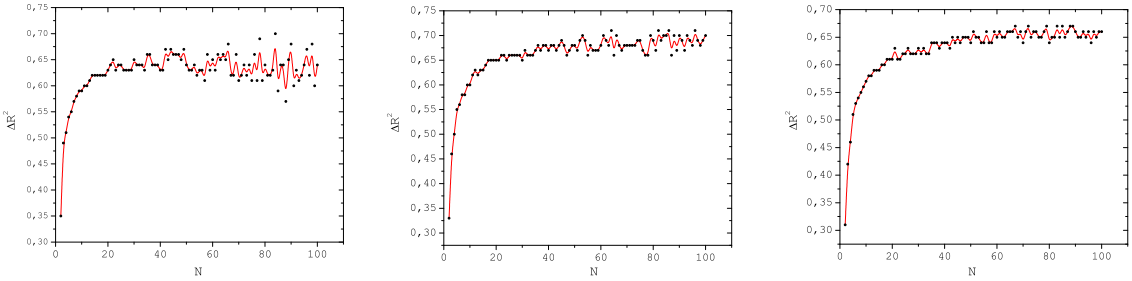

Figure 9: Mean square error of the End-to-end distance as a function of the number of steps for biased sampling with 10000 samples and N=100: (left) two dimensions, (center) three dimensions, (right) four dimensions.

In order to compare the results obtained through biased sampling with those obtained of simple sampling, we can first consider the end-to-end distance and the fractal dimension (ν) of the chain.

Before going further, note that the walk size range is larger in figure 8, compared to its simple

sampling counterpart. Using the values for walk size and dimension used in the simple sampling analysis. we fitted the value for ν. Additionally, we made the same analysis for 2d, 3d and 4d with a walk size of 100. The results of the fitting are arranged in table 3.

As observed, these results are in perfect agreement with the expected values. However, in the case of the errors in the end-to-end distance, we can observe fluctuations around certain values (see Figure 9).

Number of Samples	2d N=40	2d N=100	3d N=60	3d N=100	4d N=80	4d N=100
100	0.71 ± 0.07	0.73 ± 0.05	0.61 ± 0.06	0.60 ± 0.07	0.54 ± 0.05	0.54 ± 0.05
1000	0.72 ± 0.07	0.74 ± 0.05	0.60 ± 0.06	0.60 ± 0.05	0.54 ± 0.05	0.54 ± 0.04
10000	0.74 ± 0.04	0.74 ± 0.04	0.60 ± 0.05	0.60 ± 0.05	0.55 ± 0.04	0.55 ± 0.04

Table 3: Values of ν for different number of samples and walk size from the biased sampling algorithm.

4.2 Dynamic Algorithms

4.2.1 Reptation Method

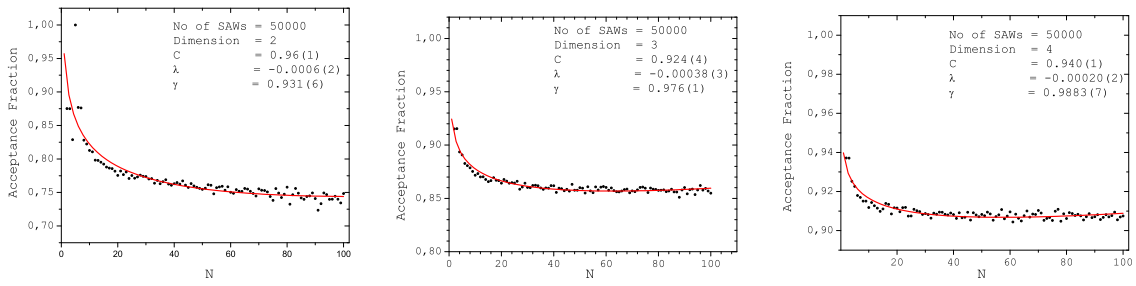

Figure 10: Acceptance fraction for the reptation algorithm with a fixed number of steps (N=100) and 50000 samples: (left) two dimensions, (center) three dimensions, (right) four dimensions.

One of the difficulties present in simple sampling was the attrition problem, that imposed certain limits in the generation of more steps for the chain. This fact is evident in the values of the attrition constant λ given in eq.5. For the case of reptation method this value is negative, hence the exponential decay component of the efficiency is not longer the dominant part (see table 4) and can be approximated as $E \approx N^{-p}$, with $p = 1 - \gamma$

By definition, the partition function for all the dynamic methods is 1, so we analyse the acceptance fraction instead.

Fig. 10 shows the dependency of the acceptance fraction with the walk size. We observe that, while attempting to produce chain sizes of 100 in 2d, 3d and 4d, the iterations produced new configurations 85%, 91% and 75% of the times.

Finally, it is worth noting that the value of the γ is close to one.

	2d	3d	4d
λ	-0.00069(6)	-0.00046(5)	-0.0009(1)
p	0.069(6)	0.024(1)	0.0117(7)

Table 4: Values of λ and the critical exponent p for the reptation method.

At first glance, it looks like there is no convergence of the value of the end-to-end distance with an increasing in the number of samples (see Fig. 11). This behaviour can be explained if we take into account that each iteration is strongly correlated with the former iteration. However, from the plots it can be seen that in order to get rid of this problem a very large statistics is needed, it means we need more than 20000 samples.

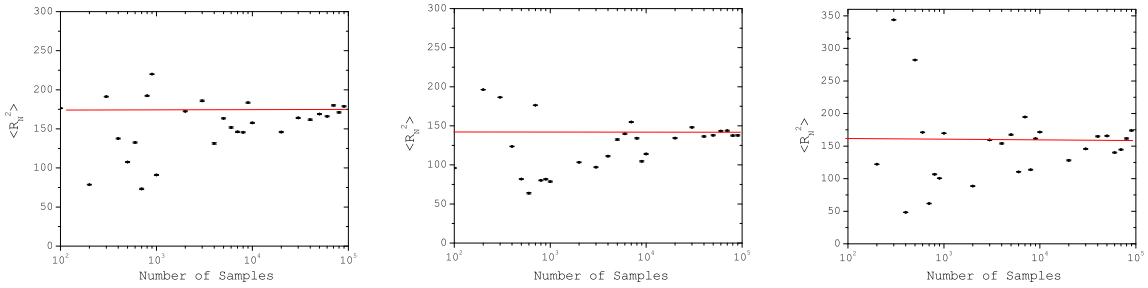

Figure 11: End-to-End distance as a function of the number of samples for the reptation algorithm for a chain of 100 steps: (left) two dimensions, (center) three dimensions, (right) four dimensions.

Taking into account the previous analysis, we can plot the end-to-end distance in order to obtain the fractal dimension of the chain ν (see Fig.18).

From the fit, we can see from table 5, that the values are in agreement with the expected ones.

Nevertheless, in the cases where the chain lies on a 2d lattice, for practical reasons it is advisable to use walk lengths less than 100 monomers ($N < 100$) in the simulation. This is due to the fact that the fractal dimension for two dimensions is outside the acceptable range for this quantity ($\nu = 3/4$) for walk lengths near 100. However, it is in perfect agreement with the expected value for a length of $N = 50$.

The relative error of this quantity shows fluctuations around some value (fig. 13), since this should tend to a constant we can say that the reptation algorithm is a biased method, since there some configurations that are favoured by this algorithm.

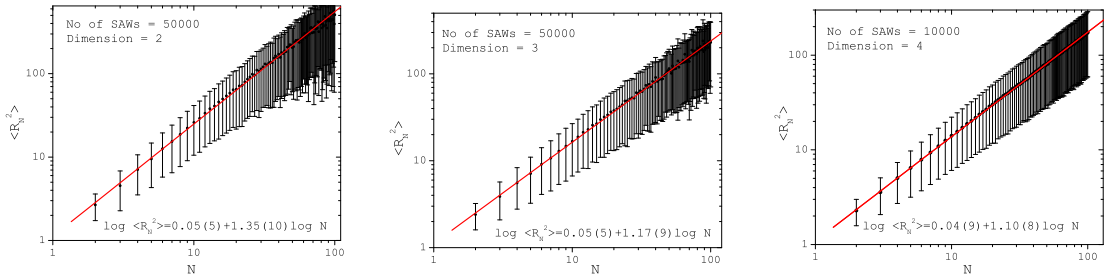

Figure 12: End-to-End distance for the reptation algorithm with 100 steps and 50000 chains: (left) two dimensions, (center) three dimensions, (right) four dimensions.

For shorter chains, for example a chain of 20 steps, the values of ν and p are arranged in table 6.

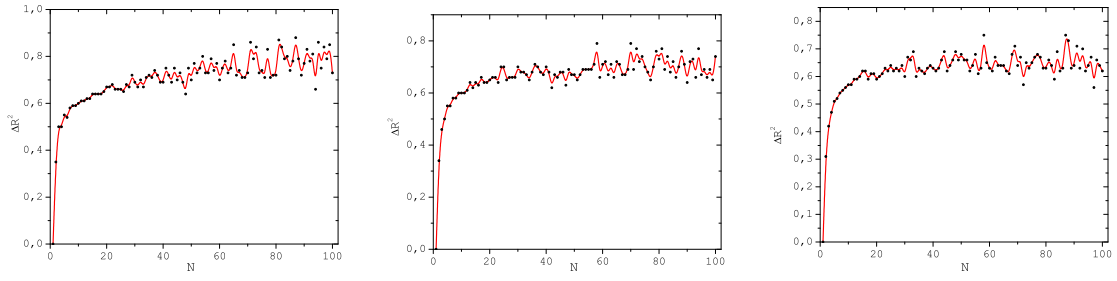

Figure 13: Mean square error of the end-to-end distance as a function of the number of steps for reptation algorithm with 100 steps and 50000 number of samples: (left) two dimensions, (center) three dimensions, (right) four dimensions.

	N=100		
Number of Samples	2d	3d	4d
50000	0.67 ± 0.05	0.58 ± 0.04	0.54 ± 0.04
10000	0.67 ± 0.05	0.58 ± 0.04	0.55 ± 0.04
1000	0.68 ± 0.04	0.62 ± 0.04	0.54 ± 0.03

Table 5: Values of ν for different number of samples and walk size from the biased sampling algorithm.

d	2	3	4
p	0.049 ± 0.004	0.025 ± 0.002	0.013 ± 0.002
ν	0.72 ± 0.12	0.61 ± 0.12	0.56 ± 0.10

Table 6: Values of the critical exponent p and of the fractal dimension for a chain of 2 steps in the reptation algorithm for different dimensions.

4.2.2 Kink-Jump Algorithm

In this algorithm, we only considered short chains due to the presence of strong fluctuations in the end-to-end distance. This method is very sensitive to the initial configuration of the chain. Therefore, it introduces a bias that is very difficult to eliminate even with many iterations. The acceptance fraction of the algorithm is displayed in fig. 14 and the critical exponents are arranged in table 7.

From the relative error ΔR^2, we still see some fluctuations in the end-to-end distance. This situation can be explained since the algorithm is highly biased and not ergodic and it is mainly used to simulate the time evolution of the whole chain in a perfect solvent. Then, it is an example of importance sampling. The values of the fractal dimension are close to the expected ones, although in the 2d case it is outside the errors bars.

d	2	3	4
p	0.256 ± 0.010	0.178 ± 0.005	0.130 ± 0.006
ν	0.705 ± 0.007	0.60 ± 0.10	0.56 ± 0.11

Table 7: Values of the critical exponent p and of the fractal dimension for the kink-jump algorithm for different dimensions.

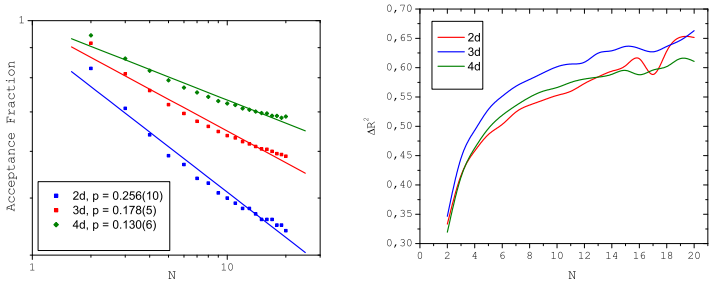

Figure 14: Acceptance fraction (left) and relative error (right) for the kink-jump algorithm with a fixed number of steps (N=20) and 10^6 iterations.

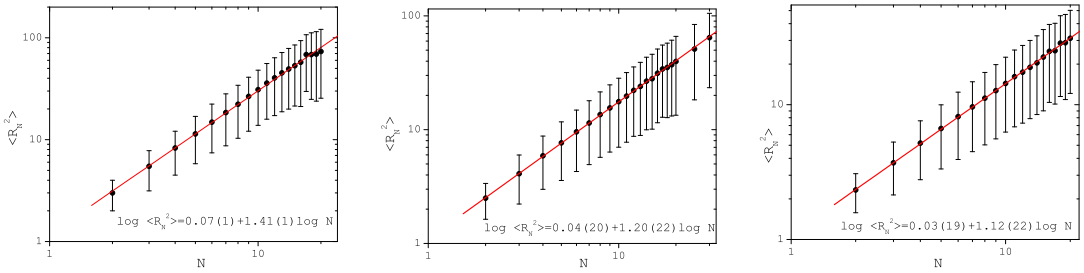

Figure 15: End-to-end distance for the kink-jump method: 2d (left), 3d (center) and 4d (right) for a chain of 20 steps and 10^6 iterations.

4.2.3 Pivot Algorithm

As in the other dynamical methods, in the pivot algorithm the fraction of successfully generated steps decrease approximately as $N^{\gamma-1}$, where $\gamma < 1$ is consigned in table 8 for the dimensions under consideration (see Fig. 16). Hence, for a chain of 100 steps, 47.9% of the attempts are successful in 2d, which is less than the fraction generated by reptation. However, unlike the previous methods, the pivot algorithm achieves convergence for $< R^2 >$ with a much smaller sample. This characteristic is shown in Fig. 17. Therefore, a sample of 10^4 chains should be enough to obtain precise values for the fractal dimension of the chains.

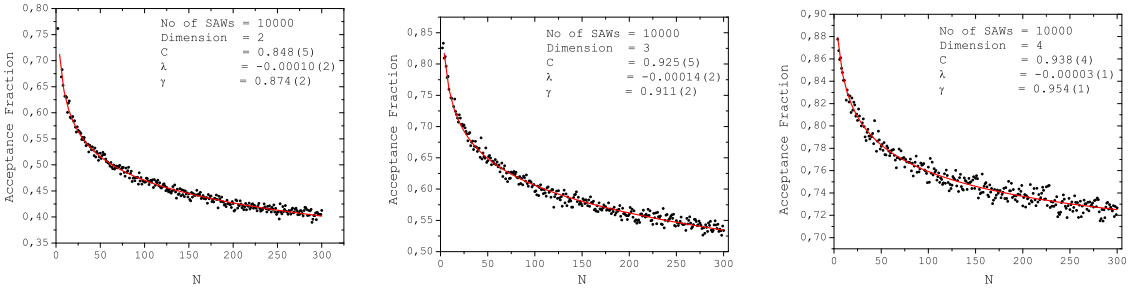

Figure 16: Acceptance fraction for the pivot algorithm with a fixed number of steps (N=300) and 10000 samples: (left) two dimensions, (center) three dimensions, (right) four dimensions.

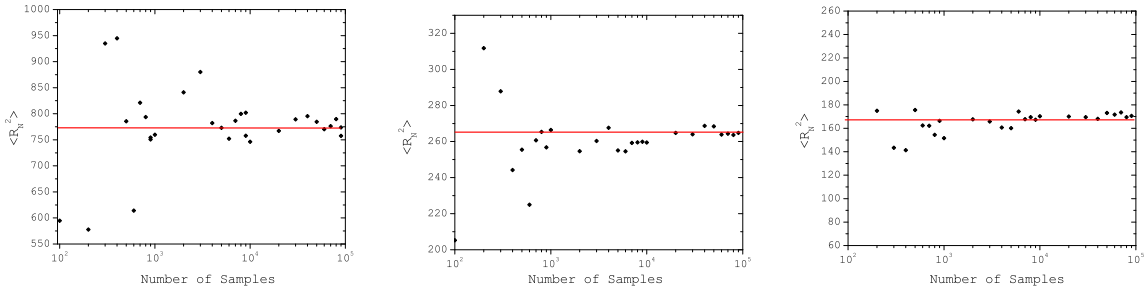

Figure 17: End-to-End distance as a function of the number of samples for the pivot algorithm: (left) two dimensions, (center) three dimensions, (right) four dimensions.

d	2	3	4
p	0.125 ± 0.001	0.0646 ± 0.0003	0.0336 ± 0.0002
ν	0.74 ± 0.03	0.60 ± 0.03	0.54 ± 0.02

Table 8: Values of the critical exponent γ and of the fractal dimension for the pivot algorithm for different dimensions. A sample of 10000 chains composed of 300 monomers was taken.

As was stated before, the pivot method shows a very quick convergence to the value of the end-to-end distance. The reason for this achievement is that the method requires much less successful symmetry applications (that is, the chain is accepted after a symmetry operation is applied) to achieve an essentially new configuration.

Practically, over all the range we used to compute the fractal dimension, the fluctuations in $< R^2 >$ are almost neglected (see Fig. 19), which imply that we can used this algorithm for simulating very long polymers chains with a reasonable statistics. This improvement is reflected in the computational time.

In both the reptation and kink jump algorithm a huge statistics is required to get the expected results; which implies and increasing in the cpu time as the number of step grows.

Finally, the algorithm has the advantage that it is ergodic, although its proof is not trivial.

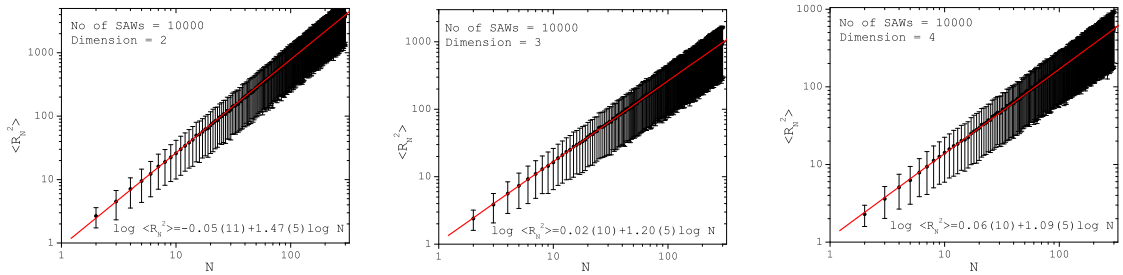

Figure 18: End-to-End distance for the pivot algorithm with 300 steps and 10000 chains: (left) two dimensions, (center) three dimensions, (right) four dimensions.

For shorter chains, for example a chain of 20 steps, the values of ν and p are arranged in table 9.

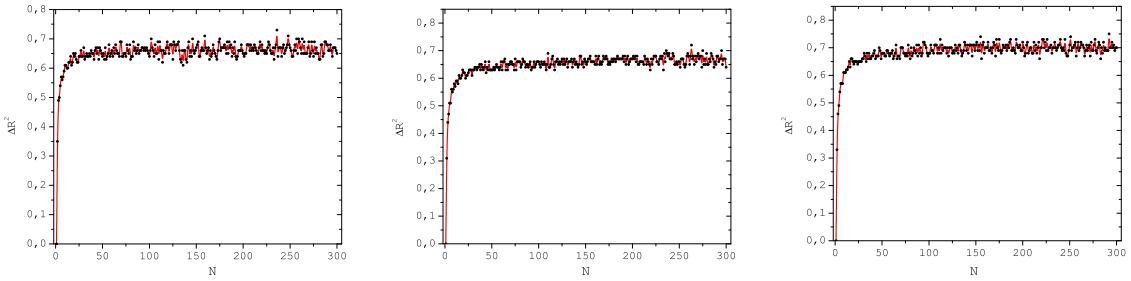

Figure 19: Relative variance of the end-to-end distance as a function of the number of steps for the pivot algorithm with 100 steps and 10000 number of samples: (left) two dimensions, (center) three dimensions, (right) four dimensions.

d	2	3	4
p	0.118 ± 0.003	0.077 ± 0.002	0.040 ± 0.001
ν	0.72 ± 0.12	0.60 ± 0.12	0.56 ± 0.11

Table 9: Values of the critical exponent p and of the fractal dimension for a chain of 20 steps in the pivot algorithm for different dimensions.

5 Conclusions

The simple method is a very basic modelling algorithm. Its efficiency is very limited, but due to its simplicity it can be considered to be the standard with whom the other methods should be compared.

The biased method is an improvement to the simple method and the weighting keeps the results unbiased, so it can be used as a substitute to the simple method in most cases.

The dynamic methods are usually more efficient, but due to their complexity, they might be biased, not ergodic, or too sensible to the initial condition. In spite of that, those are the method of choice to simulate long chains.

Nevertheless, most methods (except for the Kink-jump algorithm) give us similar results for the end-to-end mean distance and the fractal dimension in all dimensions.

From the biased sampling method, we found that the fractal dimension for random walks in (hyper)cubic lattices is 0.74(4) for two dimensions, 0.60(5) for three dimensions and 0.55(4) for four dimensions.

References

[1] Binder K. Et Al. Applications of the Monte Carlo Method in Statistical Physics. Topics in Current Physics. Springer-Verlag.

[2] Gould H., Tobochnik J.; An introduction to computer simulation Methods; Addison Wesley Publishing Company, Second edition.1996.

[3] Landau D., Binder K. A Guide to Monte Carlo Simulations in Statistical Physics. Cambridge University Press.

[4] Binder K., Heermann D., Monte Carlo Simulation in Statistical Physics, An introduction. Springer-Verlag.

[5] Batoulis J., Kremer K.; J.Phys.A: Math. Gen. **21** (1988) 127-146.

[6] Wall, F., Mandel, F.; J. Chem. Phys. **63**, 4592 (1975).

[7] Madras, N., Sokal, A.D.; Journal of Statistical Physics Vol. 50, Nos. 1/2, 1988

[8] Haggstrom, Olle. Finite Markov Chains and Algorithmic Applications. Cambridge University Press, 2002.