# "Snakes on a plane": An introduction to the study of polymer chains using Monte Carlo methods

David Moxey

January 19, 2024

**Supervisor:** Prof. Mike Allen

**Abstract**

In this report, a number of basic Monte Carlo methods for modelling polymer chains are presented (including configurational-bias Monte Carlo and the pruned-enriched Rosenbluth method). These are then used to investigate the behaviour of the collapse of polymer chains around the well-studied $\theta$-point. Additionally, a flat-histogram version of PERM is outlined and applied to the problem of polymers both tethered to and in close proximity to an adsorbing surface.

# Contents

# 1 Introduction

Experimentation in polymer physics is a difficult undertaking, given the extremely small size and relatively complex behaviour of the average polymer. Generally, advanced techniques are needed in order to study even the simplest properties of such chains. Given their importance in both chemistry and biology, coupled with their widespread usage in the manufacturing industry, it is important to be able to analyse their behaviour, and additionally predict areas of potential research interest.

Unfortunately experimentation is costly, both in terms of time and resources, and so an alternative is needed to focus this time on the most important systems. With the increasing availablity of computing power over the past twenty years, it has become viable to study, and more importantly, predict the behaviour of polymer chains by using computational techniques for increasingly complex systems.

As polymers are just a chain of atoms, it is possible to model them using standard molecular dynamics techniques, integrating the standard equations of motion over time. However, given that many polymer chains need to be generated and standard integration methods are very costly, this is not an efficient choice of algorithm.

Additionally, polymers have certain properties which make designing good algorithms extremely difficult. Given a low temperature, a polymer will tend to shrink and become compact; as the temperature increases, it becomes extremely strung out. Typically most algorithms can only be used to grow relatively short chains.

The challenge has been to create algorithms which are able to sample *all* of these states in an efficient manner. With recent advances in novel stochastic growth algorithms, it is now possible to study these chains in a much more computationally efficient way over a broad range of parameters. We also aim to study changes in the behaviour of the polymer as these variables are adjusted – the so called phase transitions of the system.

The aim of this report is to outline an introduction to computational polymer physics using these statistical methods. We will use simple models to outline the basic behaviour of an unconstrained polymer in solvent, and a polymer attached to or placed near to an attractive surface using a variety of algorithms.

In the next chapter, a brief overview of basic Monte Carlo methods is given, aided by the use of two simple examples. We then move on to the study of thermodynamic systems, and the most important properties of basic polymer chains. This is followed by a description and comparison of the algorithms involved. Finally, the algorithms are applied to the problems mentioned above, and some basic conclusions are drawn from the results obtained.

# 2 Monte Carlo Methods

## 2.1 What is a Monte Carlo method?

Monte Carlo methods are a general class of computational algorithms which may be employed to simulate a variety of mathematical and physical systems; the key distinction from the "usual" algorithms being that Monte Carlo methods are non-deterministic (or stochastic). Generally this is because the objective is to sample from some probability space, and this usually involves the use of random number generation.

The main use of a Monte Carlo method is in systems where it is technically unfeasible to use a traditional algorithm. For instance, whilst it may be possible to simulate a small system of atoms using standard ODE methods and the usual equations of motion, for large clusters of atoms

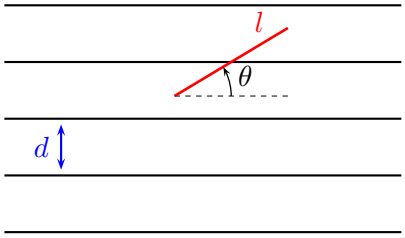



**Figure 1:** Estimating $\pi$ using needle throwing. Note that the length of the needle $l$ should be less than the distance between parallel lines, $d$.

the amount of computation this requires is, in general, impractical. However, by employing a stochastic algorithm it may be possible to dramatically reduce the amount of computational time spent on simulating the system.

Since we will be considering these systems from a purely statistical point of view, we need some method of sampling from them in order to obtain averages. Unfortunately, given the complexity of most spaces, there is no obvious way to achieve this; there are actually very few spaces that are easy to sample from. However, good uniform random number generators are relatively simple to produce and are well-documented, allowing us to sample from a uniform distribution.

Virtually all Monte Carlo methods provide us with a way of sampling from some *target* distribution, by using another *proposal* distribution which is easy to sample from. The 'closer' the proposal distribution is to the target, then the more efficient the method becomes.

To motivate the study of more advanced techniques, we now consider two specific examples; the first provides an interesting method for estimating the value of $\pi$, and the other a simple example of multi-dimensional integration using random variables.

### 2.1.1 Estimating $\pi$

Consider a sheet of paper containing a number of parallel lines which are equally spaced by a distance $d$, and a needle which is of length $l$. If dropped on the table randomly, then the needle may cross one of the lines, or it may lie between two lines. Repeating this $N$ times, one counts the number of times, $C(N)$, that the needle crosses a line. This is depicted in figure 1.

Let $\theta$ denote the (acute) angle between the lines and needle, and $x$ be the distance from the centre of the needle to the closest line. Then, we have that $0 \leq \theta \leq 2\pi$, and $0 \leq x \leq \frac{d}{2}$. If we assume that $l \leq d$, then the needle will cross the line if $x < \frac{l}{2}\sin\theta$. If we treat $x$ and $\theta$ as independent random variables which are uniformly distributed, then the probability of a needle crossing a line is given by:

$$P = \int_0^{\theta}\int_0^{\frac{l}{2}\sin\theta} \frac{2}{\pi d}\,\mathrm{d}x\,\mathrm{d}\theta$$

The maximal probability is clearly given when $d = l$, in which case $P = \frac{2}{\pi}$. Interestingly, this gives us a very easy (but clearly impractical) way of approximating $\pi$; the law of large numbers implies that:

$$\frac{2}{\pi} = \lim_{N\to\infty} \frac{C(N)}{N}$$

This result is known as the *Buffon Needle Problem* [20].

### 2.1.2 Monte Carlo Integration

Whilst it is relatively simple to evaluate many one-dimensional integrals efficiently using standard methods (classically by Gaussian quadrature), these are not suited to $n$-dimensional inte-

grals. Besides the fact that the number of operations required to evaluate the integral increases exponentially with dimension, integration over a difficult domain is a signifcant problem for most of these methods. Many physical problems require the evaluation of multi-dimensional integrals, and so it is important to find a more efficient technique.

For simplicity, let us consider the hypercube $\Omega = [0,1]^n \subset \mathbb{R}^n$. Suppose that we have an integrable function $f : \Omega \to \mathbb{R}$; our aim is to evaluate

$$I = \int_\Omega f(x)\,\mathrm{d}x$$

using a stochastic process. Now, define the function $g : \mathbb{R}^n \to \mathbb{R}$ by

$$g(x) = \begin{cases} 1 & x \in \Omega \\ 0, & x \notin \Omega. \end{cases}$$

That is, $g$ is the indicator function on the set $\Omega$. Notice that

$$\int_{\mathbb{R}^n} g(x)\,\mathrm{d}x = \int_\Omega 1\,\mathrm{d}x = 1$$

and so $g$ simply defines a probability distribution function. Indeed, in this case, $g$ is the uniform distribution. Now, we can rewrite $I$ as

$$I = \int_\Omega \frac{f(x)}{g(x)} \cdot g(x)\,\mathrm{d}x = \langle f/g \rangle_g,$$

where $\langle f \rangle_g$ is the expectation of the function $f$ with respect to $g$. Now by sampling $x_i$ independently and uniformly in $\Omega$, we obtain an estimate $I_N$ for $\langle f/g \rangle$

$$I_N = \frac{1}{N}\sum_{k=1}^{N} \frac{f(x_k)}{g(x_k)} = \frac{1}{N}\sum_{k=1}^{N} f(x_k),$$

since $g(x_k) = 1$ on $\Omega$. As $N \to \infty$, the law of large numbers guarantees that $G_N \to I$ and so, by taking large values of $N$, we can obtain an accurate approximation for $I$. Additionally, by using the central limit theorem, it can be shown that the error converges at a rate of $O(N^{-\frac{1}{2}})$ – that is, independent of dimension.

The extension to a set of the form $\Omega = [a_1, b_1] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^n$ is clear by taking the function $g$ as

$$g(x) = \begin{cases} \frac{1}{\mathrm{Vol}\Omega} & x \in \Omega \\ 0, & x \notin \Omega \end{cases}$$

and applying the same method. Given a more complex domain, we cover it by a simpler set of the above form, and by using uniform independent samples from this cover, obtain an estimate for the volume; the same method can then be applied. Note that we do not necessarily need to use the uniform distribution, as long as we know *how* to sample from it. Given the type of function involved, the distribution can be picked to optimize the method.

## 2.2 Metropolis-Hastings Sampling

We now move on to the theory of Metropolis-Hastings sampling. In order to understand these ideas, some cursory knowledge behind the theory of Markov chains is essential. The most important results are outlined briefly in this section and are taken in most part from [19], where they are explained in far greater detail.

Given a state space $S$ of some probability space with $x_0, \ldots, x_n \in S$ and $A \subseteq S$, a sequence of random variables $X_i \in S$ is called a *Markov chain* if

$$\mathbb{P}(X_{n+1} \in A \mid X_n = x_n, \ldots, X_0 = x_0) = \mathbb{P}(X_{n+1} \in A \mid X_n = x_n)\,.$$

That is, if we know the current value of the chain, then the values that the chain will take in the future will not depend on any previous values. Now, given a sequence with this property, we can obtain *transition probabilities* between states,

$$\mathbb{P}(X_{n+1} \in A \mid X_n = x) = \int_A p(x, y)\,\mathrm{d}y.$$

Here, $p(x, y)$ is called the transition kernel.

A crucial property of Markov chains is that, given certain regularity conditions, the distribution of the chain will tend towards a limit called the stationary distribution. In this case, the Markov chain is said to be *ergodic*.

Given a distribution defined by a density function $\pi(x)$ satisfying the *detailed balance condition*

$$\pi(x)p(x, y) = \pi(y)p(y, x)$$

for every $x, y$ and for the transition kernel $p$ of some ergodic Markov chain $X$, it can be proven that $\pi$ is the stationary distribution of $X$. In fact, detailed balance is merely a sufficient condition for the general balance condition

$$\pi(x) = \int \pi(y)p(x, y)\,\mathrm{d}y.$$

Markov Chain Monte Carlo (MCMC) in most part relies on an important result known as the *ergodic theorem*. Given a real function $h$ and an ergodic Markov chain $X$ with stationary distribution $\pi$, consider the average

$$h_n = \frac{1}{n}\sum_{k=1}^{n} h(X_k).$$

If $Y$ has distribution $\pi$ and $\langle |h(Y)| \rangle_\pi < \infty$ then $h_n$ converges to $\langle h(Y) \rangle_\pi$ with probability 1 as $n \to \infty$. This is effectively a law of large numbers for ergodic Markov chains.

The general idea behind most of MCMC is conceptually simple. We wish to obtain averages for some random variable $h(Y)$ from a space with probability density $\pi$, but are unsure as to how to sample from the particular distribution (or are otherwise unable to calculate $\langle h(Y) \rangle$).

However, if we can construct an ergodic Markov chain $X$ which has stationary distribution $\pi$, then we simply calculate $\frac{1}{n}\sum_{k=1}^{n} h(X_k)$. The ergodic theorem then guarantees that for large enough $n$ we will obtain a close approximation.

Finding such a Markov chain sounds quite challenging, but in fact is reasonably simple; indeed, there are a number of prescribed methods for creating such a chain. We will consider *Metropolis-Hastings* sampling, which is one of the most commonly used techniques.

Recall that our objective is to sample from some target distribution; let $S$ be the state space of this distribution. Then, for every $x \in S$, we choose a density function $q(x, \cdot)$ on $S$. Then, with $x$, $q(x, \cdot)$ becomes the transition kernel of some Markov chain; hopefully we pick $q$ such that the resulting distribution is easy to sample from (ideally, something like the uniform distribution).

If, at stage $n$ the Markov chain has $X_n = o$, we sample a new state $n$ from the distribution defined by $q$. Then, we set $X_{n+1} = n$ with probability

$$\mathrm{acc}(o \to n) = \min\left(1, \frac{\pi(n)q(n, o)}{\pi(o)q(o, n)}\right);$$

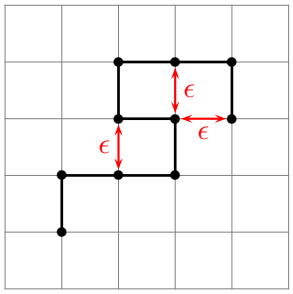



**Figure 2:** A 10-monomer on the square lattice, with nearest-neighbour energy $\epsilon$.

and otherwise, we set $X_{n+1} = o$. Whilst the acceptance probability may seem rather strange, it is chosen in such a way as to satisfy the detailed balance condition, so that

$$\begin{aligned}
\pi(x)p(x,y) &= \pi(x)q(x,y)\,\mathrm{acc}(x \to y) \\
&= \min\left(\pi(x)q(x,y), \pi(y)q(y,x)\right) \\
&= \pi(y)q(y,x)\min\left(\frac{\pi(x)q(x,y)}{\pi(y)q(y,x)}\right) \\
&= \pi(y)p(y,x)
\end{aligned}$$

This ensures that as we measure means, by the ergodic theorem, they will converge to the correct value.

# 3 Polymer Growth Algorithms

## 3.1 The self-avoiding walk

Polymers, whilst being quite large at the appropriate scale, are molecular-sized objects. They are constructed from a series of smaller molecules, known as *monomers* which are linked together in a chain by covalent bonds. Generally this is done in some form of regular pattern. Mostly, polymers are associated with plastics, although they play a key role in many biological systems.

When modelling polymers in the computational sense, many different representations can be found; generally, they can be classified into two groups – lattice, and off-lattice. In the first case, monomers are placed on the vertices of a lattice structure, with bonds between consecutively numbered monomers represented by the paths between vertices. An off-lattice method, as the name suggests, does not use such a lattice structure (but many lattice methods can be generalised to off-lattice cases). We will only consider lattice methods with simple geometries, such as the square lattice (2-dimensional) and simple cubic lattice (3-dimensional) – another popular model uses the face-centred cubic lattice.

Generally polymers do not self-intersect, so with either method, it must be ensured that the chain is not allowed to intersect itself (i.e. in the lattice case, no path contains a vertex more than once). To make this model more realistic, we also assign an attractive energy $\epsilon < 0$ between pairs of neighbouring, but non-bonded monomers; no consideration is given to any monomers lying more than one lattice unit distance away. An example of a polymer on the square lattice is given in figure 2. The major task is to devise a way of efficiently generating many self-avoiding walks (SAWs) of this type of a specific length, measuring important averages which quantify particular behaviour.

Whilst this model may or may not be an accurate physical representation of an actual polymer, there are still significant computational problems to overcome. Sampling from this space is not at first obvious. Naively, by treating the space as a probabalistic ensemble, we could perform

a complete enumeration of the space, and calculate properties for each walk along with the associated probability of obtaining the walk. (Note that the space of finite-length SAWs is unbounded, but we assume that the walk is translation invariant and hence always starts from the origin). From this, it is easy to calculate the required averages for the observed quantities.

Unfortunately, this is extremely impractical for even the shortest of chains, simply because of the number of SAWs that need to be considered. Indeed, the number of walks of length $n$, $C_n$ obeys [10]

$$C_n \sim \mu^n n^{\gamma-1}, \quad n \to \infty, \tag{1}$$

where $\mu$ is called the connective constant and $\gamma = 45/36$ for the square lattice, and the relation $\sim$ denotes an asymptotic approximation

$$f(n) \sim g(n) \text{ as } n \to \infty \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

This makes any sort of complete enumeration technically unfeasible, as we are generally interested in chains of reasonably long lengths. In fact for the simple cubic lattice, $C_n \sim 4.7^n$ and so to store 1 byte of information for all walks of length 15 would require around 10GB of computer memory. Additionally, the amount of time required to generate all random walks of length $n$ will scale exponentially as $O(4.7^n)$. Considering that numerical algorithms usually scale as $O(n \log n)$ or $O(n^p)$ for some power $p$, this is clearly not going to be an efficient way to generate walks.

However, as we have previously seen, an appropriate Monte Carlo method could allow us to sample efficiently from the space, either by using a Metropolis scheme or some other technique. In the rest of this section, we describe a number of such algorithms, provide a discussion of their implementation and the advantages and disadvantages of each.

### 3.2 Basic Statistical Mechanics

Before considering the different algorithms, it is important to understand some of the key ideas from statistical mechanics, and how they apply to our circumstances. This section is not meant to be an exhaustive overview, but an extremely quick introduction outlining some of the fundamental ideas for the setting of the algorithms.

The simulations we will perform regard any generated polymer chain as being one particular state in a thermodynamic system which is dependent upon certain parameters. Mostly, we will be interested in the temperature of the system, and the quality of the solvent in which the polymer lies.

Given a set $\Gamma$ which enumerates the states of the system, we wish to make a distribution which gives the probability of a randomly chosen configuration being in a particular state. Given $n \in \Gamma$ with energy $E_n$ at temperature $T$, it can be shown that the weighting assigned to $n$ is proportional to the *Boltzmann factor*, $e^{-E_n/k_BT}$, where $k_B$ is the *Boltzmann constant*.

It is important to note that the Boltzmann factors by themselves cannot give a probability distribution, since they are not normalized. However, by introducing a normalizing factor we can correct this issue. The probability that the system is in state $n$ is hence given by

$$p_n = \frac{C_n e^{-E_n/k_BT}}{Z(T)},$$

where $C_n$ is the number of states having energy $E_n$, and $Z(T)$ is the *partition function*, given by

$$Z(T) = \sum_{n\in\Gamma} C_n e^{-E_n/k_BT}.$$

The probabilities $p_i$ then give the *Boltzmann distribution*. It should be noted that the partition function can also depend on more than a single variable $T$, according to the particular thermodynamic system, and we will use this later. Generally, we write

$$Z(\beta) = \sum_n C_n e^{-\beta E_n}$$

with $\beta = 1/k_B T$. $Z$ is an extremely important function, as it encodes a lot of the information about the thermodynamic system without needing to explicity measure these quantities. For example, the average energy of the system is given by:

$$\langle E \rangle = \sum_n p_n E_n = \frac{1}{Z(\beta)} \sum_n C_n E_n e^{-\beta E_n} = -\frac{1}{Z} \frac{\partial}{\partial \beta} Z(\beta) = -\frac{\partial}{\partial \beta} \log Z(\beta).$$

Another important quantity is the *heat capacity*, $C_V$, which gives an indication as to the amount of heat that a body can store as temperature changes, and is defined as

$$C_V = \frac{\partial}{\partial T} \langle E \rangle = -\frac{\partial}{\partial T} \left( \frac{\partial}{\partial \beta} \log Z(\beta) \right) = \frac{1}{k_B T^2} \frac{\partial^2}{\partial \beta^2} \log Z(\beta). \tag{2}$$

In particular, the heat capacity will help us to identify areas of phase transition for a polymer chain. We call $-k_B T \log Z(\beta)$ the *free energy* of the system.

### 3.3 Simple Sampling

We now consider a series of stochastic growth algorithms, in which chains are 'grown' by laying down one monomer at a time. Amongst the simplest is a procedure called "simple sampling" [2], running as follows:

1. Place the first monomer at the origin.
2. Choose a direction from the $k$ available directions, where $k$ is the coordination number of the lattice – the number of paths connected to each vertex (for example, $k = 6$ for the simple cubic lattice and $k = 4$ for the square lattice).
3. If the direction is already taken, stop growing the chain, remove all monomers and return to the first step.
4. Repeat step 2 until the chain is grown to the desired length.
5. Collect statistics about this chain.

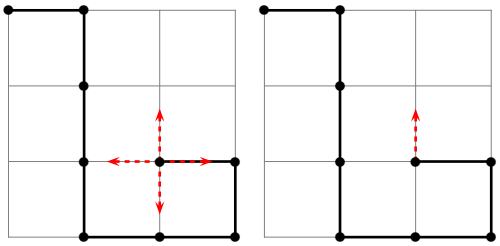
Clearly because we are choosing randomly from every direction, including those that are blocked, grown chains will be generated with equal probability and also be independent of one another. Unfortunately, for a lattice with coordination number $k$, this restriction leads to a serious problem. Namely, we have that the probability of generating a SAW of length $n$, $\mathbb{P}_{\text{saw}}(n)$ is given by

$$\mathbb{P}_{\text{saw}}(n) \sim \frac{\mu^n n^{\gamma-1}}{k^n} = n^{\gamma-1} e^{-\lambda n} \sim e^{-\lambda n}$$

for large values of $n$, where the first asymptotic expansion is given by eq. (1), and $\lambda = \log(k/\mu)$ is the attrition constant. Since $\lambda > 0$ [12], this then implies that $\mathbb{P}_{\text{saw}}(n)$ decreases exponentially with $n$. Whilst simple to implement and understand, this is clearly not a good algorithm!

### 3.4 The Rosenbluth and Rosenbluth Algorithm

The origins of the enumeration problem date back to the late 1940s, when the cubic lattice was first offered as a possible model for polymer chains. In 1955, Rosenbluth and Rosenbluth [18]



(a) Simple sampling (b) Rosenbluth sampling

**Figure 3:** Comparing simple and Rosenbluth sampling techniques. Whilst the simple sampling algorithm must consider all possible walks at every step, the Rosenbluth method allows us to ignore those sites which are already taken, leading to a much higher probability that the walk will finish.

presented one of the first statistical methods able to sample configurations for large chain lengths. Up until this point, authors had relied solely on a complete enumeration and concentrated on smaller chain lengths, generally in the region of $n \leq 20$.

The idea is a relatively simple example of *importance sampling*. In the simple sampling method, at each stage we chose a direction to continue growth at random, regardless of whether monomers were already placed there. This ensured chains were generated with equal probability, and the values we obtained for averages were sampled from the correct distribution.

On the other hand, if we deliberately choose to ignore blocked directions, then we will no longer be uniformly sampling and the results will be biased. However, by assigning a weight to each chain generated by this algorithm, the bias in any observed values can be altered to bring the averages back towards the correct value. Additionally, $\mathbb{P}_{\text{saw}}(n)$ will no longer decrease exponentially, making the algorithm much faster.

During the simulation we attempt to generate $M$ chains of length $l$, and measure an observable value $A$ for each chain generated. The algorithm, as described in [5], for generating a configuration $n$ runs as follows:

1. Place the first monomer at the origin, and denote its energy by $u_0$. Define the *Rosenbluth weight* of this monomer as $w_0 = k\exp(-\beta u_0)$, where $k$ is the coordination number of the lattice.

2. To insert the $i$-th monomer, consider the $k$ possible trial directions adjacent to the previous monomer. Let the energy of the $j$-th trial position be denoted by $u_i(j)$, and then from these $k$ directions, choose one with probability

$$p_i(j) = \frac{\exp(-\beta u_i(j))}{w_i}$$

where $\beta = 1/k_B T$ as before, and with $w_i$ being the sum of the weights of the trial directions,

$$w_i = \sum_{j=1}^{k} \exp(-\beta u_i(j)).$$

The energy $u_i(j)$ does not include any interactions with any of the subsequent monomers $i+1, \ldots, N$. Note that if the trial direction is blocked then we set $u_i(j) = \infty$, giving that direction weight 0.

3. Repeat step 2 until the entire chain is grown. Then, compute the total Rosenbluth factor of this configuration as

$$W(n) = \prod_{i=0}^{l} w_i.$$

Note that the probability of generating a particular configuration using the Rosenbluth algorithm is given by

$$P(n) = \prod_{i=1}^{l} \frac{\exp(-\beta u_i(j))}{w_i} = k^l \frac{\exp(-\beta U(n))}{W(n)}, \tag{3}$$

where $U(n)$ denotes the total energy of the chain. Note that these probabilities are normalized (that is, the sum of $P(n)$ over all possible conformations is equal to 1). Now, denote the Rosenbluth ensemble average $\langle\cdot\rangle_R$ by:

$$\langle A\rangle_R = \frac{\sum_{n=1}^{M} W(n)A(n)}{\sum_{n=1}^{M} W(n)}$$

The label $_R$ is important here, since as mentioned above, the algorithm does not generate chains with the correct Boltzmann weight and hence the samples will not be from the Boltzmann distribution.[1] However, notice that

$$\begin{aligned}
\langle A\rangle_R = \frac{\sum_n W(n)A(n)P(n)}{\sum_n W(n)P(n)} &= \frac{\sum_n W(n)A(n)\left(k^l \frac{\exp(-\beta U(n))}{W(n)}\right)}{\sum_n W(n)\left(k^l \frac{\exp(-\beta U(n))}{W(n)}\right)} \\
&= \frac{\sum_n A(n)\exp(-\beta U(n))}{\sum_n \exp(-\beta U(n))} \\
&= \langle A\rangle
\end{aligned}$$

Hence to obtain an approximation for $\langle A\rangle$, we simply calculate

$$\langle A\rangle \approx \frac{\sum_{n=1}^{M} W(n)A(n)}{\sum_{n=1}^{M} W(n)}.$$

Whilst the Rosenbluth and Rosenbluth method is certainly a better approach than attempting to completely enumerate the space, there are still two major problems to overcome. Firstly, the weights are liable to change drastically in magnitude, and so those configurations with large $W(n)$ tend to dominate the average values we obtain. (An additional technical factor is that the size of the weights can lead to overflow problems; they must be stored in double precision variables to overcome this.) Somewhat more importantly, the method is extremely susceptible to a problem with all of the growth algorithms being considered; namely, it is easy for the walk to get trapped (see figure 4).

If the walk does become trapped, then it is clear that at this step, $w_i = 0$. Importantly, the configuration *still appears* in the calculations of ensemble averages with weight zero. Subsequently, if too many walks become trapped, many more chains will need to be generated in order to obtain accurate results.

A minimal example is given in figure 3(b). We see that at each stage apart from the initial stage, there are only $k-1$ trial directions, since one will always be blocked by the preceding monomer. At the 10th stage of construction, three of the trial directions are blocked. To counter this problem, we set the respective energies being equal to $\infty$, which gives 0 as the weights for these directions. Hence, we choose the only direction available with probability 1.

[1] For further information and an example of this, see [5], example 13.

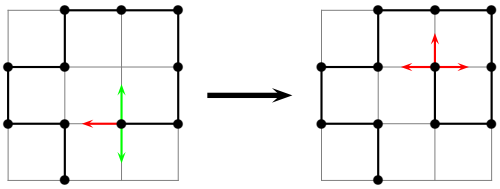


**Figure 4:** The Rosenbluth method becoming trapped; on the left, the algorithm chooses the top direction, leaving the walk blocked in by the other monomers.

### 3.5 Configurational-Bias Monte Carlo

Configurational-Bias Monte Carlo (CBMC) is an attempt to improve upon the Rosenbluth sampling technique outlined above. Since the method is biased, the algorithm does not sample from the Boltzmann distribution, relying on a re-weighting that requires a large amount of computational effort to produce correct results for longer chains.

In configurational-bias Monte Carlo, instead of re-weighting the ensemble averages, we employ a Metropolis-type rejection technique to remove the bias, effectively allowing us to sample straight from the target distribution. (Additionally, this also helps to reduce the chances of an overflow error occurring as the sum of weights no longer need to be stored in memory.)

The algorithm assumes we already have an initial starting configuration $o$, with Rosenbluth weight $W(o)$. Then run through the following steps:

1. Generate a new trial move $n$ using the Rosenbluth scheme and compute its Rosenbluth weight $W(n)$.
2. Accept the trial move from $o \to n$ with a probability
$$\mathrm{acc}(o \to n) = \min\left(1, \frac{W(n)}{W(o)}\right).$$
3. If the new configuration is accepted, set $o = n$. Repeat step 1 until $M$ chains have been generated.

The justification that this algorithm samples from the Boltzmann distribution is reasonably simple. Note that, by eq. 3, we see that the probability of generating a particular conformation is given by
$$\mathbb{P}(o \to n) = \frac{\exp(-\beta U(n))}{W(n)}, \ \mathbb{P}(n \to o) = \frac{\exp(-\beta U(o))}{W(o)}.$$
Hence, the detailed balance condition implies that
$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{W(n)}{W(o)}$$
and the acceptance criterion in step 3 guarantees this.

We measure an observable $A$ after the trial move has been generated in step 1. If the chain is accepted, then its contribution is added towards the ensemble average. Otherwise, we add the value measured from the old conformation.

### 3.6 The Pruned and Enriched Rosenbluth Method (PERM)

The pruned-enriched Rosenbluth method [16] is another attempt to correct some of the issues with the Rosenbluth method. Unlike CBMC, we do not use a Metropolis-type scheme, but again rely on the re-weighting of observed quantities.

The concept behind the improvement is relatively simple; we wish to encourage the growth of 'good' chains, and stop 'bad' chains before they waste too much computational time. Obviously we need some notion of 'good' and 'bad', and this is obtained by looking at the relative weights of the chains being generated.

If the weight is large, then it is likely that this chain will reach the desired length without becoming trapped. We encourage the growth of these chains by making copies of them and reducing their weights accordingly – this is the *enrichment* step.

Likewise, if the weight becomes too small then it is likely that it will become trapped. In this case, we *prune*; that is, stop further growth with a certain probability, and otherwise increase the weight of the chain and continue growing.

Notice that the objective is to keep the weight of the generated chains roughly constant, correcting one of the most significant problems with the Rosenbluth method. The algorithm for generating polymers of length $l$ (as stated in [17]) is outlined below:

1. Place the first monomer at the origin.
2. At the $n$-th stage of construction, define the partial weight of the chain to be

$$W_n = \prod_{k=0}^{n-1} e^{-\beta E_k}.$$

   (Recall that $E_k$ is the energy of the $k$-th monomer, and so $e^{-\beta E_k}$ is the associated Boltzmann factor.)

   Choose a free direction from those available at random (i.e. with equal probability). If there are no free directions, assign this chain weight 0 and return to step 1.
3. If $W_n > W^>$, where $W^>$ is some parameter denoting upper bound on the weight of the chain, then make two copies of the chain, each with weight $\frac{1}{2}W_n$, and continue to grow each one independently.
4. If $W_n < W^<$ then stop the growth of the polymer with probability 1/2; otherwise, set the weight of the polymer to be $2W_n$ and continue growing.
5. Repeat steps 2-5 until the desired length is reached.

Then, ensemble averages of an observable $A$ can be calculated by

$$\langle A \rangle = \frac{\sum_{n=1}^{M} A(n)W(n)}{\sum_{n=1}^{M} W(n)}$$

as in the Rosenbluth method. Intuitively, the independent growth at the branching point in step 3 would be done in parallel, growing lots of chains at the same time. However, due to the unknown number of branchings, this makes the parallelization of the algorithm difficult and somewhat impractical.

Instead, we make use of a stack. At the $n$-th stage, we place the monomers at the top of the stack, and then remove them when we get to the end of the chain. Of course, the easiest way of accomplish this is by the use of a recursive algorithm; the pseudo-code for this is outlined in the appendix.

Notice that the PERM algorithm depends on two parameters, $W^>$ and $W^<$, the upper and lower thresholds for pruning/enrichment. The efficiency of the algorithm is almost entirely dependent upon the correct choices of these parameters, and as such naive choices can lead to too much enrichment or too much pruning.

Let us assume that we set $W^{>} = W^{<} = 0$, such that no pruning occurs and only enrichment is allowed at every step. Then, assuming that the walk does not become trapped, we have that for a chain of length $N$ there will be $O(2^{N+1})$ subroutine calls. Alternatively, if too much pruning is allowed (i.e. $W^{<}$ is too high), then too few walks will be completed to obtain accurate measurements.

As such PERM can be somewhat unpredictable in terms of its efficiency, and parameters need to be chosen carefully to allow the method to work optimally. Additionally (as noted by the authors), the method performs quite badly at low temperatures unless the parameters are chosen extremely carefully. However, given the correct temperature range, the method is extremely efficient, capable of generating very high numbers of configurations in only a few hours of CPU time.

There have been several improvements to the PERM method, mostly aimed towards making the algorithm show this same performance for low temperature simulations. We consider an algorithm of this type, and also another which adapts PERM to use a Metropolis-style rejection sampling (as per the CBMC method). Neither of these algorithms have been implemented as a part of this project, but are included for completeness.

### 3.6.1 New PERM (nPERM)

nPERM [8] modifies the PERM method in two respects with the aim of finding better performance at low temperature ranges. Instead of (effectively) creating two identical copies of the same chain, we try to diversify the range of chains that PERM is generating by branching more than twice at each stage, and ensuring that these different branches are distinct.

So, at a particular stage of construction, assuming that the number of available sites $k_{\text{free}} \geq 2$, we choose $2 \leq k \leq k_{\text{free}}$ *different* sites, and ensure that the algorithm grows from each of these accordingly. (With regular PERM, and $k_{\text{free}} = 2$, there is only a 50% possibility that the chains will be distinct at each stage.)

Of course, the value of $k$ is also important for the efficiency of the algorithm. With nPERM, we pick:

$$k = \min\left(k_{\text{free}}, \lceil W^{\text{pred}}/W^{>} \rceil\right),$$

where $W^{\text{pred}}$ estimates a prediction for the weight. This is done by using a 'Markovian anticipation' technique, similar to that outlined in [6], § 4.2. With these two techniques, nPERM has offered a significant speed improvement over PERM for low-temperature runs, and also reduced the sensitivity of PERM to the parameters being used.

As noted in [8], this method can be used to search for the ground states (i.e. those states with lowest energy) of a particular type of polymer. In fact, the efficiency in this case can be improved by the use of a rejection sampling technique, as in CBMC.

### 3.6.2 Dynamic PERM

As with the Rosenbluth method, PERM is a static algorithm; that is, many chains are grown independently of one another. CBMC is *dynamic*; i.e. it uses the result from the previous conformation to generate new chains.

Dynamic PERM (dPERM) combines PERM and rejection sampling in the same way that CBMC uses the Rosenbluth method and the Metropolis scheme. A more explicit description is given in [4], including a derivation of the acceptance rule from the detailed balance condition. The main difference is that an initial pool of chains are generated using PERM instead of the Rosenbluth method.

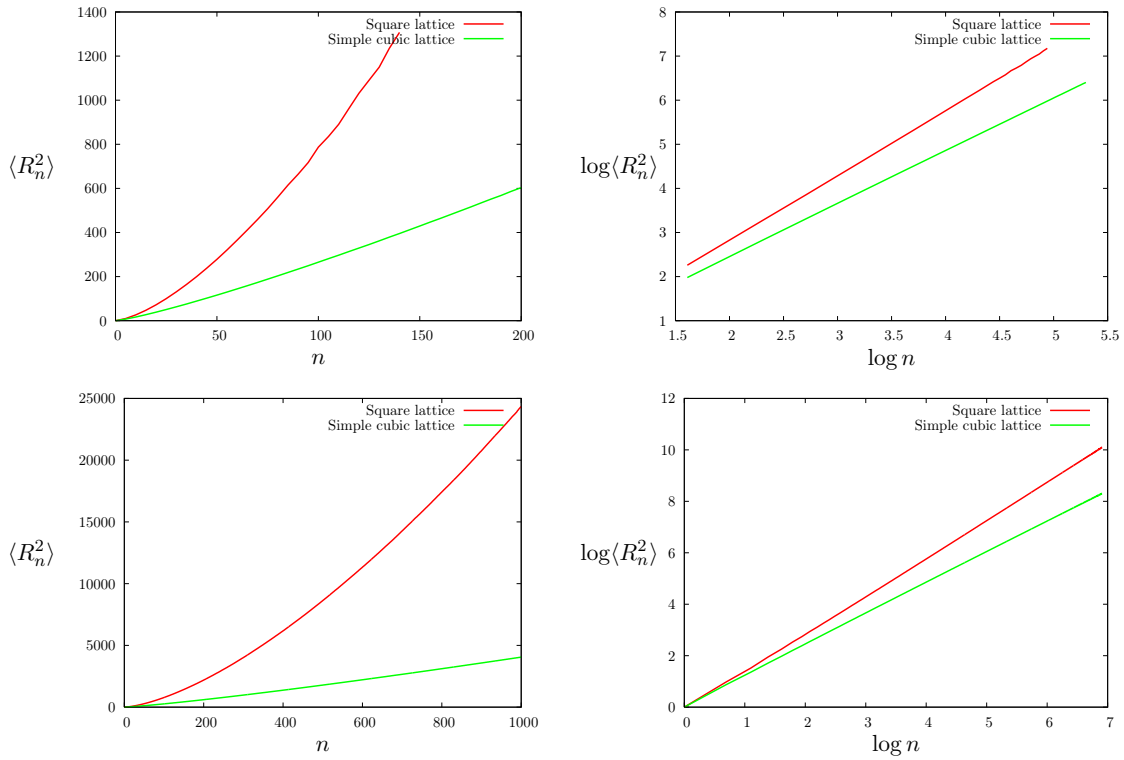



**Figure 5:** Finding the power law exponent $\nu$ for $\langle R_n^2 \rangle$ using both the CBMC (top) and PERM (bottom) algorithms.

As noted in the paper, for single chains there is no significant improvement over any of the other techniques listed so far. However, the dynamic techniques are more efficient when considering a system of polymer chains, since there is no need to grow each from scratch; we simply pick one at random and make a trial move. As mentioned above, they can also be used to improve algorithms like nPERM under certain circumstances.

# 4 Initial Results

As a part of the project, both the CBMC and PERM algorithms were implemented. In this section, we will perform two basic experiments to check the validity of the implementations against known results, and to draw some basic comparisons between the methods. Firstly, we test the generation of self-avoiding walks from the origin, with no consideration for the attractive energy term. Secondly, we investigate the commonly observed $\theta$-point collapse of polymer chains.

## 4.1 Generating self-avoiding walks

The initial tests for both the CBMC and PERM algorithms involved generating self-avoiding walks on both the square and simple cubic lattice, without any attractive energy $\epsilon$.

Whilst many properties of the self-avoiding walk are known, perhaps the simplest to measure is the mean-square end-to-end distance. For a walk starting from the origin $\vec{0}$ containing $M$ monomers with each monomer $i$ having respective lattice co-ordinates $\vec{r_i}$, we define the square end-to-end distance $R_M^2$ by

$$R_M^2 = \|\vec{r_M}\|^2,$$

where $\|\cdot\|$ denotes the standard Euclidean norm on $\mathbb{R}^n$ (we assume that the lattice is contained within Euclidean space). The mean-square end-to-end distance $\langle R_n^2 \rangle$ is given by the ensemble

average of chains of length $n$. It is believed that, for the self-avoiding walk on a lattice, $\langle R_n^2 \rangle$ obeys a simple power law in the asymptotic limit,

$$\langle R_n \rangle \sim n^{\nu} \text{ and } \langle R_n^2 \rangle \sim B n^{2\nu}, \quad n \to \infty,$$

where the exponent $\nu$ depends on the dimensionality of the particular lattice. For the square lattice, it has been proven [15] that $\nu = 3/4$. Exact results are not known for the simple cubic lattice[2], but approximations show that $\nu \approx 0.592$ [7].

For CBMC, chains of up to length 200 were generated for the simple cubic lattice and 140 for the square lattice. Due to the use of an acceptance rule in CBMC, obtaining values for $\langle R_n^2 \rangle$ in range of values $0 \leq n \leq N$ requires that a chain is grown from length $0 \to n$ and run through the acceptance rule *before* statistics are collected (otherwise we may not be sampling from the Boltzmann distribution). So to generate statistics for $\langle R_n^2 \rangle$, a subroutine call must be made for each $n$.

In contrast, since PERM uses a recursive algorithm, it is able to calculate the values of $\langle R_n^2 \rangle$ for *all* $n$ in a single call to the subroutine (see the appendix for pseudo-code). By using the upper and lower thresholds outlined in [8], the PERM algorithm was able to generate chains of up to length 1,000 for both lattices. For both algorithms, $10^7$ chains were generated to calculate accurate ensemble averages.

Initial simulations with the algorithms generated the results shown in figure 5. We visualise both $\langle R_n^2 \rangle$ against $n$, and a logarithmic plot of the same values. From the latter diagrams, it is easy to see that $2\nu \approx 3/2$ for the square lattice, and $2\nu \approx 6/5$ for the simple cubic lattice. These results were confirmed by using standard linear regression techniques.

Error calculations for both CBMC and especially PERM are not straightforward. However, to obtain a basic approximation as to the accuracy of the results, 8 independent runs were performed, each using a different seed for the required psuedorandom number generator. Each process generated around $10^8$ chains on the simple cubic lattice up to a maximum chain length of 200 monomers.

From these seperate runs, the mean values and standard deviation were calculated and are shown in table 1. As can be clearly seen, the methods certainly appear to be generating consistant and correct data, as the standard deviation is small compared to $\langle R_n^2 \rangle$.

Note that the standard deviation for the PERM results is significantly smaller than that for CBMC. This is mostly due to the fact that whilst $10^7$ initial subroutine calls were made, this number cannot be kept constant since PERM is recursive (although it remains roughly constant – see figure 9).

Additionally, whilst the acceptance rate for CBMC drops to around 10% for $n = 200$ on the simple cubic lattice, for the square lattice that figure is around 0.01%, making the algorithm extremely inefficient for even moderately long chain lengths (see figure 8).

### 4.2 Polymer collapse and the $\theta$ point

A common phenomenon in polymer physics is the collapse of polymer chains. At high temperatures polymers tend to be strung out over large areas; as the temperature is decreased past a certain critical point, there is a sudden change in the behaviour of the polymer, becoming much more compact and globular-like in shape.

[2] [9] outlines a proof that, for the simple cubic lattice, the root-mean-square value $\langle R^2 \rangle_{\text{rms}} = \sqrt{\langle R^2 \rangle}$ has $\nu = 7/12$, but this is yet to be published.

| | $\langle R_n^2 \rangle$ | |
|---|---|---|
| $n$ | CBMC | PERM |
| 5 | 7.2340± 0.0002 | 7.2340± 0.0004 |
| 10 | 16.818 ± 0.001 | 16.8180± 0.0007 |
| 25 | 50.629 ± 0.006 | 50.627 ± 0.004 |
| 50 | 116.066 ± 0.022 | 116.007 ± 0.006 |
| 75 | 188.263 ± 0.046 | 188.258 ± 0.015 |
| 100 | 265.214 ± 0.109 | 265.162 ± 0.021 |
| 150 | 429.237 ± 0.237 | 429.308 ± 0.015 |
| 200 | 603.594 ± 0.691 | 603.956 ± 0.055 |

**Table 1:** Error estimates for the CBMC and PERM methods on the simple cubic lattice.

To simulate this, we now reintroduce the attractive monomer-monomer energy $\epsilon$. In this case, it is generally understood [14] that $\langle R_n^2 \rangle$ still obeys the power law

$$\langle R_n^2 \rangle \sim n^\gamma, \quad n \to \infty,$$

but the exponent $\gamma$ now depends upon the value $\phi = -\epsilon/k_B T = -\epsilon\beta$. Note that the Boltzmann factor at each step of construction is of the form $\exp(-m\epsilon\beta) = \exp(m\phi)$, where $m$ is the number of monomer-monomer contacts, giving some indication as to why $\phi$ is an important property.

(As a technical note, since we know the upper limit on the number of monomer-monomer contacts, there can only be this number of Boltzmann factors at any particular stage. By storing these values in an array instead of calculating them explicitly at each stage, the algorithm can benefit from a significant speed increase).

In this case, as the value of $\phi$ decreases past a certain critical point, known as the $\theta$ point, we see the same collapsing polymer behaviour. Whilst there are no exact values for this point, it is believed that $\theta \approx 0.274$ [14, 13].

Both the CBMC and PERM routines were adapted to include the attractive monomer-monomer energy, and simulations over a variety of different values of $\phi$ near the $\theta$ point were recorded on the simple cubic lattice. Various chain lengths were used; for CBMC, a maximum length of 200 was imposed, whereas for PERM the limit was decreased to 400 monomers.

The results can be seen in figure 6. From the logarithmic plots, we can see that the $\theta$ point does indeed lie close to 0.274. Again, a simple regression check confirmed that the gradient is very close to 1 for this value of $\phi$.

To further check the results were valid, figure 7 shows a plot of $\langle R_n^2 \rangle / n$ (the "swelling factor" of $R_n^2$) against $1/\log n$, a result measured in [16].

### 4.3 Conclusions

Whilst both methods produced nearly identically correct results, CBMC does suffer significantly from a very high rejection rate for longer polymers. This makes obtaining accurate results extremely difficult, since many more chains need to be generated in order to get near the true values.

Figure 8 shows a plot of the percentage of configurations accepted against the chain length, $n$. As is clear from the plot, for longer chain lengths very few are accepted, leading to incorrect results. Unfortunately, this is due to the large variation of weights obtained from the Rosenbluth method, making CBMC an unattractive choice for modelling a single polymer in free space.

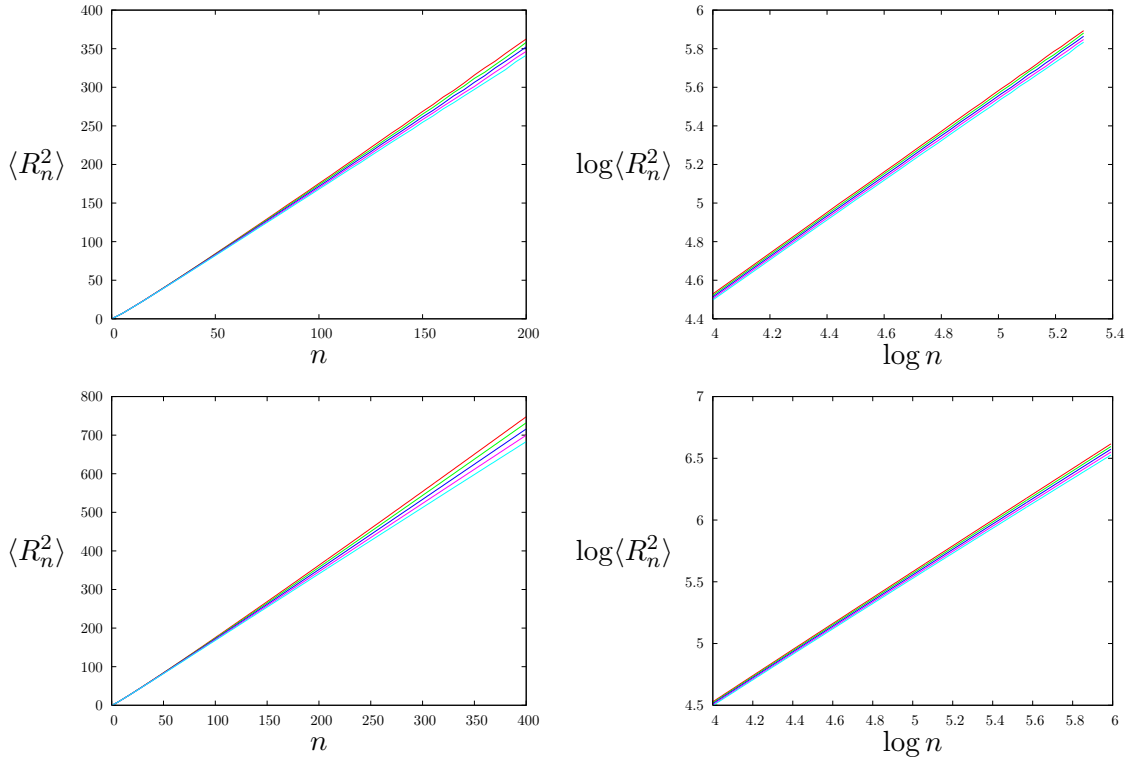



**Figure 6:** Finding the power law exponent $\nu$ for $\langle R_n^2 \rangle$ using both the CBMC (top) and PERM (bottom) algorithms. The five lines, from top to bottom, denote $\phi = 0.262, 0.266, 0.270, 0.274$ and 0.278. CBMC uses chain lengths up to 200, and PERM up to length 400. Linear regression confirms that $\phi = 0.274$ has a gradient roughly equal to 1.

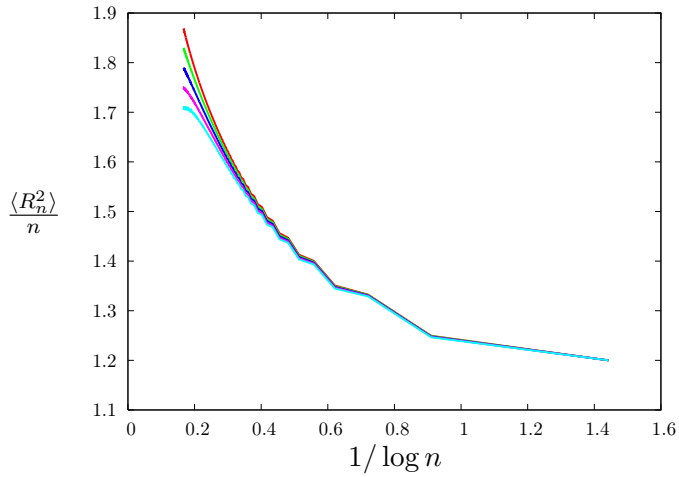



**Figure 7:** A plot of the "swelling factor" $\langle R_n^2 \rangle / n$ against $1/\log n$, as in [16]. The values of $\phi$ used are the same as in figure 6, with $\phi = 0.262$ being the top curve and $\phi = 0.278$ the bottom.

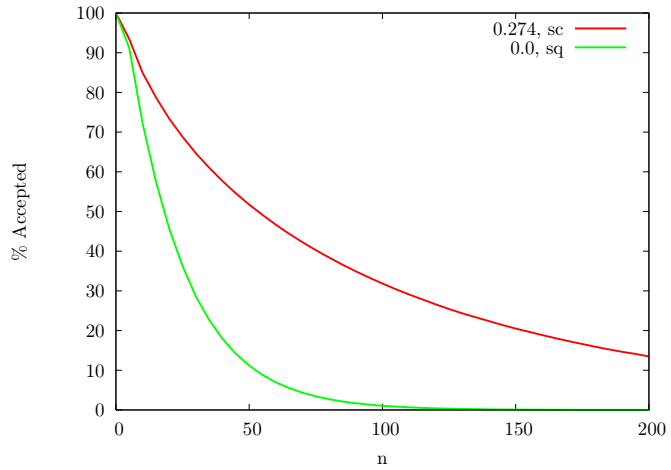



**Figure 8:** A plot of the acceptance rate for the CBMC method against chain length $n$ on the simple cubic lattice for $\phi = 0.274$ and the square lattice for $\phi = 0.0$. This shows that CBMC is inefficient given large values of $n$, as many more chains need to be generated in order to calculate correct statistics.

| | CBMC | | PERM | |
|---|---|---|---|---|
| | $n$ | Runtime | $n$ | Runtime |
| $\phi = 0.0$ | $10^8$ | 32:57:12 | $10^8$ | 00:48:24 |
| $\phi = 0.274$ | $2 \times 10^7$ | 05:44:35 | $10^8$ | 01:26:18 |

**Table 2:** CPU time required for the CBMC and PERM methods. All jobs were run using 2.6GHz AMD Opteron processors, having 512MB RAM available.

On the other hand, PERM is an excellent choice of algorithm for this system. Since the temperatures involved are relatively high, with sensible choices of parameters it can be made extremely efficient. For instance, it is not unreasonable for PERM to generate in the order of $10^9$ chains given less than half a day of CPU time on a reasonably fast machine.

As mentioned before, when dealing with an arbitrary number of chains (as opposed to a single chain in our case), CBMC may outperform PERM, since each chain will not need to be grown from scratch. As such, it is somewhat unfair to draw a comparison between the CPU time required for each algorithm. However, a brief overview of runtimes encountered is given in table 2.

# 5 Flat Histogram PERM

An inherent problem with both CBMC and PERM is their inability to generate correct statistics for low-temperature polymers. The reason for this is the same in either choice of algorithm. Given $T_1 < T_2$, we see that $\beta_1 > \beta_2$; since $\epsilon < 0$, the Boltzmann factors for monomer-monomer interaction satisfy $\exp(-\epsilon\beta_1) > \exp(-\epsilon\beta_2)$.

Since the bias introduced for both methods favours choosing directions that have larger Boltzmann factors, the generated walks become more compact as the temperature increases. This then increases the probability that the walk will become trapped at some point before the desired length.

To illustrate this point, consider figure 9, where we measure the number of chains generated at

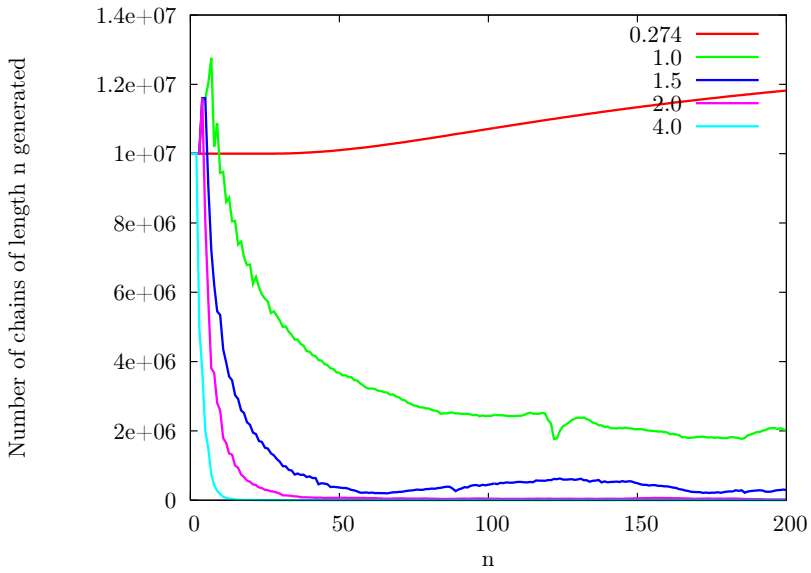



**Figure 9:** Number of chains of length $n$ plotted against $n$ for the PERM algorithm at inverse temperatures $\phi = 0.274, 1.0, 1.5, 2.0$ and $4.0$. As the temperature decreases, PERM becomes extremely inefficient. Each run here started with $10^7$ initial subroutine calls.

each step by PERM for increasing values of $\phi$. Whilst the number remains reasonably constant for $\phi = 0.274$, as soon as the temperature is decreased too much, the method suffers immensely from the small quantity of chains being generated. In this regard, at low temperatures both CBMC and PERM suffer greatly.

Whilst nPERM improves upon the performance of PERM somewhat at these lower temperatures, there is still the significant problem that some areas of the space of configurations (namely, those with many or very few contacts) will not be sampled. A new algorithm, called the "flat histogram" version of PERM – or flatPERM – has been created [17] which, as the name implies, is designed to sample evenly from the entire configuration space.

The way this is done is, in fact, relatively simple. Normally PERM is used in the thermal sense; as we have seen above, it is used to find an estimate for a the partition function $Z_n(\beta)$ at a specific inverse temperature $\beta$. However, it is very easy to modify PERM to estimate the total number of configurations of length $n$, given by $C_n$.

Furthermore, we can 'break down' this picture even further; every polymer chain must additionally have a certain number of monomer-monomer contacts $m$. This is called the *microcanonical* ensemble, as it is the simplest representation of our polymer chain. flatPERM is a modification of PERM that attempts to estimate the number of configurations of length $n$ with $m$ monomer-monomer contacts, $C_{n,m}$. This is known as the *density of states*.

## 5.1 The algorithm

In order to estimate the density of states, we first need to see how the PERM algorithm can be modified to find $C_n$. We define $a_n$ to be the number of ways that a particular configuration $\varphi_n$ of length $n$, can be extended. Then, define the weight of $\varphi_n$ to be

$$W_n = \prod_{k=0}^{n-1} a_k.$$

We can then obtain an estimate for the total number of configurations $C_n^{\text{est}}$ by looking at all generated chains of length $n$, $\varphi_n^{(i)}$, by evaluating

$$C_n^{\text{est}} = \langle W_n \rangle = \frac{1}{S} \sum_i W_n^{(i)}.$$

The mean is taken with respect to the total number of initial growth chains $S$ (and thus includes configurations which may not have reached length $n$). Although this differs slightly from the original Rosenbluth algorithm, in which we consider the Boltzmann factors of the associated monomers, we can obtain an estimate of the partition function by calculating

$$Z_n^{\text{est}}(\beta) = \sum_i W_n^{(i)} \exp(-\beta E_n^{(i)}),$$

where $E_n^{(i)}$ is the energy of $\varphi^{(i)}$. This then recovers the Boltzmann factors, effectively giving us an implementation of the Rosenbluth algorithm in a slightly different fashion. We now reintroduce the pruning and enrichment, but using the nPERM technique of weight prediction [8]. Set $r = W_n^{(i)}/C_n^{\text{est}}$. Then we enrich if $r > 1$, or else prune if $r < 1$.

Notice that unlike the original PERM algorithm, we *constantly* prune or enrich at every stage of construction. This helps us to investigate the entire configuration space, and the choice of $r$ ensures that the weights are as close as possible to $C_n^{\text{est}}$.

- $r > 1$: enrichment step. Make $c = \min(\lfloor r \rfloor, a_n)$ *distinct* copies, each with weight $\frac{1}{c} W_n^{(i)}$ as in nPERM.
- $r < 1$: pruning step. Continue growing with probability $r$ and weight $\frac{1}{r} W_n^{(i)} = C_n^{\text{est}}$; otherwise, stop growth of this chain.

At this point, we would usually choose to apply PERM to a thermal ensemble, by estimating the partition function as mentioned above. However, at this step it is extremely simple to estimate $C_{n,m}$, given by

$$C_{n,m}^{\text{est}} = \langle W_{n,m} \rangle = \frac{1}{S} \sum_i W_{n,m}^{(i)},$$

that is, the mean over all generated samples $\varphi_{n,m}^{(i)}$ of length $n$ and size $m$, with respective weights $W_{n,m}^{(i)}$. To do this, we change the pruning and enrichment procedures above by replacing $C_n^{\text{est}}$ with $C_{n,m}^{\text{est}}$ and using $r = W_{n,m}^{(i)}/C_{n,m}^{\text{est}}$.

More intuitively, through the use of pruning and enrichment to keep weights approximately constant, PERM ensures that the number of samples of length $n$ will be kept roughly constant. Indeed, figure 9 shows this is the case for $\phi = 0.274$. So, in some respects, PERM is already a flat histogram method.

However, since no attention is paid to the number of monomer-monomer contacts, it is very likely that many areas of the configuration space with very many or very few contacts will be sampled, in proportion to the number of samples obtained. By obtaining microcanonical estimates for $C_{n,m}$ and using the flat-histogram tendencies of PERM, we ensure that most of the configuration space is sampled during our run.

For an observable value $A_{n,m}^{(i)}$ of a configuration $\varphi_{n,m}^{(i)}$, estimates for the average value of $A_{n,m}$ can be obtained by collecting weighted sums of the observables,

$$A_{n,m}^{\text{est}} = \frac{\langle A_{n,m} W_{n,m} \rangle}{\langle W_{n,m} \rangle} = \frac{\sum_i A_{n,m}^{(i)} W_{n,m}^{(i)}}{\sum_i W_{n,m}^{(i)}}.$$

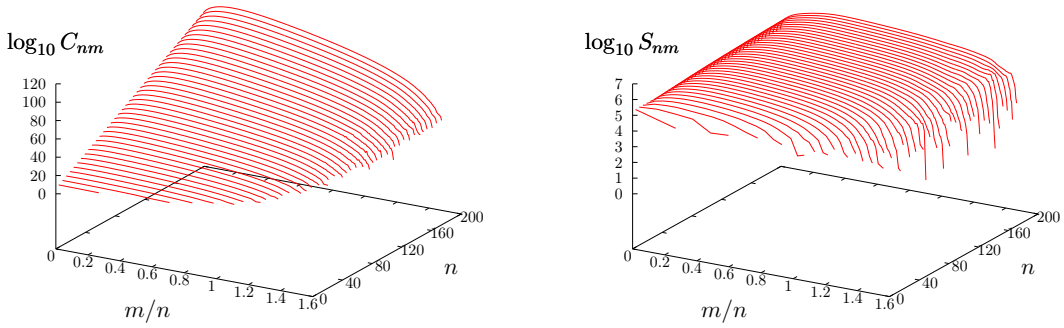



**Figure 10:** Base-10 logarithm of $C_{n,m}$ (left) and $S_{n,m}$ (right) against the internal energy $m/n$ and chain length $n$ on the simple cubic lattice.

From these values, we can obtain an estimate for the expected value of $A_n$ at an inverse temperature $\beta$ by calculating

$$A_n^{\text{est}}(\beta) = \frac{\sum_m A_{n,m}^{\text{est}} C_{n,m}^{\text{est}} \exp(-\beta E_m)}{\sum_m C_{n,m}^{\text{est}} \exp(-\beta E_m)} = \frac{\sum_m \sum_i A_{n,m}^{(i)} W_{n,m}^{(i)} \exp(-\beta E_m)}{\sum_m \sum_i W_{n,m}^{(i)} \exp(-\beta E_m)}. \tag{4}$$

Notice that the last expression is independent of $S$. Additionally, we can see that only two 2-dimensional arrays are required for the entire algorithm; one to store $\sum_i A_{n,m}^{(i)} W_{n,m}^{(i)}$, and the other to store $\sum_i W_{n,m}^{(i)}$. The latter can be used to obtain $C_{n,m}^{\text{est}}$ and hence $r$ during pruning/enrichment.

### 5.2 Initial applications

The first runs using the flatPERM algorithm checked that the program output was consistent with the results outlined in [17]. The same model was used; as before, the polymer was allowed to grow in free space with attractive monomer-monomer energy $\epsilon$. All of these runs were performed on the simple cubic lattice.

Since in this scenario we are no longer favouring walks with higher Boltzmann factors, the algorithm is effectively generating self-avoiding walks with $\epsilon = 0$. PERM is extremely efficient at doing this, and so flatPERM should provide an excellent alternative.

This time, both $C_{n,m}^{\text{est}}$ and $S_{n,m}$ were measured and the results can be seen in figure 10. The plot of $\log_{10} S_{n,m}$ clearly shows why flatPERM is called a flat-histogram method. Although $S_{n,m}$ decreases somewhat as the internal energy $m/n$ (i.e. number of contacts per chain) tends approaches 1, there are no chains with $m = n$ and so this is to be expected. Everywhere else in the configuration space is reasonably evenly covered.

The real power of the flatPERM algorithm is that we can recover statistics on the polymer at *any* given temperature in only *one* simulation; both CBMC and PERM require that a separate simulation be run for every temperature. Whilst a single flatPERM simulation may take more time to run than one PERM/CBMC, much more information can be obtained about the polymer at both very high and very low temperatures.

As an example of how this additional information can help discover and predict new phenomenon, we now consider two final examples involving attractive surfaces, and try to build a phase diagram showing the different behaviour of the polymer chain for a variety of parameters.

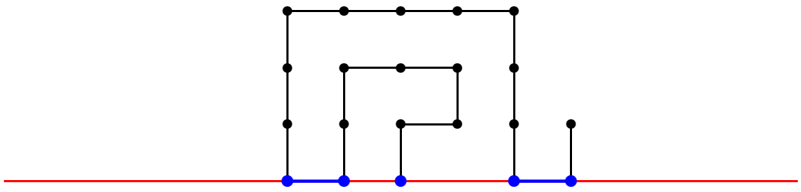


**Figure 11:** An example of a polymer tethered to an attractive surface. The long red line denotes the surface, with blue dots denoting points of contact with the surface and blue lines being bonds across the surface. The first monomer is *always* attached to the surface.

### 5.3 Tethered polymers

In this first example, we consider the problem of a polymer chain tethered to an attractive surface at a single point (as illustrated by figure 11), as in [11]. The aim is to investigate and identify changes in the behaviour of an average polymer as the attractive energies between surface/monomer and monomer/monomer are varied – i.e. discover potential areas of phase transition.

With the polymer now being allowed to interact with a surface, our microcanonical ensemble must also be parametrised by $n_s$, the number of monomer-surface interactions. Hence we will wish to approximate $C_{n,n_s,n_m}$; that is, the number of configurations of length $n$, with $n_s$ monomer-surface contacts and $n_m$ monomer-monomer contacts.

Although we only considered the density of states of a system involving a single parameter $m$ in the previous section, the extension to two parameters is reasonably clear. Simply substituting $C_{n,n_m,n_s}$ for $C_{n,m}$ and using $r = W^{(i)}_{n,n_s,n_m}/C^{\text{est}}_{n,n_m,n_s}$ will change flatPERM from one to two parameters.

We also consider an attractive monomer-surface energy $\epsilon_s < 0$, which may not be the same as the monomer-monomer energy $\epsilon_m$. Then the energy of any particular configuration $\varphi_n$ of length $n$ is given by

$$E_n(\varphi_n) = \epsilon_s n_s + \epsilon_m n_m,$$

where $n_s, n_m$ are defined as above. Hence, the partition function for all chains of length $n$ is given by

$$Z_n(\beta) = \sum_{n_m,n_s} C_{n,n_m,n_s} \exp(-\beta E_n).$$

By defining $\beta_m = \epsilon_m \beta$ and $\beta_s = \epsilon_s \beta$ we obtain

$$Z_n(\beta_m, \beta_s) = \sum_{n_m,n_s} C_{n,n_m,n_s} \exp(-\beta_m n_m - \beta_s n_s).$$

The calculation of $Z_n(\beta_m, \beta_s)$ is easy from the flatPERM algorithm; indeed, no observable data needs to be recorded apart from our estimates of $C_{n,n_m,n_s}$. However, for large values of $n_m$ and $n_s$ the exponential term involved in the summation becomes extremely large. Even if $C_{n,n_m,n_s}$ is relatively small, overflow errors can easily occur when using double precision floating point numbers.

However, if we were to deal with the logarithms of this data, the quantities involved would be much smaller and well within the bounds of even single point precision. Employing the use of logarithmic coding [3] was essential for calculating reasonably accurate values of the data. The premise is reasonably simple; given $C = A + B$, we estimate $\log C = \log(A + B)$ from the values

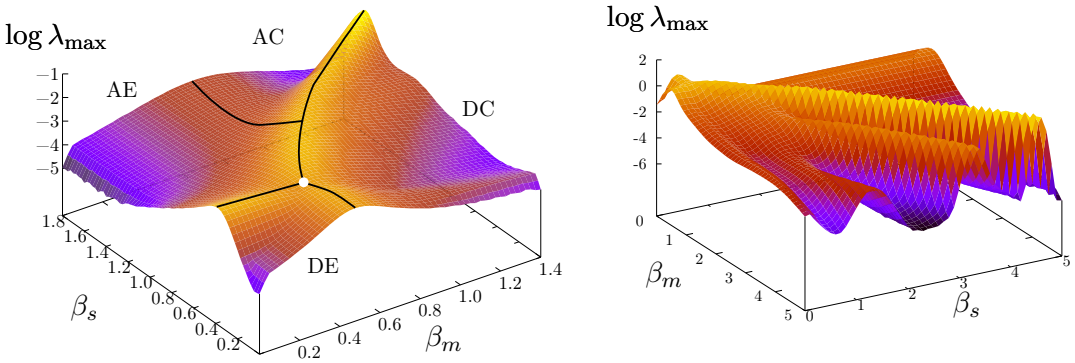



**Figure 12:** Plots of the maximum logarithm of the Hessian of $\log Z_n(\beta_m, \beta_s)$ for $n = 100$. On the left, we take $(\beta_m, \beta_s) \in [0, 1.4] \times [0, 1.8]$, and on the right we have $0 \leq \beta_m, \beta_s \leq 5$. The four phase transitions are marked on the plot, and their appropriate behaviour is listed below.

of $\log A$ and $\log B$, without needing to store $A$ and $B$ in memory. eq. (20) from [3] states that

$$\begin{aligned}\log C &= \log\left[\max(A, B) + \left(1 + \frac{\min(A, B)}{\max(A, B)}\right)\right] \\ &= \max(\log A, \log B) + \log\left[1 + \exp\left\{\min(\log A, \log B) - \max(\log A, \log B)\right\}\right].\end{aligned}$$

Whilst this logarithmic addition is reasonably expensive (involving two conditional statements, one logarithm and an exponentiation as opposed to a single much simpler CPU operation), it does offer a easy way to bypass the overflow issue. Additionally this coding is only done after the flatPERM algorithm has completed and so the efficiency is not so much of an issue.

In most papers, the specific heat capacity $C_V$ is used to identify possible areas of phase transition, as peaks in $C_V$ usually correspond to areas of different behaviour. However, for the purposes of corroborating evidence, we consider the Hessian of the free energy, $\log Z(\beta_m, \beta_s)$, where the Hessian $H(f)$ of a function $f : \mathbb{R}^2 \to \mathbb{R}$, is given by

$$H(f) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

i.e. the matrix of second derivatives of $f$.

By calculating the eigenvalues for each value of $(\beta_m, \beta_s)$ and by taking the maximum of the two, we can study the areas of maximal and minimal free energy. In turn, this tells us about the typical types of conformation we can observe at these points.

Figure 12 shows two plots. The first considers $(\beta_m, \beta_s) \in [0, 1.4] \times [0, 1.8]$, giving output for a modest range of temperatures. However, since flatPERM has considered the microcanonical states, we can extend this diagram to a much larger range of temperatures. This is shown in the second plot, covering the range $(\beta_m, \beta_s) \in [0, 5.0]^2$. We would be unable to do this using CBMC and PERM unless given a significant amount of processing time.

The first temperature range is reasonably well understood, and so provides a suitable check that the method is in fact working correctly. There are four particular areas of interest, as indicated on the diagram, identifying different types of behaviour for the polymer:

- **DE**: Desorbed, expanded – the polymer is desorbed from the surface and is in an expanded state.

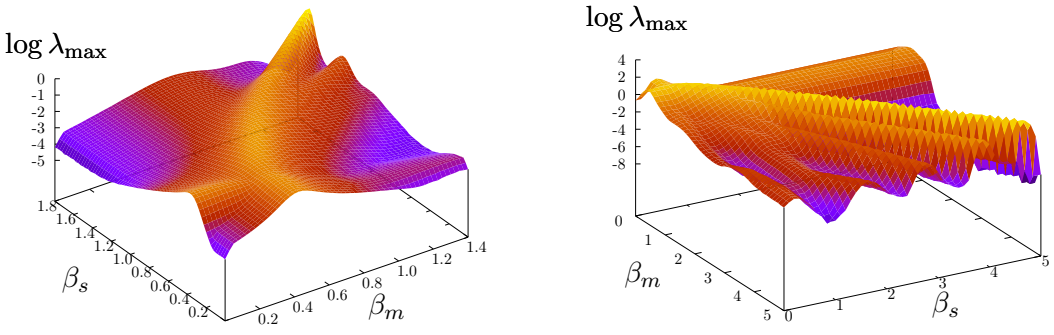



**Figure 13:** Plots of the maximum logarithm of the Hessian of $\log Z_n(\beta_m, \beta_s)$ for $n = 200$. As mentioned, there is an additional layering transition which is clearly visible from the second plot; the left-hand plot is also affected by the additional transition.

- **AE**: Adsorbed, expanded – the polymer is adsorbed on the surface, but still remains expanded (i.e. like an expanded walk on the square lattice).
- **AC**: Adsorbed, compact – similar to above, but is now much more compact (i.e. $n_m$ is large).
- **DC**: Desorbed, compact – mostly desorbed from the surface (obviously one monomer must be attached), but in a compact state.

However, the second plot describes behaviour that has not been seen inside the classically considered range of temperatures.

A detailed explanation is supplied in [11]. The behaviour is described as a 'layering transition'; at each one of the peaks, the polymer is mostly constrained to a number of two-dimensional planes lying away from the surface, like a number of square lattices layered upon one another. The higher the peaks, the larger the number of layers involved.

For instance, let us fix $\beta_m = 5$ and consider $\beta_s = 0$. Then we are in the DC phase; as $\beta_s$ is increased, we see two layering transitions from 1-3 layers, before finally entering the standard AC phase (adsorbed and contracted).

As the length of the polymer increases (i.e. in the thermodynamic limit), these layering transitions will disappear, leaving only the ones predicted by standard theory. (More and more layers will be created, effectively merging all of the smaller peaks into the main phase transition from DC $\to$ AC.

This prediction is somewhat supported by figure 13, which shows the same phase diagram, except for $n = 200$. By counting the peaks in the right-hand plot, we can see that more layering transitions have appeared.

### 5.4 Free polymers

Whilst we have studied free polymers in an infinite volume and polymers tethered at one end to a surface, it would be interesting to try to combine the two approaches. That is, we wish to study the behaviour of a polymer which is *not* attached to a surface and has the freedom to move completely away, as in [1].

Obviously this leaves us with a significant problem; if the polymer could theoretically be any distance away from the surface, then $C_{n,n_m,n_s}$ diverges. We certainly need to constrain the

polymer in some fashion, otherwise the problem is impossible to model.

Instead, we introduce a steric (i.e. non-interacting) wall, at some distance $z_w$ away from the adsorbing surface, where $z_w$ is measured in lattice units. Additionally, let us choose $\epsilon_s = \epsilon_m = -1$, and instead set the energy of a configuration $\varphi_n$ of length $n$ to be

$$E_n(\varphi_n) = -n_s - sn_m,$$

where $n_s$ and $n_m$ are as in the previous section. We add an additional parameter, $s$, which controls the quality of the solvent. If $s$ is large, then the solvent is 'bad', and the polymer will tend to contract. If $s$ is small then the solvent is 'good', and polymer will be strung out. (More about good and bad solvents can be found in [2].) We will consider what happens to the polymer over a range of temperatures for various values of $s$.

Clearly this is a problem for which flatPERM is well suited and is indeed quite similar to the problem previously considered. The partition function is again given by

$$Z_n(T, s) = \sum_{n_m, n_s} C_{n,n_m,n_s} \exp(-\beta E_n) = \sum_{n_m, n_s} C_{n,n_m,n_s} \exp(\{n_s + sn_m\}/k_B T).$$

This time, however, we consider the heat capacity $C_V$ of the polymer and attempt to build a phase diagram involving $s$ and $T$. Given $Z_n(T)$, it is possible to calculate $C_V$, but this is reasonably non-trivial. A much better approach is to note that

$$C_V = \frac{1}{k_B T^2} \mathrm{Var}(E) = \frac{1}{k_B T^2} \left( \langle E^2 \rangle - \langle E \rangle^2 \right),$$

since

$$\langle E^2 \rangle = \frac{1}{Z(\beta)} \frac{\partial^2 Z}{\partial \beta^2},$$

and hence

$$\langle E^2 \rangle - \langle E \rangle^2 = \frac{\partial^2}{\partial \beta^2} \log Z(\beta) = k_B T^2 C_V$$

by eq. (2) in section 3.2. So, by measuring the variance of the energy of the generated polymers, we obtain a value which is proportional to the heat capacity. Additionally, by placing the steric wall away by a reasonable distance, we can allow polymers to grow freely as well as obtain results about surface interactions; in these simulations, $z_w = 200$. The results can be seen in figure 14.

Here, we can see a much richer variety of behaviour split over six distinct phases. By considering $s < 0$, we additionally consider what happens when the monomer-monomer energies change from being attractive to repulsive.

The black lines indicate the local maxima in $C_V$, as in figure 12. These indicate six areas of interest which are believed to be the phase transition boundaries in the thermodynamic limit. (There are also various other local maxima which will disappear; these are not highlighted.)

The two-letter combinations AC, DC etc are as in the previous section. Here, the numbers after AC and AE indicate a layering effect; i.e. AC2 is an adsorbed, compact polymer spread over two layers. Unlike the tethered polymer, these transitions do not disappear as chain length is increased.

To consider an example of the typical behaviour of a polymer in this system, take $s = 2$. We start at $T = 0$, where the configuration is in the AC2 phase; $n_m$ is maximal, and the monomers occupy sites on the surface and the sites directly above the surface.

As the temperature is increased past $T \approx 1.2$, we see a transition into the AE2 stage, where the polymer now expands, but still only occupies two-layers. When $T$ reaches 2.75, the polymer now desorbs from the surface, but remains compact; past $T = 4$ there is a final transition into the desorbed, expanded phase.

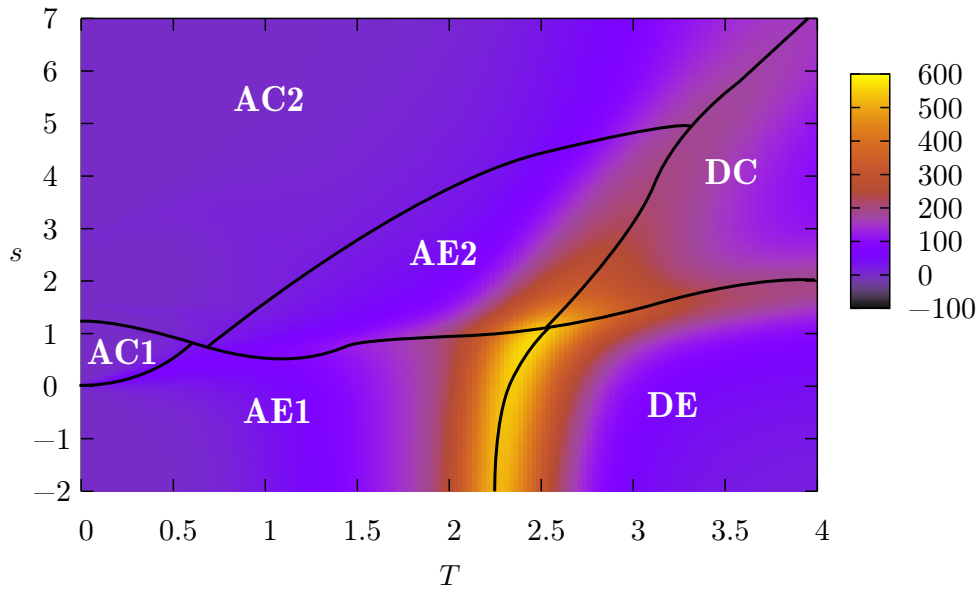



**Figure 14:** Phase diagram for the specific heat capacity $C_V$, for $0 \leq T \leq 4$ and $-2 \leq s \leq 7$. The distance between the two walls is $z_w = 200$.

## 6 Extensions and conclusions

In this report, we have only considered homopolymers – that is, polymers which are constructed from only one type of monomer. A simple extension would be to apply these algorithms to the HP model, a toy model for proteins (which are of crucial interest in biology and medicine). In this scenario, we have two types of monomer; hydrophobic (H) and polar (P). Hydrophobic monomers dislike solvent, and will bunch together. On the other hand, polar molecules mix well with the solvent.

We can simulate this behaviour by ensuring that hydrophobic molecules are attracted to each other, but all other types of interactions are disregarded. For example, we set $\epsilon_{HH} = -1$, and $\epsilon_{PP} = \epsilon_{PH} = \epsilon_{HP} = 0$, where $\epsilon$ denotes the monomer-monomer interaction energy.

An interesting problem is attempting to find the ground states of specific chains of HP polymers. The flatPERM routine is applied to find this for an 85-monomer HP chain on the square lattice in [17]. This could easily be extended further to cases where an attractive surface is included.

Obviously most polymer systems are far more complicated than the ones presented here. They usually consist of thousands of monomers of many types in a variety of different geometries. As mentioned previously, although these methods can be extended to general Euclidean space, it may be more beneficial to instead consider lattices with large co-ordination numbers.

We may also wish to study many polymer chains at the same time which are allowed to interact with one another. In this case, CBMC and dynamic PERM have a distinct advantage over PERM and flatPERM.

The main algorithm considered here, flatPERM, does rely heavily on the fact that we are in a microcanonical ensemble, and for complex systems of many parameters an efficient implementation is quite difficult. For these systems, it is best to include the least important variables in the weighting, including the more imporant parameters in the microcanonical sense. (For example, by 'least important' we mean those parameters which will not affect efficiency overly).

In summary, due to the Monte Carlo methods involved, all of these algorithms are reasonably flexible and efficient. Unlike standard molecular dynamics algorithms, they can be applied to a

wide range of problems with only a reasonably small amount of effort. Certainly the efficiency of the method is affected by the parameters we choose, and the environment it is expected to perform in, as shown even in the simple cases considered here.

Ultimately, they provide an excellent way of modelling polymers given a wide range of parameters and variety of different systems. In turn, this allows us to study and predict interesting properties for many different systems, increasing our understanding of how polymers behave and paving the way for further advances.

### Acknowledgements

I am extremely grateful to my supervisor Prof. Mike Allen for his support and advice throughout the project which was invaluable. I would also like to thank Dr. Hsiao-Ping Hsu of the Institut für Physik from Johannes Gutenberg University for example source code, which sped up development of the PERM algorithm immensely.

Additionally, my thanks go to David White and Sarah Chandler for taking the time to proof read this report and offer general advice.

Finally, I am very grateful for the computing facilities provided by staff and computer administrators at the Centre for Scientific Computing at the University of Warwick, without whose time and maintenance, none of the results presented here could have been obtained.

## 7 Appendix: PERM Pseudo-code

The code listing below is an outline of the routine that can be used to generate a SAW using the PERM algorithm. The global variables are

- `w`: Weight of the chain.
- `cpos`: Current position (i.e. monomer number).
- $Z[\cdot]$: Partition function.
- $R2[\cdot]$: Mean square end-to-end distance.

The parameters are

- $b[\cdot]$: An array containing the Boltzmann factors for a particular number of monomer-monomer interactions. Namely,
$$b(m) = e^{m\phi}$$
for $0 \leq m \leq k$.
- `W1, W2`: Upper and lower thresholds $W^{>}, W^{<}$ for pruning/enrichment respectively.
- `max_length`: Maximum length of the chain.

```
subroutine grow()
  Place a monomer at the current position, cpos;
  Measure available/taken sites around this position;
  Let w = w * b[taken];

  Collect statistics:
    Z[cpos] += + w;
    R2[cpos] += w*(R2 for this chain);

  if (avail > 0 && cpos < max_length) {
    cpos++;
```

```
    Move at random to one of the available sites;

    if (w > W1) {
      w /= 2;
      grow(); grow();
      w *= 2;
    } else if (w < W2) {
      if (random number in Uniform(0,1) < 1/2) {
        w *= 2;
        grow();
        w /= 2;
      }
      // At this point, if the random number was > 1/2 the
      // chain has been pruned and stops growing
    } else {
      grow();
    }

    Move back to the previous position
    cpos--;
  }

  Reset weight to previous value
  Remove the current monomer from its position
end subroutine
```

## References


[1] M. Bachmann and W. Janke. Conformational transitions of nongrafted polymers near an absorbing substrate. *Phys. Rev. Lett.*, 95(5):058102, 2005.

[2] J. Baschnagel, J. P. Wittmer, and H. Meyer. Monte Carlo simulation of polymers: Coarse-grained models, 2004. arXiv:cond-mat/0407717.

[3] B.A. Berg. Multicanonical simulations step by step. *Comput. Phys. Commun*, 153(3):397–406, 2005.

[4] N. Combe, T.J.H. Vlugt, P.R.T. Wolde, and D. Frenkel. Dynamic pruned-enriched Rosenbluth method. *Mol. Phys.*, 101(11):1675–1682.

[5] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 2nd edition, 2002.

[6] P. Grassberger and H. Frauenkron. PERM: A Monte Carlo strategy for simulating polymers and other things, 1998. arXiv:cond-mat/9806321.

[7] A. J. Guttmann. On the critical behaviour of self-avoiding walks. *J. Phys. A*, 20(7):1839–1854, 1987.

[8] H-P. Hsu, V. Mehra, W. Nadler, and P.Grassberger. Growth-based optimization algorithm for lattice heteropolymers. *Phys. Rev. E*, 68(2):021113, Aug 2003.

[9] I. Hueter. Proof of the conjecture that the planar self-avoiding walk has root mean square displacement exponent 3/4, 2001. arXiv:math/0108077.

[10] I. Jensen. Enumeration of self-avoiding walks on the square lattice. *J. Phys. A*, 37(21):5503–5524, 2004.

[11] J. Krawczyk, A. L. Owczarek, T. Prellberg, and A. Rechnitzer. Layering transitions for adsorbing polymers in poor solvents. *EPL*, 70(6):726–732, 2005.

[12] K. Kremer and K. Binder. Monte Carlo simulation of lattice models for macromolecules. *Comp. Phys. Rep*, 7(6):259–310, 1988.

[13] H. Mairovitch and H.A. Lim. Computer simulation study of the $\theta$-point in three dimensions. I. Self avoiding walks on the simple cubic lattice. *J. Chem. Phys*, 92(8):5144–5154, 1990.

[14] F.L. McCrackin, J. Mazur, and C.M. Guttman. Monte Carlo studies of self-interacting polymer chains with excluded volume. I. Squared radii of gyration and mean-square end-to-end distances and their moments. *Macromolecules*, 6(6):859–871, 1973.

[15] B. Nienhuis. Exact critical point and critical exponents of $o(n)$ models in two dimensions. *Phys. Rev. Lett.*, 49(15):1062–1065, Oct 1982.

[16] P.Grassberger. Pruned-enriched Rosenbluth method: Simulations of $\theta$ polymers of chain length up to 1,000,000. *Phys. Rev. E*, 56(3):3682–3693, Sep 1997.

[17] T. Prellberg and J. Krawczyk. Flat histogram version of the pruned and enriched Rosenbluth method. *Phys. Rev. Lett.*, 92(12):120602, 2004.

[18] M.N. Rosenbluth and A.W.Rosenbluth. Monte Carlo simulations of the average extension of molecular chains. *J. Chem. Phys.*, 23:356–359, 1955.

[19] E. Thönnes. Warwick University course notes for ST407: Monte Carlo methods. Available from: `http://go.warwick.ac.uk/st407`.

[20] Eric W. Weisstein. Buffon's needle problem [online]. Available from: `http://mathworld.wolfram.com/BuffonsNeedleProblem.html`.