Problem 7.27. Example of least squares fit

a. Write a program to find the least squares fit for a set of data. As a check on your program, compute the most probable values of m and b for the data shown in Table 7.2.

b. Modify the random walk program so that steps of length 1 and 2 are taken with equal probability. Use at least 10000 trials and do a least squares fit to Δx^2 as done in the text. Is your most probable estimate for ν closer to $\nu = 1/2$?

For simple random walk problems the relation $\Delta x^2 = aN^\nu$ holds for all N. However, in many random walk problems a power law relation between Δx^2 and N holds only asymptotically for large N, and hence we should use only the larger values of N to estimate the slope. Also, because we are finding the best fit for the logarithm of the independent variable N, we need to give equal weight to all intervals of $\ln N$. In the above example, we used $N = 8$, 16, 32, and 64, so that the values of $\ln N$ are equally spaced.

7.7 Applications to Polymers

Random walk models play an important role in polymer physics (cf. de Gennes). A polymer consists of N repeat units (monomers) with $N >> 1$ ($N \sim 10^3$ – 10^5). For example, polyethylene can be represented as $\cdots -\mathrm{CH_2}-\mathrm{CH_2}-CH_2-\cdots$. The detailed structure of the polymer is important for many practical applications. For example, if we wish to improve the fabrication of rubber, a good understanding of the local motions of the monomers in the rubber chain is essential. However, if we are interested in the global properties of the polymer, the details of the chain structure can be ignored.

Let us consider a familiar example of a polymer chain in a good solvent: a noodle in warm water. A short time after we place a noodle in warm water, the noodle becomes flexible, and it neither collapses into a little ball or becomes fully stretched. Instead, it adopts a random structure as shown schematically in Figure 7.6. If we do not add too many noodles, we can say that the noodles behave as a dilute solution of polymer chains in a good solvent. The dilute nature of the solution implies that we can ignore entanglement effects of the noodles and consider each noodle individually. The presence of a good solvent implies that the polymers can move freely and adopt many different configurations.

A fundamental geometrical property that characterizes a polymer in a good solvent is the mean square end-to-end distance $\langle R_N^2 \rangle$, where N is the number of monomers. (For simplicity, we will frequently write R^2 in the following.) For a dilute solution of polymer chains in a good solvent, it is known that the asymptotic dependence of R^2 is given by (7.12) with $\nu \approx 0.5874$ in three dimensions. If we were to ignore the interactions of the monomers, the simple random walk model would yield $\nu = 1/2$, independent of the dimension and symmetry of the lattice. Because this result for ν does not agree with experiment, we know that we are overlooking an important physical feature of polymers.

We now discuss a random walk that incorporates the global features of dilute linear polymers in solution. We already have introduced a model of a polymer chain consisting of straight line segments of the same size joined together at random angles (see Problem 7.13). A further idealization

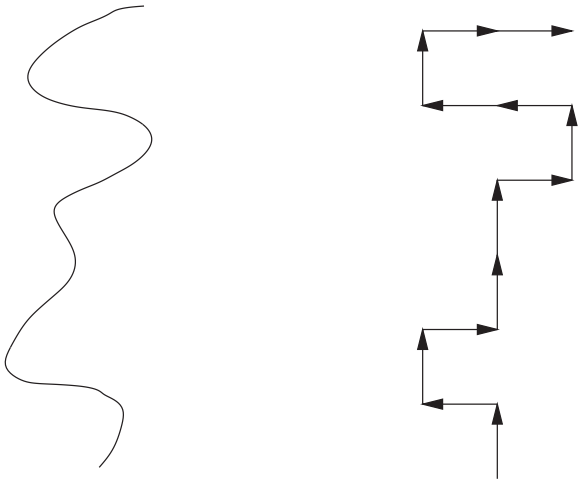

Figure 7.6: (a) Schematic illustration of a linear polymer in a good solvent. (b) Example of the corresponding self-avoiding walk on a square lattice.

is to place the polymer chain on a lattice (see Figure 7.6). A more realistic model of linear polymers accounts for its most important physical feature, that is, two monomers cannot occupy the same spatial position. This constraint is known as the *excluded volume* condition, which is ignored in a simple random walk. A well known lattice model for a linear polymer chain that incorporates this constraint is known as the *self-avoiding* walk (SAW). This model consists of the set of all N step walks starting from the origin subject to the global constraint that no lattice site can be visited more than once in each walk; this constraint accounts for the excluded volume condition.

Self-avoiding walks have many applications, such as the physics of magnetic materials and the study of phase transitions, and they are of interest as purely mathematical objects. Many of the obvious questions have resisted rigorous analysis, and exact enumeration and Monte Carlo simulation have played an important role in our current understanding. The result for ν in two dimensions for the self-avoiding walk is known to be exactly $\nu = 3/4$. The proportionality constant in (7.12) depends on the structure of the monomers and on the solvent. In contrast, the exponent ν is independent of these details and depends only on the spatial dimension.

We consider Monte Carlo simulations of the self-avoiding walk in two dimensions in Problems 7.28–7.30. Another algorithm for the self-avoiding walk is considered in Project 7.41.

Problem 7.28. The two-dimensional self-avoiding walk

Consider the self-avoiding walk on the square lattice. Choose an arbitrary site as the origin and assume that the first step is "up." The walks generated by the three other possible initial directions only differ by a rotation of the whole lattice and do not have to be considered explicitly. The second step can be in three rather than four possible directions because of the constraint that the walk cannot return to the origin. To obtain unbiased results, we generate a random number to choose

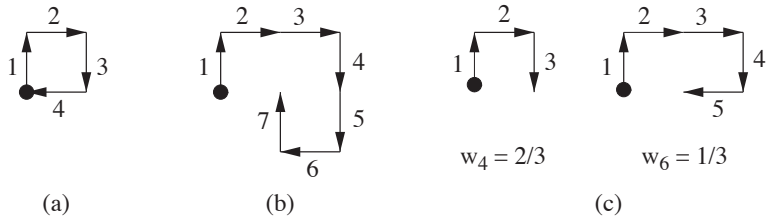

Figure 7.7: Examples of self-avoiding walks on a square lattice. The origin is denoted by a filled circle. (a) A $N = 3$ walk. The fourth step shown is forbidden. (b) A $N = 7$ walk that leads to a self-intersection at the next step; the weight of the $N = 8$ walk is zero. (c) Two examples of the weights of walks in the enrichment method.

one of the three directions. Successive steps are generated in the same way. Unfortunately, the walk will very likely not continue indefinitely. To obtain unbiased results, we must choose at random one of the three steps, even though one or more of these steps might lead to a self-intersection. If the next step does lead to a self-intersection, the walk must be terminated to keep the statistics unbiased. An example of a three step walk is shown in Figure 7.7a. The next step leads to a self-intersection and violates the constraint. In this case we must start a new walk at the origin.

a. Write a program that implements this algorithm and record the fraction $f(N)$ of successful attempts at constructing polymer chains with N total monomers. Represent the lattice as a array so that you can record the sites that already have been visited. What is the qualitative dependence of $f(N)$ on N? What is the maximum value of N that you can reasonably consider?

b. Determine the mean square end-to-end distance $\langle R_N^2 \rangle$ for values of N that you can reasonably consider with this sampling method.

The disadvantage of the straightforward sampling method in Problem 7.28 is that it becomes very inefficient for long chains, that is, the fraction of successful attempts decreases exponentially. To overcome this attrition, several "enrichment" techniques have been developed. We first discuss a relatively simple algorithm proposed by Rosenbluth and Rosenbluth in which each walk of N steps is associated with a weighting function $w(N)$. Because the first step to the north is always possible, we have $w(1) = 1$. In order that all allowed configurations of a given N are counted equally, the weights $w(N)$ for $N > 1$ are determined according to the following possibilities:

1. All three possible steps violate the self-intersection constraint (see Figure 7.7b). The walk is terminated with a weight $w(N) = 0$, and a new walk is generated at the origin.

2. All three steps are possible and $w(N) = w(N - 1)$.

3. Only m steps are possible with $1 \leq m < 3$ (see Figure 7.7c). In this case $w(N) = (m/3)w(N-1)$, and one of the m possible steps is chosen at random.

The desired unbiased value of $\langle R^2 \rangle$ is obtained by weighting R_i^2, the value of R^2 obtained in the ith trial, by the value of $w_i(N)$, the weight found for this trial. Hence we write

$$\langle R^2 \rangle = \frac{\sum_i w_i(N) R_i^2}{\sum_i w_i(N)}, \tag{7.46}$$

where the sum is over all trials.

Problem 7.29. Rosenbluth and Rosenbluth enrichment method

Incorporate the Rosenbluth method into your Monte Carlo program and compute R^2 for $N = 4$, 8, 16, and 32. Estimate the exponent ν from a log-log plot of R^2 versus N. Can you distinguish your estimate for ν from its random walk value $\nu = 1/2$?

The Rosenbluth and Rosenbluth procedure is not particularly efficient because many walks still terminate, and thus we do not obtain many walkers for large N. Grassberger improved this algorithm by increasing the population of walkers with high weights and reducing the population of walkers with low weights. The idea is that if $w(N)$ for a given trial is above a certain threshold, we add a new walker and give the new and old walker half of the original weight. If $w(N)$ is below a certain threshold, then we eliminate half of the walkers with weights below this threshold (for example, every second walker) and double the weights of the remaining half. It is a good idea to adjust the thresholds as the simulation runs in order to maintain a relatively constant number of walkers.

More recently Prellberg and Krawczyk further improved the Rosenbluth and Rosenbluth enrichment method so that there is no need to provide a threshold value. After each step the average weight of the walkers, $\langle w(N) \rangle$ is computed for a given trial and the ratio $r = w(N)/\langle w(N) \rangle$ is used to determine whether to add walkers (enrichment) or eliminate walkers (pruning). If $r > 1$, then $c = \min(r, m)$ copies of the walker are made each with weight $w(N)/c$. If $r < 1$, then remove this walker with probability $1 - r$. This algorithm leads to an approximately constant number of walkers and is related to the Wang-landau method which we will discuss in Problem 16.30.

Another enrichment algorithms is the "reptation" method (see Wall and Mandel). For simplicity, consider a model polymer chain in which all bond angles are $\pm 90°$. As an example of this model, the five independent $N = 5$ polymer chains are shown in Figure 7.8. (Other chains differ only by a rotation or a reflection.) The reptation method can be stated as follows:

1. Choose a chain at random and remove the tail link.
2. Attempt to add a link to the head of the chain. There is a maximum of two directions in which the new head link can be added.
3. If the attempt violates the self-intersection constraint, return to the original chain and interchange the head and tail. Include the chain in the statistical sample.

The above steps are repeated many times to obtain an estimate of R^2.

As an example of the reptation method, consider chain a of Figure 7.8. A new link can be added in two directions (see Figure 7.9a), so that on the average we find, $a \to \frac{1}{2}c + \frac{1}{2}d$. In contrast, a link can be added to chain b in only one direction, and we obtain $b \to \frac{1}{2}e + \frac{1}{2}b$, where the tail and

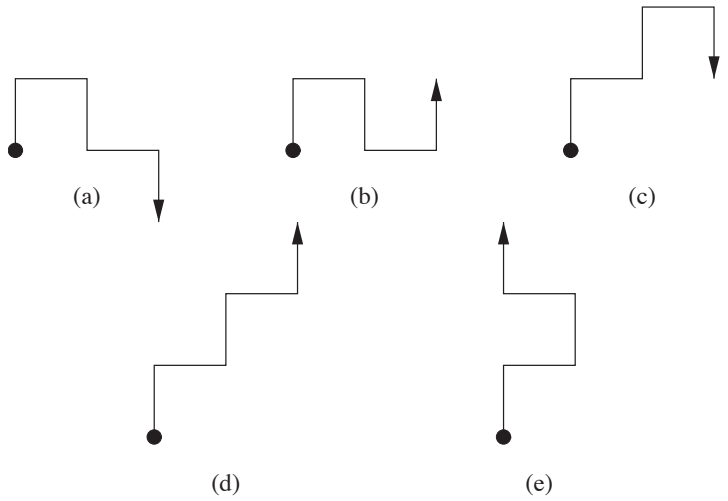

Figure 7.8: The five independent possible walks of $N = 5$ steps on a square lattice with $\pm 90^\circ$ bond angles. The tail and head of each walk are denoted by a circle and arrow respectively.

head of chain b have been interchanged (see Figure 7.9b). Confirm that $c \to \frac{1}{2}e + \frac{1}{2}a$, $d \to \frac{1}{2}c + \frac{1}{2}d$, and $e \to \frac{1}{2}a + \frac{1}{2}b$, and that all five chains are equally probable. That is, the transformations in the reptation method preserve the proper statistical weights of the chains without attrition. There is just one problem: unless we begin with a double ended "cul-de-sac" configuration such as shown in Figure 7.10, we will never obtain such a configuration using the above transformation. Hence, the reptation method introduces a small statistical bias, and the calculated mean end-to-end distance will be slightly larger than if all configurations were considered. However, the probability of such trapped configurations is very small, and the bias can be neglected for most purposes.

*__Problem 7.30.__ The reptation method

a. Adopt the $\pm 90^\circ$ bond angle restriction and calculate by hand the exact value of $\langle R^2 \rangle$ for $N = 5$. Then write a Monte Carlo program that implements the reptation method. Generate one walk of $N = 5$ and use the reptation method to generate a statistical sample of chains. As a check on your program, compute $\langle R^2 \rangle$ for $N = 5$ and compare your result with the exact result. Then extend your Monte Carlo computations of $\langle R^2 \rangle$ to larger N.

b. Modify the reptation model so that the bond angle also can be 180°. This modification leads to a maximum of three directions for a new bond. Compare your results with those from part (a).

In principle, the dynamics of a polymer chain undergoing collisions with solvent molecules can be simulated by using a molecular dynamics method. However, in practice only relatively small chains can be simulated in this way. An alternative approach is to use a Monte Carlo model that simplifies the effect of the random collisions of the solvent molecules with the atoms of the chain. Most of these models (cf. Verdier and Stockmayer) consider the chain to be composed of beads

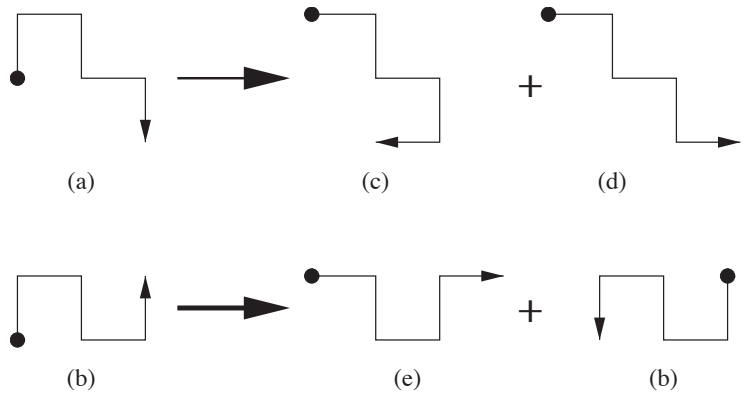

Figure 7.9: The possible transformations of chains a and b. One of the two possible transformations of chain b violates the self-intersection restriction and the head and tail are interchanged.

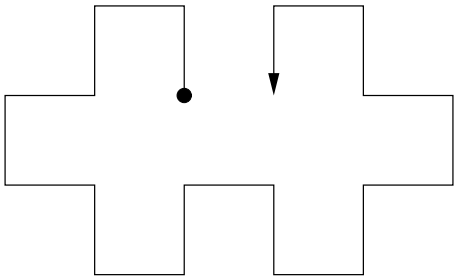

Figure 7.10: Example of a double cul-de-sac configuration for the self-avoiding walk that cannot be obtained by the reptation method.

connected by bonds and restrict the positions of the beads to the sites of a lattice. For simplicity, we assume that the bond angles can be either $\pm 90°$ or $180°$. The idea is to begin with an allowed configuration of N beads ($N-1$ bonds). A possible starting configuration can be generated by taking successive steps in the positive y direction and positive x directions. The dynamics of the Verdier-Stockmayer algorithm is summarized by the following steps.

1. Select at random a bead (occupied site) on the polymer chain. If the bead is not an end site, then the bead can move to a nearest neighbor site of another bead if this site is empty and if the new angle between adjacent bonds is either $\pm 90°$ or $180°$. For example, bead 4 in Figure 7.11 can move to position $4'$ while bead 3 cannot move if selected. That is, a selected bead can move to a diagonally opposite unoccupied site only if the two bonds to which it is attached are mutually perpendicular.

2. If the selected bead is an end site, move it to one of two (maximum) possible unoccupied sites so that the bond to which it is connected changes its orientation by $\pm 90°$ (see Figure 7.11).

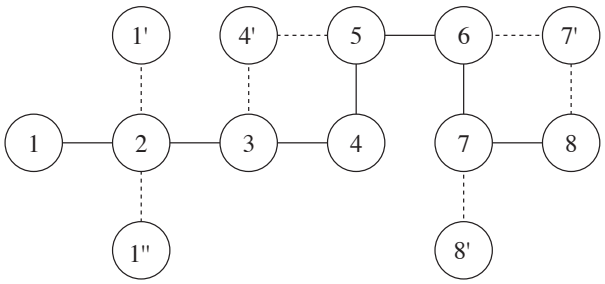

Figure 7.11: Examples of possible moves of the simple polymer dynamics model considered in Problem 7.31. For this configuration beads 2, 3, 5, and 6 cannot move, while beads 1, 4, 7, and 8 can move to the positions shown if they are selected. Only one bead can move at a time. This figure is adopted from the article by Verdier and Stockmayer.

3. If the selected bead cannot move, retain the previous configuration.

The physical quantities of interest include $\langle R^2 \rangle$ and the mean square displacement of the center of mass of the chain $\langle r^2 \rangle = \langle x^2 \rangle - \langle x \rangle^2 + \langle y^2 \rangle - \langle y \rangle^2$, where x and y are the coordinates of the center of mass. The unit of time is the number of Monte Carlo steps per bead; in one Monte Carlo step per bead each bead has one chance on the average to move to a different site.

Another efficient method for simulating the dynamics of a polymer chain is the bond fluctuation model (see Carmesin and Kremer).

Problem 7.31. The dynamics of polymers in a dilute solution

a. Consider a two-dimensional lattice and compute $\langle R^2 \rangle$ and $\langle r^2 \rangle$ for various values of N. How do these quantities depend on N? (The first published results for three dimensions were limited to 32 Monte Carlo steps per bead for $N = 8$, 16, and 32 and only 8 Monte Carlo steps per bead for $N = 64$.) Also compute the probability density $P(R)$ that the end-to-end distance is R. How does this probability compare to a Gaussian distribution?

b.* Two configurations are strongly correlated if they differ by only the position of one bead. Hence, it would be a waste of computer time to measure the end-to-end distance and the position of the center of mass after every single move. Ideally, we wish to compute these quantities for configurations that are approximately statistically independent. Because we do not know *a priori* the mean number of Monte Carlo steps per bead needed to obtain configurations that are statistically independent, it is a good idea to estimate this time in our preliminary calculations. The correlation time, τ, is the time needed to obtain statistically independent configurations and can be obtained by computing the equilibrium averaged time-autocorrelation function for a chain of fixed N:

$$C(t) = \frac{\langle R^2(t'+t)R^2(t') \rangle - \langle R^2 \rangle^2}{\langle R^4 \rangle - \langle R^2 \rangle^2}. \tag{7.47}$$

$C(t)$ is defined so that $C(t = 0) = 1$ and $C(t) = 0$ if the configurations are not correlated. Because the configurations will become uncorrelated if the time t between the configurations