Chapter 9

Random walks and the Metropolis algorithm

> Nel mezzo del cammin di nostra vita, mi ritrovai per una selva oscura, ché la diritta via era smarrita. (Divina Commedia, Inferno, Canto I, 1-3)*Dante Alighieri*
>
> The way that can be spoken of is not the constant way. (Tao Te Ching, Book I, I.1)*Lao Tzu*

9.1 Motivation

In the previous chapter we discussed technical aspects of Monte Carlo integration such as algorithms for generating random numbers and integration of multidimensional integrals. The latter topic served to illustrate two key topics in Monte Carlo simulations, namely a proper selection of variables and importance sampling. An intelligent selection of variables, good sampling techniques and guiding functions can be crucial for the outcome of our Monte Carlo simulations. Examples of this will be demonstrated in the chapters on statistical and quantum physics applications. Here we make a detour however from this main area of applications. The focus is on diffusion and random walks. The rationale for this is that the tricky part of an actual Monte Carlo simulation resides in the appropriate selection of random states, and thereby numbers, according to the probability distribution (PDF) at hand. With appropriate there is however much more to the picture than meets the eye.

Suppose our PDF is given by the well-known normal distribution. Think of for example the velocity distribution of an ideal gas in a container. In our simulations we could then accept or reject new moves with a probability proportional to the normal distribution. This would parallel our example on the sixth dimensional integral in the previous chapter. However, in this case we would end up rejecting basically all moves since the probabilities are exponentially small in most cases. The result would be that we barely moved from the initial position. Our statistical averages would then be significantly biased and most likely not very reliable.

Instead, all Monte Carlo schemes used are based on Markov processes in order to generate new random states. A Markov process is a random walk with a selected probability for making a move. The new move is independent of the previous history of the system. The Markov process is used repeatedly in Monte Carlo simulations in order to generate new random states. The reason for choosing a Markov process is that when it is run for a long enough time starting with a random state, we will eventually reach the most likely state of the system. In thermodynamics, this means that after a certain number of

Markov processes we reach an equilibrium distribution. This mimicks the way a real system reaches its most likely state at a given temperature of the surroundings.

To reach this distribution, the Markov process needs to obey two important conditions, that of ergodicity and detailed balance. These conditions impose then constraints on our algorithms for accepting or rejecting new random states. The Metropolis algorithm discussed here abides to both these constraints and is discussed in more detail in Section 9.5. The Metropolis algorithm is widely used in Monte Carlo simulations of physical systems and the understanding of it rests within the interpretation of random walks and Markov processes. However, before we do that we discuss the intimate link between random walks, Markov processes and the diffusion equation. In section 9.3 we show that a Markov process is nothing but the discretized version of the diffusion equation. Diffusion and random walks are discussed from a more experimental point of view in the next section. There we show also a simple algorithm for random walks and discuss eventual physical implications. We end this chapter with a discussion of one of the most used algorithms for generating new steps, namely the Metropolis algorithm. This algorithm, which is based on Markovian random walks satisfies both the ergodicity and detailed balance requirements and is widely in applications of Monte Carlo simulations in the natural sciences. The Metropolis algorithm is used in our studies of phase transitions in statistical physics and the simulations of quantum mechanical systems.

9.2 Diffusion equation and random walks

Physical systems subject to random influences from the ambient have a long history, dating back to the famous experiments by the British Botanist R. Brown on pollen of different plants dispersed in water. This lead to the famous concept of Brownian motion. In general, small fractions of any system exhibit the same behavior when exposed to random fluctuations of the medium. Although apparently non-deterministic, the rules obeyed by such Brownian systems are laid out within the framework of diffusion and Markov chains. The fundamental works on Brownian motion were developed by A. Einstein at the turn of the last century.

Diffusion and the diffusion equation are central topics in both Physics and Mathematics, and their ranges of applicability span from stellar dynamics to the diffusion of particles governed by Schrödinger's equation. The latter is, for a free particle, nothing but the diffusion equation in complex time!

Let us consider the one-dimensional diffusion equation. We study a large ensemble of particles performing Brownian motion along the x-axis. There is no interaction between the particles.

We define $w(x,t)dx$ as the probability of finding a given number of particles in an interval of length dx in $x \in [x, x+dx]$ at a time t. This quantity is our probability distribution function (PDF). The quantum physics equivalent of $w(x,t)$ is the wave function itself. This diffusion interpretation of Schrödinger's equation forms the starting point for diffusion Monte Carlo techniques in quantum physics.

9.2.1 Diffusion equation

From experiment there are strong indications that the flux of particles $j(x,t)$, viz., the number of particles passing x at a time t is proportional to the gradient of $w(x,t)$. This proportionality is expressed mathematically through

$$j(x,t) = -D\frac{\partial w(x,t)}{\partial x}, \tag{9.1}$$

where D is the so-called diffusion constant, with dimensionality length^2 per time. If the number of particles is conserved, we have the continuity equation

$$\frac{\partial j(x,t)}{\partial x} = -\frac{\partial w(x,t)}{\partial t}, \tag{9.2}$$

which leads to

$$\frac{\partial w(x,t)}{\partial t} = D\frac{\partial^2 w(x,t)}{\partial x^2}, \tag{9.3}$$

which is the diffusion equation in one dimension.

With the probability distribution function $w(x,t)dx$ we can use the results from the previous chapter to compute expectation values such as the mean distance

$$\langle x(t)\rangle = \int_{-\infty}^{\infty} xw(x,t)dx, \tag{9.4}$$

or

$$\langle x^2(t)\rangle = \int_{-\infty}^{\infty} x^2 w(x,t)dx, \tag{9.5}$$

which allows for the computation of the variance $\sigma^2 = \langle x^2(t)\rangle - \langle x(t)\rangle^2$. Note well that these expectation values are time-dependent. In a similar way we can also define expectation values of functions $f(x,t)$ as

$$\langle f(x,t)\rangle = \int_{-\infty}^{\infty} f(x,t)w(x,t)dx. \tag{9.6}$$

Since $w(x,t)$ is now treated as a PDF, it needs to obey the same criteria as discussed in the previous chapter. However, the normalization condition

$$\int_{-\infty}^{\infty} w(x,t)dx = 1 \tag{9.7}$$

imposes significant constraints on $w(x,t)$. These are

$$w(x=\pm\infty,t) = 0 \qquad \frac{\partial^n w(x,t)}{\partial x^n}|_{x=\pm\infty} = 0, \tag{9.8}$$

implying that when we study the time-derivative $\partial\langle x(t)\rangle/\partial t$, we obtain after integration by parts and using Eq. (9.3)

$$\frac{\partial\langle x\rangle}{\partial t} = \int_{-\infty}^{\infty} x\frac{\partial w(x,t)}{\partial t}dx = D\int_{-\infty}^{\infty} x\frac{\partial^2 w(x,t)}{\partial x^2}dx, \tag{9.9}$$

leading to

$$\frac{\partial\langle x\rangle}{\partial t} = Dx\frac{\partial w(x,t)}{\partial x}|_{x=\pm\infty} - D\int_{-\infty}^{\infty}\frac{\partial w(x,t)}{\partial x}dx, \tag{9.10}$$

implying that

$$\frac{\partial\langle x\rangle}{\partial t} = 0. \tag{9.11}$$

This means in turn that $\langle x\rangle$ is independent of time. If we choose the initial position $x(t=0)=0$, the average displacement $\langle x\rangle = 0$. If we link this discussion to a random walk in one dimension with equal probability of jumping to the left or right and with an initial position $x=0$, then our probability

distribution remains centered around $\langle x\rangle = 0$ as function of time. However, the variance is not necessarily 0. Consider first

$$\frac{\partial\langle x^2\rangle}{\partial t} = Dx^2\frac{\partial w(x,t)}{\partial x}|_{x=\pm\infty} - 2D\int_{-\infty}^{\infty} x\frac{\partial w(x,t)}{\partial x}dx, \tag{9.12}$$

where we have performed an integration by parts as we did for $\frac{\partial\langle x\rangle}{\partial t}$. A further integration by parts results in

$$\frac{\partial\langle x^2\rangle}{\partial t} = -Dxw(x,t)|_{x=\pm\infty} + 2D\int_{-\infty}^{\infty} w(x,t)dx = 2D, \tag{9.13}$$

leading to

$$\langle x^2\rangle = 2Dt, \tag{9.14}$$

and the variance as

$$\langle x^2\rangle - \langle x\rangle^2 = 2Dt. \tag{9.15}$$

The root mean square displacement after a time t is then

$$\sqrt{\langle x^2\rangle - \langle x\rangle^2} = \sqrt{2Dt}. \tag{9.16}$$

This should be contrasted to the displacement of a free particle with initial velocity v_0. In that case the distance from the initial position after a time t is $x(t) = vt$ whereas for a diffusion process the root mean square value is $\sqrt{\langle x^2\rangle - \langle x\rangle^2} \propto \sqrt{t}$. Since diffusion is strongly linked with random walks, we could say that a random walker escapes much more slowly from the starting point than would a free particle. We can vizualize the above in the following figure. In Fig. 9.1 we have assumed that our distribution is given by a normal distribution with variance $\sigma^2 = 2Dt$, centered at $x = 0$. The distribution reads

$$w(x,t)dx = \frac{1}{\sqrt{4\pi Dt}}\exp{(-\frac{x^2}{4Dt})}dx. \tag{9.17}$$

At a time $t = 2$s the new variance is $\sigma^2 = 4D$s, implying that the root mean square value is $\sqrt{\langle x^2\rangle - \langle x\rangle^2} = 2\sqrt{D}$. At a further time $t = 8$ we have $\sqrt{\langle x^2\rangle - \langle x\rangle^2} = 4\sqrt{D}$. While time has elapsed by a factor of 4, the root mean square has only changed by a factor of 2. Fig. 9.1 demonstrates the spreadout of the distribution as time elapses. A typical example can be the diffusion of gas molecules in a container or the distribution of cream in a cup of coffee. In both cases we can assume that the the initial distribution is represented by a normal distribution.

9.2.2 Random walks

Consider now a random walker in one dimension, with probability R of moving to the right and L for moving to the left. At $t = 0$ we place the walker at $x = 0$, as indicated in Fig. 9.2. The walker can then jump, with the above probabilities, either to the left or to the right for each time step. Note that in principle we could also have the possibility that the walker remains in the same position. This is not implemented in this example. Every step has length $\Delta x = l$. Time is discretized and we have a jump either to the left or to the right at every time step. Let us now assume that we have equal probabilities for jumping to the left or to the right, i.e., $L = R = 1/2$. The average displacement after n time steps is

$$\langle x(n)\rangle = \sum_i^n \Delta x_i = 0 \qquad \Delta x_i = \pm l, \tag{9.18}$$

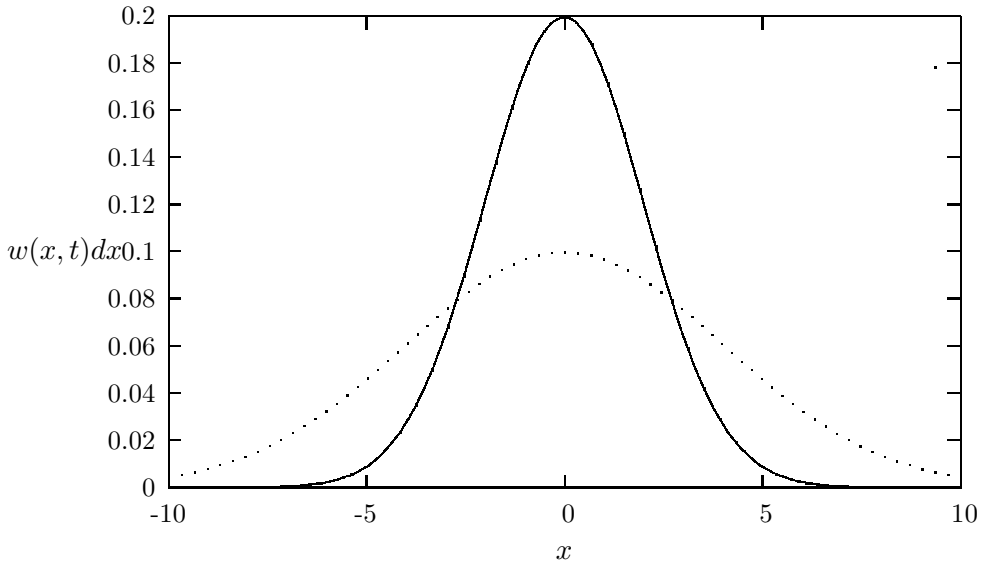

Figure 9.1: Time development of a normal distribution with variance $\sigma^2 = 2Dt$ and with $D = 1\text{m}^2/\text{s}$. The solid line represents the distribution at $t = 2\text{s}$ while the dotted line stands for $t = 8\text{s}$.

Figure 9.2: One-dimensional walker which can jump either to the left or to the right. Every step has length $\Delta x = l$.

since we have an equal probability of jumping either to the left or to right. The value of $\langle x(n)^2\rangle$ is

$$\langle x(n)^2\rangle = \left(\sum_i^n \Delta x_i\right)^2 = \sum_i^n \Delta x_i^2 + \sum_{i\neq j}^n \Delta x_i \Delta x_j = l^2 n. \tag{9.19}$$

For many enough steps the non-diagonal contribution is

$$\sum_{i\neq j}^N \Delta x_i \Delta x_j = 0, \tag{9.20}$$

since $\Delta x_{i,j} = \pm l$. The variance is then

$$\langle x(n)^2\rangle - \langle x(n)\rangle^2 = l^2 n. \tag{9.21}$$

It is also rather straightforward to compute the variance for $L \neq R$. The result is

$$\langle x(n)^2\rangle - \langle x(n)\rangle^2 = 4LRl^2 n. \tag{9.22}$$

In Eq. (9.21) the variable n represents the number of time steps. If we define $n = t/\Delta t$, we can then couple the variance result from a random walk in one dimension with the variance from the diffusion equation of Eq. (9.15) by defining the diffusion constant as

$$D = \frac{l^2}{\Delta t}. \tag{9.23}$$

In the next section we show in detail that this is the case.

The program below demonstrates the simplicity of the one-dimensional random walk algorithm. It is straightforward to extend this program to two or three dimensions as well. The input is the number of time steps, the probability for a move to the left or to the right and the total number of Monte Carlo samples. It computes the average displacement and the variance for one random walker for a given number of Monte Carlo samples. Each sample is thus to be considered as one experiment with a given number of walks. The interesting part of the algorithm is described in the function mc_sampling. The other functions read or write the results from screen or file and are similar in structure to programs discussed previously. The main program reads the name of the output file from screen and sets up the arrays containing the walker's position after a given number of steps. The corresponding program for a two-dimensional random walk (not listed in the main text) is found under programs/chapter9/program2.cpp

http://folk.uio.no/mhjensen/fys3150/2005/programs/chapter9/program1.cpp

```
%\begin{lstlisting}[title={programs/chapter9/program1.cpp}]
/*
  1-dim random walk program.
  A walker makes several trials steps with
  a given number of walks per trial
*/
#include <iostream>
#include <fstream>
#include <iomanip>
#include "lib.h"
using namespace  std;
```

```
// Function to read in data from screen, note call by reference
void initialise(int&, int&, double&) ;
// The Mc sampling for random walks
void  mc_sampling(int, int, double, int *, int *);
// prints to screen the results of the calculations
void  output(int, int, int *, int *);

int main()
{
  int max_trials, number_walks;
  double move_probability;
  // Read in data
  initialise(max_trials, number_walks, move_probability) ;
  int *walk_cumulative = new int [number_walks+1];
  int *walk2_cumulative = new int [number_walks+1];
  for (int walks = 1; walks <= number_walks; walks++){
    walk_cumulative[walks] = walk2_cumulative[walks] = 0;
  } // end initialization of vectors
  // Do the mc sampling
  mc_sampling(max_trials, number_walks, move_probability,
              walk_cumulative, walk2_cumulative);
  // Print out results
  output(max_trials, number_walks, walk_cumulative,
         walk2_cumulative);
  delete [] walk_cumulative; // free memory
  delete [] walk2_cumulative;
  return 0;
} // end main function
```

The input and output functions are

```
void initialise(int& max_trials, int& number_walks, double& move_probability
    )
{
  cout << "Number of Monte Carlo trials =";
  cin >> max_trials;
  cout << "Number of attempted walks=";
  cin >> number_walks;
  cout << "Move probability=";
  cin >> move_probability;
}  // end of function initialise


void output(int max_trials, int number_walks,
            int *walk_cumulative, int *walk2_cumulative)
{
  ofstream ofile("testwalkers.dat");
  for( int  i = 1; i <=  number_walks; i++){
    double xaverage = walk_cumulative[i]/((double) max_trials);
    double x2average = walk2_cumulative[i]/((double) max_trials);
    double variance = x2average - xaverage*xaverage;
    ofile << setiosflags(ios::showpoint | ios::uppercase);
    ofile << setw(6) << i;
```

```
    ofile << setw(15) << setprecision(8) << xaverage;
    ofile << setw(15) << setprecision(8) << variance << endl;
  }
  ofile.close();
}  // end of function output
```

The algorithm is in the function mc_sampling and tests the probability of moving to the left or to the right by generating a random number.

```
void mc_sampling(int max_trials, int number_walks,
                 double move_probability, int *walk_cumulative,
                 int *walk2_cumulative)
{
  long idum;
  idum=-1;  // initialise random number generator
  for (int trial=1; trial <= max_trials; trial++){
    int position = 0;
    for (int walks = 1; walks <= number_walks; walks++){
      if (ran0(&idum) <= move_probability) {
        position += 1;
      }
      else {
        position -= 1;
      }
      walk_cumulative[walks] += position;
      walk2_cumulative[walks] += position*position;
    }  // end of loop over walks
  } // end of loop over trials
}   // end mc_sampling function
```

Fig. 9.3 shows that the variance increases linearly as function of the number of time steps, as expected from the analytic results. Similarly, the mean displacement in Fig. 9.4 oscillates around zero.

Exercise 9.1
Extend the above program to a two-dimensional random walk with probability $1/4$ for a move to the right, left, up or down. Compute the variance for both the x and y directions and the total variance.

9.3 Microscopic derivation of the diffusion equation

When solving partial differential equations such as the diffusion equation numerically, the derivatives are always discretized. Recalling our discussions from Chapter 3, we can rewrite the time derivative as

$$\frac{\partial w(x,t)}{\partial t} \approx \frac{w(i,n+1)-w(i,n)}{\Delta t}, \tag{9.24}$$

whereas the gradient is approximated as

$$D\frac{\partial^2 w(x,t)}{\partial x^2} \approx D\frac{w(i+1,n)+w(i-1,n)-2w(i,n)}{(\Delta x)^2}, \tag{9.25}$$

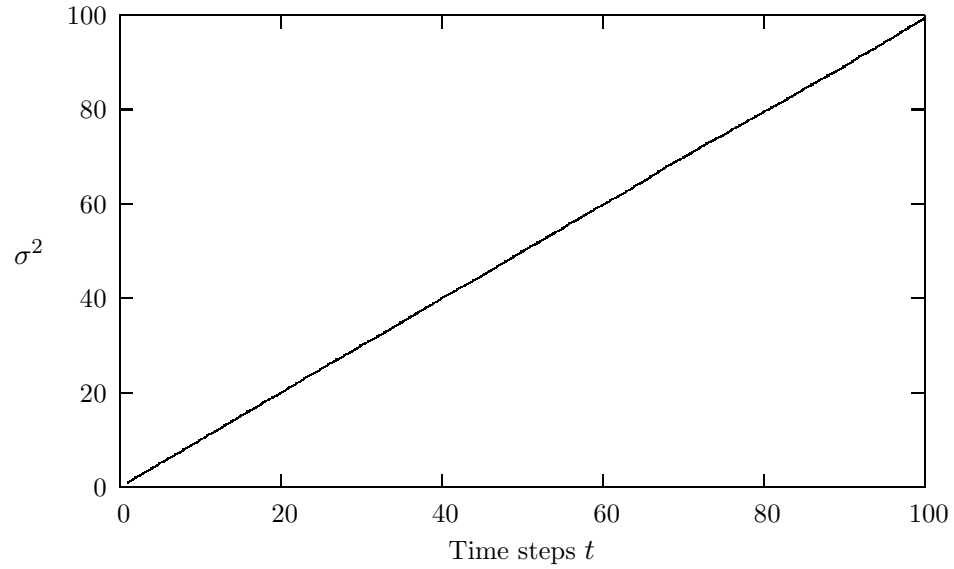

Figure 9.3: Time development of σ^2 for a random walker. 100000 Monte Carlo samples were used with the function ran1 and a seed set to -1.

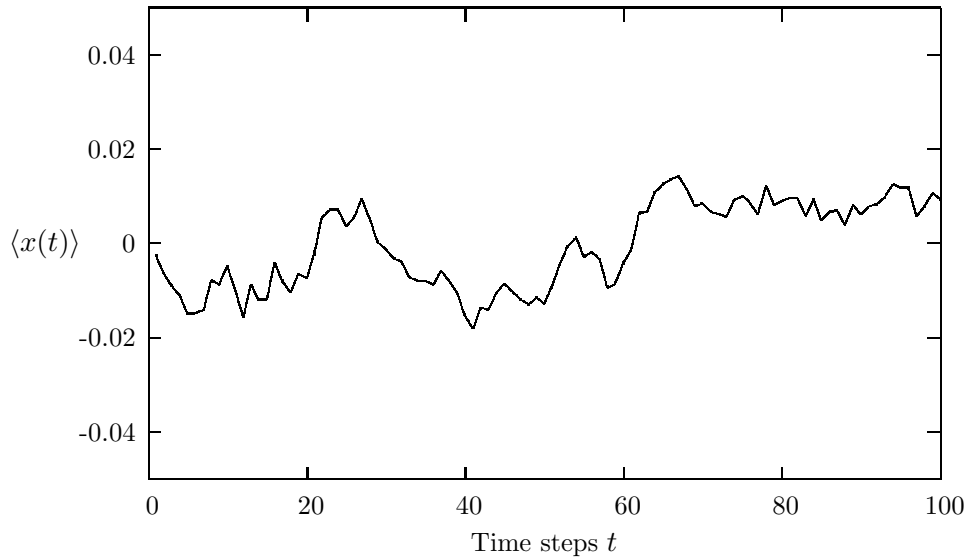

Figure 9.4: Time development of $\langle x(t) \rangle$ for a random walker. 100000 Monte Carlo samples were used with the function ran1 and a seed set to -1.

resulting in the discretized diffusion equation

$$\frac{w(i,n+1)-w(i,n)}{\Delta t}=D\frac{w(i+1,n)+w(i-1,n)-2w(i,n)}{(\Delta x)^2}, \tag{9.26}$$

where n represents a given time step and i a step in the x-direction. We will come back to the solution of such equations in our chapter on partial differential equations, see Chapter 15. The aim here is to show that we can derive the discretized diffusion equation from a Markov process and thereby demonstrate the close connection between the important physical process diffusion and random walks. Random walks allow for an intuitive way of picturing the process of diffusion. In addition, as demonstrated in the previous section, it is easy to simulate a random walk.

9.3.1 Discretized diffusion equation and Markov chains

A Markov process allows in principle for a microscopic description of Brownian motion. As with the random walk studied in the previous section, we consider a particle which moves along the x-axis in the form of a series of jumps with step length $\Delta x = l$. Time and space are discretized and the subsequent moves are statistically indenpendent, i.e., the new move depends only on the previous step and not on the results from earlier trials. We start at a position $x = jl = j\Delta x$ and move to a new position $x = i\Delta x$ during a step $\Delta t = \epsilon$, where $i \geq 0$ and $j \geq 0$ are integers. The original probability distribution function (PDF) of the particles is given by $w_i(t = 0)$ where i refers to a specific position on the grid in Fig. 9.2, with $i = 0$ representing $x = 0$. The function $w_i(t = 0)$ is now the discretized version of $w(x, t)$. We can regard the discretized PDF as a vector. For the Markov process we have a transition probability from a position $x = jl$ to a position $x = il$ given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i-j| = 1 \\ 0 & \text{else} \end{cases} \tag{9.27}$$

We call W_{ij} for the transition probability and we can represent it, see below, as a matrix. Our new PDF $w_i(t = \epsilon)$ is now related to the PDF at $t = 0$ through the relation

$$w_i(t = \epsilon) = W(j \rightarrow i)w_j(t = 0). \tag{9.28}$$

This equation represents the discretized time-development of an original PDF. It is a microscopic way of representing the process shown in Fig. 9.1. Since both W and w represent probabilities, they have to be normalized, i.e., we require that at each time step we have

$$\sum_i w_i(t) = 1, \tag{9.29}$$

and

$$\sum_j W(j \rightarrow i) = 1. \tag{9.30}$$

The further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$. Note that the probability for remaining at the same place is in general not necessarily equal zero. In our Markov process we allow only for jumps to the left or to the right.

The time development of our initial PDF can now be represented through the action of the transition probability matrix applied n times. At a time $t_n = n\epsilon$ our initial distribution has developed into

$$w_i(t_n) = \sum_j W_{ij}(t_n)w_j(0), \tag{9.31}$$

and defining

$$W(il - jl, n\epsilon) = (W^n(\epsilon))_{ij} \tag{9.32}$$

we obtain

$$w_i(n\epsilon) = \sum_j (W^n(\epsilon))_{ij} w_j(0), \tag{9.33}$$

or in matrix form

$$\hat{w}(n\epsilon) = \hat{W}^n(\epsilon)\hat{w}(0). \tag{9.34}$$

The matrix $\hat{W}$ can be written in terms of two matrices

$$\hat{W} = \frac{1}{2}\left(\hat{L} + \hat{R}\right), \tag{9.35}$$

where $\hat{L}$ and $\hat{R}$ represent the transition probabilities for a jump to the left or the right, respectively. For a 4×4 case we could write these matrices as

$$\hat{R} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{9.36}$$

and

$$\hat{L} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{9.37}$$

However, in principle these are infinite dimensional matrices since the number of time steps are very large or infinite. For the infinite case we can write these matrices $R_{ij} = \delta_{i,(j+1)}$ and $L_{ij} = \delta_{(i+1),j}$, implying that

$$\hat{L}\hat{R} = \hat{R}\hat{L} = I, \tag{9.38}$$

and

$$\hat{L} = \hat{R}^{-1} \tag{9.39}$$

To see that $\hat{L}\hat{R} = \hat{R}\hat{L} = 1$, perform e.g., the matrix multiplication

$$\hat{L}\hat{R} = \sum_k \hat{L}_{ik}\hat{R}_{kj} = \sum_k \delta_{(i+1),k}\delta_{k,(j+1)} = \delta_{i+1,j+1} = \delta_{i,j}, \tag{9.40}$$

and only the diagonal matrix elements are different from zero.

For the first time step we have thus

$$\hat{W} = \frac{1}{2}\left(\hat{L} + \hat{R}\right), \tag{9.41}$$

and using the properties in Eqs. (9.38) and (9.39) we have after two time steps

$$\hat{W}^2(2\epsilon) = \frac{1}{4}\left(\hat{L}^2 + \hat{R}^2 + 2\hat{R}\hat{L}\right), \tag{9.42}$$

and similarly after three time steps

$$\hat{W}^3(3\epsilon) = \frac{1}{8}\left(\hat{L}^3 + \hat{R}^3 + 3\hat{R}\hat{L}^2 + 3\hat{R}^2\hat{L}\right). \tag{9.43}$$

Using the binomial formula

$$\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\hat{a}^k\hat{b}^{n-k} = (a+b)^n, \tag{9.44}$$

we have that the transition matrix after n time steps can be written as

$$\hat{W}^n(n\epsilon)) = \frac{1}{2^n}\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\hat{R}^k\hat{L}^{n-k}, \tag{9.45}$$

or

$$\hat{W}^n(n\epsilon)) = \frac{1}{2^n}\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\hat{L}^{n-2k} = \frac{1}{2^n}\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\hat{R}^{2k-n}, \tag{9.46}$$

and using $R_{ij}^m = \delta_{i,(j+m)}$ and $L_{ij}^m = \delta_{(i+m),j}$ we arrive at

$$W(il-jl,n\epsilon) = \begin{cases} \frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i-j) \end{array}\right) & |i-j| \le n \\ 0 & \text{else} \end{cases}, \tag{9.47}$$

and $n+i-j$ has to be an even number. We note that the transition matrix for a Markov process has three important properties:

- It depends only on the difference in space $i-j$, it is thus homogenous in space.
- It is also isotropic in space since it is unchanged when we go from (i,j) to $(-i,-j)$.
- It is homogenous in time since it depends only the difference between the initial time and final time.

If we place the walker at $x = 0$ at $t = 0$ we can represent the initial PDF with $w_i(0) = \delta_{i,0}$. Using Eq. (9.34) we have

$$w_i(n\epsilon) = \sum_j (W^n(\epsilon))_{ij}w_j(0) = \sum_j \frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i-j) \end{array}\right)\delta_{j,0}, \tag{9.48}$$

resulting in

$$w_i(n\epsilon) = \frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i) \end{array}\right) \qquad |i| \le n \tag{9.49}$$

Using the recursion relation for the binomials

$$\left(\begin{array}{c} n+1 \\ \frac{1}{2}(n+1+i)) \end{array}\right) = \left(\begin{array}{c} n \\ \frac{1}{2}(n+i+1) \end{array}\right) + \left(\begin{array}{c} n \\ \frac{1}{2}(n+i)-1 \end{array}\right) \tag{9.50}$$

we obtain, defining $x = il$, $t = n\epsilon$ and setting

$$w(x,t) = w(il,n\epsilon) = w_i(n\epsilon), \tag{9.51}$$

$$w(x,t+\epsilon) = \frac{1}{2}w(x+l,t) + \frac{1}{2}w(x-l,t), \tag{9.52}$$

and adding and subtracting $w(x,t)$ and multiplying both sides with l^2/ϵ we have

$$\frac{w(x,t+\epsilon)-w(x,t)}{\epsilon} = \frac{l^2}{2\epsilon}\frac{w(x+l,t)-2w(x,t)+w(x-l,t)}{l^2}, \tag{9.53}$$

and identifying $D = l^2/2\epsilon$ and letting $l = \Delta x$ and $\epsilon = \Delta t$ we see that this is nothing but the discretized version of the diffusion equation. Taking the limits $\Delta x \to 0$ and $\Delta t \to 0$ we recover

$$\frac{\partial w(x,t)}{\partial t} = D\frac{\partial^2 w(x,t)}{\partial x^2},$$

the diffusion equation.

9.3.2 Continuous equations

Hitherto we have considered discretized versions of all equations. Our initial probability distribution function was then given by

$$w_i(0) = \delta_{i,0},$$

and its time-development after a given time step $\Delta t = \epsilon$ is

$$w_i(t) = \sum_j W(j \to i) w_j(t=0).$$

The continuous analog to $w_i(0)$ is

$$w(\mathbf{x}) \to \delta(\mathbf{x}), \tag{9.54}$$

where we now have generalized the one-dimensional position x to a generic-dimensional vector $\mathbf{x}$. The Kroenecker δ function is replaced by the δ distribution function $\delta(\mathbf{x})$ at $t = 0$.

The transition from a state j to a state i is now replaced by a transition to a state with position $\mathbf{y}$ from a state with position $\mathbf{x}$. The discrete sum of transition probabilities can then be replaced by an integral and we obtain the new distribution at a time $t + \Delta t$ as

$$w(\mathbf{y}, t+\Delta t) = \int W(\mathbf{y},\mathbf{x},\Delta t) w(\mathbf{x},t) d\mathbf{x}, \tag{9.55}$$

and after m time steps we have

$$w(\mathbf{y}, t+m\Delta t) = \int W(\mathbf{y},\mathbf{x},m\Delta t) w(\mathbf{x},t) d\mathbf{x}. \tag{9.56}$$

When equilibrium is reached we have

$$w(\mathbf{y}) = \int W(\mathbf{y},\mathbf{x},t) w(\mathbf{x}) d\mathbf{x}. \tag{9.57}$$

We can solve the equation for $w(\mathbf{y},t)$ by making a Fourier transform to momentum space. The PDF $w(\mathbf{x},t)$ is related to its Fourier transform $\tilde{w}(\mathbf{k},t)$ through

$$w(\mathbf{x},t) = \int_{-\infty}^{\infty} d\mathbf{k} \exp(i\mathbf{k}\mathbf{x}) \tilde{w}(\mathbf{k},t), \tag{9.58}$$

and using the definition of the δ-function

$$\delta(\mathbf{x}) = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\mathbf{k} \exp(i\mathbf{k}\mathbf{x}), \tag{9.59}$$

we see that

$$\tilde{w}(\mathbf{k},0) = 1/2\pi. \tag{9.60}$$

We can then use the Fourier-transformed diffusion equation

$$\frac{\partial \tilde{w}(\mathbf{k},t)}{\partial t} = -D\mathbf{k}^2\tilde{w}(\mathbf{k},t), \tag{9.61}$$

with the obvious solution

$$\tilde{w}(\mathbf{k},t) = \tilde{w}(\mathbf{k},0)\exp\left[-(D\mathbf{k}^2t)\right) = \frac{1}{2\pi}\exp\left[-(D\mathbf{k}^2t)\right]. \tag{9.62}$$

Using Eq. (9.58) we obtain

$$w(\mathbf{x},t) = \int_{-\infty}^{\infty} d\mathbf{k}\exp\left[i\mathbf{k}\mathbf{x}\right]\frac{1}{2\pi}\exp\left[-(D\mathbf{k}^2t)\right] = \frac{1}{\sqrt{4\pi Dt}}\exp\left[-(\mathbf{x}^2/4Dt)\right], \tag{9.63}$$

with the normalization condition

$$\int_{-\infty}^{\infty} w(\mathbf{x},t)d\mathbf{x} = 1. \tag{9.64}$$

It is rather easy to verify by insertion that Eq. (9.63) is a solution of the diffusion equation. The solution represents the probability of finding our random walker at position $\mathbf{x}$ at time t if the initial distribution was placed at $\mathbf{x} = 0$ at $t = 0$.

There is another interesting feature worth observing. The discrete transition probability W itself is given by a binomial distribution, see Eq. (9.47). The results from the central limit theorem, see Sect. 8.2.2, state that transition probability in the limit $n \to \infty$ converges to the normal distribution. It is then possible to show that

$$W(il - jl, n\epsilon) \to W(\mathbf{y},\mathbf{x},\Delta t) = \frac{1}{\sqrt{4\pi D\Delta t}}\exp\left[-((\mathbf{y}-\mathbf{x})^2/4D\Delta t)\right], \tag{9.65}$$

and that it satisfies the normalization condition and is itself a solution to the diffusion equation.

9.3.3 Numerical simulation

In the two previous subsections we have given evidence that a Markov process actually yields in the limit of infinitely many steps the diffusion equation. It links therefore in a physical intuitive way the fundamental process of diffusion with random walks. It could therefore be of interest to visualize this connection through a numerical experiment. We saw in the previous subsection that one possible solution to the diffusion equation is given by a normal distribution. In addition, the transition rate for a given number of steps develops from a binomial distribution into a normal distribution in the limit of infinitely many steps. To achieve this we construct in addition a histogram which contains the number of times the walker was in a particular position x. This is given by the variable probability , which is normalized in the output function. We have omitted the initialization function, since this identical to program1.cpp or program2.cpp of this chapter. The array probability extends from $-$number_walks to +number_walks

`http://folk.uio.no/mhjensen/fys3150/2005/programs/chapter9/program2.cpp`

```
%\begin{lstlisting}[title={programs/chapter9/program3.cpp}]
/*
  1-dim random walk program.
  A walker makes several trials steps with
```

```
  a given number of walks per trial
*/
#include <iostream>
#include <fstream>
#include <iomanip>
#include "lib.h"
using namespace  std;

// Function to read in data from screen, note call by reference
void initialise(int&, int&, double&) ;
// The Mc sampling for random walks
void  mc_sampling(int, int, double, int *, int *, int *);
// prints to screen the results of the calculations
void  output(int, int, int *, int *, int *);

int main()
{
  int max_trials, number_walks;
  double move_probability;
  // Read in data
  initialise(max_trials, number_walks, move_probability) ;
  int *walk_cumulative = new int [number_walks+1];
  int *walk2_cumulative = new int [number_walks+1];
  int *probability = new int [2*(number_walks+1)];
  for (int walks = 1; walks <= number_walks; walks++){
    walk_cumulative[walks] = walk2_cumulative[walks] = 0;
  }
  for (int walks = 0; walks <= 2*number_walks; walks++){
    probability[walks] = 0;
  } // end initialization of vectors
  // Do the mc sampling
  mc_sampling(max_trials, number_walks, move_probability,
              walk_cumulative, walk2_cumulative, probability);
  // Print out results
  output(max_trials, number_walks, walk_cumulative,
         walk2_cumulative, probability);
  delete [] walk_cumulative; // free memory
  delete [] walk2_cumulative; delete [] probability;
  return 0;
} // end main function
```

The output function contains now the normalization of the probability as well and writes this to its own file.

```
void output(int max_trials, int number_walks,
            int *walk_cumulative, int *walk2_cumulative, int * probability)
{
  ofstream ofile("testwalkers.dat");
  ofstream probfile("probability.dat");
  for( int  i = 1; i <=  number_walks; i++){
    double xaverage = walk_cumulative[i]/((double) max_trials);
    double x2average = walk2_cumulative[i]/((double) max_trials);
    double variance = x2average - xaverage*xaverage;
```

```
    ofile << setiosflags(ios::showpoint | ios::uppercase);
    ofile << setw(6) << i;
    ofile << setw(15) << setprecision(8) << xaverage;
    ofile << setw(15) << setprecision(8) << variance << endl;
  }
  ofile.close();
  // find norm of probability
  double norm = 0.;
  for( int  i = -number_walks; i <=  number_walks; i++){
    norm += (double) probability[i+number_walks];
  }
  // write probability
  for( int  i = -number_walks; i <=  number_walks; i++){
    double histogram = probability[i+number_walks]/norm;
    probfile << setiosflags(ios::showpoint | ios::uppercase);
    probfile << setw(6) << i;
    probfile << setw(15) << setprecision(8) << histogram << endl;
  }
  probfile.close();
}  // end of function output
```

The sampling part is still done in the same function, but contains now the setup of a histogram containing the number of times the walker visited a given position x.

```
void mc_sampling(int max_trials, int number_walks,
                 double move_probability, int *walk_cumulative,
                 int *walk2_cumulative, int *probability)
{
  long idum;
  idum=-1;  // initialise random number generator
  for (int trial=1; trial <= max_trials; trial++){
    int position = 0;
    for (int walks = 1; walks <= number_walks; walks++){
      if (ran0(&idum) <= move_probability) {
        position += 1;
      }
      else {
        position -= 1;
      }
      walk_cumulative[walks] += position;
      walk2_cumulative[walks] += position*position;
      probability[position+number_walks] += 1;
    }  // end of loop over walks
  } // end of loop over trials
}   // end mc_sampling function
```

Fig. 9.5 shows the resulting probability distribution after n steps We see from Fig. 9.5 that the probability distribution function resembles a normal distribution.

Exercise 9.2
Use the above program and try to fit the computed probability distribution with a normal distribution using your calculated values of σ^2 and $\langle x \rangle$.

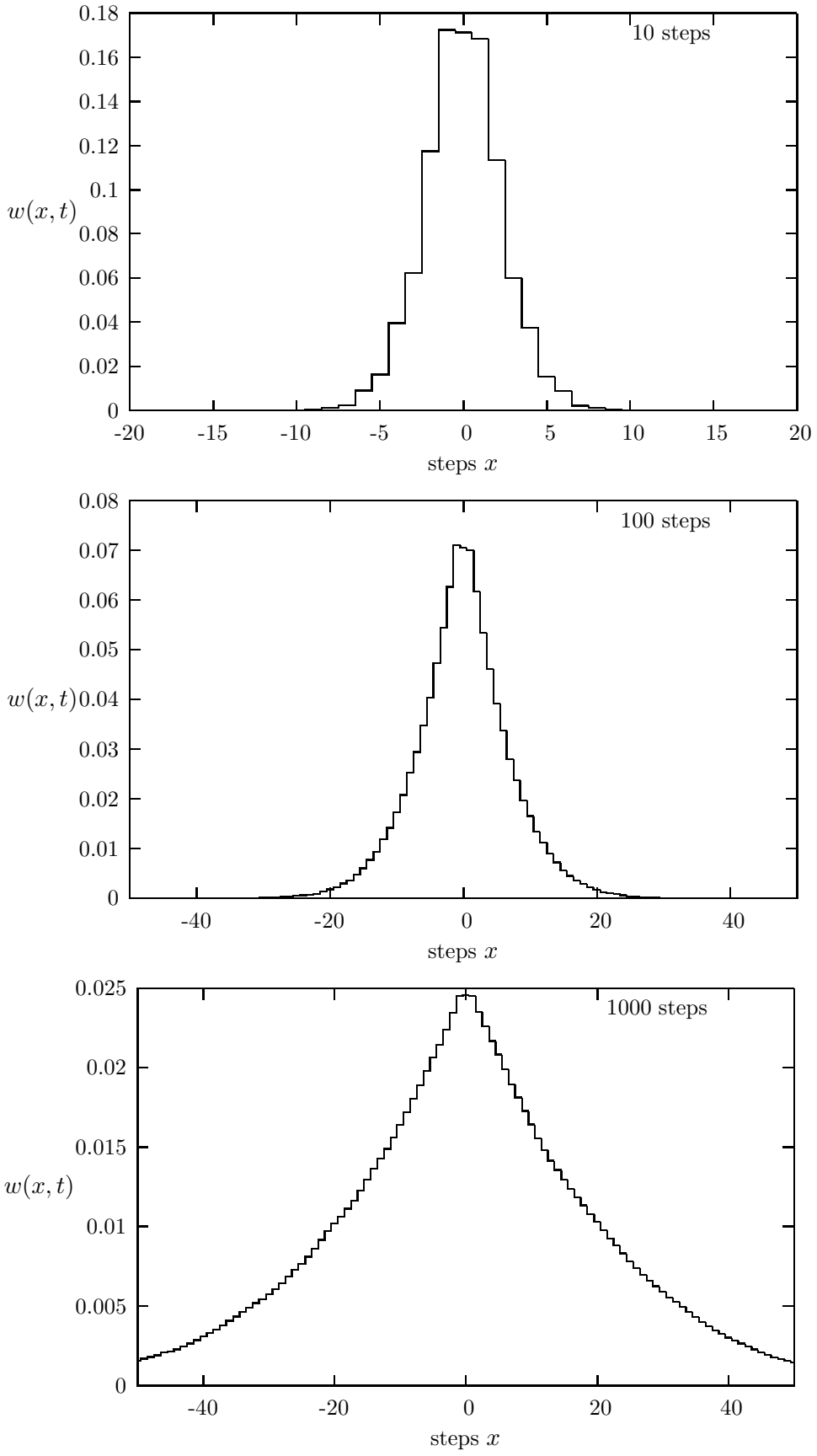

Figure 9.5: Probability distribution for one walker after 10, 100 and 1000 steps.

9.4 Entropy and Equilibrium Features

We use this section to motivate, in a physically intuitive way, the importance of the ergodic hypothesis via a discussion of how a Markovian process reaches an equilibrium situation after a given number of random walks. It serves then the scope of bridging the gap between a Markovian process and our discussion of the Metropolis algorithm in the next section.

To achieve this, we will use the program from the previous section, see programs/chapter9/program3.cpp and introduce the concept of entropy S. We discuss the thermodynamical meaning of the entropy and its link with the second law of thermodynamics in the next chapter. Here it will suffice to state that the entropy is a measure of the disorder of the system, thus a system which is fully ordered and stays in its fundamental state (ground state) has zero entropy, while a disordered system has a large and nonzero entropy.

The definition of the entropy S (as a dimensionless quantity here) is

$$S = -\sum_i w_i ln(w_i), \tag{9.66}$$

where w_i is the probability of finding our system in a state i. For our one-dimensional random walk case discussed in the previous sections it represents the probability for being at position $i = i\Delta x$ after a given number of time steps. In order to test this, we start with the previous program but assume now that we have N random walkers at $i = 0$ and $t = 0$ and let these random walkers diffuse as function of time. This means simply an additional loop. We compute then, as in the previous program example, the probability distribution for N walkers after a given number of steps i along x and time steps j. We can then compute an entropy S_j for a given number of time steps by summing over all probabilities i. We show this in Fig. 9.6. The code used to compute these results is in programs/chapter9/program4.cpp. Here we have used 100 walkers on a lattice of length from $L = -50$ to $L = 50$ employing periodic boundary conditions meaning that if a walker reaches the point $x = L$ it is shifted to $x = -L$ and if $x = -L$ it is shifted to $x = L$. We see from Fig. 9.6 that for small time steps, where all particles N are in the same position or close to the initial position, the entropy is very small, reflecting the fact that we have an ordered state. As time elapses, the random walkers spread out in space (here in one dimension) and the entropy increases as there are more states, that is positions accesible to the system. We say that the system shows an increased degree of disorder. After several time steps, we see that the entropy reaches a constant value, a situation called a steady state. This signals that the system has reached its equilibrium situation and that the random walkers spread out to occupy all possible available states. At equilibrium it means thus that all states are equally probable and this is not baked into any dynamical equations such as Newton's law of motion. It occurs because the system is allowed to explore all possibilities. An important hypothesis, which has never been proven rigorously but for certain systems, is the ergodic hypothesis which states that in equilibrium all available states of a closed system have equal probability. This hypothesis states also that if we are able to simulate long enough, then one should be able to trace through all possible paths in the space of available states to reach the equilibrium situation. Our Markov process should be able to reach any state of the system from any other state if we run for long enough. Markov processes fullfil the requirement of ergodicity since all new steps are independent of the previous ones and the random walkers can thus explore with equal probability all possible positions. In general however, we

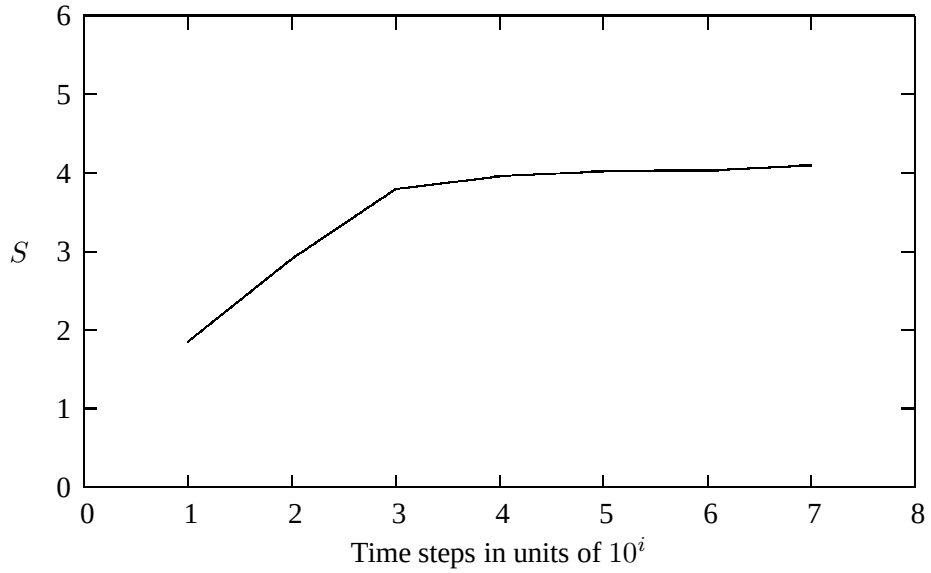

Figure 9.6: Entropy S_j as function of number of time steps j for a random walk in one dimension. Here we have used 100 walkers on a lattice of length from $L = -50$ to $L = 50$ employing periodic boundary conditions meaning that if a walker reaches the point $x = L$ it is shifted to $x = -L$ and if $x = -L$ it is shifted to $x = L$.

know that physical processes are not independent of each other. The relation between ergodicity and physical systems is an unsettled topic.

The Metropolis algorithm which we discuss in the next section is based on a Markovian process and fullfils the requirement of ergodicity. In addition, in the next section we impose the criterion of detailed balance.

9.5 The Metropolis algorithm and detailed balance

Let us recapitulate some of our results about Markov chains and random walks.

– The time development of our PDF $w(t)$, after one time-step from $t = 0$ is given by

$$w_i(t = \epsilon) = W(j \rightarrow i)w_j(t = 0).$$

This equation represents the discretized time-development of an original PDF. We can rewrite this as a

$$w_i(t = \epsilon) = W_{ij}w_j(t = 0).$$

with the transition matrix W for a random walk given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i - j| = 1 \\ 0 & \text{else} \end{cases}$$

We call W_{ij} for the transition probability and we represent it as a matrix.

– Both W and w represent probabilities and they have to be normalized, meaning that that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \rightarrow i) = 1.$$

The further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$.

– We can thus write the action of W as

$$w_i(t+1) = \sum_j W_{ij} w_j(t), \tag{9.67}$$

or as vector-matrix relation

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{W}}\hat{\mathbf{w}}(t), \tag{9.68}$$

and if we have that $||\hat{\mathbf{w}}(t+1) - \hat{\mathbf{w}}(t)|| \rightarrow 0$, we say that we have reached the most likely state of the system, the so-called steady state or equilibrium state. Another way of phrasing this is

$$\mathbf{w}(t=\infty) = \mathbf{W}\mathbf{w}(t=\infty). \tag{9.69}$$

An important condition we require that our Markov chain should satisfy is that of detailed balance. In statistical physics this condition ensures that it is e.g., the Boltzmann distribution which is generated when equilibrium is reached. The definition for being in equilibrium is that the rates at which a system makes a transition to or from a given state i have to be equal, that is

$$\sum_i W(j \rightarrow i) w_j = \sum_i W(i \rightarrow j) w_i. \tag{9.70}$$

However, the condition that the rates should equal each other is in general not sufficient to guarantee that we, after many simulations, generate the correct distribution. We therefore introduce an additional condition, namely that of detailed balance

$$W(j \rightarrow i) w_j = W(i \rightarrow j) w_i. \tag{9.71}$$

At equilibrium detailed balance gives thus

$$\frac{W(j \rightarrow i)}{W(i \rightarrow j)} = \frac{w_i}{w_j}. \tag{9.72}$$

We introduce the Boltzmann distribution

$$w_i = \frac{\exp\left(-\beta(E_i)\right)}{Z}, \tag{9.73}$$

which states that probability of finding the system in a state i with energy E_i at an inverse temperature $\beta = 1/k_B T$ is $w_i \propto \exp\left(-\beta(E_i)\right)$. The denominator Z is a normalization constant which ensures that the sum of all probabilities is normalized to one. It is defined as the sum of probabilities over all microstates j of the system

$$Z = \sum_j \exp\left(-\beta(E_i)\right). \tag{9.74}$$

From the partition function we can in principle generate all interesting quantities for a given system in equilibrium with its surroundings at a temperature T. This is demonstrated in the next chapter.

With the probability distribution given by the Boltzmann distribution we are now in the position where we can generate expectation values for a given variable A through the definition

$$\langle A\rangle = \sum_j A_j w_j = \frac{\sum_j A_j \exp\left(-\beta(E_j)\right.}{Z}. \tag{9.75}$$

In general, most systems have an infinity of microstates making thereby the computation of Z practically impossible and a brute force Monte Carlo calculation over a given number of randomly selected microstates may therefore not yield those microstates which are important at equilibrium. To select the most important contributions we need to use the condition for detailed balance. Since this is just given by the ratios of probabilities, we never need to evaluate the partition function Z. For the Boltzmann distribution, detailed balance results in

$$\frac{w_i}{w_j} = \exp\left(-\beta(E_i - E_j)\right). \tag{9.76}$$

Let us now specialize to a system whose energy is defined by the orientation of single spins. Consider the state i, with given energy E_i represented by the following N spins

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We are interested in the transition with one single spinflip to a new state j with energy E_j

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

This change from one microstate i (or spin configuration) to another microstate j is the configuration space analogue to a random walk on a lattice. Instead of jumping from one place to another in space, we 'jump' from one microstate to another.

However, the selection of states has to generate a final distribution which is the Boltzmann distribution. This is again the same we saw for a random walker, for the discrete case we had always a binomial distribution, whereas for the continuous case we had a normal distribution. The way we sample configurations should result in, when equilibrium is established, in the Boltzmann distribution. Else, our algorithm for selecting microstates has to be wrong.

Since we do not know the analytic form of the transition rate, we are free to model it as

$$W(i\rightarrow j) = g(i\rightarrow j)A(i\rightarrow j), \tag{9.77}$$

where g is a selection probability while A is the probability for accepting a move. It is also called the acceptance ratio. The selection probability should be same for all possible spin orientations, namely

$$g(i\rightarrow j) = \frac{1}{N}. \tag{9.78}$$

With detailed balance this gives

$$\frac{g(j\rightarrow i)A(j\rightarrow i)}{g(i\rightarrow j)A(i\rightarrow j)} = \exp\left(-\beta(E_i - E_j)\right), \tag{9.79}$$

but since the selection ratio is the same for both transitions, we have

$$\frac{A(j \rightarrow i)}{A(i \rightarrow j)} = \exp\left(-\beta(E_i - E_j)\right) \tag{9.80}$$

In general, we are looking for those spin orientations which correspond to the average energy at equilibrium.

We are in this case interested in a new state E_j whose energy is lower than E_i, viz., $\Delta E = E_j - E_i \leq 0$. A simple test would then be to accept only those microstates which lower the energy. Suppose we have ten microstates with energy $E_0 \leq E_1 \leq E_2 \leq E_3 \leq \cdots \leq E_9$. Our desired energy is E_0. At a given temperature T we start our simulation by randomly choosing state E_9. Flipping spins we may then find a path from $E_9 \rightarrow E_8 \rightarrow E_7 \cdots \rightarrow E_1 \rightarrow E_0$. This would however lead to biased statistical averages since it would violate the ergodic hypothesis discussed in the previous section. This principle states that it should be possible for any Markov process to reach every possible state of the system from any starting point if the simulations is carried out for a long enough time.

Any state in a Boltzmann distribution has a probability different from zero and if such a state cannot be reached from a given starting point, then the system is not ergodic. This means that another possible path to E_0 could be $E_9 \rightarrow E_7 \rightarrow E_8 \cdots \rightarrow E_9 \rightarrow E_5 \rightarrow E_0$ and so forth. Even though such a path could have a negligible probability it is still a possibility, and if we simulate long enough it should be included in our computation of an expectation value.

Thus, we require that our algorithm should satisfy the principle of detailed balance and be ergodic. One possible way is the Metropolis algorithm, which reads

$$A(j \rightarrow i) = \begin{cases} \exp\left(-\beta(E_i - E_j)\right) & E_i - E_j > 0 \\ 1 & else \end{cases} \tag{9.81}$$

This algorithm satisfies the condition for detailed balance and ergodicity. It is implemented as follows:

- Establish an initial energy E_b
- Do a random change of this initial state by e.g., flipping an individual spin. This new state has energy E_t. Compute then $\Delta E = E_t - E_b$
- If $\Delta E \leq 0$ accept the new configuration.
- If $\Delta E > 0$, compute $w = e^{-(\beta \Delta E)}$.
- Compare w with a random number r. If $r \leq w$ accept, else keep the old configuration.
- Compute the terms in the sums $\sum A_s w_s$.
- Repeat the above steps in order to have a large enough number of microstates
- For a given number of MC cycles, compute then expectation values.

The application of this algorithm will be discussed in detail in the next two chapters.

9.6 Physics project: simulation of the Boltzmann distribution

In this project the aim is to show that the Metropolis algorithm generates the Boltzmann distribution

$$P(\beta) = \frac{e^{-\beta E}}{Z}, \tag{9.82}$$

with $\beta = 1/kT$ being the inverse temperature, E is the energy of the system and Z is the partition function. The only functions you will need are those to generate random numbers.

We are going to study one single particle in equilibrium with its surroundings, the latter modeled via a large heat bath with temperature T.

The model used to describe this particle is that of an ideal gas in **one** dimension and with velocity $-v$ or v. We are interested in finding $P(v)dv$, which expresses the probability for finding the system with a given velocity $v \in [v, v + dv]$. The energy for this one-dimensional system is

$$E = \frac{1}{2}kT = \frac{1}{2}v^2, \tag{9.83}$$

with mass $m = 1$. In order to simulate the Boltzmann distribution, your program should contain the following ingredients:

- Reads in the temperature T, the number of Monte Carlo cycles, and the initial velocity. You should also read in the change in velocity δv used in every Monte Carlo step. Let the temperature have dimension energy.
- Thereafter you choose a maximum velocity given by e.g., $v_{max} \sim 10\sqrt{T}$. Then you construct a velocity interval defined by v_{max} and divided it in small intervals through v_{max}/N, with $N \sim 100 - 1000$. For each of these intervals your task is to find out how many times a given velocity during the Monte Carlo sampling appears in each specific interval.
- The number of times a given velocity appears in a specific interval is used to construct a histogram representing $P(v)dv$. To achieve this you should construct a vector $P[N]$ which contains the number of times a given velocity appears in the subinterval $v, v + dv$.

In order to find the number of velocities appearing in each interval we will employ the Metropolis algorithm. A pseudocode for this is

```
for( montecarlo_cycles=1; Max_cycles; montecarlo_cycles++) {
   ...
   // change speed as function of delta v
   v_change = (2*ran1(&idum) -1 )* delta_v;
   v_new = v_old+v_change;
   // energy change
   delta_E = 0.5*(v_new*v_new - v_old*v_old) ;
   ......
   // Metropolis algorithm begins here
     if ( ran1(&idum) <= exp(-beta*delta_E)  ) {
         accept_step = accept_step + 1 ;
         v_old = v_new ;
         .....
     }
   // thereafter we must fill in  P[N] as a function of
```

```
    // the new speed
      P[?] = ...

    // upgrade mean velocity , energy and variance
       ...
    }
```

a) Make your own algorithm which sets up the histogram $P(v)dv$, find mean velocity, energy, energy variance and the number of accepted steps for a given temperature. Study the change of the number of accepted moves as a function of δv. Compare the final energy with the analytic result $E = kT/2$ for one dimension. Use $T = 4$ and set the intial velocity to zero, i.e., $v_0 = 0$. Try different values of δv. A possible start value is $\delta v = 4$. Check the final result for the energy as a function of the number of Monte Carlo cycles.

b) Make thereafter a plot of $ln(P(v))$ as function of E and see if you get a straight line. Comment the result.

9.7 Physics project: Random Walk in two dimensions

For this project you can build upon program programs/chapter9/program2.cpp (or the f90 version). You will need to compute the expectation values $\langle x(N)\rangle$, $\langle y(N)\rangle$ and

$$\langle \Delta R^2(N)\rangle = \langle x^2(N)\rangle + \langle y^2(N)\rangle - \langle x(N)\rangle^2 - \langle y(N)\rangle^2$$

where N is the number of time steps.

a) Enumerate all random walks on a square lattice for $N = 2$ and obtain exact results for $\langle x(N)\rangle$, $\langle y(N)\rangle$ and $\langle \Delta R^2(N)\rangle$. Verify your results by comparing your Monte Carlo simulations with the exact results. Assume that all four directions are equally probable.

b) Do a Monte Carlo simulation to estimate $\langle \Delta R^2(N)\rangle$ for $N = 10, 40, 60$ and 80 using a reasonable number of trials for each N. Assume that we have the asymptotic behavior

$$\langle \Delta R^2(N)\rangle \sim N^{2\nu},$$

and estimate the exponent ν from a log-log plot of $\langle \Delta R^2(N)\rangle$ versus N. If $\nu \approx 1/2$, estimate the magnutide of the self-diffusion coefficient D given by

$$\langle \Delta R^2(N)\rangle \sim 2dDN,$$

with d the dimension of the system.

c) Compute now the quantities $\langle x(N)\rangle$, $\langle y(N)\rangle$, $\langle \Delta R^2(N)\rangle$ and

$$\langle R^2(N)\rangle = \langle x^2(N)\rangle + \langle y^2(N)\rangle,$$

for the same values of N as in the previous case but now with the step probabilities $2/3$, $1/6$, $1/6$ and $1/6$ corresponding to right, left, up and down, respectively. This choice corresponds to a biased random walk with a drift to the right. What is the interpretation of $\langle x(N)\rangle$ in this case? What is the dependence of $\langle \Delta R^2(N)\rangle$ on N and does $\langle R^2(N)\rangle$ depend simply on N?

d) Consider now a random walk that starts at a site that is a distance $y = h$ above a horisontal line (ground). If the probability of a step down towards the ground is bigger than the probability of a step up, we expect that the walker will eventually reach a horisontal line. This walk is a simple model of the fall of a rain drop in the presence of a random breeze. Assume that the probabilities are 0.1, 0.6, $0,15$ and 0.15 corresponding to up, down, right and left, respectively. Do a Monte Carlo simulation to determine the mean time τ for the walker to reach any site on the line at $x = 0$. Find the functional dependence of τ on h. Can you define a velocity in the vertical direction? Since the walker does not always move vertically, it suffers a net displacement Δx in the horizontal direction. Compute $\langle \Delta x^2 \rangle$ and find its dependence on h and τ.