Lectures II & III: Random Numbers, Markov Chains, Diffusion and the Metropolis Algorithm

Morten Hjorth-Jensen

[1]Department of Physics and Center of Mathematics for Applications
University of Oslo, N-0316 Oslo, Norway

[2]Department of Physics and Astronomy, Michigan State University
East Lansing, Michigan, USA

January 28 - February 2

Outline

Outline

Outline

’Iacta Alea est’, the die is cast!

Plan for the lectures

1. January 28: Introduction to Monte Carlo methods, probability distributions and Monte Carlo Integration.
2. January 29: Random numbers, Markov chains, diffusion and the Metropolis algorithm.
3. January 30: Applications in sociology, simulations of phase transitions in physics and quantum physics.
4. All material taken from my text on Computational Physics, see `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06/undervisningsmateriale/LectureNotes/`.

Reminder from yesterday: What is Monte Carlo?

1. **Monte Carlo** methods are nowadays widely used, from the integration of multi-dimensional integrals to solving ab initio problems in chemistry, physics, medicine, biology, or even Dow-Jones forecasting. Computational finance is one of the novel fields where Monte Carlo methods have found a new field of applications, with financial engineering as an emerging field.
2. **Numerical methods** that are known as Monte Carlo methods can be loosely described as statistical simulation methods, where statistical simulation is defined in quite general terms to be any method that utilizes sequences of random numbers to perform the simulation.

Reminder from yesterday: Monte Carlo Keywords

Consider it is a numerical experiment

- Be able to generate random variables following a given probability distribution function (PDF). The starting point for any calculation is the derivation of random numbers based on the uniform distribution.
- Sampling rule for accepting a move
- Compute standard deviation and other expectation values
- Techniques for improving errors

Exercises for Lecture I

(a) Calculate the integral

$$I = \int_0^1 e^{-x^2} dx,$$

using brute force Monte Carlo with $p(x) = 1$ and importance sampling with $p(x) = ae^{-x}$ where a is a constant.

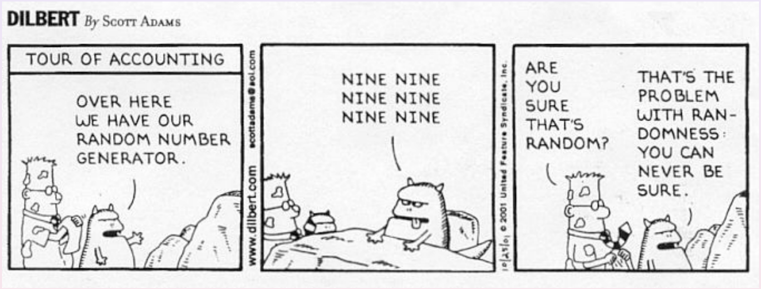
(b) Calculate the integral

$$I = \int_0^{\pi} \frac{1}{x^2 + cos^2(x)} dx,$$

with $p(x) = ae^{-x}$ where a is a constant. Determine the value of a which minimizes the variance.

(c) Try to parallelize the last code as well.

Random Numbers


DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.
www.dilbert.com
scottadams@aol.com
© 2001 United Feature Syndicate, Inc.

Random Numbers

- The codes for the random number generators ran0, ran1, ran2 and ran3 discussed here are at `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06/undervisningsmateriale/Programs/` and go to either the Fortran95 or C++ libraries (see also version with Blitz++).
- The random number generators ran0, ran1, ran2 and ran3, taken from the text *Numerical Recipes*, see `http://www.nr.com`.
- For parallel random number generators, there is the so-called Scalable Parallel Random Number Generators Library (SPRNG) for ASCI Monte Carlo Computations, see `http://sprng.cs.fsu.edu/` for software and detailed information.

Random Numbers

Most used are so-called ’Linear congruential’

$$N_i = (aN_{i-1} + c)MOD(M),$$

and to find a number in $x \in [0, 1]$

$$x_i = N_i/M$$

M is called the period and should be as big as possible. The start value is N_0 and is called the seed.

- The random variables should result in the uniform distribution
- No correlations between numbers (zero covariance)
- As big as possible period M
- Fast algo

Random Numbers

The problem with such generators is that their outputs are periodic; they will start to repeat themselves with a period that is at most M. If however the parameters a and c are badly chosen, the period may be even shorter.
Consider the following example

$$N_i = (6N_{i-1} + 7)\mathrm{MOD}(5),$$

with a seed $N_0 = 2$. This generator produces the sequence $4, 1, 3, 0, 2, 4, 1, 3, 0, 2, ... \dots$, i.e., a sequence with period 5. However, increasing M may not guarantee a larger period as the following example shows

$$N_i = (27N_{i-1} + 11)\mathrm{MOD}(54),$$

which still, with $N_0 = 2$, results in $11, 38, 11, 38, 11, 38, \dots$, a period of just 2.

Random Numbers

Typical periods for the random generators provided in the program library are of the order of $\sim 10^9$ (ran0) or larger (ran1, ran2 and ran3). Other random number generators which have become increasingly popular are so-called shift-register generators. In these generators each successive number depends on many preceding values (rather than the last values as in the linear congruential generator). For example, you could make a shift register generator whose lth number is the sum of the $l-i$th and $l-j$th values with modulo M,

$$N_l = (aN_{l-i} + cN_{l-j})\mathrm{MOD}(M).$$

Such a generator again produces a sequence of pseudorandom numbers but this time with a period much larger than M. It is also possible to construct more elaborate algorithms by including more than two past terms in the sum of each iteration. One example is the generator of Marsaglia and Zaman (Computers in Physics **8** (1994) 117) which consists of two congruential relations

$$N_l = (N_{l-3} - N_{l-1})\mathrm{MOD}(2^{31} - 69),$$

followed by

$$N_l = (69069N_{l-1} + 1013904243)\mathrm{MOD}(2^{32}),$$

which according to the authors has a period larger than 2^{94}.

Random Numbers

Using modular addition, we could use the bitwise exclusive-OR ($\oplus$) operation so that

$$N_l = (N_{l-i}) \oplus (N_{l-j})$$

where the bitwise action of $\oplus$ means that if $N_{l-i} = N_{l-j}$ the result is 0 whereas if $N_{l-i} \neq N_{l-j}$ the result is 1. As an example, consider the case where $N_{l-i} = 6$ and $N_{l-j} = 11$. The first one has a bit representation (using 4 bits only) which reads 0110 whereas the second number is 1011. Employing the $\oplus$ operator yields 1101, or $2^3 + 2^2 + 2^0 = 13$.

In Fortran, the bitwise $\oplus$ operation is coded through the intrinsic function $\mathrm{IEOR}(m, n)$ where m and n are the input numbers, while in C it is given by $m \wedge n$.

Random Numbers

The function *ran*0 implements

$$N_i = (aN_{i-1})\mathrm{MOD}(M).$$

Note that $c = 0$ and that it cannot be initialized with $N_0 = 0$.
Problem: since a and N_{i-1} are integers and their multiplication could become greater than the standard 32 bit integer, there is a trick via Schrage's algorithm which approximates the multiplication of large integers through the factorization

$$M = aq + r,$$

where we have defined

$$q = [M/a],$$

and

$$r = M \ \mathrm{MOD}\ a.$$

where the brackets denote integer division. In the code below the numbers q and r are chosen so that $r < q$.

Random Numbers

To see how this works we note first that

$$(aN_{i-1})\mathrm{MOD}(M) = (aN_{i-1} - [N_{i-1}/q]M)\mathrm{MOD}(M),$$

since we can add or subtract any integer multiple of M from aN_{i-1}. The last term $[N_{i-1}/q]M\mathrm{MOD}(M)$ is zero since the integer division $[N_{i-1}/q]$ just yields a constant which is multiplied with M. Rewrite as

$$(aN_{i-1})\mathrm{MOD}(M) = (aN_{i-1} - [N_{i-1}/q](aq + r))\mathrm{MOD}(M),$$

Random Numbers

It gives

$$(aN_{i-1})\mathrm{MOD}(M) = (a(N_{i-1} - [N_{i-1}/q]q) - [N_{i-1}/q]r))\,\mathrm{MOD}(M),$$

yielding

$$(aN_{i-1})\mathrm{MOD}(M) = (a(N_{i-1}\mathrm{MOD}(q)) - [N_{i-1}/q]r))\,\mathrm{MOD}(M).$$

- $[N_{i-1}/q]r$ is always smaller or equal $N_{i-1}(r/q)$ and with $r < q$ we obtain always a number smaller than N_{i-1}, which is smaller than M.
- $N_{i-1}\mathrm{MOD}(q)$ is between zero and $q-1$ then $a(N_{i-1}\mathrm{MOD}(q)) < aq$.
- Our definition of $q = [M/a]$ ensures that this term is also smaller than M meaning that both terms fit into a 32-bit signed integer. None of these two terms can be negative, but their difference could.

Random Numbers

```
/*  ran0() is an "Minimal" random number generator of Park and Miller
** Set or reset the input value
** idum to any integer value (except the unlikely value MASK)
** to initialize the sequence; idum must not be altered between
** calls for sucessive deviates in a sequence.
** The function returns a uniform deviate between 0.0 and 1.0.
*/
double ran0(long &idum)
{
   const int a = 16807, m = 2147483647, q = 127773;
   const int r = 2836, MASK = 123459876;
   const double am = 1./m;
   long     k;
   double   ans;
   idum ^= MASK;
   k = (*idum)/q;
   idum = a*(idum - k*q) - r*k;
   // add m if negative difference
   if(idum < 0) idum += m;
   ans=am*(idum);
   idum ^= MASK;
   return ans;
} // End: function ran0()
```

Random Numbers

Important tests of random numbers are the standard deviation σ and the mean $\mu = \langle x \rangle$.
For the uniform distribution with N points we have that the average $\langle x^k \rangle$ is

$$\langle x^k \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i^k p(x_i),$$

and taking the limit $N \to \infty$ we have

$$\langle x^k \rangle = \int_0^1 dx p(x) x^k = \int_0^1 dx x^k = \frac{1}{k+1},$$

since $p(x) = 1$. The mean value μ is then

$$\mu = \langle x \rangle = \frac{1}{2}$$

while the standard deviation is

$$\sigma = \sqrt{\langle x^2 \rangle - \mu^2} = \frac{1}{\sqrt{12}} = 0.2886.$$

Random Numbers

Number of x-values for various intervals generated by 4 random number generators, their corresponding mean values and standard deviations. All calculations have been initialized with the variable *idum* $= -1$.

x-bin	ran0	ran1	ran2	ran3
0.0-0.1	1013	991	938	1047
0.1-0.2	1002	1009	1040	1030
0.2-0.3	989	999	1030	993
0.3-0.4	939	960	1023	937
0.4-0.5	1038	1001	1002	992
0.5-0.6	1037	1047	1009	1009
0.6-0.7	1005	989	1003	989
0.7-0.8	986	962	985	954
0.8-0.9	1000	1027	1009	1023
0.9-1.0	991	1015	961	1026
μ	0.4997	0.5018	0.4992	0.4990
σ	0.2882	0.2892	0.2861	0.2915

Random Numbers

Since our random numbers, which are typically generated via a linear congruential algorithm, are never fully independent, we can then define an important test which measures the degree of correlation, namely the so-called auto-correlation function C_k

$$C_k = \frac{\langle x_{i+k}x_i\rangle - \langle x_i\rangle^2}{\langle x_i^2\rangle - \langle x_i\rangle^2},$$

with $C_0 = 1$. Recall that $\sigma^2 = \langle x_i^2\rangle - \langle x_i\rangle^2$. The non-vanishing of C_k for $k \neq 0$ means that the random numbers are not independent. The independence of the random numbers is crucial in the evaluation of other expectation values. The expectation values which enter the definition of C_k are given by

$$\langle x_{i+k}x_i\rangle = \frac{1}{N-k}\sum_{i=1}^{N-k} x_i x_{i+k}.$$

The correlation function is related to the covariance.

Brownian Motion and Markov Processes

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. The Metropolis algorithm is widely used in Monte Carlo simulations and the understanding of it rests within the interpretation of random walks and Markov processes.

Brownian Motion and Markov Processes

In a random walk one defines a mathematical entity called a **walker**, whose attributes completely define the state of the system in question. The state of the system can refer to any physical quantities, from the vibrational state of a molecule specified by a set of quantum numbers, to the brands of coffee in your favourite supermarket.
The walker moves in an appropriate state space by a combination of deterministic and random displacements from its previous position.

This sequence of steps forms a **chain**.

A simple Example

The obvious case is that of a random walker on a one-, or two- or three-dimensional lattice (dubbed coordinate space hereafter)
Consider 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

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

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

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

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.

Brownian Motion and Markov Processes

We wish to study the time-development of a PDF after a given number of time steps. We define our PDF by the function $w(t)$. In addition we define a transition probability W. The time development of our PDF $w(t)$, after one time-step from $t = 0$ is given by

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

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 left or right (cannot stay in the same position) given by

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

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

Brownian Motion and Markov Processes

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

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

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

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

or as vector-matrix relation

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

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

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

Brownian Motion and Markov Processes, a simple Example

Consider the simple 3×3 matrix $\hat{W}$

$$\hat{W} = \begin{pmatrix} 1/4 & 1/8 & 2/3 \\ 3/4 & 5/8 & 0 \\ 0 & 1/4 & 1/3 \end{pmatrix},$$

and we choose our initial state as

$$\hat{w}(t=0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The first iteration is

$$w_i(t=\epsilon) = W(j \to i)w_j(t=0),$$

resulting in

$$\hat{w}(t=\epsilon) = \begin{pmatrix} 1/4 \\ 3/4 \\ 0 \end{pmatrix}.$$

Brownian Motion and Markov Processes, a simple Example

The next iteration results in

$$w_i(t = 2\epsilon) = W(j \rightarrow i)w_j(t = \epsilon),$$

resulting in

$$\hat{w}(t = 2\epsilon) = \begin{pmatrix} 5/23 \\ 21/32 \\ 6/32 \end{pmatrix}.$$

Note that the vector $\hat{w}$ is always normalized to 1. We find the steady state of the system by solving the linear set of equations

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

Brownian Motion and Markov Processes, a simple Example

This linear set of equations reads

$$\begin{aligned} W_{11}w_1(t=\infty)+W_{12}w_2(t=\infty)+W_{13}w_3(t=\infty) &= w_1(t=\infty) \\ W_{21}w_1(t=\infty)+W_{22}w_2(t=\infty)+W_{23}w_3(t=\infty) &= w_2(t=\infty) \\ W_{31}w_1(t=\infty)+W_{32}w_2(t=\infty)+W_{33}w_3(t=\infty) &= w_3(t=\infty) \end{aligned} \tag{1}$$

with the constraint that

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

yielding as solution

$$\hat{w}(t=\infty)=\left(\begin{array}{c} 4/15 \\ 8/15 \\ 3/15 \end{array}\right).$$

Brownian Motion and Markov Processes, a simple Example

Convergence of the simple example

Iteration	w_1	w_2	w_3
0	1.00000	0.00000	0.00000
1	0.25000	0.75000	0.00000
2	0.15625	0.62625	0.18750
3	0.24609	0.52734	0.22656
4	0.27848	0.51416	0.20736
5	0.27213	0.53021	0.19766
6	0.26608	0.53548	0.19844
7	0.26575	0.53424	0.20002
8	0.26656	0.53321	0.20023
9	0.26678	0.53318	0.20005
10	0.26671	0.53332	0.19998
11	0.26666	0.53335	0.20000
12	0.26666	0.53334	0.20000
13	0.26667	0.53333	0.20000
$\hat{w}(t = \infty)$	0.26667	0.53333	0.20000

Exercise: make a small program where you perform these iterations,but change the initial vector and study the convergence.

Brownian Motion and Markov Processes, what is happening?

We have after t-steps

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^{\mathbf{t}}\hat{\mathbf{w}}(0),$$

with $\hat{\mathbf{w}}(0)$ the distribution at $t = 0$ and $\hat{\mathbf{W}}$ representing the transition probability matrix. We can always expand $\hat{\mathbf{w}}(0)$ in terms of the right eigenvectors $\hat{\mathbf{v}}$ of $\hat{\mathbf{W}}$ as

$$\hat{\mathbf{w}}(0) = \sum_i \alpha_i \hat{\mathbf{v}}_i,$$

resulting in

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^t\hat{\mathbf{w}}(0) = \hat{\mathbf{W}}^t \sum_i \alpha_i \hat{\mathbf{v}}_i = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i,$$

with λ_i the i^{th} eigenvalue corresponding to the eigenvector $\hat{\mathbf{v}}_i$.

Brownian Motion and Markov Processes, what is happening?

If we assume that λ_0 is the largest eigenvector we see that in the limit $t \rightarrow \infty$, $\hat{\mathbf{w}}(t)$ becomes proportional to the corresponding eigenvector $\hat{\mathbf{v}}_0$. This is our steady state or final distribution.
In our discussion below in connection with the entropy of a system and tomorrow's lecture on physics applications, we will relate these properties to correlation functions such as the time-correlation function.

That will allow us to define the so-called *equilibration time*,viz the time needed for the system to reach its most likely state. Form that state and on we can can compute contributions to various statistical variables.

Brownian Motion and Markov Processes, what is happening?

We anticipate parts of tomorrow's discussion:
We can relate this property to an observable like the mean magnetization of say a magnetic material. With the probabilty $\hat{\mathbf{w}}(t)$ we can write the mean magnetization as

$$\langle \mathcal{M}(t) \rangle = \sum_{\mu} \hat{\mathbf{w}}(t)_{\mu} \mathcal{M}_{\mu},$$

or as the scalar of a vector product

$$\langle \mathcal{M}(t) \rangle = \hat{\mathbf{w}}(t)\mathbf{m},$$

with **m** being the vector whose elements are the values of $\mathcal{M}_{\mu}$ in its various microstates μ.
Recall our definition of an expectation value with a discrete PDF $p(x_i)$:

$$E[x^k] = \langle x^k \rangle = \frac{1}{N}\sum_{i=1}^{N} x_i^k p(x_i),$$

provided that the sums (or integrals) $\sum_{i=1}^{N} p(x_i)$ converge absolutely (viz , $\sum_{i=1}^{N} |p(x_i)|$ converges)

Brownian Motion and Markov Processes, what is happening?

We rewrite the last relation as

$$\langle \mathcal{M}(t) \rangle = \hat{\mathbf{w}}(t)\mathbf{m} = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i \mathbf{m}_i .$$

If we define $m_i = \hat{\mathbf{v}}_i \mathbf{m}_i$ as the expectation value of $\mathcal{M}$ in the i^{th} eigenstate we can rewrite the last equation as

$$\langle \mathcal{M}(t) \rangle = \sum_i \lambda_i^t \alpha_i m_i .$$

Since we have that in the limit $t \to \infty$ the mean magnetization is dominated by the largest eigenvalue λ_0, we can rewrite the last equation as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \lambda_i^t \alpha_i m_i .$$

Brownian Motion and Markov Processes, what is happening?

We define the quantity

$$\tau_i = -\frac{1}{log\lambda_i},$$

and rewrite the last expectation value as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \alpha_i m_i e^{-t/\tau_i}.$$

The quantities τ_i are the correlation times for the system. They control also the time-correlation functions to be discussed tomorrow.

The longest correlation time is obviously given by the second largest eigenvalue τ_1, which normally defines the correlation time discussed above. For large times, this is the only correlation time that survives. If higher eigenvalues of the transition matrix are well separated from λ_1 and we simulate long enough, τ_1 may well define the correlation time. In other cases we may not be able to extract a reliable result for τ_1.

Diffusion from Markov Chain

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

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

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

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

which leads to

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

which is the diffusion equation in one dimension.

Diffusion from Markov Chain

With the probability distribution function $w(x,t)dx$ we can compute expectation values such as the mean distance

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

or

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

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

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

Diffusion from Markov Chain

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

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

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

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

implying that when we study the time-derivative $\partial\langle x(t)\rangle/\partial t$, we obtain after integration by parts and using Eq. (34)

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

leading to

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

implying that

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

Diffusion from Markov Chain

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

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

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

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

leading to

$$\langle x^2\rangle=2Dt,$$

and the variance as

$$\langle x^2\rangle-\langle x\rangle^2=2Dt.$$

The root mean square displacement after a time t is then

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

What does it mean?

Our results 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.

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 $x = 0$. 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.

Random Walks

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

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

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

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

Random Walks

For many enough steps the non-diagonal contribution is

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

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

$$\langle x(n)^2 \rangle - \langle x(n) \rangle^2 = l^2 n.$$

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

$$\langle x(n)^2 \rangle - \langle x(n) \rangle^2 = 4LRl^2 n.$$

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 by defining the diffusion constant as

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

Random Walks, simple Algo

It is rather straightforward to implement the **sampling rule** for the random walk in one dimension:

```
  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
```

Excercise: Extend this to two and three-dimensions. Codes at `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06/undervisningsmateriale/Programs/` and go to chapter 9 for both C++ and Fortran95 codes.

Diffusion from Markov Chain

When solving partial differential equations such as the diffusion equation numerically, the derivatives are always discretized. We can rewrite the time derivative as

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

whereas the gradient is approximated as

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

resulting in the discretized diffusion equation

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

where n represents a given time step and i a step in the x-direction.

Explicit PDE Scheme

This results in a matrix-vector multiplication

$$w_{j+1} = \hat{W} w_j$$

with the matrix $\hat{W}$ given by

$$\hat{W} = \begin{pmatrix} 1-2\alpha & \alpha & 0 & 0\dots \\ \alpha & 1-2\alpha & \alpha & 0\dots \\ \dots & \dots & \dots & \dots \\ 0\dots & 0\dots & \alpha & 1-2\alpha \end{pmatrix}$$

which means we can rewrite the original partial differential equation as a set of matrix-vector multiplications

$$w_{j+1} = \hat{W} w_j = \cdots = \hat{W}^{j+1} w_0,$$

where w_0 is the initial vector at time $t = 0$ defined by the initial value $g(x)$.

Explicit PDE Scheme

$$\alpha = \Delta t/\Delta x^2$$

For convergence spectral radius $\rho(\hat{W})$ of our matrix $\hat{W}$ satisfies the condition

$$\rho(\hat{W}) < 1,$$

The spectral radius is defined as

$$\rho(\hat{W}) = \max\left\{|\lambda| : \det(\hat{W} - \lambda\hat{I})\right\},$$

which is interpreted as the smallest number such that a circle with radius centered at zero in the complex plane contains all eigenvalues of $\hat{A}$. If the matrix is positive definite, the condition is always satisfied.
This requirement results in

$$-1 < 1 - \alpha 2\left(1 - cos(\theta)\right) < 1,$$

which is satisfied only if $\alpha < (1 - cos(\theta))^{-1}$ resulting in $\alpha \leq 1/2$ or $\Delta/\Delta x^2 \leq 1/2$.

Diffusion from Markov Chain

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 a grid, 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.

Diffusion from Markov Chain

For the Markov process we have a transition probability from a position $x = jl$ to a position $x = il$ given by

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

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

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

Diffusion from Markov Chain

This equation represents the discretized time-development of an original PDF. Since both W and w represent probabilities, they have to be normalized, i.e., we require that at each time step we have

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

and

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

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.

Diffusion from Markov Chain

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

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

and defining

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

we obtain

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

or in matrix form

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

Diffusion from Markov Chain

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

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

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

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

and

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

Diffusion from Markov Chain

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

$$\hat{L}\hat{R} = \hat{R}\hat{L} = 1,$$

and

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

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

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

and only the diagonal matrix elements are different from zero.

Diffusion from Markov Chain

For the first time step we have thus

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

and using the properties in Eqs. (51) and (51) we have after two time steps

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

and similarly after three time steps

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

Diffusion from Markov Chain

Using the binomial formula

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

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

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

or

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

Diffusion from Markov Chain

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

$$W(il - jl, n\epsilon) = \begin{cases} \frac{1}{2^n} \begin{pmatrix} n \\ \frac{1}{2}(n+i-j) \end{pmatrix} & |i-j| \leq n \\ 0 & \text{else} \end{cases},$$

and $n + i - j$ has to be an even number.

Diffusion from Markov Chain

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. (49) we have

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

resulting in

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

Diffusion from Markov Chain

Using the recursion relation for the binomials

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

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

$$w(x,t) = w(il, n\epsilon) = w_i(n\epsilon),$$

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

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

Diffusion from Markov Chain

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

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.

Continuous Equation

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

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

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 $\mathbf{x}$.

Continuous Equation

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

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

and after m time steps we have

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

When equilibrium is reached we have

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

Continuous Equation

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

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

and using the definition of the δ-function

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

we see that

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

Continuous Equation

We can then use the Fourier-transformed diffusion equation

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

with the obvious solution

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

We then obtain

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

with the normalization condition

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

Continuous Equation

It is rather easy to verify by insertion that the above 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. The results from the central limit theorem, see yesterday's lecture, state that the transition probability in the limit $n \to \infty$ converges to the normal distribution. It is then possible to show that

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

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

Entropy and Equilibrium

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

$$S = -\sum_i w_i ln(w_i),$$

where w_i is the probability of finding our system in a state i. For our one-dimensional randow walk it represents the probability for being at position $i = i\Delta x$ after a given number of time steps. Assume now that we have N random walkers at $i = 0$ and $t = 0$ and let these random walkers diffuse as function of time. We compute then 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. 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 $x = L$.

Entropy

```
// loop over all time steps
  for (int step=1; step <= time_steps; step++){
    // move all walkers with periodic boundary conditions
    for (int walks = 1; walks <= walkers; walks++){
      if (ran0(&idum) <= move_probability) {
if ( x[walks] +1 > length) {
  x[walks] = -length;
}
else{
  x[walks] += 1;
}
      }
      else {
if ( x[walks] -1 < -length) {
  x[walks] = length;
        }
else{
  x[walks] -= 1;
}
      }
    }  // end of loop over walks
  } // end of loop over trials
```

Entropy

```
// at the final time step we compute the probability
// by counting the number of walkers at every position
for ( int i = -length; i <= length; i++){
  int count = 0;
  for( int j = 1; j <= walkers; j++){
    if ( x[j] == i ) {
      count += 1;
    }
  }
  probability[i+length] = count;
}
```

Entropy

```
// Writes the results to screen
void output(int length, int time_steps, int walkers, int *probability)
{
  double entropy, histogram;
  // find norm of probability
  double norm = 1.0/walkers;
  // compute the entropy
  entropy = 0.; histogram = 0.;
  for( int  i = -length; i <=  length; i++){
    histogram = (double) probability[i+length]*norm;
    if ( histogram > 0.0) {
    entropy -= histogram*log(histogram);
    }
  }
    cout << setiosflags(ios::showpoint | ios::uppercase);
    cout << setw(6) << time_steps;
    cout << setw(15) << setprecision(8) << entropy << endl;
}  // end of function output
```

Entropy

At small time steps 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.

Detailed Balance

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

$$\sum_i W(j \rightarrow i) w_j = \sum_i W(i \rightarrow j) w_i.$$

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

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

At equilibrium detailed balance gives thus

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

Ergodicity

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. If any state in a distribution which has a probability different from zero and this state cannot be reached from any given starting point if we simulate long enough, then the system is not ergodic.

The Boltzmann Distribution as Example

We introduce the Boltzmann distribution

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

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

$$Z = \sum_j \exp\left(-\beta(E_i)\right).$$

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.

Boltzmann Distribution as Example

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

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

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

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

Boltzmann Distribution as Example

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

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

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

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

We saw previously that 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.

Boltzmann Distribution as Example

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

$$W(i \rightarrow j) = g(i \rightarrow j)A(i \rightarrow j),$$

where g is a selection probability while A is the probability for accepting a move. It is also called the acceptance ratio.

Boltzmann Distribution as Example

The selection probability should be same for all possible spin orientations, namely

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

With detailed balance this gives

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

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

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

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

Boltzmann Distribution as Example

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 above. 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.

Boltzmann Distribution as Example

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 $E_9 \longrightarrow E_7 \longrightarrow E_8 \cdots \longrightarrow E_9 \longrightarrow E_5 \longrightarrow E_0$ 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.

Metropolis Algorithm

The equation for detailed balance

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

is general and we could replace the Boltzmann distribution with other distributions as well. It specifies the ratio of pairs of acceptance probabilities, which leaves us with quite some room to manouvre.

- We give the largest of the two acceptance ratios the value 1 and adjust the other to satisfy the constraint.
- If $E_\mu < E_\nu$ then the larger of the two acceptance ratios is $A(\nu \to \mu)$ and we set to 1.
- Then we must have

$$A(\mu \to \nu) = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

if $E_\nu - E_\mu > 0$ and 1 otherwise. And that is the **Metropolis** algorithm.

Implementation

- 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 $\sum A_s P_s$.
- 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.

Test of the Metropolis Algorithm

Want to show that the Metropolis algorithm generates the Boltzmann distribution

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

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

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

with mass $m = 1$.

Test of the Metropolis Algorithm

- Read 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 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 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 construct a vector $P[N]$ which contains the number of times a given velocity appears in the subinterval $v, v+dv$.

Test of the Metropolis Algorithm

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

Codes at `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06/undervisningsmateriale/Projects/` and go to project 3 and solution for C++ implementation.

Test of the Metropolis Algorithm, analytic Results

The partition function of the system of interest is:

$$Z = \int_{-\infty}^{+\infty} e^{-\beta v^2/2} dv = \sqrt{2\pi}\beta^{-1/2}$$

The mean velocity

$$\langle v \rangle = \int_{-\infty}^{+\infty} v e^{-\beta v^2/2} dv = 0$$

The expressions for $\langle E \rangle$ and σ_E assume the following form:

$$\langle E \rangle = \int_{-\infty}^{+\infty} \frac{v^2}{2} e^{-\beta v^2/2} dv = -\frac{1}{Z}\frac{\partial Z}{\partial \beta} = \frac{1}{2}\beta^{-1} = \frac{1}{2}T$$

$$\langle E^2 \rangle = \int_{-\infty}^{+\infty} \frac{v^4}{4} e^{-\beta v^2/2} dv = \frac{1}{Z}\frac{\partial^2 Z}{\partial \beta^2} = \frac{3}{4}\beta^{-2} = \frac{3}{4}T^2$$

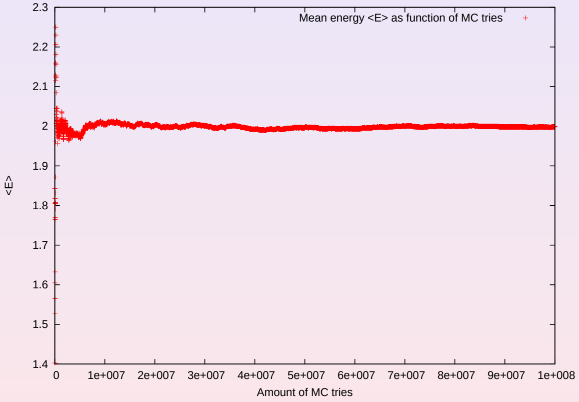
and

$$\sigma_E = \langle E^2 \rangle - \langle E \rangle^2 = \frac{1}{2}T^2$$

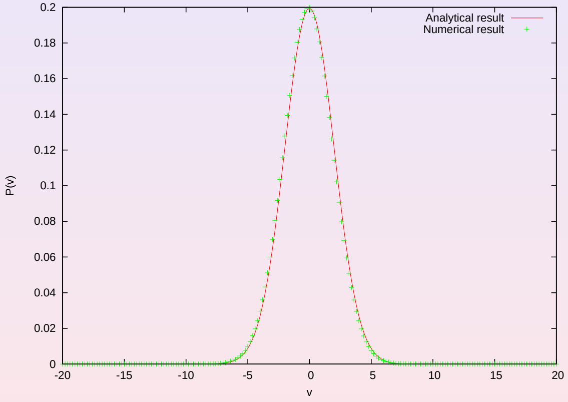
Test of the Metropolis Algorithm, Results

Observable	Analytical value	Numerical value
$\langle v \rangle$	0.00000	-0.00679
$\langle E \rangle$	2.00000	1.99855
σ_E	8.00000	8.06669

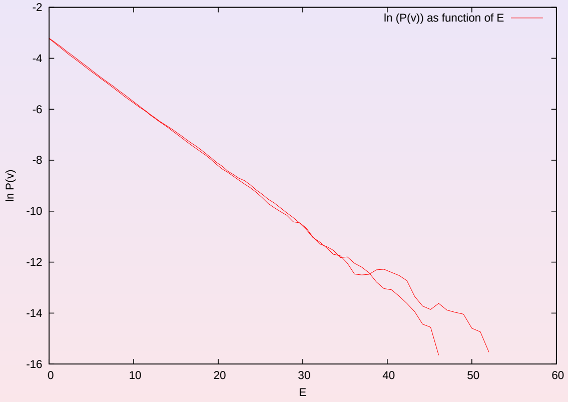
Test of the Metropolis Algorithm



Test of the Metropolis Algorithm



Test of the Metropolis Algorithm


ln (P(v)) as function of E
ln P(v)
E