CLASSICAL MONTE CARLO & METROPOLIS ALGORITHM

Monte Carlo (MC) simulations are, probably, the most powerful numerical tool to get the answers for large systems. MC is a universal technique, and can be applied to virtually any problem, but unfortunately its convergence is severely affected by the sign problem (to be discussed at the end of this section). We will start with the simplest application of MC methods to discrete and continuous *classical* systems. Quantum systems will be covered later in the course.

Classical Monte Carlo

Suppose one has to evaluate the ratio of two N-dimensional sums (or integrals) of the form

$$\langle A\rangle = Z^{-1}\sum_{i_1}\sum_{i_2}\dots\sum_{i_N} A(i_1,i_2,\dots,i_N)\,W(i_1,i_2,\dots,i_N)\;, \tag{1}$$

$$Z = \sum_{i_1}\sum_{i_2}\dots\sum_{i_N} W(i_1,i_2,\dots,i_N)\;, \tag{2}$$

where functions A and W are arbitrary. Let us also introduce the notion of **configuration**, ν, which is just the collection of all summation indices: $\nu \equiv \{i_1, i_2, \dots, i_N\}$. For the moment, assume that W is positive definite; then the combination $p(\nu) = W(i_1, i_2, \dots, i_N)/Z$ may be interpreted as the **configuration weight** because it is positive and normalized to unity. The expression for $\langle A\rangle$ may be considered then as the average of quantity $A(\nu)$ over all possible configurations were each configuration is included with the probability $p(\nu)$. The short hand notation is

$$\langle A\rangle = \frac{\sum_\nu A(\nu)W(\nu)}{\sum_\nu W(\nu)} \tag{3}$$

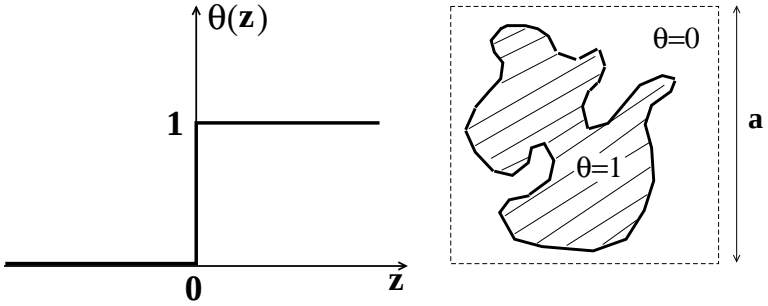
The connection between Eq. (3) and statistical physics is apparent. In stat.mech one is interested in various quantities averaged over the equilibrium statistics of the system states. If all system states are enumerated by the (multi-dimensional) index ν and their energies are E_ν, then the probability of ν to happen in the equilibrium is given by the normalized Gibbs distribution

$p(\nu) = e^{-E_\nu}/Z$, i.e. $W(\nu) = e^{-E_\nu}$ and the normalization constant Z is nothing but the partition function.

Even if there is no denominator in Eq. (1) and one needs to know just $\sum_\nu A(\nu)$, then the formal strategy is to introduce $W(\nu) \equiv 1$ and $Z = \sum_\nu W(\nu) = \sum_\nu$. The calculation is then reduced to exactly the same form provided $\sum_\nu$ is easy to get analytically. A typical example is provided by the MC simulation of the multi-dimensional body volume. Imagine that some nasty function $F(x_1, x_2, \ldots, x_N) = F(\nu) = 0$ specifies the body surface (all points with $F < 0$ belong to the body) and our task is to calculate its volume, i.e. we need

$$V_b = \int dx_1 \int dx_2 \ldots \int dx_N \; \theta(-F(\nu)) \; , \tag{4}$$

where $\theta(z)$ is the step-function: $\theta(z < 0) = 0$ $\theta(z > 0) = 1$,



Well, we may include the body into the box of linear size a and volume $V_0 = a^N$, and write V_b identically as an average of $\theta(\nu)$ over V_0

$$V_b \equiv a^N \; \langle \, \theta(-F(\nu)) \rangle_\nu \quad \text{where} \quad (\nu \in V_0) \; , \tag{5}$$

The second factor is the average of $\theta(-F(\nu))$ over all configurations in the enclosing box.

There are many other examples of how expressions similar to Eq. (1) appear in different fields of science, but what concerns us now is the problem of having N so large that it is not possible to sum over all terms in a reasonable time even with the use of supercomputers. For example, if each variable takes only two values $i_k = \pm 1$, but there are $N = 100$ of them, then the total number of terms in the sum will be 2^{100} or roughly 10^{30}, and even Tera-flop (10^{12} operations per second) computers will fail to do the job. In a human life-time only a tiny fraction of all configurations can be accounted for. What is the hope then that the answer can be found with some reason-

able and controlled accuracy? [At this point we have to forget about exact answer!].

There are two separate considerations of why it is possible to find $\langle A \rangle$ with high accuracy.

Consideration #1. This is similar to the direct election of the president between two candidates. Yes, every vote counts, and there are exponentially many individual vote combinations. However we would like to know just one number—what is the percentage of "in favor of person $\mathcal{A}$"? It is crucial that the number of combinations with this number being $> 50\%$ is also exponentially large. You will probably agree that if many, say 100, elections are held in a raw and person $\mathcal{A}$ wins all of them then we have determined the public opinion who should be the president pretty accurately. So, we do not need to know all the vote combinations (in fact we have now idea about most of them), all we need is a **large representative set** of votes.

Exactly same reason makes MC simulations a useful numerical tool. The W and A functions typically take the same (or very close) value for exponentially many configurations ν, let's call them a subgroup $set_A\{\nu\}$. Now, if we split all configurations into subgroups $set_A\{\nu\}$, then all we need for a good estimate of the average value of A is a large representative set of configurations from different subgroups. The body volume problem nicely illustrates how it works. For all configurations in the body we have $\theta = 1$, for all other configurations we have $\theta = 0$. The algorithm of MC simulation is very simple then:

1. Initialize counters; Attempt=0; Result=0.
2. Use random numbers to seed a point inside the V_0 volume with the uniform probability density, e.g. $x_1 = a \cdot rndm()$,$x_2 = a \cdot rndm()$, etc.
3. update counters; Attempt=Attempt+1; Result=Result+$\theta(\nu)$; and go back to point 2. From time to time you may check how the calculation is going by printing $V_0 \cdot (Result/Attempt)$.

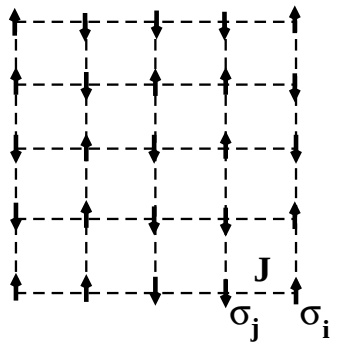
Of course, all we do in this algorithm is computing the fraction of volume occupied by the body which is also the probability if hitting it by generating points in V_0 at random.

Problem . Implement this algorithm and calculate the volume inside the 5-dimensional unit sphere $F = x_1^2+x_2^2+x_3^2+x_4^2+x_5^2-1 = 0$**.**

The other example is the simulation of magnetization for the Ising model which describes a system of lattice spin variables coupled by nearest neighbor (short hand notation is "n.n.") couplings

$$H = -J \sum_{<ij>} \sigma_i \sigma_j - h \sum_i \sigma_i \ . \tag{6}$$

Here J is the coupling energy, $< ij >$ stands for the nearest neighbor pairs of spins on a simple cubic one- two- three- or higher-dimensional square lattice, h is the effective magnetic field ($h = \mu B$, where μ is the magnetic moment), and $\sigma_i = \pm 1$ describes two spin states on each lattice site. The two-dimensional arrangement of spins is shown to the right.



Magnetization is defined as the difference between the number of up- and down-spins

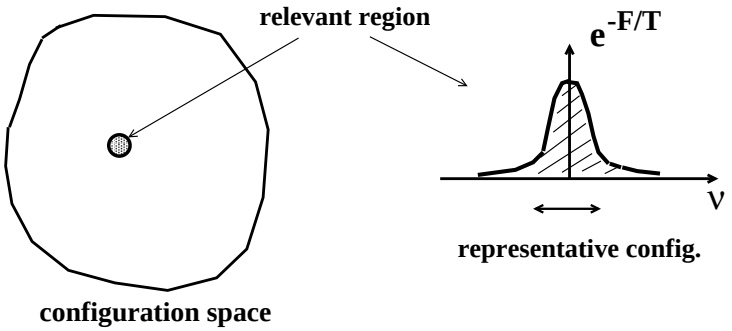
$$M = \sum_i \sigma_i \ . \tag{7}$$

Suppose we have to find the average modulus of magnetization. Then, in our formal notation, $\nu = \sigma_1, \sigma_2, \ldots \sigma_N$, $A(\nu) = |M(\sigma_1, \sigma_2, \ldots)|$, and $W(\nu) = \exp\{-H(\sigma_1, \sigma_2, \ldots)/T\}$. We notice, that for N spins, which have in total 2^N different configurations, there are only $N/2 + 1$ different values of $|M| = 0, 2, 4, \ldots, N$. To understand what $\langle\, |M| \rangle$ is, a representative set of configurations of order N will be sufficient. It should be appreciated that $N/2^N$ is an incredibly small fraction of configurations (in state of the art simulations N is as large as $N = 500\,000\,000$!)

Consideration #2.

In many cases (e.g. in statistical physics) the structure of the W-function is such that only a tiny fraction of configurations determine the answer. All the other configurations have extremely small W, so small that it beats their large number (in other words the number of "unrepresentative" configurations does not compensate the small value of W, and their contribution to the answer remains exponentially small). What is necessary in this case, is a clever procedure of summing up only the most important terms. Note, that this is **not** an optimization problem because one has to add many terms to get the right answer, and the dominant contribution may come not from

the maximum value of W, but from the maximum of W times the number of configurations with roughly the same W, i.e., from the maximum of $We^S \sim e^{-F/T}$ where the following statistical physics notations were used:
number of configurations with the given energy $= e^S$;
S=entropy
Helmholtz free energy $F = E - TS$



This observation is both a "blessing" and a "trouble". On one hand, we understand that we can discard most of configurations and deal with a much smaller number of terms. However, even if the relevant region is only a tiny fraction, e.g. as small as $\sqrt{2^N}$ (an enormous reduction from 2^N !!!), still it is very large and the consideration #1 should be used. On the other hand, it is not possible any more to select configurations at random without thinking since with near certainty the selected configuration will be having vanishingly small W and completely irrelevant for the answer. It would be nice, of course, if we can select configurations with probabilities proportional to their weight, since then configurations with large weights are most likely to be selected. But how to achieve this goal in the general case? The solution is provided by the **Metropolis algorithm** (MetA) (1953).

Metropolis algorithm

The algorithm of summing up only the most important terms in such a way that in the limit of infinitely long calculation the result converges to the exact value was found by **Metropolis et al.** when the time was just right for the computer age. In the spirit of the body-volume calculation, MetA suggests to replace the original sum over all configurations with the

stochastic sum

$$\frac{\sum_\nu A(\nu)W(\nu)}{\sum_\nu W(\nu)} \quad \Longrightarrow \quad \frac{\sum'_\nu A_\nu}{\sum'_\nu}$$

where configurations to be included into the sum $\sum'$ are generated using random numbers, and one actually includes them into the sum with certain probabilities. The following two rules (laws) must be satisfied:

• *Starting from any initial configuration the process of stochastic generation of new configurations to be included into the sum has to allow all other configurations to be generated in a long run - this is the* **ergodicity** *requirement. It is very important to satisfy it, since we are not going to miss any of the terms without being able to look at them. This rule is also very important for the proof that the simulation converges to the exact answer (which is the sum of all terms).*
• *The probability to have a configuration ν in the stochastic sum is proportional to W_ν*

Let us first prove that in the limit of infinite calculation time the two answers - full sum and the stochastic sum - will agree with each other exactly. Next, we will discuss how one can arrange a simple random process so that the probability of the configuration to be included into the sum is proportional to its weight. Indeed, in the infinite time limit we can write:

$$\frac{\sum'_\nu A_\nu}{\sum'_\nu} \; = \; \frac{\sum_\nu A_\nu M_\nu}{\sum_\nu M_\nu} \quad \Longrightarrow \quad \frac{c\sum_\nu A_\nu W(\nu)}{c\sum_\nu W(\nu)}$$

The first equality follows from the ergodicity requirement that all configurations will be accounted and M_ν is telling us how many times. The second transformation for the number of times each configuration appeared in the sum is possible because M_ν is proportional to the configuration weight, i.e. $M_\nu = cW(\nu)$. Canceling the c-factors we arrive at the original expression for $\langle A\rangle$, see Eq. (1).

The algorithm of including/accepting configurations into the stochastic sum is very simple.

1. Initialize counters: Z=0; Result=0;
   Choose any configuration ν to be the first one to be included into the sum

2. Include the configuration into the sum: $Z = Z{+}1$; Result=Result$+A(\nu)$;

3. Suggest another configuration $\nu\,'$ which derives from ν by, e.g., suggesting to change the value of one of the variables i_k. The variable i_k and its new value may be selected at random, e.g. play random numbers to pick which variable to change $k = [rndm * N] + 1$, and if i_k can take any value from the list $a_1, a_2, \ldots, a_K$ with K-entries, then, play random numbers to pick the new value for i_k to be the list entry number $s = [rndm * K] + 1$. In this example the suggested configuration change $\nu \to \nu'$ is $\{i_1, \ldots, i_k, \ldots, \} \to \{i_1, \ldots, i'_k, \ldots, \}$, where $i'_k = a_s$. The procedure of getting ν' from ν is called an **update**. At this point the algorithm is very flexible, and many different updates can be designed. This freedom may be used to maximize the efficiency of the algorithm, or to minimize the programming work. Moreover, in the same code several updates may be used (more details later).

4. Now we have to figure out how to include the new configuration into the sum with probability proportional to it's weight. Since we start from ν being already accepted into the sum, the probability to accept $\nu\,'$ may be determined by simply comparing the $W(\nu)$ and $W(\nu\,')$ weights. In particular, we have to make sure that for any accepted ν we accept ν' on average $W(\nu\,')/W(\nu)$ times. The requirement $M_\nu = cW(\nu)$ is established then through a series of ratios

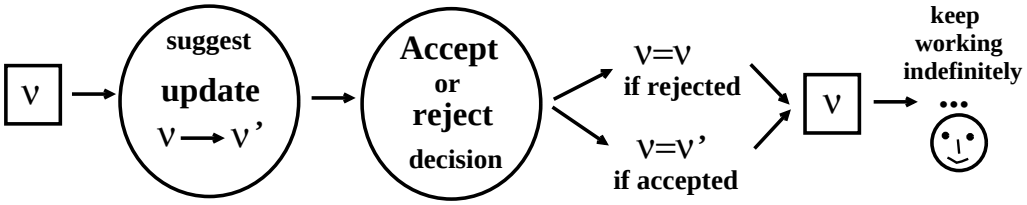
$$M_{\nu'} = M_\nu \frac{W(\nu')}{W(\nu)} = W(\nu') \frac{M_\nu}{W(\nu)}$$

$$M_{\nu''} = M_{\nu'} \frac{W(\nu'')}{W(\nu')} = W(\nu'') \frac{M_{\nu'}}{W(\nu')} = W(\nu'') \frac{M_\nu}{W(\nu)}$$

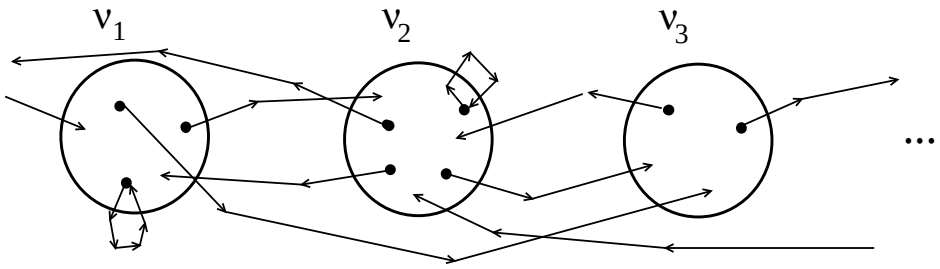
$$M_{\nu'''} = M_{\nu'''} \frac{W(\nu''')}{W(\nu'')} = W(\nu''') \frac{M_\nu}{W(\nu)}$$

   i.e. $M_\nu / W(\nu) = const$ as required!

5. Depending on the decision to accept or reject the update, modify the configuration accordingly: if $\nu\,'$ was rejected, then do nothing, $\nu = \nu$; if $\nu\,'$ is accepted then implement the change $\nu = \nu\,'$. Proceed to point 2 above in a cycle.



The only crucial equation which is at the heart of all Monte Carlo simulations is the so called **balance equation** which tells us the exact relation between the probability of accepting the new configuration into the sum and the ratio of configuration weights. Imagine an infinite set of computers each performing exactly the same MC simulation as described above (this trick is used to avoid waiting infinitely long time for the simulation to converge to the exact answer!); alternatively, imagine that we run the same simulation on a multi=∞-processor computer and each processor is working independently. Since the set of computers is very large, for each configuration we may count how many computers are having it at a given moment of time; call it M_ν. Since the simulation is done according to the rules, at any moment we have $M_\nu = cW(\nu)$ to make sure that averaging over all computers we correctly reproduce $\langle A \rangle$. This, in particular, means that the distribution M_ν is static and time independent. During the next operational cycle, all computers will suggest to update their current configurations, and many will accept the changes.



In this figure an ensemble update generates a flow of computers between different configuration states, but all flows cancel each other so that M_ν remain the same. Mathematically, the cancellation of all flows can be written

as

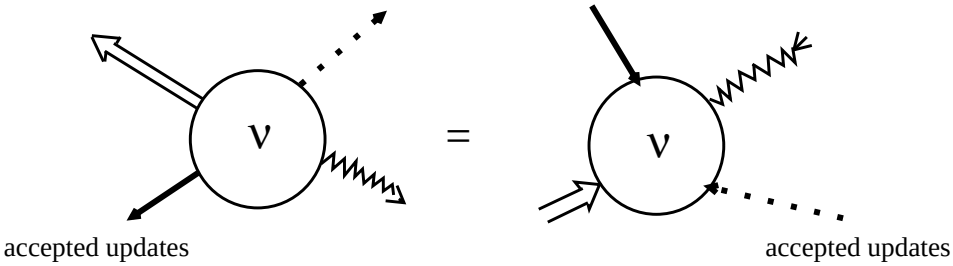
$$M_\nu \sum_{\nu\,',u} p_u P_u^{acc}(\nu \to \nu\,') = \sum_{\nu\,',u} M_{\nu\,'} p_u P_u^{acc}(\nu\,' \to \nu) \; . \tag{8}$$

The l.h.s. of this equation reads <<the decrease in the number of configurations ν is given by their number times the sum of all probabilities which change ν to any $\nu\,'$ using any of the updating procedures (here "u" is the index of the updating procedure)>>. Correspondingly, p_u is the probability of applying update "u" in the simulation, and $P_u^{acc}(\nu \to \nu')$ is the probability that the modification suggested by "u" is actually accepted—this quantity is also called the **acceptance probability**. The r.h.s of Eq. (8) counts all cases which result in the accepted ν when updates are performed on other configurations.

Finally, we substitute here $M_\nu = cW(\nu)$ to get

$$W_\nu \sum_{\nu\,',u} p_u P_u^{acc}(\nu \to \nu\,') = \sum_{\nu\,',u} W_{\nu\,'} p_u P_u^{acc}(\nu\,' \to \nu) \; . \tag{9}$$

Eq. (9) is called the **balance equation** because is requires that flows "out" and "in" balance each other.



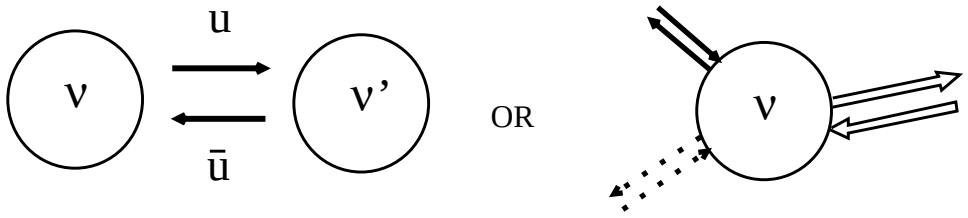
The solution of the balance Eq. depends on the updating scheme. The updating scheme by itself is an art because there are infinitely many ways one can design updates. The simplest way to satisfy Eq. (9), which is also general enough for most practical applications, is to *balance updates in pairs*. If "u" can transform ν to $\nu\,'$ and $\bar{u}$ can perform the opposite transformation of $\nu\,'$ to ν, then the pair is balanced if

$$W_\nu p_u P_u^{acc}(\nu \to \nu\,') = W_{\nu\,'} p_{\bar{u}} P_{\bar{u}}^{acc}(\nu\,' \to \nu) \; , \tag{10}$$

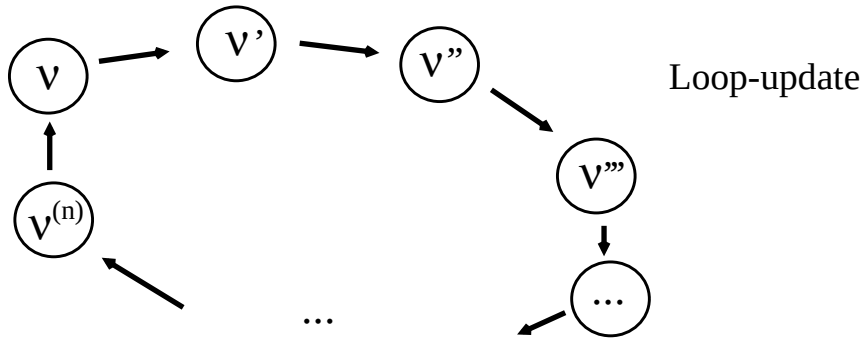
(no summation !), or

$$\frac{P_u^{acc}(\nu \to \nu\,')}{P_{\bar{u}}^{acc}(\nu\,' \to \nu)} = \frac{W_{\nu\,'}}{W_\nu} \, \frac{p_{\bar{u}}}{p_u} \equiv R \; , \tag{11}$$

where R is called the **acceptance ratio**. Eq. (11) is called the **detailed balance Eq.** for obvious reasons: if each term in the sum is balanced then the whole sum is balanced as well. The idea is illustrated below



One may also try other schemes where updates form loops, like



and solve a chain of balance equations. It is possible to do so in special cases, but one has to be very accurate in doing this.

Spin-flip algorithm for the Ising model

It is probably best at this point to illustrate how MetA works by employing the case of the Ising model. Suppose we are interested in the average energy (not magnetization modulus, just for a change), so $A(\nu) = E(\nu) = -\sum_{<ij>} J\sigma_i\sigma_j$, and $W(\nu) = e^{-\beta E(\nu)}$. It is sufficient to use only one update to generate an ergodic process. This update consists of the following steps
(i) *choose at random any site in the lattice; well, this is familiar* $k = [rndm * N] + 1$.
(ii) *suggest to flip the spin on the selected site,* $\sigma_k{}' = -\sigma_k$*, i.e,* ν' *is different from* ν *by the value of only one spin variable. By repeating this update*

many-many times we may, obviously, generate any other configuration.

All we need to start calculating is the solution of the detailed balance Eq. for the acceptance probabilities. Since we have only one update, and this update is self-similar (going backwards from $\nu\,'$ to ν we use the same updating scheme), the balance equation says

$$\frac{P_{acc}(\sigma_k \to \sigma_k\,')}{P_{acc}(\sigma_k\,' \to \sigma_k)} = e^{-\beta[E(\nu\,')-E(\nu)]}\,\frac{N}{N} = e^{-\beta[E(\nu\,')-E(\nu)]}\,. \tag{12}$$

The $1/N$ factors relate to the probability of selecting site k out of N sites (formally, we have to consider updating different spins as different updates). Denoting the energy difference as

$$\Delta E_k = E(\nu\,') - E(\nu) = 2J\sigma_k \sum_{<kj>} \sigma_j\ , \tag{13}$$

we finally get

$$\frac{P_{acc}(\sigma_k \to \sigma_k\,')}{P_{acc}(\sigma_k\,' \to \sigma_k)} = e^{-\beta\Delta E_k} \equiv R\,. \tag{14}$$

In this particular example the **acceptance ratio** is the ratio of configuration weights, nothing else.

There are two quantities to be determined from one equation. The solution is not unique then. One choice is

$$\begin{aligned} P_{acc}(\sigma_k \to \sigma_k\,') &= \frac{R}{1+R} = \frac{e^{-\beta E(\nu\,')}}{e^{-\beta E(\nu)} + e^{-\beta E(\nu\,')}} \\ P_{acc}(\sigma_k\,' \to \sigma_k) &= \frac{1}{1+R} = \frac{e^{-\beta E(\nu)}}{e^{-\beta E(\nu)} + e^{-\beta E(\nu\,')}} \end{aligned} \tag{15}$$

Easy to check that the balance Eq. is satisfied, and P's can be considered as probabilities, i.e. they both are ≤ 1.

The other (more elegant from my point of view) solution is

$$P_{acc}(\sigma_k \to \sigma_k\,') = \begin{cases} R & \text{if} \quad R < 1 \\ 1 & \text{if} \quad R \geq 1 \end{cases} \tag{16}$$

$$P_{acc}(\sigma_k\,' \to \sigma_k) = \begin{cases} 1 & \text{if} \quad R \leq 1 \\ 1/R & \text{if} \quad R > 1 \end{cases} \tag{17}$$

It also has an advantage that acceptance rates are larger, and it is easier to do numerically the step that follows next.

If the random number generated is smaller than $P_{acc}(\sigma_k{}' \to \sigma_k)$

$$rndm < P_{acc} \ , \tag{18}$$

the new configuration is accepted (obviously we do not need to generate rndm if $P_{acc} = 1$).

"Accepted" means that we have to update the configuration file and may add the new configuration to the stochastic sum. We may also choose to include into the sum only every second accepted configuration; the final answer may not depend on this since all we do then is $M_\nu \to M_\nu/2$. More generally, we may include into the sum only every m's accepted configuration. Remember, however, that m has to be set once and for all in a given MC simulation—if you change m during the simulation the balance Eq. will be violated unless you figure out how to incorporate changing m into the balance Eq. If the system is large, it usually makes sense to perform at least N updates before adding the configuration to the sum. The frequency of adding accepted configurations to the statistics is arbitrary since the final result does not depend on it. However in certain cases it may be time consuming to evaluate the configuration value $A(\nu)$, and to optimize the calculation time one has to spend at least half of the time on updates. For the average energy in the Ising model this problem does not exist, because $E(\nu') = E(\nu) + \Delta E_k$, and after each update we know both the previous configuration energy $E(\nu)$ (by simply keeping it in the memory) and the energy difference, so we may update also the current energy variable.

The simple updating scheme discussed above can be modified a little using the so called **heat bath** method. Instead of suggesting to flip the selected spin variable, we may try to play random numbers so that the probabilities of having σ_k up or down are reproducing the exact statistics of σ_k when the neighboring spins are frozen. In other words, we do the best we can for σ_k in its present environment. So, after σ_k is selected, we calculate $B = (J/T)\sum_{kj}\sigma_j$, and with probability p_1 we propose the new value to be $\sigma'_k = 1$ and with probability $p_{-1} = 1 - p_1$ we propose $\sigma'_k = -1$, where

$$p_\sigma = \frac{e^{\sigma B}}{e^{\sigma B} + e^{-\sigma B}} = \frac{e^{\sigma B}}{2\cosh B} \ . \tag{19}$$

Note, that the current value of σ_k does not play any role in the selection process.

We may now write the acceptance ratio for the "heat bath" scheme

$$R = e^{\sigma'_k B - \sigma_k B} \left(\frac{N}{N}\right) \left(\frac{p_{\sigma_k}}{p_{\sigma'_k}}\right) \equiv 1 \; .$$

Whatever is suggested is definitely accepted now! The second factor is the ratio of probabilities to select updates which transform $\{\sigma_1, \ldots, \sigma_k, \ldots\}$ to $\{\sigma_1, \ldots, \sigma'_k, \ldots\}$ and $\{\sigma_1, \ldots, \sigma'_k, \ldots\}$ to $\{\sigma_1, \ldots, \sigma_k, \ldots\}$ [sorry, for boring reminders of what we do explicitly]. The result we got is not surprising since we literally do what "the doctor prescribed" - suggest configurations with probabilities proportional to their weight in the most straightforward form.

The heat bath scheme for the Ising model is not, however, more efficient than the single flip Metropolis method. In fact, some of the heat bath updates do nothing by leaving the spin value unchanged; in the Metropolis scheme this would be considered as a rejection. The ratio of rejections is given by Eq. (19) and it is larger than in the single flip method for any B. However, when spins take more than two different values, e.g. in continuous spin or Potts models, then the heat bath algorithm becomes more efficient than the single flip one.

Problem. Write down the spin-flip MetA for the Ising model in any language you know. You have to develop it yourself because many problems later in the course will be based it. Consider a two-dimensional square lattice with $L = 6$ spins per dimension, choose any value of J/T in the range $(1, 3)$ and evaluate the average magnetization modulus.

We now turn to the question of how fast the MC scheme converges to the final answer.

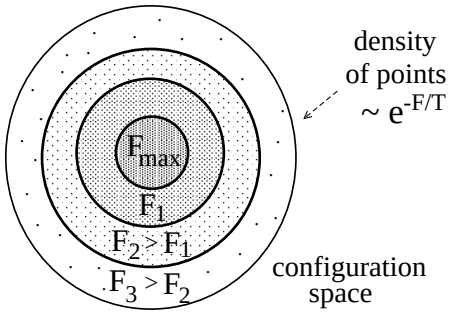
Example of the Fortran code for the Ising model:
(Random number generator is separate from this code)

```
integer, parameter :: N=6                  !this is linear system size
real, parameter :: JT=1.5                  ! this is the ratio J/T
integer, parameter :: L=100000             ! how often to print (in updates)
integer*2 :: s(N,N)                        ! spin configuration
integer :: counter                         ! counter for printing
real :: Z,A,M,dE                           ! sum counters
real :: R                                  ! acceptance ratio

s=1                                        ! initialize configuration
M=N*N                                      ! initial value of magnetization
Z=0 ; A=0; counter=0                       ! initial Z, A, and counter

DO                                         ! Main MC cycle
Z=Z+1; A=A+ABS(M)                          ! include to the sum
counter=counter+1;                         ! increase counter
k1=rndm()*N+1; k2=rndm()*N+1;              ! lattice point for update

dE=0                                       ! calculating the energy change
I=k1-1; if(I>0) dE=dE+s(I,k2)              !
I=k1+1; if(I<N+1) dE=dE+s(I,k2)            !
I=k2-1; if(I>0) dE=dE+s(k1,I)              !
I=k2+1; if(I<N+1) dE=dE+s(k1,I)            !
dE=dE*JT*2.*s(k1,k2)                       ! energy change calculated

R=exp(-dE)                                 ! acceptance ratio
if( R>1. .OR. rndm() <R ) then             ! decision to accept
s(k1,k2)=-s(k1,k2)                         ! update configuration
M=M+2.*s(k1,k2)                            ! update M value
endif                                      ! update done

if(mod(counter,L)==0) then                 ! print result after L updates
print*, '< A >=', A/Z                      ! print
counter=0; endif                           ! reset counter
ENDDO                                      ! looks like we are done

END
```

Convergence and autocorrelation time

As discussed above, the wonder of MC simulations is that the result becomes accurate even when only an incredibly small fraction of relevant configurations is sampled. This is because A_ν is not sensitive to the detailed structure of ν, and enormous number of different configurations result in very much the same A_ν. It means that looking at some of them already gives the right idea about the answer. Of course, sampling only some of the configurations we know the answer only approximately. Moreover, the simulation result is approaching the exact answer only when the set of accepted configurations covers the whole relevant region (i.e., the region around the maximum of $e^{-F/T}$)

Having many points in each region is sufficient for an accurate estimate of what is the $\langle A \rangle$ value.



Thus, starting from some initial configuration, and performing local updates (by local I mean updates which change only one, or several variables around some point and leave the rest of the configuration intact) it will take some time before the set of accepted configurations will cover the relevant region. Only then we may hope that our result is meaningful. Let the simulation result after time t_0 be

$$< A >_{0,t_0} = < A >_{\text{exact}} + \delta A_{t_0}$$

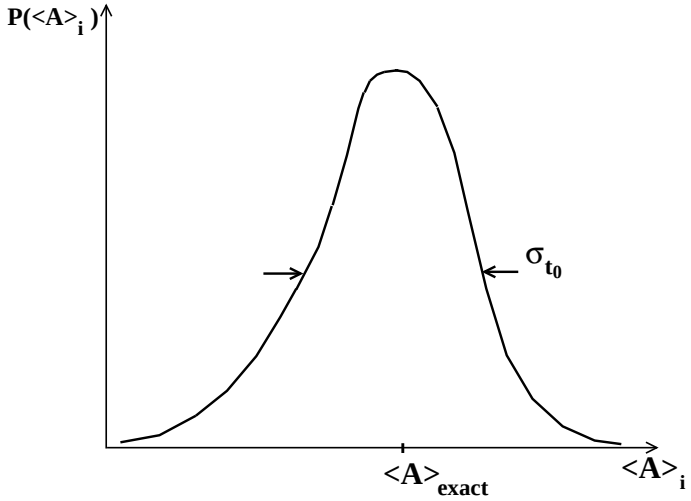
Imagine that t_0 is long enough (how to define "long" quantitatively will be discussed later), and we have a reasonably good sampling of the relevant region in the phase space. Let us now double the calculation time and evaluate $< A >_{t_0,t_0+t_0}$ - the average over configurations generated during the period of time between t_0 and $t_0 + t_0$. This average is no different from the first one except that now we are dealing with another set of configurations from the same region of the configuration space, i.e., we have

$$< A >_{t_0,t_0+t_0} = < A >_{\text{exact}} + \delta A_{t_0,t_0+t_0}$$

with another random deviation $\delta A_{t_0,t_0+t_0}$. Repeating this procedure many times and denoting

$$< A >_{(n-1)t_0,nt_0} =< A >_n =< A >_{\text{exact}} + \delta A_n$$

we obtain a collection of independent random variables, call them "block averages") with the same average $< A >_{\text{exact}}$ and standard deviation σ_{t_0} (at this point the distribution of block averages is not necessarily Gaussian)



The central limit theorem then says that the average over the entire simulation, i.e. over time period $t = nt_0$

$$< A >_{0,t} \equiv < A >= \frac{\sum_{i=1}^{n} < A >_i}{n}$$

has a much smaller standard deviation from the exact answer

$$\sigma_t^2 = \frac{1}{n^2} \sum_{i=1}^{n} \sigma_{t_0}^2 = \sigma_{t_0}^2 / n \ ,$$

or

$$\sigma_t = \sigma_{t_0} \sqrt{t_0/t} \ , \tag{20}$$

and its distribution is Gaussian. It means that MC results are getting more and more accurate with time, but only as $1/\sqrt{t}$. Still, it is possible to achieve accuracy better than 10^{-4} in most calculations, and up to 10^{-6} in the state of the art procedures. For example, the critical temperature of the 3D Ising model is known now to accuracy 0.2216546(1) (the number in parenthesis gives the standard deviation from the reported answer in terms of last digit, we have to read this as 0.2216546 ± 0.0000001).

By definition σ_t may be calculated as the dispersion of block-averaged results

$$\sigma_t = \sqrt{\frac{\sum_{i=1}^{n}(< A >_i - < A >)^2}{n^2}}\ , \tag{21}$$

and this is how the calculation error may be obtained in MC simulations. There is one unknown parameter though in the procedure, namely t_0. We assumed that t_0 is sufficiently long so that $< A >_n$ are statistically independent. If $< A >_n$ are correlated, then the whole analysis is wrong! One method of estimating error bars which avoids this uncertainty is based on making larger and larger blocks as the simulation progresses.

The blocking method

Let us look at the simulation data in prime terms: instead of working with the list of block averages $< A >_1, < A >_2, \ldots, < A >_n$ we may consider the list of block numerators $R_1, R_2, \ldots, R_n$ with each block containing Z_B accepted configurations, $R_i = \sum_{\nu \in i-th\ block} A_\nu$. By definition, $< A >_i = R_i/Z_B$. We may now calculate not just one errorbar, $\sigma_t^{(n)}$ using Eq. (21) for n blocks, but also $\sigma_t^{(m)}$ for the smaller number m of larger blocks. If $n = m * j$ then we simply "glue" j blocks together to form "superblocks"

$$< B >_i = \frac{1}{j \cdot Z_B} \sum_{k=1+(i-1)\cdot j}^{i \cdot j} R_k \tag{22}$$

and use the $\{< B >_i\}$-sequence to estimate the errorbar

$$\sigma_t^{(m)} = \frac{1}{m} \sqrt{\sum_{i=1}^{m}(< B >_i - < A >)^2}\ , \tag{23}$$

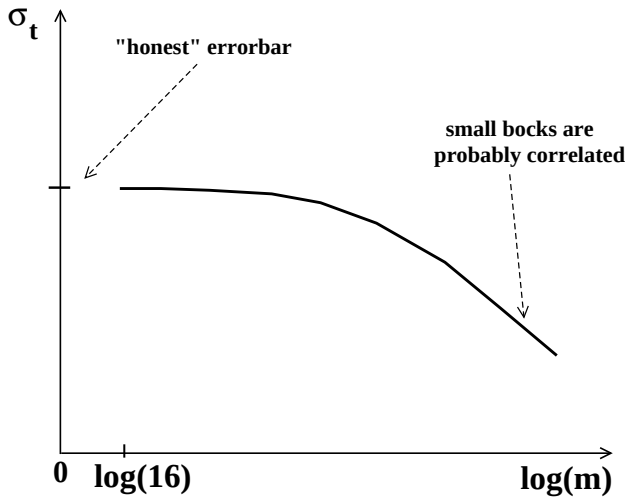
For large n, e.g. for $n = 2^L$, we may construct larger and larger superblocks by doubling their sizes many times. To calculate the dispersion we need many blocks, so it is wise to keep the number of blocks sufficiently large, say > 16.

Now, if the block numerators R_i are statistically independent, the value of $\sigma_t^{(m)}$ will not depend on m. But sufficiently large blocks will eventually become statistically independent, so the expectation is that $\sigma_t^{(m)}$ will saturate

to the correct errorbar value to matter whether the initial choice of t_0 was good or bad. More elaborately, one may form superblocks using

$$< B >_i = \frac{1}{j \cdot Z_B} \sum_{k=1}^{j} R_{i+(k-1)*m} \ , \tag{24}$$

i.e. by combining blocks well separated in time, or just by selecting j blocks at random. In either case we expect this



Whatever the method, it has to produce the same "honest" errorbar in the long run simulation obeying the sqrt-law (20).

The bootstrap and jackknife methods

The idea of making large randomized superblocks to estimate errorbars has other solutions. In the **bootstrap** method superblocks are formed by combining n blocks selected at random from the list $\{R_i\}$. Some of the prime blocks may contribute to the superblock more than once, some may not contribute at all. After a large number of superblocks are generated, say m, one estimates the errorbar, $\sigma_t^{(bootstrap)}$, from

$$\sigma_t^{(bootstrap)} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (< B >_i - < A >)^2} \equiv \sqrt{\overline{(\delta B)^2}} \ . \tag{25}$$

The bar over $(\delta B)^2$ stands for the average over the selection process. The last equality makes it clear that the answer is independent of $m >> 1$, which

should be the case since m is arbitrary. The advantage of the bootstrap method is that its error bar does not depend on the choice of t_0 for the prime block.

If block averages *are* statistically independent then another method of estimating errorbars is called the **jackknife** [I do not like it, since I do not see any difference with Eq. (21) which is the most "natural" procedure]. In the **jackknife** method, superblocks are formed by combining all but one block

$$< B >_i = \frac{1}{(n-1) \cdot Z_B} \sum_{k \neq i}^{n} R_k \ , \tag{26}$$

and the dispersion of superblock answers gives the errorbar as

$$\sigma_t^{(jackknife)} = \sqrt{\sum_{i=1}^{n} (< B >_i - < A >)^2} \ . \tag{27}$$

Problem. Prove that the bootstrap method reproduces correct answer for the errorbar when contributions R_i are statistically independent and the number of superblocks is large.

Problem. Develop your code for the two-dimensional Ising model further to be able to do the analysis of errorbars for the magnetization modulus using the bootstrap method. For system parameters $L = 16$, $J/T = 0.6$, and the size (number of updates performed) of the smallest block equal to $L \times L = 256$ make a plot of $\sigma_t^{(bootstrap)}$ (here $t = n =$# of blocks) and check whether it is decaying as $\sim 1/\sqrt{n}$ for large n. Make sure that in your simulation n is large enough, e.g. 10^6 or larger.

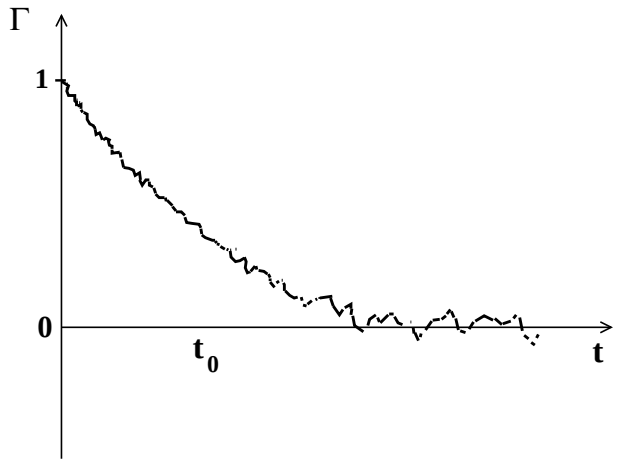
Autocorrelation time

Since configurations derive one from another (although not in a deterministic way, but still only a small part of the cnf. is updated in local schemes) it is not at all obvious when $< A >_n$ will become statistically independent, and we need some means to check it. In fact, block averages are strongly correlated if t_0 is small. To study these correlations we keep a long record of subsequent values of A_ν, i.e., $A_1 = A_{\nu_1}, A_2 = A_{\nu_2}, \ldots, A_M = A_{\nu_M}$, taken at equal number of MC updates apart from each other. Let us denote the number of updates between the records as Δt and use it as a time unit, i.e. if ν_{i+1} is obtained from ν_i after performing 1000 updates, then the time unit is $\Delta t = 1000$ updates. Next, we look how A_i fluctuate in time around the average

$$\Gamma_j = \Gamma(t/\Delta t) = \frac{\sum_{i=1}^{M-j} \delta A_i \, \delta A_{i+j}}{\sum_{i=1}^{M-j} \delta A_i^2} \ , \quad \text{where} \quad \delta A_i = A_i - < A > \ .$$

The so called **autocorrelation** function Γ tells us how different values of A_ν are correlated in time. The typical form of the autocorrelation function is

We see that correlations die away after some characteristic time t_0. We may define the characteristic time formally as $t_0 = \Delta t/2 + \sum_{i=1}^{\infty} \Gamma_i \Delta t$ (in reality we sum up to some very large number and verify that the result has converged)



This definition, which works for any shape of Γ_i, is also known as an **integrated autocorrelation time**. If the time decay of Γ is purely exponential, $\Gamma = e^{-t/\tau}$, then the integrated autocorrelation time is the same as τ.

In a system with N degrees of freedom we expect correlations to persist at least up to N updates until we have a chance to change every degree

of freedom (otherwise the two configurations will share identical values for some of the variables and will be obviously correlated). This sets a natural time unit which is called a Monte Carlo sweep, $\Delta t = \text{one sweep} = N$ updates. Usually the autocorrelation time is measured in the MC sweeps, and $t_0 \gg 1$ would mean that we have to address each degree of freedom many times before we get an independent value A_i. In practice, the study of autocorrelations takes a lot of CPU time since very large $M \sim 10^6 \div 10^7$ are necessary to suppress noise in the autocorrelation function.

In general, t_0 will depend on the quantity measured, i.e., $t_0(A) \neq t_0(B)$. Also, it may happen that there are several time scales in $\Gamma(t)$, especially when the system is inhomogeneous, or disordered. The other important remark concerns algorithms used to update MC configurations. Some of them have very long autocorrelation times and thus are ineffective. Clever algorithm may improve efficiency by a factor of thousand or more; the most famous example of this kind is the invention of the cluster algorithm by Swendsen and Wang (for Ising-like models).

Problem. Using your code for the two-dimensional Ising model (take system parameters $L = 16$, $J/T = 0.6$), make a list of 10^6 magnetization measurements separated by one MC sweep, and study the correlation function Γ and the integrated autocorrelation time t_0.

Thermalization problem

Starting MC simulation from some arbitrary configuration we are unlikely to hit the relevant region. For a while the algorithm will work its way to the state of thermal equilibrium. This initial period is called the thermalization stage. Any data collected during thermalization process should be trashed because they are artifacts of the initial condition and algorithm details. In fact, it is wise not to update Z and counters for other calculated quantities for a while, and wait until the thermal equilibrium is reached. In most cases the thermalization time is much longer than t_0, as long as 100 or even $1000t_0$. It does not hurt much even if we spend on themalization a significant fraction of the total CPU time, say 20%. Indeed, the convergence of errorbars in MC simulations is proportional to $1/\sqrt{t}$, thus the errorbars for the calculation time t and $0.8\,t$ relate as $\sqrt{0.8} \approx 0.9$, a 10% difference only.

It is hard to express t_{therm} in terms of t_0 because the latter is the property of the equilibrium state, and the former depends also on the algorithm performance for states far from equilibrium. The "based on experience" rule is to monitor the convergence of errorbars using the blocking method and wait until the superblock doubling does not change the estimate for σ_t. Next, t_{therm} may be set equal or longer than this waiting time and the actual simulation is started. One may also take advantage of "well thermalized" initial configurations from previous MC runs, especially when the new simulation parameters are only slightly different.

Other quantities to measure.

Energy and magnetization modulus are the simplest quantities to measure for the Ising model. They also have very simple **MC estimators**. By MC estimator for quantity A is understood an expression $A(\nu)$ which allows to calculate $\langle A \rangle$ using Eq. (3). Sounds almost self evident, especially for energy and magnetization which have estimators $E = -J \sum_{<ij>} \sigma_i \sigma_j$, and $M = \sum_i \sigma_i$. However, **specific heat** is already more tricky: formally $C = d\langle E \rangle / dT$; it may be calculated by subtracting average energies at two close values of temperature

$$C \approx \frac{\langle E \rangle_{T_1} - \langle E \rangle_{T_2}}{T_1 - T_2} \ .$$

This is not the best we can do however, because the procedure is not exact for large $|T_1 - T_2|$, and the error bars shoot up for small $|T_1 - T_2|$. However, if we do the differentiation analytically, then

$$\begin{aligned} C &= \frac{d}{dT} \frac{\sum_\nu E(\nu) e^{-E(\nu)/T}}{\sum_\nu e^{-E(\nu)/T}} \qquad (28) \\ &= \frac{1}{T^2} \left(\frac{\sum_\nu E^2(\nu) e^{-E(\nu)/T}}{Z} - \frac{\left(\sum_\nu E(\nu) e^{-E(\nu)/T} \right)^2}{Z^2} \right) = \frac{1}{T^2} \left(\langle E^2 \rangle - \langle E \rangle^2 \right) \ , \end{aligned}$$

the specific heat is expressed in terms of the energy dispersion, and both terms in this expression have MC estimators.

At the phase transition the **magnetic susceptibility** per particle $\chi = L^{-d}\, dM/dH|_{H=0}$ (here d is the dimension of space) diverges. The trick of deriving the estimator for χ is to pretend that we calculate magnetization

in finite magnetic field, differentiate $M(H)$, and set $H = 0$ in the final expression (the analogy with the specific heat is straightforward):

$$\begin{aligned}
\chi &= \frac{1}{L^d}\frac{d}{dH}\frac{\sum_\nu M(\nu)e^{-[E(\nu)-hM(\nu)]/T}}{\sum_\nu e^{[-[E(\nu)-hM(\nu)]/T}}\Big|_{H=0} \\
&= \frac{1}{TL^d}\left(\frac{\sum_\nu M^2(\nu)e^{-E(\nu)/T}}{Z} - \frac{\left(\sum_\nu M(\nu)e^{-E(\nu)/T}\right)^2}{Z^2}\right) = \\
&= \frac{1}{TL^d}\left(\langle M^2\rangle - \langle M\rangle^2\right) \equiv \langle M^2\rangle/TL^d \,, \qquad (29)
\end{aligned}$$

The last identity follows from the fact that $\langle M\rangle \equiv 0$ by symmetry. In fact, the differentiation with respect to some Hamiltonian parameter is a very useful general trick to derive MC estimators, and we will see more examples later.

In the standard Metropolis scheme the calculation of **entropy**, S, is a problem. Later, in the discussion of re-weighing (or **importance sampling**) techniques and the entropy sampling/flat histogram approach, we will see how S can be simulated. However, we may also make use of the thermodynamic relation between entropy and the specific heat integral

$$S = S(T_0) + \int_{T_0}^{T} \frac{C(T)dT}{T} \,.$$

If we are not interested in the value of $S(T_0)$ then we may stop here. If absolute values of $S(T)$ are of interest, then (assuming that the ground state is non-degenerate and $S(T = 0) = 0$ —the third law of thermodynamic) we have to integrate all the way to $T = 0$

$$S = \int_{0}^{T} \frac{C(T)dT}{T} \,.$$

This approach requires that we evaluate specific heat at many T points. The integration itself is not a big problem, and can be done using linear- or higher-order interpolation techniques.

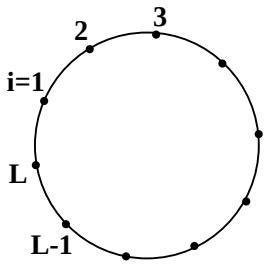
The other quantity of interest in MC simulations is the spin-spin correlation function which tells us how the direction of one spin, say at the origin of the coordinate system, influences other spins. Formally:

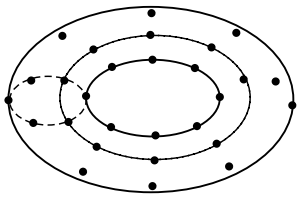
$$g(i,j) = \langle\, \sigma_i \cdot \sigma_j \rangle \,,$$

The problem with $g(i,j)$ is not in what is its MC estimator (this is obvious) but how to make it work efficiently. In the d-dimensional system of linear size L, the correlation function contains L^{2d} points!

First, let us introduce the so-called **periodic boundary conditions**. They are used to get rid of the edge effects and make the system translation invariant, i.e. when every point in the system is no different from any other point. Periodic boundary conditions for the system with the linear sizes $L_1, L_2, \ldots, L_d$ mean that spins $\sigma(i_1, \ldots, i_\mu + L_\mu, dots)$ and $\sigma(i_1, \ldots, i_\mu, dots)$ are identical for any direction $\mu = 1, \ldots, d$.

In one dimension, the geometry of the sample is a circle, i.e. the clockwise neighbor of spin $\sigma(L)$ is spin $\sigma(1)$. In two dimensions, the geometry of the sample is a torus.





In higher dimensions you simply have to imagine that starting from any lattice point i and going exactly L_μ steps in the direction μ you end up back at i. If the system is translation invariant then $g(i,j)$ may depend on $i-j$ only, since all points have identical properties, $g(i,j) = g(0, j-i) \equiv g(j-i)$. This alone saves us a lot of memory, since we need only $N = L^d$ points to consider. Using simple cubic lattice rotation symmetries it is possible to further reduce the number of points.

The other problem comes from the count of how many operations are required to make the measurement of $g(j-i)$ for a given configuration. If we address all pairs of spins and add the product $\sigma_i \cdot \sigma_j$ to the counter for $g(j-i)$, i.e. $g(j-i) = g(j-i) + \sigma_i \cdot \sigma_j$ we will perform $N(N-1)/2$ operations. If measurements are performed every MC sweep (N updates), then most of the CPU time will be spent on the g-measurement, and not on the simulation itself. This is unacceptable. One solution is to measure less frequently, for example once every $N(N-1)/2$ updates, but this might be much longer than the autocorrelation time. The other solution is pick at random only N spin pairs in the g-measurement instead of looking at all of them. The ultimate solution is to use the **fast Fourier transform** technique.

Let us Fourier transform $g(i)$:

$$g(j) = N^{-1} \sum_k \tilde{g}(k) e^{ikj} \ , \quad \tilde{g}(k) = \sum_j g(j) e^{-ikj} \ ,$$

where the sum over the multi-dimensional (in the general case) Fourier index, or "momentum", $k = (k_1, k_2, \ldots, k_d)$ contains exactly L_μ points in each direction, $k_\mu = 2\pi n_\mu / L_\mu$, $n_\mu = 0, 1, \ldots, L_\mu - 1$. This discrete Fourier transform works because

$$N^{-1} \sum_j e^{ipj} e^{-ikj} \equiv \delta_{p,k} = \begin{cases} 1, & \text{if } p = k \\ 0, & \text{otherwise} \end{cases}$$

We have then (using translation invariance "backwards")

$$\begin{aligned} \tilde{g}(k) &= \sum_j g(j) e^{-ikj} = N^{-1} \sum_{i,j} \langle \sigma_i \cdot \sigma_j \rangle e^{-ik(j-i)} \\ &= N^{-1} \langle |\tilde{\sigma}(k)|^2 \rangle \ . \end{aligned} \tag{30}$$

The advantage of this relation is that $\tilde{g}(k)$ may be calculated using MC estimator for the Fourier coefficients of the spin distribution $\tilde{\sigma}(k)$. Thus in the simulation we collect statistics of $|\tilde{\sigma}(k)|^2$ by making Fourier transforms of σ_j, and after the calculation is finished we perform the inverse Fourier transform for $\tilde{g}(k)$.

One may immediately object that we really do not gain anything because in the measurement we will need to evaluate N Fourier coefficients and each of them is the sum over N spins. This is true provided we calculate $\tilde{\sigma}(k)$ naively by taking sum after sum over and over again. In fact, it is possible to get $\tilde{\sigma}(k)$ in just $N \log_2 N$ operations by the fast Fourier transform (FFT) technique, Gauss=genius (1805). Below I will assume that N is a power of 2, i.e. $N = 2^I$. If this is not the case, then simply add a certain number of zeros to the end of the σ_j file making it longer and matching the nearest power of 2. This is how it works in one dimension, Danielson and Lanczos (1942) [it can be generalized to higher dimensions and there are well developed codes in libraries to do it; FFT is not the MC technique so in this course I will be very brief]. First, split $\tilde{\sigma}(k)$ into even- and odd-sites contributions

$$\tilde{\sigma}(k = 2\pi n/N) = \sum_{j=1}^{N} \sigma(j) e^{i2\pi nj/N}$$

$$
\begin{aligned}
&= \sum_{j=1}^{N/2} \sigma(2j) e^{i2\pi n 2j/N} + \sum_{j=1}^{N/2} \sigma(2j-1) e^{i2\pi n(2j-1)/N} \\
&= \sum_{j=1}^{N/2} \sigma(2j) e^{i2\pi nj/(N/2)} + W^n \sum_{j=1}^{N/2} \sigma(2j-1) e^{i2\pi nj/(N/2)} \\
&\equiv \tilde{\sigma}^{(e)}(n) + W^n \tilde{\sigma}^{(o)}(n) \ , \quad W = e^{-i2\pi/N} \ . \qquad (31)
\end{aligned}
$$

Well, still this requires N operations to do, but we immediately observe that $\tilde{\sigma}^{(e)}(n)$ and $\tilde{\sigma}^{(o)}(n)$ are periodic with the period $N/2$, not N!, and thus we may calculate it only for half of the momentum points. The beauty of the Danielson and Lanczos lemma is that it can be taken recursively to the next level

$$
\begin{aligned}
\tilde{\sigma}^{(e)}(n) &= \tilde{\sigma}^{(ee)}(n) + W^n \tilde{\sigma}^{(eo)}(n) \ , \\
\tilde{\sigma}^{(o)}(n) &= \tilde{\sigma}^{(oe)}(n) + W^n \tilde{\sigma}^{(oo)}(n) \ ,
\end{aligned}
$$

and the next level functions, e.g. $\tilde{\sigma}^{(ee)}(n)$, are all periodic with the period $N/4$. It means that the second level functions should be calculated only for $N/4$ momentum points. If we continue further along the tree of iterations, at step I (remember that $N = 2^I$), we will have to take the one spin sums, i.e. for any I-long word consisting of "e" and "o"

$$
\tilde{\sigma}^{(I-long\ word)}(n) = \sigma(j) \ , \quad \text{for some } j
$$

If the eo-word is written in the binary representation with $e = 0$ and $o = 1$, then $j = (binaryword) + 1$. The total number of operations we have to perform to complete the transform is then, counting backwards from the I-th level and going up

N - at the lowest level we have N single values functions
N - one level up we have $N/2$ functions each having 2 values
N - one level up we have $N/4$ functions each having 4 values
.....
N - one function $\tilde{\sigma}(k)$ which has N points

Total=$N \log_2(N)$.

Well, may be a factor of two larger to explain the code which pairs to combine at each level. Still, the scaling is almost linear, and thus the measurement can be performed almost every autocorrelation time.

The other useful formula relates $g(j)$ and the susceptibility

$$\chi = \langle M^2 \rangle / TN = \langle \sum_i \sigma(i) \sum_j \sigma(j) \rangle / TN = \sum_j g(j)/T = \tilde{g}(k=0)/T \; .$$

This is one of the "faces" of the so-called fluctuation/dissipation theorem which relates linear response coefficients and correlation functions. In essence, system fluctuations away from the average position tell us how easy it would be to shift the average if we apply external forces.

Re-weighing (importance sampling), histogram, and multiple histogram data analysis.

The central idea of the Metropolis algorithm is to make sure that the scheme samples configurations in the relevant region and deviations from this region are naturally penalized by larger free energy barriers. If efficiency is not an issue, or overcoming large energy barriers becomes crucial, one may consider other schemes which allow configurations to "wonder" more freely away from the relevant region. The general theory of how this may be achieved is extremely simple.

Suppose, we decided to include configurations into the stochastic sum with probabilities proportional to some arbitrary function $P(\nu)$, and *not* proportional to $W(\nu)$. Now $M(\nu) = cP(\nu)$, and the question is now we should modify estimators in stochastic sums to ensure that our final results do not change and still converge to the exact answer. Let's seek the solution in the form

$$\langle A \rangle = \frac{\sum'_\nu A(\nu) X(\nu)}{\sum'_\nu X(\nu)} \; .$$

This is not just a "wild guess"—we know that if $P(\nu) = const$, i.e. all configurations are equally likely to appear in the sum, then $X(\nu) = W(\nu)$ reproduces the original expression we started with. Repeating the "infinite simulation time" proof we did for the Metropolis algorithm, we get

$$\frac{\sum'_\nu A(\nu) X(\nu)}{\sum'_\nu X(\nu)} \Longrightarrow \frac{\sum_\nu A(\nu) X(\nu) M(\nu)}{\sum_\nu X(\nu) M(\nu)} = \frac{\sum_\nu A(\nu) X(\nu) P(\nu)}{\sum_\nu X(\nu) P(\nu)} \; .$$

The solution of our problem is immediate, use

$$X(\nu) = W(\nu)/P(\nu) \; , \tag{32}$$

and this will guarantee that the answer remains unchanged.

Often, $P(\nu)$ is presented as $P(\nu) = W(\nu) \cdot Y(\nu)$ to underline the difference between configuration weight and the probability of the configuration to be included into the stochastic sum. To implement this change, all we need to do is to "re-weigh" the standard acceptance ratio $R \Longrightarrow R\,[Y(\nu')/Y(\nu)]$, as well as the estimators for the numerator and denominator, i.e. now

$$\langle\, A\rangle = \frac{\sum'_\nu A(\nu) Y^{-1}(\nu)}{\sum'_\nu Y^{-1}(\nu)}\ . \tag{33}$$

I will call this a "re-weighed simulation" or a simulation with importance smapling..

Histogram method takes advantage of this flexibility and uses it to calculate quantity A not just for one set of the system parameters used in the simulation, but for the immediate vicinity of this parameter set as well. In the example below, which is also the standard application of the histogram method, we will consider system temperature as a parameter. However, similar considerations apply to any other parameter in the Hamiltonian without limitations. This is how it works.

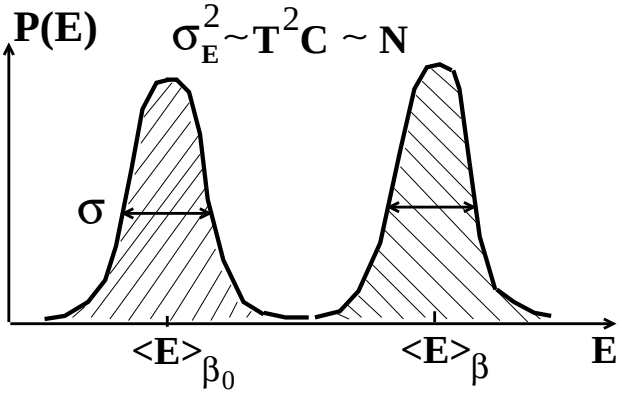
Let the simulation is performed at inverse temperature β_0 as usual with $M(\nu) = cW(\nu, \beta_0)$, or $Y(\nu, \beta_0) = 1$. The result is $A(\beta_0)$. Equivalently, this same simulation may be considered also as a simulation at some other temperature β and $Y(\nu, \beta) = W(\nu, \beta_0)/W(\nu, \beta)$.Thus the same set of configurations may be analyzed as if it was a re-weighted simulation for temperature β

$$A(\beta) = \frac{\sum'_\nu A(\nu) W(\nu, \beta)/W(\nu, \beta_0)}{\sum'_\nu W(\nu, \beta)/W(\nu, \beta_0)} = \frac{\sum'_\nu A(\nu) e^{-E(\nu)(\beta-\beta_0)}}{\sum'_\nu e^{-E(\nu)(\beta-\beta_0)}}\ . \tag{34}$$

This is a wonderful tool because for the CPU time of one simulation we may get many points in temperature at once! The only downside here is that we either have to store all $A(\nu)$ and $E(\nu)$ values during the simulation process (well, one per autocorrelation time), or, decide ahead of time at what points exactly around β_0 we would like to know $A(\beta)$. The last method is more economic and is easy to implement when we update counters:
instead of Z=Z+1; A=A+$A_{estimator}$
we use an array of, say K, temperature points $\{\beta(j)\}$ to do update counters arrays

DO j=1,K
Y=exp($E_{estimator}$*(β_0 - β (j)))
Z(j)=Z(j)+Y
A(j)=A(j)+ $A_{estimator}$*Y
ENDDO

We should recognize, however, that histogram re-weighting has its limits and will not work if β and β_0 differ significantly. From Eq. (29) we know that energy fluctuations in a given simulation are controlled by the specific heat. Since $C \sim N$ and $\langle E \rangle \sim N$ the energy distribution is very narrow in relative units.



It is given by the Gaussian $\exp\{-(E - \langle E \rangle)^2 \beta^2 / 2C\}$. For $|E - \langle E \rangle|\beta \gg \sqrt{C}$ the MC statistics will be very poor, or none at all. Correspondingly, doing simulation at β_0 there is no way of knowing what system properties are at temperature β if the corresponding energy distributions $P(E, \beta_0)$ and $P(E, \beta)$ do not overlap.

The shift of $\langle E \rangle$ with temperature is given by $C(T - T_0)$, and it gets comparable to the distribution dispersion when

$$C(T - T_0) \sim T\sqrt{C}\ , \quad \text{or} \quad \frac{|T - T_0|}{T} \sim 1/\sqrt{C} \sim 1/\sqrt{N}\ . \tag{35}$$

This Eq. sets the range of temperatures which can be addressed by the histogram method. For large N the method is not really giving us extra information for a wide range of parameters. Still, some techniques for calculating critical temperatures and critical indices work best if we can fix temperature with accuracy up to seven meaningful digits! Using histogram method we should not worry about this ahead of time (I mean before we know what T_c is) because we can always re-weight our simulation to the required temperature value in the final analysis.

Suppose now that we perform a series of simulations at temperatures T_i which are sufficiently close to each other, i.e. $T_{i+1} - T_i$ are within the range given by Eq. (35). Of course, we can re-weight each simulation separately

and then combine results. This is not the optimal procedure, though, since different energy distributions overlap, and the accuracy may be improved if we know how to utilize data from all simulations at once. The **multiple histogram** method allows us to achieve this goal.

Before we discuss how it works ...
Lemma: Suppose we have a collection of independent statistical variables x_i with distributions characterized by exactly same average $\langle\, x_i\rangle = \langle\, x\rangle$ but different dispersions σ_i. What we are modeling here is the situation when the same quantity is calculated using MC methods many times with calculations being completely independent from each other and having different errorbars. The problem is in combining results in the most optimal way, so that the final error bar for the variable

$$\frac{x_{all} = \sum_i x_i w_i}{\sum_i w_i \ ,} \tag{36}$$

is as small as possible. The relation between x_{all} and x_i must be linear in order to satisfy the requirement that $\langle\, x_{all}\rangle = \langle\, x\rangle$, and be able to to reduce to the obvious $x_{all} = x_i$ when all variables except some x_i have very large error bars (and thus are better not to be considered at all!). The solution is

$$w_i = 1/\sigma_i^2 \ . \tag{37}$$

Proof: For the sum of independent variables $x_i w_i / \sum_j w_j$ we have

$$\sigma_{all}^2 = \frac{\sum_i \sigma_i^2 w_i^2}{\left(\sum_i w_i\right)^2} \ .$$

To minimize σ_{all} with respect to w_i we require that all derivatives $\partial\sigma_{all}^2/\partial w_i$ are zero

$$\partial\sigma_{all}^2/\partial w_i = 2\frac{w_i\sigma_i^2}{\left(\sum_j w_j\right)^2} - 2\frac{\sum_j \sigma_j^2 w_j^2}{\left(\sum_j w_j\right)^3} = 0 \ ,$$

or

$$w_i = \frac{1}{\sigma_i^2}\,\frac{\sum_j \sigma_j^2 w_j^2}{\sum_j w_j} \ = \frac{const}{\sigma_i^2} \ . \tag{38}$$

Since the value of *const* is arbitrary here [it cancels in Eq. (36)] we simply choose Eq. (37) as the correct solution. Now back to the multiple histogram method.

Let us introduce the notion of the density of states

$$\rho(E) = \sum_{\nu} \delta_{E,E(\nu)} = e^{S(E)} \ , \tag{39}$$

which counts how many states have energy E. The log of this quantity is also known as a microcanonical entropy. We may use it to write the partition function and average energy as one dimensional energy sums or integrals:

$$Z = \sum_{E} \rho(E) e^{-\beta E} \ , \tag{40}$$

$$\langle\, E\rangle = Z^{-1} \sum_{E} E\rho(E) e^{-\beta E} \ , \tag{41}$$

If we know $\rho(E)$, which is by definition temperature independent, we may use it to calculate many thermodynamic properties at any temperature we want. The central piece of the multiple histogram method is to find the most optimal way of obtaining $\rho(E)$ from a series of simulations performed at a fixed set of $\beta(j)$.

The probability of having a configuration with energy E is given by

$$p_\beta(E) = \rho(E) \frac{e^{-\beta E}}{Z} \ .$$

On another hand, in a given simulation performed this probability may be estimated as the fraction of time the measured configuration energy was E

$$p_\beta(E) = H_\beta(E)/H_\beta \ .$$

Here H is the total number of measurements, and $H(E)$ is the number of measurements with $E(\nu) = E$, collected, e.g., as a histogram —this is where the name of the method originates. From the last two Eqs. we get as estimate for the density of states as

$$\rho(E) = \frac{H_\beta(E)}{H_\beta} \, Z(\beta) e^{\beta E} \ . \tag{42}$$

In a given simulation performed at temperature β_i we will get some value for the density of states as

$$\rho_i(E) = \frac{H_i(E)}{H_i} \, Z_i e^{\beta_i E} \ . \tag{43}$$

which is fluctuating around the exact T-independent value $\rho(E)$.

To use the Lemma proved above and to combine results from different simulations we need to know errorbars on $\rho_i(E)$. If the simulation at temperature T_i is repeated infinitely many times (with the same number of measurements H_i in each simulation) and we average all of them then we would get the exact answer

$$\rho(E) = \langle\, H_i(E) \rangle\, \frac{Z_i e^{\beta_i E}}{H_i}\ .$$

Note, that the l.h.s. does *not* depend on index "i". The number of measurements in the histogram bin $H_i(E)$ fluctuates from simulation to simulation around its average $p_i(E)H_i$ and is subject to the standard Poisson statistics. It means that the errorbar on the measured quantity, Eq. (43) is

$$\sigma_i^2(E) = p_i(E)H_i\, \left(\frac{Z_i e^{\beta_i E}}{H_i}\right)^2$$

Substituting here the initial relation between $p_\beta(E)$ and $\rho(E)$ (see above), we get

$$\sigma_i^2(E) = \rho(E)\, \frac{Z_i e^{\beta_i E}}{H_i}\ .$$

Now we may apply our lemma to write the optimal estimate for the density of states

$$\rho_{all}(E) = \frac{\sum_i \rho_i(E)/\sigma_i^2}{\sum_i 1/\sigma_i^2} = \frac{\sum_i H_i(E)}{\sum_i H_i\, (e^{-\beta_i E}/Z_i)}\ . \tag{44}$$

Nice, but we do not know what exact partition functions Z_i are! Fortunately, $\rho_{all}(E)$ can be used to calculate Z_i, see Eq. (40), and we can make everything a selfconsistent loop of relations:

$$Z_i = \sum_E \rho_{all}(E) e^{-\beta_i E} = \sum_E \frac{\sum_j H_j(E)}{\sum_j H_j\, (e^{(\beta_i - \beta_j)E}/Z_j)}\ . \tag{45}$$

This equation can be solved by iterations: guess something about $\{Z_j\}$, substitute the guess to the r.h.s. of Eq. (45) and get new values of $\{Z_j\}$, substitute ... , if the result does not change after being iterated then it is the solution.

Several remarks are due on the practical implementation of the method. One is the overflow problem.

(i) Partition function is a macroscopic exponential and may easily exceed the largest number permitted by the computer, e.g. 10^{300}. To deal with this problem we may work instead with $z_i = \ln Z_i$. Products and ratios of capital "Z"s are sums and differences of "z"s—easy. The only thing to remember in programming *everything* in terms of logarithms is how to handle sums. If $Z_1 > Z_2$ then

$$\ln(Z_1 + Z_2) = \ln Z_1(1 + Z_2/Z_1) = \ln Z_1 + \ln(1 + Z_2/Z_1)$$

and, further on,

$$\ln(Z_1 + Z_2) = \ln Z_1 + \ln\left(1 + e^{\ln Z_2 - \ln Z_1}\right) .$$

Since $Z_1 > Z_2$, we always have $\ln Z_2 - \ln Z_1 < 0$ and the last exponent does not cause overflow [we may even replace it with zero if $\ln Z_2 - \ln Z_1 < -100$ because computer roundoff errors are larger anyway. This procedure works for any sum of positive definite numbers; just single out the contribution with the largest logarithm. Let it be Z_i:

$$\ln(\sum_j Z_j) = \ln Z_i + \ln\left(\sum_j e^{\ln Z_j - \ln Z_i}\right) .$$

Not a big deal to program ...

(ii) The density of states may be considered as an auxiliary property used to relate different quantities to each other. In the realistic simulation we may "skip" calculating it and go directly to the answer. Sometimes this is even a better way of doing things because it allows to eliminate systematic errors introduced by grouping configuration energies into finite-size bins. Using definitions of H_i and $H_i(E)$ we rewrite identically

$$\begin{aligned} Z_k &= \sum_E \frac{\sum_i H_i(E)}{\sum_j H_j\, e^{(\beta_k-\beta_j)E}/Z_j} = \sum_{E,i} H_i(E) \frac{1}{\sum_j (H_j/Z_j)\, e^{(\beta_k-\beta_j)E}} \\ &\longrightarrow \sum_{i,\nu_i} \frac{1}{\sum_j (H_j/Z_j)\, e^{(\beta_k-\beta_j)E_{\nu_i}}} . \end{aligned} \tag{46}$$

In this implementation we do not have to keep the histograms, but we have to keep the energy records for all runs. The advantage is in avoiding **systematic, programmer-induced** errors in the final answer. For example, if the configuration space is continuous then collecting energy values into discrete

bins is an approximate procedure. Of course, the bin size may be reduced to such a small value that most bins will remain "empty" by the end of the simulation, and only a small fraction of bins will have $H_i(E) = 1$— this is exactly what Eq. (46) describes.

The multiple histogram method may be used to calculate any quantity A over a broad range of temperatures. After solving for $\{Z_j\}$ iteratively, one calculates

$$Z_\beta = \sum_{i,\nu_i} \frac{1}{\sum_j (H_j/Z_j)\, e^{(\beta-\beta_j)E_{\nu_i}}} \ , \tag{47}$$

for any other value of T, and, keeping in memory all records $\{A_{\nu_i}\}$ finds

$$\langle\, A\rangle_\beta = Z_\beta^{-1} \sum_{i,\nu_i} \frac{A_{\nu_i}}{\sum_j (H_j/Z_j)\, e^{(\beta-\beta_j)E_{\nu_i}}} \ . \tag{48}$$

Just a reminder: the multiple-histogram method is a generalization of the simple re-weighting technique for the case when many points can be re-weighted in temperature and their combined results are summed in the most optimal way.

Sign Problem.

Let's go back to the first formula in this Section and formally consider W_ν functions which are *not* positive-definite (for whatever reason; in the quantum mechnics Section we will see how sign-alternating W_ν happen naturally). We may separate the sign of W_ν from W_ν explicitly and write

$$\langle A\rangle = \frac{\sum_\nu A_\nu\, Sign_\nu\, W_\nu}{\sum_\nu Sign_\nu\, W_\nu} \ ,$$

where W_ν is now positive definite by definition. This allows us to proceed with the standard MC simulation which replaces full sums with stochastic sums and generates configurations according to their positive definite weights:

$$\langle A\rangle = \frac{\sum'_\nu A_\nu\, Sign_\nu}{\sum'_\nu Sign_\nu} \ = \ \frac{\sum'_\nu A_\nu\, Sign_\nu}{\sum'_\nu} \ \frac{\sum'_\nu}{\sum'_\nu Sign_\nu} \ = \ \frac{\langle ASign\rangle}{\langle Sign\rangle} \ .$$

The trouble comes in cases with $\langle Sign\rangle \to 0$ in the large system size limit (typicaly $\langle Sign\rangle$ approaches zero *exponentially fast*!!!). Finite $\langle A\rangle$ is obtained from the ratio of two quantities both converging to zero. Since both the

numerator and denominator have finite MC errorbars we have no idea what the correct answer is; we simply look at the ratio of errorbars which fluctuates wildly. Indeed (assuming for a moment that averages have small errorbars)

$$\langle A\rangle + \delta_A = \frac{\langle ASign\rangle + \delta_{AS}}{\langle Sign\rangle + \delta_S} \approx \frac{\langle ASign\rangle}{\langle Sign\rangle}\left(1 + \frac{\delta_{AS}}{\langle ASign\rangle} + \frac{\delta_S}{\langle Sign\rangle}\right)$$

or,

$$\frac{\delta_A}{\langle A\rangle} \approx \frac{\delta_{AS}}{\langle ASign\rangle \to 0} + \frac{\delta_S}{\langle Sign\rangle \to 0} \ .$$

There is no generic solution to the sign-problem. It prevents MC methods from studies of such interesting systems as interacting fermions, frustrated bosonis and magnetic systems, real time dynamics, etc. It does not mean, however, that if you see the formulation of the problem which has a sign, you have to ignore it right away. It may happen that sign-problem exists in one formulation and is absent in another! Then the sign-free formulation may be used for the MC simulation. There are many examples of this kind, though they are always case specific. To illustrate the point consider arbitrary quantum mechanical system and write the partition function in the Hamiltonian *eigenfunction* representation, $H\Psi_\alpha = E_\alpha \Psi_\alpha$:

$$Z = \sum_\alpha e^{-E_\alpha/T} \ .$$

This expression contains only positive definite terms! But there is no way to know eigenfunctions and eigenvalues for the complex many-body system in the first place(!), and thus this formal setup is not suitable for the simulation. Still, the lesson we learn is that the sign-problem may be reduced or completely eliminated by the proper choice of the basis set. Unfortunately, our choices are quite limited and there is no generic prescription of how to get the basis set leading to the sign-free formulation of the problem.

Any solution of the sign-problem for specific models is a breakthrough if it allows to calculate answers in time which scales as some power of the system size.