Equilibration time and Correlation time

In order to make use of the Markov chain Monte Carlo, one starts with some initial condition, typically a "cold start" (T=0, all spins aligned) or a "hot start" (T=∞, spins at random), then must run the Markov chain for "a while" until the system reaches equilibrium at the desired temperature T: this time is known as equilibration time.

How do we know this equilibration time? Typically this is determined empirically, by monitoring how statistical properties, e.g. the magnetization $\langle S_z \rangle$, depend on time along the chain: it goes from the value characteristic of the initial condition to something that fluctuates around a definite value, with transient behavior in between.

The danger of identifying the equilibration time this way is that we may fall into a metastable state, where statistical values seem to fluctuate around what seems equilibrium, but in reality after a while things will start flowing again. In order to avoid this, it is a good idea to try this method a couple of times with different initial conditions, e.g., $T=0$, $T=∞$, or different random seeds when starting at $T=∞$, and see how equilibration time depends on the initial condition. One might get some sort of instability.

From looking at these different paths the equilibrium converge to the same asymptotic magnetization, one can be reasonably safe to...
Conclude some equilibration time \( T_{eq} \). The natural time scale to use here is to measure times in units of Monte Carlo steps per spin, i.e. the relevant "time" variable is:

\[
\tau = \frac{n}{N^2}
\]

where \( n \) is total number of Monte Carlo steps, and \( N^2 \) is the number of spins (in 2D grid of size \( N \times N \)). In this way, it makes sense to compare different grid sizes.

Once in equilibrium, our Markov chain will give consecutive states in equilibrium, but correlated. Since the new state is obtained from the previous one by just a single spin flip, most of the configuration of the spins are the same, thus it makes no sense to use consecutive near state of our Markov chain to estimate expectation values: If in one state \( x_1 \) our variable of interest \( X \) fluctuates up compared to its mean (unknown) \( \langle X \rangle \), then in \( x_2 \) \( X \) will also be very likely large than \( \langle X \rangle \), thus averaging over nearby states in the chain will lead to biased estimate of thermal expectation value. What we need is a measure of the correlation time, that is, the characteristic time after which two states in the Markov chain can be considered independent. There are many ways of estimating this, a simple one is using the correlation function of the magnetization, time-displaced:

\[
\gamma_c(t) = \int dt' \left[ m(t) - \langle m \rangle \right] \left[ m(t+t') - \langle m \rangle \right] = \int dt' \left[ \langle m(t)m(t+t') \rangle - \langle m \rangle^2 \right]
\]

What we mean by this is, discretizing

\[
\gamma_c(n) = \frac{1}{N_{c-m}} \sum_{i=0}^{N_{c-m}} m(i) m(i+m) - \left( \frac{1}{N_{c-m}} \sum_{i=0}^{N_{c-m}} m(i) \right)^2 \left( \frac{1}{N_{c-m}} \sum_{i=0}^{N_{c-m}} m(i+m) \right)
\]

Note the last term, we have two factors that in ideal case will approximate \( \langle m \rangle \) both, but this way of approximating \( \langle m \rangle \)
makes $\chi(n)$ a little bit better behaved, since we use exactly the same starting samples as in first term. Note that as $n \to \infty$, where $N$ is the number of realizations in the chain starting with $i=0$, (which corresponds to time after equilibration time!) which is when we start "taking data."

Clearly, if we take $n$ small (where $n=1$ corresponds to calculating correlation between one MC step and the next), $\chi(n)$ is going to be pretty large, as dependencies on each realization in the chain are highly correlated because they differ in only a few spin flips. What happens typically is that for long times, we have

$$\frac{\chi(n)}{\chi(0)} \sim e^{-t/\tau}$$

where $\tau$ is the correlation time we are after. [In fact there are as many correlation times as states in the system, so that for large $n$ we are only concerned about the largest of them, which we call $\tau$, all others have died out when $t=\tau$.]

The correlation time depends on temperature, in fact for the Metropolis algorithm such dependence is quite strong. What happens near the critical temperature is that in order to get an uncorrelated state from a previous one, it is necessary to generate a realization where clusters of all sizes up to $\xi$ (and $\xi \to \infty$ at $T=\tilde{T}_c$) are at different locations in the grid. Now, this requires flipping large clusters, but in the Metropolis algorithm this is painfully slow, because we are flipping one spin at a time. Indeed at

$$T=\tilde{T}_c = \frac{2J}{\ln(4+\sqrt{2})} = 2.269 J$$

The cost of flipping a spin surrounded by 4 spins pointing in some direction is $8J$, then probability of accepting such a move is
\[ A(\mu; v) = e^{-8.51 T_c} = 0.0294 \]

i.e. about 3%. Thus, even though flipping an entire cluster would only cost energy proportional to its perimeter, at worst, flipping by single spins fails forever, with most flipflops happening only at the boundary of a cluster (flipping inside, if accepted, get very quickly flipped back in a later move since it is so energetically favourable to flip back). Therefore in Metropolis algorithm a new uncorrelated realization is generated by boundaries of clusters moving until they occupy approximately new random locations in grid space. If one plots the connected time-correlation function, it looks something like this at different temperatures.

\[ \frac{X(t)}{X(0)} = \begin{cases} 1 & T < T_c \\ \frac{1}{t} & T > T_c \end{cases} \]

This increase of the correlation time as \(T \to T_c\) is known as critical slowing down. Empirically it is found that the dependence of the correlation time on temperature can be written as a relation to the correlation length (it must be since \(\xi\) is the only scale in the problem at critical temperature, assuming infinite grid!)

\[ T \sim \xi^z \]

where \(z\) is often known as the dynamical exponent. For the Metropolis algorithm \(z = 2.17\) (5/2D). Now, in a finite grid \(\xi\) can never be larger than \(L\) (box side), this means that in fact one has something like

\[ T \sim \min(911)^{z} \]
Near the critical temperature, we have $T \propto L^2$, and the CPU time taken to perform a certain number of Monte Carlo steps per site increases like the number of states $\sim L^d$ where $d$ is dimensionality of the system. Thus the time to generate a new statistically independent realization scales as

$$T_{CPU} \sim L^{d+2}$$

near $T=T_c$, e.g. for 2D Ising $T_{CPU} \sim L^4$. This is quite a serious problem. Clearly, as we discussed above, one could reduce $T$ significantly if the algorithm allows for flipping entire clusters of spins at once, rather than proceeding by flipping a single spin each Monte Carlo step. Such algorithms are known as cluster-flipping algorithms. The two most popular are those of Swendsen and Wang (1987) and Wolff (1989).

**Wolff algorithm**

<table>
<thead>
<tr>
<th>$k$</th>
<th>$Z(\text{Metropolis})$</th>
<th>$Z(\text{Wolff})$</th>
<th>$Z(\text{SW})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.147 ± 0.01</td>
<td>0.25 ± 0.01</td>
<td>0.25 ± 0.01</td>
</tr>
<tr>
<td>2</td>
<td>2.02 ± 0.02</td>
<td>0.33 ± 0.01</td>
<td>0.54 ± 0.02</td>
</tr>
<tr>
<td>3</td>
<td>2.02 ± 0.02</td>
<td>0.33 ± 0.01</td>
<td>0.54 ± 0.02</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
<td>0.25 ± 0.01</td>
<td>0.86 ± 0.02</td>
</tr>
</tbody>
</table>

We will just discuss the Wolff algorithm, which is somewhat easier to code.

The idea is to start from a "seed spin" chosen at random, and then "grow" a cluster, by adding to all neighbor spins which point in the same direction as the seed spin, with probability $P_{add}$. Once we have checked all neighbors we move into the new spins of the cluster and proceed the same way (if a neighbor is already in cluster, we don't change anything, i.e. it is not we add them with probability $P_{add}$ or keep as it points in same direction. If a spin was considered before and rejected it gets another chance to join the cluster). Once we are done growing the cluster, all spins are flipped.

The question is what is, given $P_{add}$, the best choice of acceptance ratio to make the algorithm satisfy detailed balance, and what $P_{add}$ should be in order for the average acceptance ratio as close to 1 as possible.
Consider the states $\mu$ and $\nu$ above, before and after flipping. Notice that in each of these states, there are spins outside the cluster that point in the same direction as the cluster; these "bonds" are broken as we flip the cluster.

Consider one move that takes us from $\mu$ to $\nu$, and consider also the reverse move that takes us from $\nu$ to $\mu$. (Starting with some seed spin and adding the others to the cluster in the same order as the forward move.) The probability of choosing the seed is exactly the same in the two directions, as is the probability of adding each spin to the cluster. The only thing that changes between forward and reverse move is the number of bonds that need to be broken in each case.

Let $m$ be the number of bonds that need to be broken in the forward move, and in the number in the inverse move. Broken bonds are given by spins that point in some direction were not added to the algorithm to the growing cluster. Since the probability of not adding is $1-\text{Padd}$, we have that $g(\mu \rightarrow \nu) \propto (1-\text{Padd})^m$ and $g(\nu \rightarrow \mu) \propto (1-\text{Padd})^m$, then the condition of detailed balance reads

$$\frac{g(\mu \rightarrow \nu) A(\mu \rightarrow \nu)}{g(\nu \rightarrow \mu) A(\nu \rightarrow \mu)} = (1-\text{Padd})^{m-m} \quad \frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = e^{-\beta (E_\nu - E_\mu)} = e^{-\beta 2J(m-m)}$$

$$\Rightarrow \quad \frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = \left[ e^{2\beta J (1-\text{Padd})} \right]^{m-m}$$
This is the condition on the acceptance ratios. Now notice that if one chooses

$$P_{\text{add}} = 1 - e^{-2\beta J}$$

the acceptance ratios can be taken as unity, both forward and backward, which is the best value they can take. Therefore, every move of the chain is accepted and we still satisfy detailed balance. Therefore, the Wolff algorithm can be summarized as follows,

1) Choose a seed spin at random from the lattice

2) Look at neighbors; if they point in the same direction, add them to the cluster with probability

$$P_{\text{add}} = 1 - e^{-2\beta J}$$

3) For each spin added to cluster, examine its neighbors and repeat procedure in 2) until one has considered all neighbors of cluster for inclusion.

Note: if neighbors are already in cluster, we don't do anything. If they are not, they are added with probability $P_{\text{add}}$ even if they were considered before (and fail to add) by virtue of being neighbor of another cluster spin (i.e., they got a second chance to be added).

Note that $P_{\text{add}}$ decreases with increasing temperature, so at high $T$, one has very small clusters, as it makes sense since uncorrelated states, differ by flipping only small clusters ($\phi$ is very small). At low $T$, the size of clusters is large, and one flips a large number of spins.

In these limits the Wolff algorithm is a bit slower than Metropolis (for some correlation time) due to overhead in the complexity of the procedure. However, when $T \approx T_c$, the Wolff algorithm produces uncorrelated states at a much faster rate, e.g., for 100x100 lattice the ratio of Wolff is of order a thousand.
Note, however, that at \( T \approx T_c \) the fluctuations become large due to phase transitions (there is nothing we can do about this, it's physics!), and so we would need a lot of independent realizations in order to measure thermal averages accurately enough. It is very hard due to the nature of phase transitions to infer to or critical exponents with any accuracy by measuring things close to the critical point. In order to overcome this problem, we turn to the R6, which maps a system at \( T \approx T_c \) with \( \gamma \rightarrow \infty \) to another one at \( T' \) with smaller \( \gamma \) (in units of lattice spacing).

**Monte Carlo R6**

Let us explain the method by the following procedure (should sound familiar given what we discussed about R6 in previous classes):

1) Perform a MC simulation (using your favorite algorithm) at temperature \( T \) in grid of size \( L \times L \) (say in 2D). Calculate the internal energy per spin \( u \) by averaging over grid.

2) Take the spins of the MC simulation and define a spin blocking (coarse graining procedure) to obtain new spins \( s' \) at \( L_2 \times L_2 \) lattice, separated by \( b=2 \). You can for example use majority rule to define \( s' \) (with a tie breaker if \( S_i = 0 \) by tossing a random number). Calculate the renormalized internal energy per spin \( u' \).

3) Vary the temperature \( T \) until you find the point at which \( u' = u \). This is the fixed point, and gives the critical temperature.
4) To find critical exponent $v$, find $T'$ by using that

$$U'(T) = U(T') \Rightarrow T' = U^{-1}[U'(T)] \quad \text{(EVT for Trasversal)}$$

From the curve $T'(T)$, find the exponent $v$ from usual linearization:

$$v = \frac{\log b}{\log \frac{dT'}{dT}|_{T_c}}$$