

5.7. The Markov Chain Monte Carlo (MCMC) method

[Thijssen: Computational physics p. 275- and others]

There exists a completely different approach to doing importance sampling of a distribution than the one presented earlier.

In the importance sampling and combined analytical-rejection random number generation methods presented earlier in sections 4 and 5, points were generated in a distribution using some schemes to select more points in the regions where the function is the strongest.

- But in both methods each point generated was completely independent of the previous ones
 - I.e. the points were in random order, **uncorrelated**.

The Markov chain Monte Carlo (MCMC) method takes the opposite extreme approach of generating points such that each point is directly dependent on the previous one.

- Thus of course the points are correlated with each other

- But they are generated in such a way that they lead to a desired distribution when they are ran long enough

5.7.1. Markov chains and ergodicity

Consider a system which evolves from one state to the next

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_3 \cdots \quad (1)$$

where \mathbf{x} denotes the momentaneous state of the system.

The state space \mathbf{x} may be any discrete or continuous space.

A Markov chain is a sequence of states \mathbf{x}_i , where i is an integer, which has the property that the probability $P(\mathbf{x}_s)$ to reach a certain state x_s is dependent on the immediately previous state of the system, but no other states. That is,

$$P(\mathbf{x}_s) = T(\mathbf{x}_{s-1} \rightarrow \mathbf{x}_s) \quad (2)$$

where T is the transition probability.

- By contrast, in an uncorrelated system $P(\mathbf{x}_s) = \text{constant}$.

Now we are interested in the concept of generating a distribution $f(\mathbf{x})$ using a Markov chain MC simulation. It is obvious that in order for it to be possible to generate a distribution using a Markov chain, every possible \mathbf{x} value where the distribution is accessible needs to be reachable by the chain in a finite number of steps, regardless of the starting point of the chain. This property is called **ergodicity**.

5.7.1.1. Example

The above was a bit mathematical. Here is an example of what it all can be in practice.

- $f(\mathbf{x})$ may simply be a one-dimensional function $f(x)$ over real numbers x
- The transition step may be $P(-1, 1)\Delta x_{\max}$ where $P(-1, 1)$ is a uniform random number between -1 and 1.

This system is obviously ergodic if $f(x) > 0$ everywhere and $\Delta x_{\max} > 0$.

But consider the following function:

$$f(x) = \begin{cases} (x - 1)e^{-x}, & x > 1 \\ 0, & -1 < x < 1 \\ (1 - x)e^x, & x < -1 \end{cases} \quad (3)$$

and a Markov chain where the probability to reach a point is $\leq f(x)$.

Then if $\Delta x_{\max} < 2$ this system is obviously non-ergodic!

5.7.2. The MCMC algorithm

Let us now for the remainder of this section assume we have an ergodic system.

We want to generate points in a distribution $f(\mathbf{x})$ whose probability density $\rho(\mathbf{x})$ is known, i.e.

$$f(\mathbf{x}) \propto \rho(\mathbf{x}) \quad \text{for all } \mathbf{x} . \quad (4)$$

- ρ does not need to be normalized, and we do not need to know the proportionality constant

For concreteness we shall assume that \mathbf{x} is a vector in cartesian space of some dimensionality M , and the transition is a displacement of this vector.

- But the algorithm can in a rather obvious way be generalized to deal with an arbitrary space and an arbitrary kind of displacement

The Metropolis version of the MCMC method to generate points with the probability distribution ρ is as follows.

- 0 a° Select an initial configuration \mathbf{x} for which $\rho(\mathbf{x}) > 0$
- 0 b° Choose a maximum displacement value Δx_{\max}
- 0 c° Calculate the initial $\rho = \rho(\mathbf{x})$
- 1° Store the current state as $\mathbf{x}_0 = \mathbf{x}$
- 2° Generate a random number vector $\mathbf{u} = (u_1, u_2, \dots, u_M)$ where each u is a uniform random number **between -1 and 1**
- 3° Generate a new trial state $\mathbf{x}' = \mathbf{x} + \mathbf{u}\Delta x_{\max}$
- 4° Calculate the value of the function in the trial state $\rho' = \rho(\mathbf{x}')$
- 5° Choose whether to move to the new state as follows:
 - 6° If $\rho' \geq \rho$ accept the state
 - 7° If $\rho' < \rho$: accept the state only if $u < \frac{\rho'}{\rho}$ where u is a random number $\in [0, 1]$
 - 8 a° If the state is accepted, go to the new state: $\mathbf{x} = \mathbf{x}'$
 - 8 a° If the state is rejected, return to the previous state: $\mathbf{x} = \mathbf{x}_0$
- 9° Calculate the new $\rho = \rho(\mathbf{x})$
- 10° Do statistics of the current value of \mathbf{x} or properties dependent on it
- 11° Return to step 1

- Note that it is extremely important that steps 9 and 10 are carried out regardless of whether the state is accepted or not!

-
- It can be proven rigorously that this works, but we will not do it

- On the course “MC simulations in physics” this will be done.

- But one can intuitively understand fairly well why this works:

- Imagine being in a minimum of $\rho(\mathbf{x})$ at some \mathbf{x}_{\min} . Then the new trial state ρ' is guaranteed to be larger than ρ , and it will always be accepted.

- Imagine being next to a minimum of $\rho(\mathbf{x})$ at some $\mathbf{x}_{\text{larger}}$. Then if the trial state ρ' would happen to go to the minimum \mathbf{x}_{\min} , $\rho'/\rho < 1$ and we have a smaller probability of accepting the move.

- Combining the above two examples, we see that the system is more likely to go to, or remain in, states with larger ρ values !

One thing is very clear here: the sequence of states $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ are highly correlated! Thus for a low statistics this is probably a very poor way of generating points in a distribution

- The proof of the algorithm actually only shows that for an infinite number of steps the correct distribution is reached, if the system is ergodic.
 - But the proof does not tell anything about how large a number of steps is enough in practice!

Another thing of interest is that the algorithm only deals with the ratio of ρ'/ρ (steps 6° and 7° can be rewritten as comparing ρ'/ρ with 1).

- Thus clearly the algorithm is independent of the proportionality constant between $f(\mathbf{x})$ and $\rho(\mathbf{x})$
 - It works even if the proportionality constant is not known!
 - On the other hand it thus can not be used to integrate the function $f(\mathbf{x})$

The algorithm can be used to generate random numbers distributed as some arbitrary distribution $f(\mathbf{x})$ (compare with section 4.6).

But it is even more useful because it can be used to integrate properties which depend on the probability distribution of the points \mathbf{x} .

The simplest example is just calculating the average of points x distributed according to $f(x)$. This is normally achieved using

$$\langle x \rangle = \frac{\int x f(x) dx}{\int f(x) dx} \quad (5)$$

However, in the MCMC method the x points are distributed directly as $f(x)$. This means that we can achieve the calculation of the average simply by taking the average over all x points generated during the MCMC run! ¹

5.7.3. Other versions of the algorithm

The Metropolis version of the MCMC method is the original algorithm used. However, it is by no means the only one.

Let us introduce the concept of an *acceptance criterion* $A_{\mathbf{x}\mathbf{x}'}$, which gives the probability that a

¹This is of course also true for any other method which generates points x distributed as $f(x)$.

transition $\mathbf{x} \rightarrow \mathbf{x}'$ is accepted. In the Metropolis version presented above this criterion is

$$A_{\mathbf{x}\mathbf{x}'} = \min \left(1, \frac{\rho(\mathbf{x}')}{\rho(\mathbf{x})} \right) \quad (6)$$

In fact one can show that any acceptance criterion which fulfills the so called **detailed balance** condition

$$\frac{A_{\mathbf{x}\mathbf{x}'}}{A_{\mathbf{x}'\mathbf{x}}} = \frac{\rho(\mathbf{x}')}{\rho(\mathbf{x})} \quad (7)$$

leads to the desired result.

It is easy to show that the Metropolis equation fulfills this.

It is clear that the Metropolis scheme is not the only possible way of fulfilling detailed balance. In fact there is a wide range of other criteria used – but none has proved to be overall superior to the original one.

Also the transition type can of course be much different from the one presented above. If we denote a generalized transition type from state \mathbf{x} to state \mathbf{x}' with $T_{\mathbf{x}\mathbf{x}'}$, then the so called

Metropolis-Hastings method says that the transition probability is

$$A_{\mathbf{x}\mathbf{x}'} = \min \left(1, \frac{\rho(\mathbf{x}')T_{\mathbf{x}\mathbf{x}'}}{\rho(\mathbf{x})T_{\mathbf{x}'\mathbf{x}}} \right) \quad (8)$$

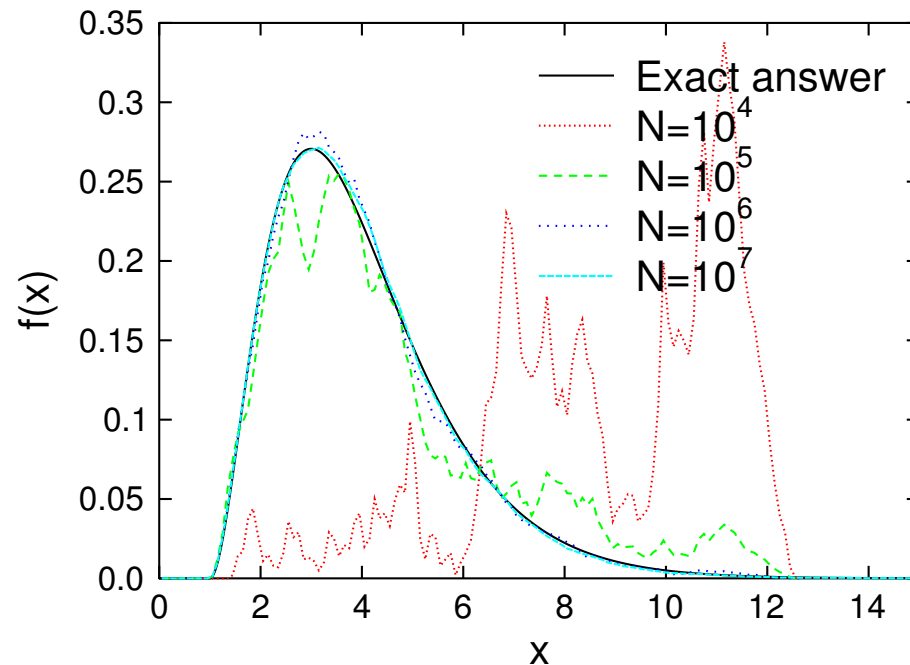
- In the original formulation the transitions are symmetric and hence $T_{\mathbf{x}\mathbf{x}'}/T_{\mathbf{x}'\mathbf{x}} = 1$ and we get back the original algorithm.

5.7.4. Example

As an example, I wrote a code which generates points with the MCMC method in the one-dimensional distribution

$$f(x) = \begin{cases} (x-1)^2 e^{-(x-1)}, & x > 1 \\ 0 & x \leq 1 \end{cases} \quad (9)$$

and runs it for different numbers of MC steps N . Here is an illustration of the result:



The figure shows clearly that for small numbers of steps, $N < 10^5$, the method gives absolutely horrible results. But for $N > 10^6$ it gives a result practically indistinguishable from the exact result.

The code also calculated $\langle x \rangle$, for which the analytical answer is exactly 4. Here are the results for $\langle x \rangle$ as a function of N :

N	$\langle x \rangle$
10^4	8.812
10^5	4.542
10^6	3.983
10^7	3.963
10^8	3.997
Exact	4.000

It is also important to realize that one cannot calculate the uncertainty of the results of the MCMC method in the same manner as for ordinary MC integration. The reason is that the simple equations for calculating an error require that the points are uncorrelated, which they by definition are not in the MCMC method!

Of course one can always repeat the MCMC runs a few times with different seeds and calculate the error as the error over the different runs.