Basic Diagrammatic Monte Carlo

Goal: Implementation of various Monte Carlo algorithms with focus on Diagrammatic Monte Carlo.

Monte Carlo (MC) algorithms are ubiquitous. They are used in various fields ranging from statistics, mathematics and physics to economics and finance. The general term "MC method" does not refer to a specific numerical technique but to a whole class of algorithms used to solve problems too complex to solve analytically or deterministically [1]. A common feature of all MC methods is their stochastic nature, i.e. the use of random numbers.

Feynman diagrams are a powerful tool in many-body physics which appear in perturbative expansions of quantum statistical averages [2]. Mathematically this means, that we can write a function $Q(\{y\})$ as an (infinte) diagrammatic series

$$Q(\{y\}) = \sum_{n=0}^{N} \sum_{\xi_n} \int dx_1 ... dx_n D_n^{\xi_n}(\{y\}; x_1, ..., x_n). \tag{1}$$

Here $Q(\{y\})$ is our function of interest depending on a set of external parameters $\{y\}$, n is the running index of the series, ξ_n indexes different diagrams of the same order n and $x_1, \ldots, x_n$ are integration/summation variables. For $n = 0$, $D_n^{\xi_n}$ is a function of $\{y\}$ only, i.e. $Q^{(0)}(\{y\}) = D_n^{\xi_n}(\{y\})$. Each function $D_n^{\xi_n}$ can be mapped to a certain Feynman diagram/graph, where graph lines and vertices represent propagators (Green's functions G) and interaction potentials V, so that a diagram is given as a product of Gs and Vs. Figure 1 shows a typical Feynman diagram. Solid lines are electron propagators, the wiggly line is a phonon propagator and vertices are interaction potentials.

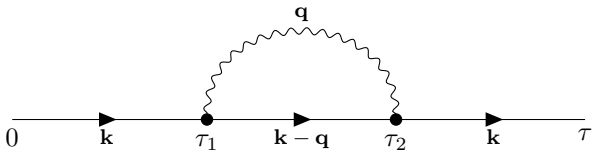


Figure 1: 2^{nd}-order diagram of a one-electron Green's function $G(\mathbf{k}, \tau)$.

The Diagrammatic Monte Carlo (DMC) method refers to the stochastic sampling of Feynman diagrams. It allows one to simulate quantities given as a diagrammatic series similar to Eq. 1. We use the word "simulate" because it does not evaluate the integrals explicitly (as one would expect from a MC technique involving integrals), but rather generates random samples $(n, \xi_n, \{y\}, x_1, ..., x_n)$ according to their weights $D_n^{\xi_n}$. This will become more clear as we work out some examples.

In this project we will work through several tasks all involving MC methods. Starting from the very basics, we will add more and more complexity to the tasks, so that you can become comfortable with implementing your own MC codes. Although the overall goal is to write a DMC code and apply it to some interesting physical systems, we will at first focus only on the technical details and solve purely mathematical toy models with DMC. The reason for this is that usually the theory behind those many-body quantum systems can be quite demanding and one can easily get lost in the physics before even starting to write the first few lines of code. So let's start with the basics.

Task 1: Definitions and Notations

The first task is to familiarize yourself with common notations and definitions used in MC methods. Important concepts are probability, probability density function, cumulative distribution function, joint probability, marginal probability, random sampling, discrete and continuous distributions, mean, expectation value, variance, standard deviation, and so on. If you are already

familiar with all of these concepts, then you can just skip this task. If not, I can recommend Chapters 2 and 3 in [1] or Part 4 in [3].

Task 2: Estimation of π

In this task we will write our first MC program. The goal is to estimate π using two different MC strategies. In both of them we consider the unit circle in the upper right plane ($x > 0$ and $y > 0$):

- Area method: The area underneath the circle (see Figure) is $A_c = \pi/4$ and the area of the unit square is $A = 1$. We can find an estimator for π by generating N uniformly distributed numbers in the unit square and counting the number of points N_c which lie within the circle. If we let $N \to \infty$ then we have $N_c/N = A_c/A$. Rewriting in terms of A_c gives us an estimate for $\pi/4$.
- MC integration: The second method uses simple MC integration. The area underneath the circle is $A_c = \pi/4 = \int_0^1 f(x)dx$ with $f(x) = \sqrt{1-x^2}$. We can rewrite this as $A_c = \int_0^1 f(x)/p(x)p(x)dx$ with $p(x)$ being an arbitrary probability density defined on $[0, 1]$. This lets us interpret the definite integral as an expectation value of $f(x)/p(x)$ with respect to the probabilty density $p(x)$. Such an expectation value has a straightforward MC estimator: $\langle f(x)/p(x)\rangle_p \approx 1/N \sum_{i=1}^{N} f(x_i)/p(x_i)$, where the x_i are generated randomly from $p(x)$.

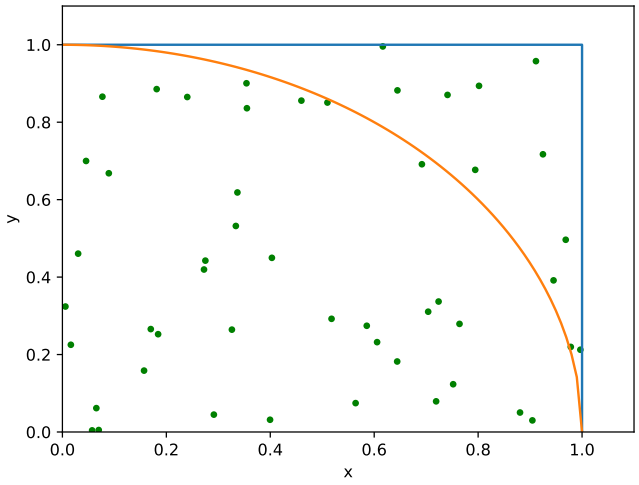


Figure 2: Unit square and unit circle with uniformly distributed random points.

Your tasks are:

1. Implement both methods to get an estimate for π or $\pi/4$ (use a uniform distribution for $p(x)$, i.e. $p(x) = 1/(b-a)$ with $a = 0$ and $b = 1$).
2. Observe how the estimates converge to the exact result with increasing N.
3. Perform M such experiments with a reasonable N for both methods and plot the results in a histogram.
4. How are the results distributed? Around which mean value? Calculate the overall mean and variance for both methods and determine which is more effective.

(Side note: The area method actually also performs a MC integration. It integrates the function $f(x, y) = 1$ with support only on the unit disk.)

Task 3: CDF inversion

In this task we will simulate a simple exponential function $Q(\tau, \alpha) = \exp(-\alpha\tau)$ with α fixed at some value. Note that we are using the notation from Eq. 1 because it can be identified with the 0^{th}-order term of a diagrammatic expansion if we let $\{y\} = \{\tau, \alpha\}$. Since our function is very simple and we already know its analytical form, there usually is no need to simulate it in practice. Nevertheless, we will do it to practice CDF inversion.

- CDF inversion: CDF inversion is a direct sampling method. It lets us generate random samples from an arbitrary probability density $f(x)$ as long as we know its cumulative distribution function (CDF) $F(x)$ and its inverse $F^{-1}(x)$. Let r be a uniform random number on $[0, 1)$, i.e. $r \sim U(0, 1)$. Then we can generate a random sample from $f(x)$ by calculating
$$r = \frac{\int_{x_{min}}^{x} f(x')dx'}{\int_{x_{min}}^{x_{max}} f(x')dx'} \tag{2}$$
and solving for x.

Your tasks are:

1. Let $\alpha = 1$ and let $\tau \in [0, 5]$. Simulate $Q(\tau, \alpha)$ by generating random samples from it using CDF inversion and by putting them into a histogram.
2. Plot the histogram and compare it with the exact function $Q(\tau, \alpha)$. How can you normalize the histogram correctly?
3. Appoximate the following integrals with MC integration:
   - $I_1 = \int_0^5 \tau Q(\tau, \alpha)d\tau$
   - $I_2 = \int_0^5 \tau^2 Q(\tau, \alpha)d\tau$

   Use $Q(\tau, \alpha)$ as the (unnormalized) probability density. Estimate also the error of your approximation by calculating the sample variance/standard deviation (see Chapter 3 in [1] for more details). Compare your results with exact ones.

Task 4: Markov Chain Monte Carlo

In this task we will perform the same simulation as in Task 3, but instead of directly sampling $Q(\tau, \alpha)$, we will generate samples using a Markov Chain Monte Carlo (MCMC) algorithm. In this and all subsequent tasks we will be working with the Metropolis-Hastings algorithm.

- MCMC: MCMC lets us sample from complex, even unnormalized, distributions. Suppose we want to generate samples from the (possibly unnormalized) distribution $f(x)$. Let x_i be an arbitrary point within the domain of f. Then the Metropolis-Hastings algorithm tells us to propose a new point x_j from the proposal distribution $p(x)$ and accept this new point with probability
$$A_{i\to j} = \min\left\{1, \frac{f(x_j)p(x_i)}{f(x_i)p(x_j)}\right\}. \tag{3}$$
The so generated $x_1, x_2, \ldots$ will, after a certain "equilibration" phase, be distributed according to our target distribution $f(x)$. Unfortunately, due to the very nature of Markov chains, the x_i are now correlated and not independent anymore.

Your tasks are:

1. Read Chapter 2 in [1] or Chapter 12 in [3] (if you haven't done it already) or any other introduction to MCMC.
2. Perform the same simulation as in Task 3 but with MCMC. Use a uniform distribution for $p(x) = U(0, 5)$. Keep track of the number of accepted and rejected steps.
3. If the acceptance percentage is too low, try to improve it by changing $p(x)$. How can we make the acceptance percentage equal to 1, i.e. how can we accept every proposed step?
4. Calculate the same integrals as in Task 3 but with MCMC. Use $p(x) = U(0, 5)$. Don't forget about the error estimation!

5. The samples that we generate with MCMC are correlated. This leads to an underestimation of the error bars if we naively use the usual formula for the variance/standard deviation. To overcome this problem, we will use the so called "blocking method". You can read the details in Chapter 3.4 in [1]. Try implementing a simple blocking method to get more reliable error estimates.

Task 5: Diagrammatic Monte Carlo I

In this task we will establish the connection between what we did in the last 2 tasks and Diagrammatic Monte Carlo (DMC). In order to do this, we need to represent our function of interest $Q(\tau, \alpha) = \exp(-\alpha\tau)$ in terms of diagrams. We can do this with a straight line of length τ:

$$= D_0^{\xi_0}(\tau, \alpha) = \exp(-\alpha\tau) \tag{4}$$

The weight of the diagram is $D_0^{\xi_0}(\tau, \alpha)$. DMC generates new diagrams randomly according to their weight. Usually it is done via MCMC and a Metropols-Hastings like algorithm:

(i) Find a way to represent the diagrams $D_n^{\xi_n}$ on the computer. In our case this is quite easy, since a diagram is fully characterized by its length τ and the value of α.

(ii) Start from a random diagram. The easiest is usually just a 0^{th}-order diagram. In our case this would be some arbitrary $D_0^{\xi_0}(\tau_0, \alpha_0)$. (Note: We worked with a fixed α so far, but for the sake of generality we will assume now that it is not fixed.)

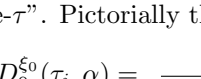
(iii) Propose a new diagram and accept it with a Metropolis-Hastings acceptance step. In our case we propose $D_0^{\xi_0}(\tau_1, \alpha_1)$, where the new α_1 and τ_1 are generated from the distribution $p(\tau, \alpha)$. Then we accept the new diagram with probability:

$$A_{0\to1} = \min\left\{1, \frac{D_0^{\xi_0}(\tau_1, \alpha_1)p(\tau_0, \alpha_0)}{D_0^{\xi_0}(\tau_0, \alpha_0)p(\tau_1, \alpha_1)}\right\}. \tag{5}$$

(iv) Make the desired measurements. In our case this is the simple histogram for $Q(\tau, \alpha)$ and the integrals I_1 and I_2.

(v) Go back to (iii) and repeat.

At the heart of every DMC code are updates which allow us to propose a new diagram given the current one. These updates need to fulfill the ergodicity requirement of the underlying Markov Chain, i.e. we should theoretically be able to produce every possible diagram of the diagrammatic series. For us, to fulfill ergodicity, it is enough to have an update which changes the length τ (if we keep α fixed, otherwise we would also need an update that changes α). Lets call this update "change-τ". Pictorially this looks as follows:

$D_0^{\xi_0}(\tau_i, \alpha) =$ 

$\longleftrightarrow$

$= D_0^{\xi_0}(\tau_j, \alpha)$ (6)

Note the double arrow. This means that for every update we implement, we also need an inverse update, which lets us "undo" the previous update (required by detailed-balance). Our simple change-τ update doesn't need an extra inverse update because itself is its inverse.

Your tasks are:

1. Read Chapter 12 in [4] to get a better understanding of DMC.

2. Restructure/rewrite your code from Task 4 to incorporate the idea of diagrams and DMC. Try to make it as general as possible, so that it is easier to extend the code to other diagrams and more sophisticated updates.

3. Test your code by comparison with the previous tasks. The results should be the same.

4. Introduce an update to change α, called "change-α". Select the change-τ and the change-α updates with equal probability. Run your code for $\alpha \in \{0.5, 1.0\}$. Of course, this could also be done in two independent runs (and also more efficiently) but it is a good practice opportunity for implementing updates. Plot the histograms for both α values.

$$D_0^{\xi_0}(\tau, \alpha_i) = \underset{0 \qquad \alpha_i \qquad \tau}{\text{———————}} \quad \longleftrightarrow \quad \underset{0 \qquad \alpha_j \qquad \tau}{\text{———————}} = D_0^{\xi_0}(\tau, \alpha_j) \tag{7}$$

Task 6: Diagrammatic Monte Carlo II

So far we have only worked with 0^{th}-order diagrams. In this task we will add a 2^{nd}-order diagram. The function we are interested in is now given as

$$Q(\tau, \alpha, V) = \exp(-\alpha\tau) + \sum_\beta \int_0^\tau d\tau_2 \int_0^{\tau_2} d\tau_1 e^{-\alpha\tau_1} V e^{-\beta(\tau_2-\tau_1)} V e^{-\alpha(\tau-\tau_2)}, \tag{8}$$

where we will take $\tau \in [0, 5]$, $\alpha = 1$, $V = 0.5$ and $\beta \in \{\beta_1 = 0.25, \beta_2 = 0.75\}$. (Note: We call it 2^{nd}-order, although we have 3 integration variable τ_1, τ_2 and β. This is has to do with the number of vertices and is just notation.) The 0^{th}-order diagram is the same as in the last task. The second order diagram can be pictured as follows:

$$\underset{0 \quad \alpha \quad \tau_1 \quad \beta \quad \tau_2 \quad \alpha \quad \tau}{\text{———●- - - -●———}} = D_2^{\xi_2}(\tau, \alpha, V; \tau_1, \tau_2, \beta) = e^{-\alpha\tau_1} V e^{-\beta(\tau_2-\tau_1)} V e^{-\alpha(\tau-\tau_2)} \tag{9}$$

Each of the 2 vertices in the diagram contribute a factor of V to its weight. Now in order to perform a DMC for this kind of problem, we need two more updates which allow us to change the order of the diagram, i.e. one update to go from $0 \to 2$ and its inverse to go from $2 \to 0$. They can be implemented as follows:

$$\underset{0 \qquad \alpha_i \qquad \tau}{\text{———————}} \quad \longleftrightarrow \quad \underset{0 \quad \alpha \quad \tau_1 \quad \beta \quad \tau_2 \quad \alpha \quad \tau}{\text{———●- - - -●———}} \tag{10}$$

- $0 \to 2$: Lets call this update "add-β". Starting point is $D_0^{\xi_0}(\tau, \alpha)$. If we want to propose a $D_2^{\xi_2}(\tau, \alpha, V; \tau_1, \tau_2, \beta)$ diagram, we need values for τ_1, τ_2 and β. The simplest way is to generate them uniformly, i.e. let $\tau_1 \sim U(0, 5) = p_1(\tau_1)$, $\tau_2 \sim U(\tau_1, 5) = p_2(\tau_2)$ and $\beta = \beta_1$ with probability $0.5 = p_3(\beta_1)$ and $\beta = \beta_2$ with probability $0.5 = p_3(\beta_2)$. Then the acceptance probability for the Metropolis-Hastings algorithm is

$$A_{0\to2} = \min\left\{1, \frac{p_{rem}}{p_{add}} \frac{D_2^{\xi_2}(\tau, \alpha, V; \tau_1, \tau_2, \beta) d\tau_1 d\tau_2}{D_0^{\xi_0}(\tau_0, \alpha_0) p_1(\tau_1) p_2(\tau_2) p_3(\beta) d\tau_1 d\tau_2}\right\}. \tag{11}$$

  p_{add} and p_{rem} are so called context factors. It can be quite tricky to get them right (see Chapter 12 in [4] for more information). In our case they are simply the probabilities with which we address the add-β and remove-β updates. If we address them with the same probabilty, then the ratio is $p_{rem}/p_{add} = 1$. The differentials cancel each other out. If the current diagram is already 2^{nd}-order, just reject the update right away.

- $2 \to 0$: Lets call this update "remove-β". It is the inverse of add-β. We don't have to propose new values. Its acceptance probability is the inverse of $A_{0\to2}$, i.e.

$$A_{2\to0} = \min\left\{1, \frac{p_{add}}{p_{rem}} \frac{D_0^{\xi_0}(\tau_0, \alpha_0) p_1(\tau_1) p_2(\tau_2) p_3(\beta) d\tau_1 d\tau_2}{D_2^{\xi_2}(\tau, \alpha, V; \tau_1, \tau_2, \beta) d\tau_1 d\tau_2}\right\}. \tag{12}$$

  If the current diagram is already 0^{th}-order, just reject the update right away.

Your tasks are:

1. Take your code from Task 5 and extend it to handle also 2^{nd}-order diagrams.
2. Implement add-β and remove-β updates.

3. Try to simulate $Q(\tau, \alpha, V)$ for the values given at the beginning of the task. Compare your your results with analytically exact ones.

4. How can you normalize the histogram correctly? In the current case we can still calculate the normalization analytically, but this is usually not possible. Can you come up with a different method to normalize our histogram?

Task 7: Diagrammatic Monte Carlo III (Advanced, Optional)

In this last task we will improve our histogram. So far we have used a simple binning strategy for the histogram, i.e. we discretized the τ-space into bins and everytime we generated a diagram/τ-value we put it in the corresponding bin. This introduces a certain discretization error, which we could decrease by making the bins smaller and smaller but this would increase the statistical error because less and less points would fall into a certain bin. To get rid of the error, we want to derive an exact MC estimator q_{τ_0}, such that for a given α and V value, we have

$$\left\langle q_{\tau_0} \right\rangle_{MC} = C * Q(\tau_0, \alpha, V), \tag{13}$$

i.e. the expectation value of our estimator should be the exact value of the function at a specific τ_0. C is just a normalization constant.

Your tasks are:

1. Read Section 3.C in [5]. Derive the estimator for our case.

2. Implement the estimator in your code and test it with $Q(\tau, \alpha)$ from Task 5 as well as $Q(\tau, \alpha, V)$ from Task 6.

Outlook

I hope these tasks can introduce you to MC methods in general and prepare you to work on your own DMC code. I assumed that you had no prior experience with implementing MC code yourself. If you already have implemented something similar and the tasks are too trivial for you, then let me know. We can think of some interesting physical system and jump right into writing a full DMC code.
If, on the other hand, the tasks are too complicated, don't hesitate to contact me. We can then work out the problems together or I can send you some more references. Especially, if you are stuck somewhere or your results don't seem to be correct. Before you spend days or weeks trying to figure out what and if something is wrong, just let me know right away. I think it would be beneficial for both of us, if we discuss all the problems that arise.

After you completed the tasks, we will leave the mathematical toy models and start working on more interesting physical systems. But we can disuss these, when the time comes.

Anyway, if you have any questions I am happy to help. Have fun!

References

[1] J. Gubernatis, N. Kawashima & P. Werner. "Quantum Monte Carlo Methods". Cambridge University Press. (2016)

[2] K. Van Houcke, E. Kozik, N. Prokof'ev & B. Svistunov. "Diagrammatic Monte Carlo". Physics Procedia, 6, 95–105. (2010)

[3] M. Hjorth-Jensen. "Computational Physics". Lecture Notes. (2015)

[4] H Fehske, R. Schneider & A. Weiße. "Computational Many-Particle Physics". Springer. (2008)

[5] A. Mishchenko *et al.* "Diagrammatic quantum Monte Carlo study of the Fröhlich polaron". Physical Review B, 62(10), 6317. (2000)