PHY407 – University of Toronto
Lecture 11: Monte Carlo Simulations

Nicolas Grisouard, nicolas.grisouard@utoronto.ca

30 November 2020

Supporting textbook chapters for week 11: Chapters 10.3&4

Fill out the online evaluations!!!

1 Previously, in PHY407...

Q on the chat last week: > *is MonteCarlo used to find the normalization factor in QM?*

See here for applications of MC techniques in QM:

https://en.wikipedia.org/wiki/Quantum_Monte_Carlo

Last week: * How to draw random numbers: PRNGs * how to "fake" random draws (linear congruential ggenerator for illustration); * how to test: statistical properties of distributions; * Python's (pseudo) random number generator is the Mersenne Twister * Transformation of distributions, e.g., for uniformly distributed distribution $p(x) = a\exp(-ax)$ obtained from

$$x = -\frac{1}{a}\ln(1-z).$$

- Monte Carlo integration:
  - when functions are pathological (fast variations),
  - when integrating over a lot of dimensions,
  - when integration domains are complicated.
- MC integration techniques:

Importance sampling

- Hit or Miss integration and mean value method have errors that vary as $N^{-1/2}$
- Importance sampling chooses weights that favour largest integration values:

$$I = \int_a^b f(x)\mathrm{d}x = \left\langle \frac{f(x)}{w(x)} \right\rangle_w \int_a^b w(x)\mathrm{d}x,$$

$$\left\langle \frac{f(x)}{w(x)} \right\rangle_w = \frac{\int_a^b \left[\frac{f(x)}{w(x)}\right] w(x)\mathrm{d}x}{\int_a^b w(x)\mathrm{d}x} \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{w(x_i)},$$

Week 10, topics: * Monte Carlo simulation

Fill out the online evaluations!!!

Monte Carlo simulations:

Any simulation that uses random numbers to simulate random physical processes to estimate something about the outcome of that process.

We focus on statistical mechanics here.

2 Statistical mechanics: a review

- For a system in equilibrium at temperature T (canonical ensemble), the probability of finding the system in any particular microstate i is given by the Boltzmann distribution,

$$P(E_i) = \frac{\exp\left[-E_i/(k_B T)\right]}{Z}, \quad Z = \sum_{i=1}^{ALL} \exp\left[-E_i/(k_B T)\right]$$

  where E_i is the energy of microstate i, and k_B is Boltzmann's constant.

- System at temperature T undergoes transitions between microstates with probability of being in a particular microstate $P(E_i)$
- To calculate a macroscopic property during a measurement (total energy, magnetization…) $\Rightarrow$ average over the many microstates that the system visits during the measurement.
- If we want to measure a quantity "X" over the macrostate:

$$\langle X \rangle = \sum_{i=1}^{ALL} X_i P(E_i)$$

  where X_i is the value of the quantity in the i^{th} microstate and P is the probability of finding the system in that microstate.

- Simple example: single mole of gas has $N_A \approx 6 \times 10^{23}$ molecules. Assume each molecule had only 2 possible quantum states (gross underestimation), then the total number of microstates of the mole of gas is 2^{N_A}, which is huge.

3 Monte Carlo simulation in Stat. Mech.

$$\text{Recall} \quad P(E_i) = \frac{\exp\left[-E_i/(k_B T)\right]}{Z}, \quad Z = \sum_{i=1}^{ALL} \exp\left[-E_i/(k_B T)\right]$$

3.1 Setting the problem

- Huge number of terms in sum $\Rightarrow$ use Monte Carlo summation.
- Two difficulties to overcome:
  1. properly sampling which terms to sum over (*solution: importance sampling*),
  2. estimating Z (*solution: Markov Chain Monte Carlo*)

Difficulty #1: why do we have to choose which terms to sum over?

(We will get to difficulty #2 later)

- Randomly sample the terms in the sum and only use those as an estimate. Replace:

$$\langle X \rangle = \sum_{i=1}^{ALL} X_i P(E_i)$$

- with a sum over N randomly sampled microstates:

$$\langle X \rangle = \frac{\sum_{i=1}^{N} X_i P(E_i)}{\sum_{i=1}^{N} P(E_i)}$$

- the denominator is needed to ensure the total probability over the sampled states is 1.
- It is only worth keeping the big terms in the sum if we want to compute this:

$$\langle X \rangle = \sum_{i=1}^{ALL} X_i P(E_i)$$

- There are a lot of states with $P(E_i)$ really small, with $E_i \gg k_B T$, which is the case for most of the states:

$$P(E_i) = \frac{\exp\left[-E_i/(k_B T)\right]}{Z}$$

- To get a good estimate for the sum, need to preferentially choose terms where the integrand is non-negligible.
- So we should use importance sampling!

3.2 Importance sampling for Stat. Mech.

- For an integral

$$I = \int_a^b f(x)\mathrm{d}x = \left\langle \frac{f(x)}{w(x)} \right\rangle_w \int_a^b w(x)\mathrm{d}x \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{w(x_i)} \int_a^b w(x)\mathrm{d}x,$$

- For a sum:

$$\langle X \rangle = \sum_{i=1}^{N} X_i P(E_i) \approx \frac{1}{N} \sum_{k=1}^{N} \frac{X_k P(E_k)}{w_k} \sum_{i=1}^{ALL} w_i.$$

- What to choose for weight w to reduce the variance?
- $P(E_i)$ of course!

$$\langle X \rangle \approx \frac{1}{N} \sum_{k=1}^{N} \underbrace{\frac{X_k P(E_k)}{P(E_k)}}_{=X_k} \underbrace{\sum_{i=1}^{ALL} P(E_i)}_{=1}.$$

$$\langle X \rangle \approx \frac{1}{N} \sum_{k=1}^{N} X_k.$$

* Looks simple and no different from regular MC, * but recall that the X_k's are drawn from non-uniform distribution: we randomly choose terms in the sum based on their Boltzmann probabilities.

- One thing left to deal with: How do we pick states with probability $P(E_k)$? Recall:

$$P(E_i) = \frac{\exp\left[-E_i/(k_B T)\right]}{Z}, \quad Z = \sum_{i=1}^{ALL} \exp\left[-E_i/(k_B T)\right]$$

- To do it this way, we need Z, which is a sum over all states. But if we could do this, we wouldn't need Monte Carlo in the first place!

3.3 Markov chain method

3.3.1 Elevator Pitch

Mish-mashing https://en.wikipedia.org/wiki/Markov_chain and https://en.wikipedia.org/wiki/Markov_property,

> A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. [...] (sometimes characterized as "memorylessness"). In simpler terms, it is a process for which predictions can be made regarding future outcomes based solely on its present state [...]. In other words, conditional on the present state of the system, its future and past states are independent.

- Random walks (Brownian motion) are Markov chains.
- Here: events are jumps in energy states, one after another.
- Solution: Use the Markov chain method.
  - Text goes into details on how to implement this method with a Metropolis algorithm.
  - Crucial key: Metropolis does not compute probability to be in one state, but probability to transition between two states (Z cancels out in the process).
  - I will summarize it algorithmically first, then briefly outline why it works mathematically.

3.3.2 Algorithm

1. Choose a random starting state i
2. Calculate the energy of that state E_i
3. Choose a transition to a new state j uniformly at random from allowed set
4. Calculate the energy of this new state, E_j
5. Calculate the acceptance probability for this transition:
   - $P_a = 1$ if $E_j \leq E_i$ (always accept a lower energy state)
   - $P_a = \exp\left(-\frac{E_j - E_i}{k_B T}\right)$ if $E_j > E_i$ (sometimes accept a higher energy state, more often for high T).
6. Accept/reject the move according to the acceptance probability
7. Measure the quantity X you want in its current state (new or old i) & store it
8. Repeat from step 2.

- How to implement the probability of the event in the previous slide?
  - $P_a = 1$ if $E_j \leq E_i$ (always accept a lower energy state)
  - $P_a = \exp\left(-\frac{E_j - E_i}{k_B T}\right)$ if $E_j > E_i$ (sometimes accept a higher energy state, more often for hight T).
- Draw a random number in $[0, 1)$. The statement

  ```
  if random() < exp(-(Ej-Ei)/kT):
  ```

  will introduce what to do if the move is accepted (`elif` will introduce what to do if rejected).
- E.g., at very high T, exp ≈ 1 and almost all moves are accepted.
- E.g., at low T, say, $\exp(-(E_j - E_i)/(k_B T)) = 1\%$, then `random()` has 1% chance of drawing a number that is $< 1\%$.
- If $E_j \leq E_i$, then exp ≥ 1 and `if` statement automatically accepts.

3.3.3 Why does it work?

- Why do the probability transitions move-by-move (Metropolis algorithm) end up creating a system where each microstate has a probability $P(E_i)$, the Boltzmann distribution?
- Let τ_{ij} a transition probability from μ-state i to μ-state j, such that

$$\frac{\tau_{ij}}{\tau_{ji}} = \frac{P(E_j)}{P(E_i)}.$$

  For example, a small ratio above means that "j is much less probable than i" and equivalently, "probability of $j \to i$ is much higher than $i \to j$", in equal amounts.
- "In equal amounts" is crucial: it means that if you start from any initial state, your system will progressively evolve towards one where all μ-states follow Boltzmann.
- Does it always converge? Could it converge to another distribution? If you love linear algebra and eigenvectors, see proof in Appendix D of Newman (it is actually quite elegant).

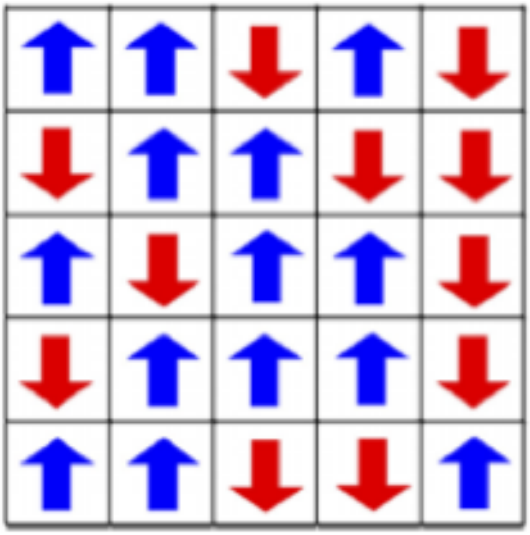
3.4 Example: Ising model

3.4.1 Elevator Pitch

- Simple model of ferromagnetism, but demonstrates many of the physical characteristics of fancier models.
- Assume an object is made up of a collection of dipoles (e.g. electron spins) and the net magnetization is the sum of the magnetization of all the spins
- Ising model:
  - assume the spins can only point up or down.
  - the spins interact and favor parallel alignment of pairs of spins
  - the interactions are non-zero only between nearest neighbours (i.e. distance dependent).
- The macroscopic energy E and magnetization M given by

$$E = -J \sum_{\langle ij \rangle} s_i s_j \quad \text{and} \quad M = \sum_i s_i$$

  where $s = +1$ if spin is up & $s = -1$ if spin is down.
- Notice that the lowest energy occurs if the spins all line up.
- Spins can randomly flip as the system visits a set of allowable states given its temperature. At any particular moment the system may look like



3.4.2 Example in 1D

- Create array of dipoles, initial state: random spin at each location.
- Calculate energy & magnetization of state
- Implement Metropolis algorithm:
  - create new state: flip 1 spin randomly
  - calculate new total energy
  - calculate acceptance probability
  - decide whether to accept or reject new state
  - store 'new' energy & magnetization

    - repeat
- After this is a starter code for lab 11.

Oh, and by the way…

Fill out the online evaluations!!!

[ ]:
```
# %load ising_1D_start.py
```

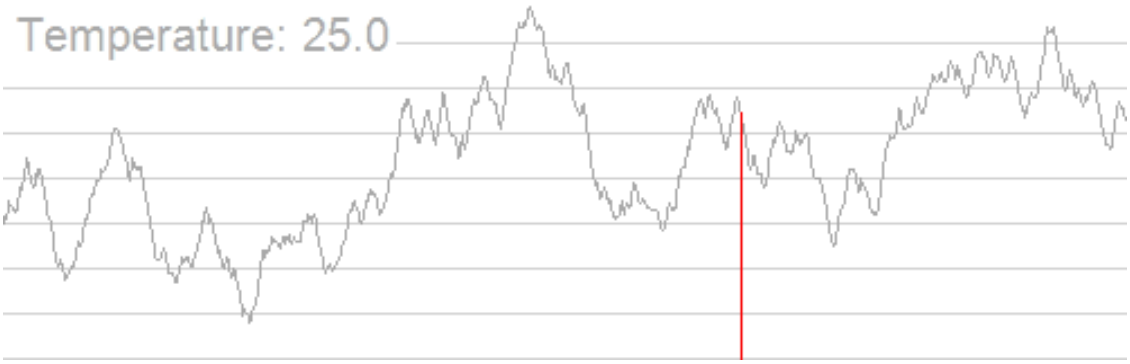
[ ]:
```
import matplotlib.pyplot as plt
# plot energy, magnetization
fg, ax = plt.subplots(2, 1, sharex=True)
ax[0].plot(magnet)
ax[0].set_ylabel('Magnetization')
ax[0].grid()
ax[1].plot(energy)
ax[1].set_xlabel('Number of flipping attempts')
ax[1].set_ylabel('Energy')
ax[1].grid()

plt.tight_layout()
plt.show()
```

4 Simulated annealing

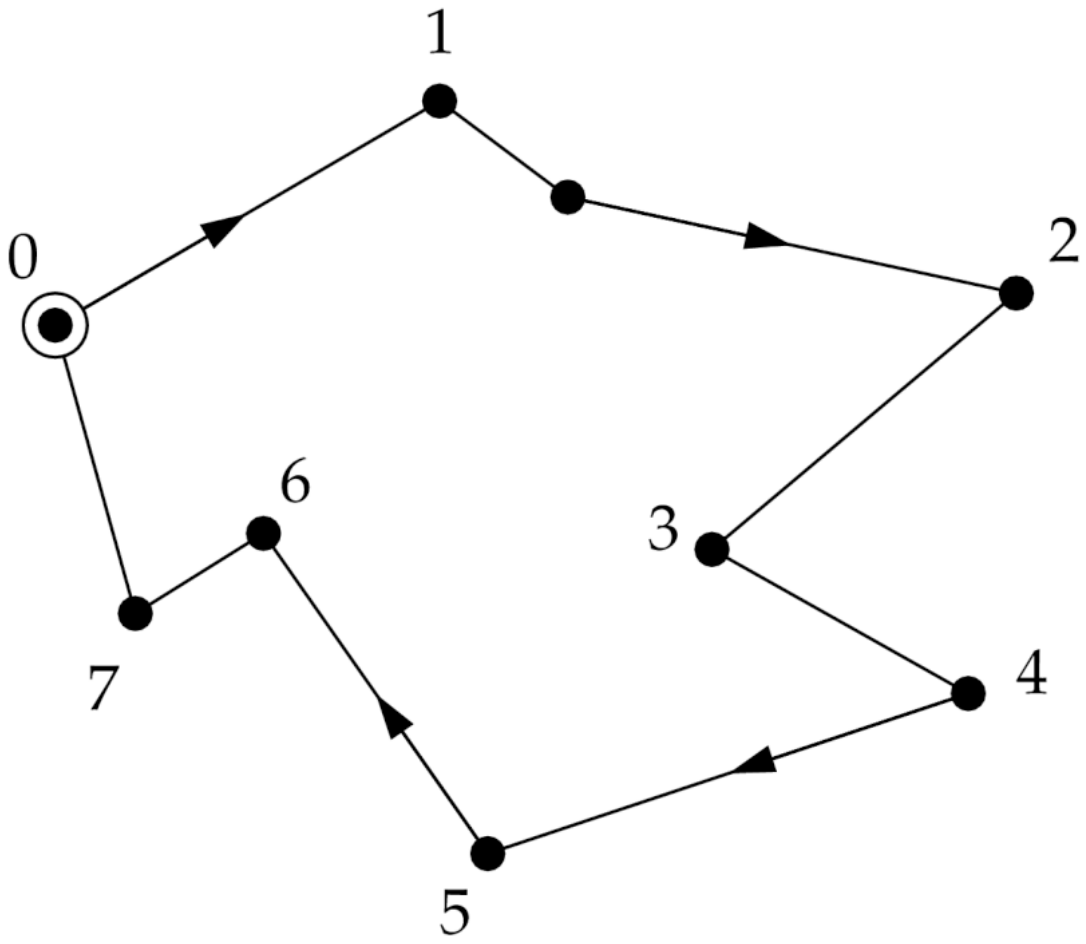
4.1 Elevator Pitch

- Using Monte Carlo simulations to find **global** minima/maxima.
- In week 4 we talked about ways of finding local minima (e.g., golden ratio search).
- How it works: rewrite max/min problem as looking for a "ground state energy" of a system.
    - Function f that you want the max/min of: make this the energy function.
    - how could you find ground state: reduce temperature until you reach the ground state.
- Issue: if you reduce temperature too quickly: might get caught in a local min instead of the global min.
- Solution: reduce temperature slowly. This way system has time to explore many microstates and find a good approximation to the global minimum.
- Visual Analogy: particle in a bumpy potential. Too low energy: get stuck in nearest local minimum. Keep low energy but allow some random 'kicks' in energy: can kick out of local minimum and continue heading to global minimum (see this figure).



4.2 Example: travelling salesman

- Famous NP-hard problem (https://en.wikipedia.org/wiki/NP-hardness): what is the shortest route to visit a given set of locations on a map?
- Want global minimum of distance
- Start with random route, swap 2 cities, use Metropolis algorithm to determine whether to keep the swap
- "energy" in this case is the total distance of the route
- You can explore this problem using code from the book (`salesman.py`).



Fill out the online evaluations!!!

[ ]: