PHY407: Computational Physics
Fall, 2017

Lecture 11: Random Processes, Part 2

FILL OUT THE ONLINE EVALUATIONS

Summary & Status

- ☑ Weeks 1-3: Programming basics, numerical errors, numerical integration and differntiation.
- ☑ Weeks 4-5: Solving linear & nonlinear systems and Fourier transforms.
- ☑ Week 6: ODEs Part 1: RK4, Leapfrog, Verlet, adaptive time stepping; customizing python output
- ☑ Week 7: ODEs Part 2: Bulirsch-Stoer, Boundary Value Problems/shooting,
- ☑ Weeks 8-9: PDEs Part 1: Elliptic equation solvers, leapfrog time stepping, FTCS, Crank-Nicholson, Spectral Methods
- ☑ Week 10: Stochastic methods, Part 1: random numbers, monte carlo integration
- ❑ Week 11: Stochastic methods, Part 2: recap, statistical mechanics ideas, simulated annealing approach to optimization.
- ❑ Week 12 (no lab): recap, discussion, extensions.

PHY407: Computational Physics
Fall, 2017

Lecture 11: Random methods/stochastics Part 2

- Recap from last week
- Statistical mechanics ideas
- The “Metropolis” algorithm
- Simulated Annealing

FILL OUT THE ONLINE EVALUATIONS

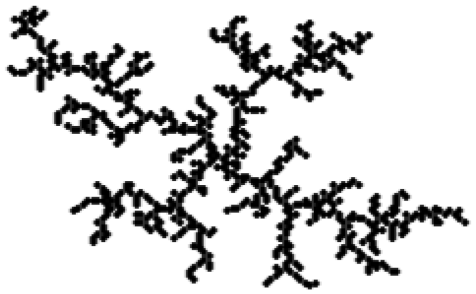
Recap from last week

- Python's (pseudo) random number generator is the Mersenne Twister
- Transformation of distributions, e.g.

For uniformly distributed z, Poisson distribution $p(x) = \mu e^{-\mu x}$ obtained from

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

- Using random number sequence to generate fractal structures.
- Monte Carlo integration.



Monte Carlo: Importance Sampling

- Name two reasons we need Monte Carlo integration…
- 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:

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

$$\left\langle \frac{f(x)}{w(x)} \right\rangle_w = \frac{\int_a^b \left(f(x) \middle/ w(x) \right) w(x)\,dx}{\int_a^b w(x)\,dx} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{w(x_i)}, \quad x_i \text{ from } p(x) \propto w(x)$$

FILL OUT THE ONLINE EVALUATIONS

Review: Statistical Mechanics

- For a system in equilibrium at temperature T, the probability of finding the system in any particular microstate ‘i’ is given by:

$$\boxed{P(E_i) = \frac{\exp(-\beta E_i)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

- where E_i is the energy of state i, and k_B is Boltzmann’s constant.

- System undergoes transitions between microstates with probability of being in a particular microstate $P(E_i)$

- To calculate a macroscopic property during a measurement, like the total energy, or magnetization, we need to average over the many microstates that the system visits during the measurement.

Review: Statistical Mechanics

- For a system in equilibrium at temperature T, the probability of finding the system in any particular microstate ‘i’ is given by:

$$\boxed{P(E_i) = \frac{\exp(-\beta E_i)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

Review: Statistical Mechanics

- For a system in equilibrium at temperature T, the probability of finding the system in any particular microstate ‘i’ is given by:

$$\boxed{P(E_i) = \frac{\exp(-\beta E_i)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

At given T, what determines the probability of a particular microstate?

Review: Statistical Mechanics

- For a system in equilibrium at temperature T, the probability of finding the system in any particular microstate 'i' is given by:

$$\boxed{P(E_i) = \frac{\exp(-\beta E_i)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

At given T, what determines the probability of a particular microstate?

E

What states are more likely? Those with high E or those with low E?

Review: Statistical Mechanics

- For a system in equilibrium at temperature T, the probability of finding the system in any particular microstate ‘i’ is given by:

$$\boxed{P(E_i) = \frac{\exp(-\beta E_i)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

At given T, what determines the probability of a particular microstate?

E

What states are more likely? Those with high E or those with low E?

low E (e.g. E << k_BT) $\rightarrow$ exp(-E/k_BT) bigger than state with higher E

The Problem of Large Numbers

- If we want to measure a quantity ‘ *X* ’ over the macrostate:

$$\boxed{< X >= \sum_{i=1}^{ALL} X_i P(E_i)}$$

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.

- Typically, this sum has an enormous number of terms.

- Simple example: single mole of gas has 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: $\boxed{2^{10^{23}}}$ which is HUGE.

Monte Carlo Summation in Stat. Mech.

- Huge number of terms in sum → use Monte Carlo summation
- Randomly sample the terms in the sum and only use those as an estimate. Replace:

$$\boxed{<X>=\sum_{i=1}^{ALL} X_i P(E_i)}$$

- with a sum over 'N' randomly sampled microstates:

$$\boxed{<X>=\frac{\sum_{k=1}^{N} X_k P(E_k)}{\sum_{k=1}^{N} P(E_k)}}$$

- the denominator is needed to ensure the total probability over the sampled states is 1.

Monte Carlo Summation in Stat. Mech.

- It is only worth keeping the big terms in the sum if we want to compute this:

$$\boxed{< X >= \sum_{i=1}^{ALL} X_i P(E_i)}$$

- There are a lot of states with *P(E$_i$)* really small, with *E$_i$>>k$_B$T*, which is the case for most of the states:

$$\boxed{P(E_i) = \frac{\exp(-E_i / k_B T)}{Z}}$$

- 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!

Importance Sampling

- For an integral:

$$I = \int_a^b f(x)\,dx = \int_a^b w(x)\frac{f(x)}{w(x)}\,dx$$

$$\Rightarrow I = \left\langle \frac{f(y)}{w(y)} \right\rangle_w \int_a^b w(x)dx \approx \frac{1}{N}\sum_{k=1}^{N}\frac{f(y_k)}{w(y_k)}\int_a^b w(x)dx$$

$\updownarrow$ (compare directly)

- For a sum (what we have):

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

- What to choose for weight w to reduce the variance?

Importance Sampling

- For an integral:

$$I = \int_a^b f(x)\,dx = \int_a^b w(x)\frac{f(x)}{w(x)}\,dx$$

$$\Rightarrow I = \left\langle \frac{f(y)}{w(y)} \right\rangle_w \int_a^b w(x)\,dx \approx \frac{1}{N}\sum_{k=1}^{N}\frac{f(y_k)}{w(y_k)}\int_a^b w(x)\,dx$$

$\updownarrow$ (compare directly)

- For a sum (what we have):

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

- What to choose for weight w to reduce the variance?

- P(E_i)! (Let's keep terms with with large P)

$$\boxed{< X > \approx \frac{1}{N}\sum_{k=1}^{N}\frac{X_k P(E_k)}{P(E_k)}\sum_{i=1}^{ALL} P(E_i)}$$

Importance Sampling in Stat. Mech.

$$\boxed{< X > \approx \frac{1}{N}\sum_{k=1}^{N}\frac{X_k P(E_k)}{P(E_k)}\sum_{i=1}^{ALL} P(E_i)}$$

- Two ways we can simplify this expression. What are they?
- (1)
- (2)

Importance Sampling in Stat. Mech.

$$\boxed{< X > \approx \frac{1}{N}\sum_{k=1}^{N}\frac{X_k P(E_k)}{P(E_k)}\sum_{i=1}^{ALL}P(E_i)}$$

- Two ways we can simplify this expression. What are they?
- (1) Cancel out the P’s in the first sum:

$$\boxed{\Rightarrow < X > \approx \frac{1}{N}\sum_{k=1}^{N}X_k\sum_{i=1}^{ALL}P(E_i)}$$

- (2) Sum of probabilities over all states is 1:

$$\boxed{\Rightarrow < X > \approx \frac{1}{N}\sum_{k=1}^{N}X_k}$$

Importance Sampling in Stat. Mech.

$$\boxed{<X> = \frac{1}{N}\sum_{k=1}^{N} X_k}$$

wow that's much simpler!

but remember: you are now choosing your k's from a non-uniform distribution p

- our sampling points *k* come from:

$$\boxed{p_k = \frac{w_k}{\sum_{i=1}^{ALL} w_i} = \frac{P(E_k)}{\sum_{i=1}^{ALL} P(E_i)} = P(E_k)}$$

- So essentially, to find the macroscopic <*X*> we randomly choose terms in the sum based on their Boltzmann probabilities.

Markov Chain Method

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

$$\boxed{P(E_k) = \frac{\exp(-\beta E_k)}{Z}, \qquad Z = \sum_{i=1}^{ALL} \exp(-\beta E_i), \qquad \beta = \frac{1}{k_B T}}$$

- 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!

- Solution: Use the Markov Chain Method:
    - text goes into details on how to implement this method with a Metropolis algorithm
    - I will summarize it algorithmically…

Markov Chain Method: The Algorithm

1. Choose a random starting state
2. Calculate the energy of that state E_i
3. Choose a transition to a new state uniformly at random from allowed set
4. Calculate the energy of this new state
5. Calculate the acceptance probability for this transition:

$$\boxed{P_a = \begin{cases} 1 & \text{if } E_j \leq E_i \\ \exp[-\beta(E_j - E_i)] & \text{if } E_j > E_i \end{cases}}$$

- Always accept a lower energy state.
- Sometimes accept a higher energy state.
- Accept higher energy more often for higher temperature.

6. Accept/reject the move according to the acceptance probability
7. Measure the quantity you want ‘X’ in its current state & store it
8. Repeat from step 2.

Ok, lets try applying it…

Ising Model

- 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).

Ising Model

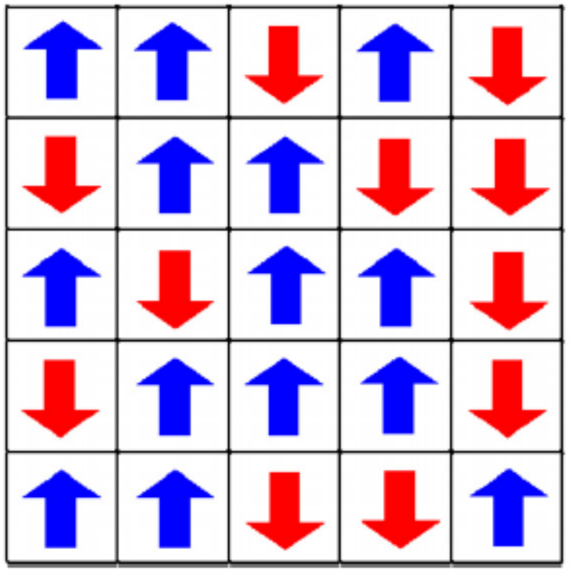
- The macroscopic energy given by:

$$\boxed{E = -J \sum_{<ij>} s_i s_j}$$

  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:

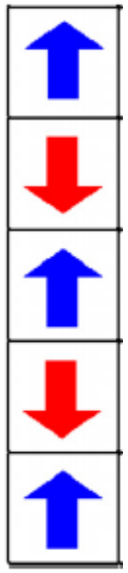


Example: 1D Ising Model

- create array of dipoles, initial state: random spins
- Calculate energy & magnetization of state

- use 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

- All right, lets do it…



FILL OUT THE ONLINE EVALUATIONS

Simulated Annealing

- Using Monte Carlo simulations to find GLOBAL minima/maxima.

- In Ch. 6 we talked about ways of finding local mimima (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: make this the energy function.

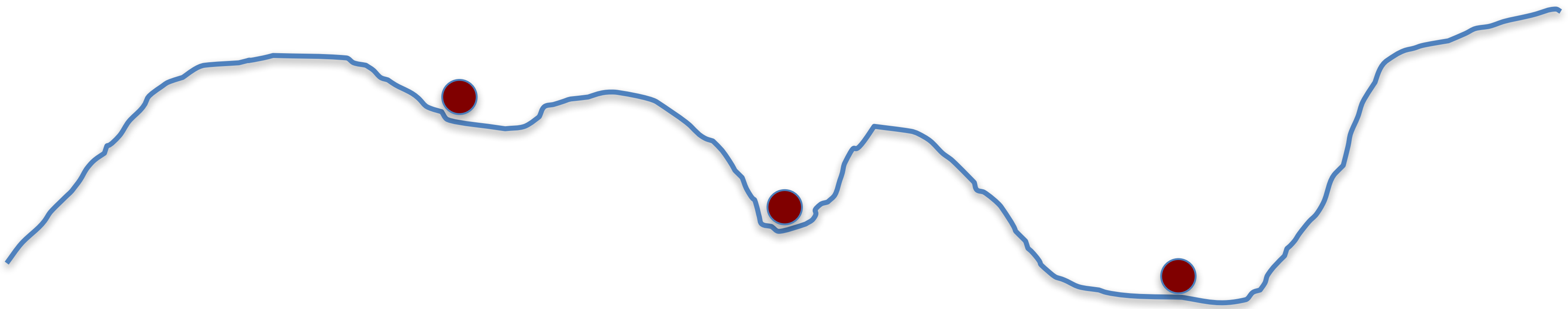
  - how could you find ground state: reduce temperature until you reach the ground state.

Simulated Annealing

- 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.

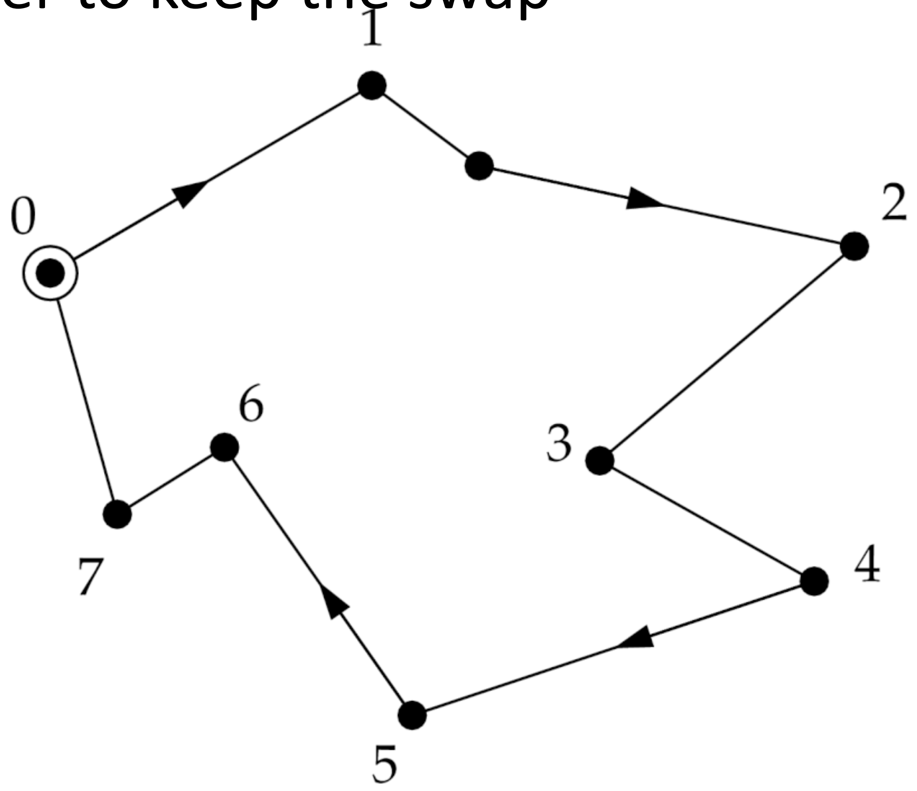
- Visual Analogy: particle in a bumpy potential. Too low energy: get stuck in nearest local minimum. Keep low energy but allow some random 'kicks' in energy: can kick out of local minimum and continue heading to global minimum.



Example: Traveling Salesman

- Famous NP-hard problem: 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.



show salesman.py

PHY407: Computational Physics Fall, 2017

Lecture 11: Random methods/stochastics Part 2

- Recap from last week
- Statistical mechanics ideas
- The “Metropolis” algorithm
- Simulated Annealing

FILL OUT THE ONLINE EVALUATIONS

Summary & Status

- ☑ Weeks 1-3: Programming basics, numerical errors, numerical integration and differntiation.
- ☑ Weeks 4-5: Solving linear & nonlinear systems and Fourier transforms.
- ☑ Week 6: ODEs Part 1: RK4, Leapfrog, Verlet, adaptive time stepping; customizing python output
- ☑ Week 7: ODEs Part 2: Bulirsch-Stoer, Boundary Value Problems/shooting,
- ☑ Weeks 8-9: PDEs Part 1: Elliptic equation solvers, leapfrog time stepping, FTCS, Crank-Nicholson, Spectral Methods
- ☑ Week 10: Stochastic methods, Part 1: random numbers, monte carlo integration
- ☑ Week 11: Stochastic methods, Part 2: recap, statistical mechanics ideas, simulated annealing approach to optimization.
- ❑ **Week 12 (no lab): recap, discussion, extensions.**