The Metropolis-Hastings algorithm

The first thoughts and attempts I made . . . were suggested by a question which occurred to me in 1946 as I was convalescing from an illness and playing solitaires. The question was what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method that 'abstract thinking' might not to be to lay it out say one hundred times and simply observe and count the number of successful plays.

This was already possible to envisage with the beginning of the new era of fast computers, and I immediately thought of problems of neutron diffusion and other questions of mathematical physics. . .

Stanislaw Ulam, 1983

9.1 Introduction

In the final years of World War II, a team of physicists, engineers and early computer scientists were working together at Los Alamos. Most people know this is when the first nuclear tests took place during the development of the nuclear bomb. Many people also know that the first electronic general purpose computer, ENIAC, was developed. Far fewer people know that this was also the birthplace of one of the most important algorithms in computational science the Monte Carlo Method.

The general motivation for a statistical-based technique was to allow physicists to solve problems that were too complex to solve analytically. The particular motivation was to model the behaviour of neutrons in a nuclear reactor.

Numerical simulations are often useful in scenarios where there are easily characterised interactions or effects, but the combined result is not analytically solvable. This can frequently occur when effects are probabilistic, which is often the case in particle physics and thermodynamics. Monte Carlo simulations are ideally suited to these types of problems because of the probabilistic nature of particle interactions, as well as the discrete nature of the particles themselves.

After WWII ended Nicholas Metropolis and Stanislaw Ulam continued working on computational methods for solving complex problems, publishing *The Monte Carlo Method* in 1949. The MH algorithm was named after Metropolis as he was the first author on a 1953 paper that presented an improved method for sampling from probability distributions. However, there is some controversy about who made the primary contributions to that work (see the references). The method was generalised by Keith Hastings in 1970.

In this lab we will start with a simple sampling method called rejection sampling, and then improve on it by implementing the Metropolis-Hastings algorithm.

9.2 Lab objectives

By the end of this lab sessions you should be able to:

1. Understand the concept of sampling from a distribution.
2. Implement rejection sampling for a given distribution.
3. Understand and implement the Metropolis algorithm

If you still don't know how to do any of these things once you have completed the lab, please ask your tutor for help. You should also use these lists of objectives when you are revising for the end of semester exam.

9.3 Walkthrough: The Maxwell-Boltzmann distribution

In the previous lab we looked at how to select random numbers from *uniform* and *normal* distributions. These are useful for many experiments, but in most physical systems the distribution of a particular parameter is more complex. For example, say we want to run a simulation of gas particles in an empty room, at a particular temperature.

In a classical ideal gas in which the particles do not interact with each other except for very brief collisions, the probability that a molecule of mass m has a speed in the range v to $v + dv$ is $P(v)$, where:

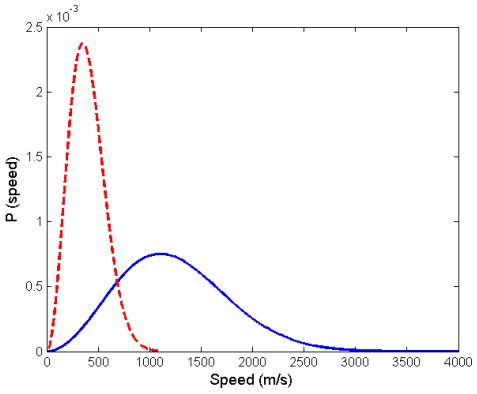
$$P(v) = \left(\frac{m}{2\pi kT}\right)^{3/2} 4\pi v^2 e^{-\frac{mv^2}{2kT}} \tag{9.1}$$

where m is the particle mass (in kg), k is Boltzmann's constant ($1.3806488 \times 10^{-23}$ JK^{-1}) and T is the temperature (in K). This is known as the Maxwell-Boltzmann distribution.

Question 1
Write a function `mb` that takes a vector of speeds, the mass of an atom `m` and the temperature `T` and returns a vector of probabilities as defined by Equation 9.1. Use your function to plot the speed distribution for (a) helium, and (b) argon at room temperature (21°C = 294.15 K). The mass of a helium atom is 6.64648×10^{-27} kg, and the mass of an argon atom is 6.6335×10^{-26} kg. Use a speed range of 0 to 4000 ms^{-1}.

Because the probability that a molecule of mass m has a speed in the range v to $v + dv$ is given by $P(v)dv$, the area under your curve should be equal to one. Check this by summing the output from your `mb` function. *Note: Record the maximum probabilities, you will need them later.*

Argon = red dashed line
Helium = blue solid line



Question 2
Explain what these plots tell you about the speeds of helium and argon atoms.

For two gases at the same temperature the helium atoms move at higher speeds (on average) than the argon atoms (for two gases at the same temperature). Their speeds are also distributed over a wider range.

Question 3
The mean speed of gas molecules in the room is given by:

$$v_{av} = \sqrt{\frac{8kT}{\pi m}} \tag{9.2}$$

and the most probable speed of gas molecules is given by:

$$v_{prob} = \sqrt{\frac{2kT}{m}} \tag{9.3}$$

Calculate the mean speed and most probable speed for helium and argon atoms at $T = 294.15$. Discuss how these relate to your plot in Question 1.

Helium: mean = 1247 m/s
Helium: most probable = 1105 m/s
Argon: mean = 395 m/s
Argon: most probable = 350 m/s

Question 4
What do you expect would happen to these distributions if we reduced the temperature in the room. Run your program again with a temperature $T = 120$ K to test your predictions.

Distributions should peak at lower speeds.

We've now explored the Maxwell-Boltzmann distribution at a statistical level. If we wanted to simulate this system (a gas in a room), we would need to be able to create a set of particles with velocities that follow this distribution. We know how to select random numbers from uniform and normal distributions, how can we do something similar for the Maxwell-Boltzmann distribution (or in fact for any other known distribution)? Perhaps the most straightforward technique is called **rejection sampling**. As you'll see, you've actually come across this already in the previous lab, when we simulated throwing darts at a circle inside a square to estimate the value of π.

9.4 Activity: Rejection sampling

Sampling is the process of obtaining representative examples from a statistical population. In experiments, sampling is used to try to determine the underlying distribution. For example, if you roll a die you have a $\frac{1}{6}$ chance of getting any one of the six numbers. If you are *simulating* a game that involves rolling a single die, you don't just want to know that there is a $\frac{1}{6}$ chance of rolling a 5, you need to know if and when a 5 actually occurs.

To sample from the Maxwell-Boltzmann distribution, we are going to implement a rejection sampling algorithm. Rejection sampling works like this:

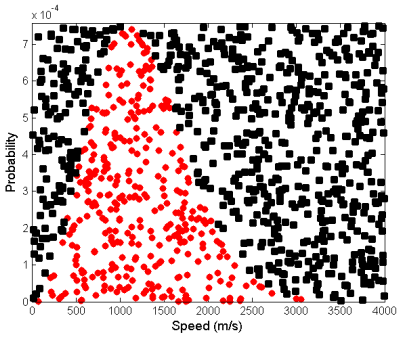
1. Use `rand()` to pick a uniformly distributed random speed v between 0 and v_{max};
2. Use `rand()` to pick a probability p between 0 and P_{max};
3. If $p \leq P(v)$ mark this v as accepted
4. Otherwise mark this v as rejected.

The speeds that are accepted form the final sample, reflecting the underlying distribution they are drawn from.

Question 5
Write a program to implement rejection sampling for the Maxwell-Boltzmann distribution according to the steps above. Assume we have a room of helium gas at $T = 294.15$ K and choose speeds in the range 0 to 4000 ms^{-1}. You can obtain P_{max} from your results in Question 1.

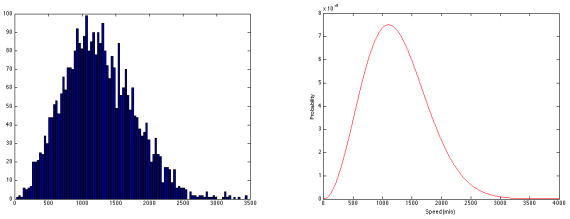
Plot a scatter plot of probability p vs. speed v showing the result for 1000 samples. Plot the accepted samples in red and the rejected samples in black. What do you observe?

It looks like the Maxwell-Boltzmann distribution.



Question 6 (PHYS2911 and PHYS2921)
Optional for PHYS2011 Plot a histogram of the accepted samples and plot with the theoretical distribution (on a separate plot). What happens if you run your program with 10 000 samples? Explain your results.
Hint: 100 bins is a reasonable value for your histogram.

If you use more samples, the histogram approaches the theoretical distribution.



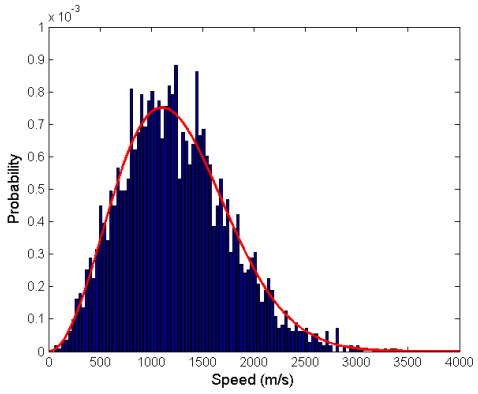
Question 7 (PHYS2911 and PHYS2921)
Optional for PHYS2011

To do a direct comparison of your results with the theoretical distribution, you need a normalising factor. You can calculate this with the following steps:

a) Calculate the area under your histogram (A_{hist}) by multiplying the number of counts in each bin by the bin width.

b) Recall that the area under the theoretical curve is 1. So if we multiply the number in each bin by the normalising factor $1/A_{hist}$, the area under the histogram would be 1. However it is easier simply to multiply the theoretical values by A_{hist} to see if your results match.

Sketch your histogram and matching theoretical curve on the same plot,



Hint: The `hist` *function can be used to return the bins and counts, rather than just plotting them directly. The syntax for this is* `[N,X] = hist(...)` *where N is a vector containing counts and X is a vector containing the bin centres. Run* `help hist` *for more information. You can plot a histogram specifying the bins and counts using* `bar`.

Checkpoint 1:

9.5 Activity: The Metropolis-Hastings algorithm

One of the limitations of rejection sampling is that it is a very inefficient method, particularly in higher dimensional space. This means you have to select a very large number of samples to get a significant sample of accepted values.

The Metropolis-Hastings (MH) algorithm is a Monte-Carlo based method that allows you to sample from *any* arbitrary distribution, as long as you are able to characterize the distribution. The MH algorithm simulates a random walk over the possible outcomes, spending more time in the places where the probability distribution is large. Each step of the walk is a sample from the probability distribution.

The key steps of the MH algorithm are:

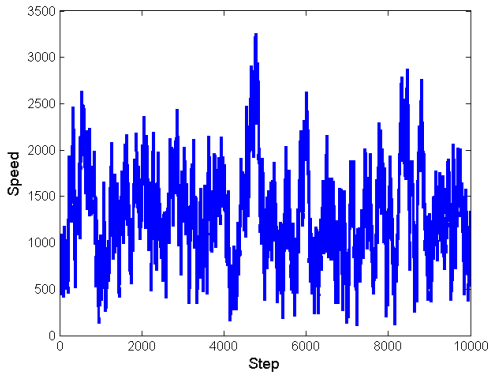
1. Use `rand()` to pick an initial random speed v_1 between v_{min} and v_{max}.
2. Estimate the next speed (v_2) by using a probability that depends only on the current one (v_1). A typical choice for this *proposal distribution* is a normal distribution centered at the current speed.
3. Calculate the *acceptance ratio* α which is defined as which is defined as the ratio of the probability of the next speed over the probability of the current one, i.e. $\alpha = P(v_2)/P(v_1)$. If $\alpha > 1$, replace it by $\alpha = 1$.
4. Use `rand()` to pick a probability p between 0 and 1. If $p \leq \alpha$, the new speed v_2 will be accepted and replace v_1. (Note, it $\alpha = 1$, then it is guaranteed to be accepted.)
5. Record the speed as one of the samples of the probability distribution P.

As before the probabilities $P(v)$ can be calculated from the Maxwell-Boltzmann distribution given in Equation 9.1. It is worth noting that the only place the probability distribution enters the algorithm is in the acceptance ratio α. Because $\alpha = P(v_2)/P(v_1)$ this method doesn't even require the probability distribution to be normalized.

The MH algorithm is a **Markov Chain Monte Carlo** method (MCMC) because the new speed chosen depends only on the current speed.

Question 8
Implement the Metropolis-Hastings algorithm to generate a sample of speeds drawn from the Maxwell-Boltzmann distribution. Assume the gas is helium at 294.15 K. For Step 1, choose $v_1 = 1000\ \text{ms}^{-1}$ instead of a random speed; for step 2, use `v2 = 100*randn(1) + v1` (a normal distribution). Run your simulation for 10 000 steps and plot the speed vs. step number. Does your plot look like what you expected?

Hint: Depending on how you implemented your `mb` *function, you may need to modify it so that it ensures the probability of points with v <= 0 are equal to zero.*

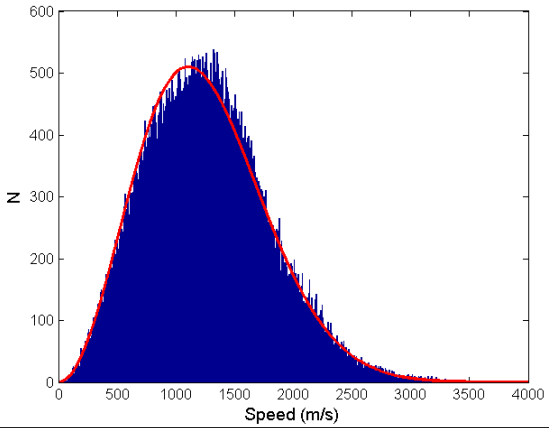


The random walk spends more time close to 1000, which is expected based on the probability distribution.

Question 9

Make a plot of a histogram showing the speed samples over 100 000 steps. How does your plot compare to the probability distribution function?

If students haven't used the hint in Q8, this plot may show negative speeds. You should get them to incorporate the hint.



9.6 PHYS2921: Understanding the algorithm

To better understand what is going on in this algorithm, we can test the effect of making small changes to the algorithm. Modify your code to experiment with each of the following scenarios.

Question 10

Explain what happens if the next proposed speed is always accepted (irrespective of the value of α).

The result is simply a random walk as there is no dependence on the target probability distribution.

Question 11

Describe what happens if the next proposed speed is only accepted if $\alpha > 1$ (if the proposed point is less likely, there is no chance of accepting it).

The algorithm immediately goes to the part of the probability distribution with the highest likelihood and stays there.

Checkpoint 2:

9.7 Further information and references

You can read more about the development of the Monte Carlo method in the post-WWII period in this article:
`http://library.lanl.gov/cgi-bin/getfile?15-13.pdf`

And something about the sociology of publication in the 1950s from this transcript of an interview with Rosenbluth, one of the authors on the Metropolis et al. 1953 paper
`http://www.aip.org/history/ohilist/28636_1.html`

A history of the development of the Metropolis-Hastings algorithm is given in this article:
`http://www.jstor.org/stable/pdfplus/30037292.pdf`