Lecture 8

Generating a non-uniform probability distribution

Discrete outcomes

Last week we discussed generating a non-uniform probability distribution for the case of finite discrete outcomes. An algorithm to carry out the non-uniform probability is to define an array A as follows:

$$A_n = \sum_{j=1}^{n} P_j \tag{1}$$

where $A_0 = 0$. Note that $Z_N = 1$. A_n is just the sum of the probabilities from $1 \to n$. Now, "throw" a random number r that has uniform probability between 0 and 1. Find the value of k, such that $A_{k-1} < r < A_k$. Then the outcome is z_k. The following code should work after initiallizing the array $A[i]$:

```
i=0;
r=random()/m;
found=0;
while(found=0)
{
i=i+1;
if(A[i] > r) found=1;
if (i=N) found=1;
}
outcome=A[i];
```

Continuous Outcomes

In the realm of quantum mechanics the outcome of nearly every experiment is probabilistic. Discrete outcomes might be the final spin state of a system. In the discrete case, the probability for any particular outcome is a unitless number between 0 and 1. Many outcomes are however, continuous. Examples of continuous outcomes include: a particle's position, momentum, energy, cross section, to name a few.

Suppose the continuous outcome is the quantity x. Then one cannot describe the probability simply as the function $P(x)$. Since x is continuous, there are an infinite number of possibilities no matter how close you get to x. One can only describe the randomness as the probability for the outcome x to lie in a certain range. That is, the probability for the outcome to be between x_1 and x_2 $P(x_1 \to x_2)$ can be written as

$$P(x_1 \to x_2) = \int_{x_1}^{x_2} P(x)\, dx \tag{2}$$

The continuous function $P(x)$ can be interpreted as follows: the probability that the outcome lies between x' and $x' + \Delta x$ is $P(x')\Delta x$ (in the limit as $\Delta x \to 0$). The probability function $P(x)$ has units of 1/length, and is refered to as a **probability density**. In three dimensions, the probability density will be a function of $x, y,$ *and* z. One property that $P(x)$ must satisfy is

$$\int P(x) dx = 1 \tag{3}$$

where the integral is over all possible x. In three dimensions this becomes:

$$\int \int \int P(\vec{r}) dV = 1 \tag{4}$$

In quantum mechanics, the absolute square of a particle's state in coordinate space, $\Psi^*(x)\Psi(x)$ is a probability density. The function $\Psi(\vec{r})$ is a probability density amplitude. Likewise, the differential cross section is a probabililty density in θ, with $f(\theta)$ being a probability density amplitude.

How does one produce a non-uniform probability for a continuous outcome from a random number generator that has a uniform probability distribution? We can use the same approach that we did for the discrete case. For the discrete case it was most useful to define A_n:

$$A_n = \sum_{j=1}^{n} P_j \tag{5}$$

Taking the limit to the continuous case, $P_j \to P(x')dx'$, and $A_n \to A(x)$. Suppose x lies between a and b, $(a \leq x \leq b)$. Then we have

$$A(x) = \int_a^x P(x')dx' \tag{6}$$

Note that $A(a) = 0$ and $A(b) = 1$ as before.

We can "throw" a random number with non-uniform probability density $P(x)$ as follows. First "throw" a random number r between zero and one with uniform probability. Then solve the following equation

$$r = A(x) \tag{7}$$

for x. x will be random with the probability density $P(x)$. This was the same method we used before in the case of discrete outcomes. After "throwing" r, the random outcome was the first value n (for A_n) above r.

Another way of understanding this method for the case when $a \leq x \leq b$ is to start with the graph of $P(x)$, which goes from a to b. Then divide the $a \to b$ segment on the x-axis up into N equal pieces of length $\Delta x = (b-a)/N$. Now we have N discrete outcomes, $x_j = a + j\delta x$, where j is an integer and goes from one to N. The probability of each outcome equals the area under the curve in between x and $x + \Delta x$: $P_j = P(x_j)\Delta x$. Now we define A_n as before:

$$A_n = \sum_{j=1}^{n} P_j = \sum_{j=1}^{n} P(x_j)\Delta x \tag{8}$$

which is the equation we used in the discrete case. Taking the limit as $N \to \infty$ yields the continuous case.

Let's do an example. Suppose we want to throw a random number x between zero and two that has a probability that is proportional to x. That is, $P(x) \propto x$, or $P(x) = Cx$. First we need to normalize $P(x)$, that is to find the constant C such that $\int_0^2 P(x)dx = 1$.

$$\begin{aligned} \int_0^2 Cxdx &= 1 \\ C &= \frac{1}{2} \end{aligned}$$

Now we determine $A(x)$:

$$\begin{aligned} A(x) &= \int_0^x \frac{1}{2}x'dx' \\ A(x) &= \frac{x^2}{4} \end{aligned}$$

Now we "throw" a random number r between zero and one with uniform probability. Then set $r = x^2/4$ and solve for x in terms of r:

$$x = 2\sqrt{r} \tag{9}$$

If r has a uniform probability between zero and one, then x will have a (non-uniform) probability density of $x/2$.

The following computer code could be used to generate this non-uniform probability distribution for x:

```
r=random()/m;
x=2*sqrt(r);
```

"Throwing" a Gaussian Probability distribution

A Gaussian probability distribution in one dimension is the following function for $P(x)$:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{10}$$

This probability density distribution is important in physics, because often experimental data will have random errors that follow this "normal" distribution. The average value of x is μ, with a standard deviation about this value of σ. Note, that here, x ranges from $-\infty$ to $+\infty$.

To "throw" a Gaussian probability distribution in x, one would have to solve the following integral

$$A(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{\infty}^{x} e^{-(x'-\mu)^2/(2\sigma^2)} dx' \tag{11}$$

for $A(x)$. As far as I know, there is no analytic solution to this integral. However, there is a nice "trick" to produce a Gaussian probability. If one throws a two-dimensional Gaussian probability distribution, the integral can be solved analytically. I'll show you the result first, then show you how it is derived.

To throw a Gaussian distribution in x, with mean μ and standard deviation σ, do the following. Throw two random numbers, r_1 and r_2, each with a uniform probability distribution between zero and one. The random number x equals $x = \mu + s\sqrt{-2\ ln(1-r_1)}\ sin(2\pi r_2)$. A simple code to carry this algorithm out is:

```
r1 = random()/m;
r2 = random()/m;
r = s * sqrt(-2*ln(1-r1)) ;
theta = 2*pi*r2;
x = mu + r * sin(theta);
```

This algorithm is derived by tossing a two dimensional Gaussian probability in the x-y plane. If we "toss" a random point in the x-y plane that has a Gaussian probability density in x and y about the origin with standard deviation for both x and y as σ, the probability density is

$$P(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \ \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/(2\sigma^2)} \tag{12}$$

The expression $P(x, y)$ is an area density, with units of 1/area. The complete expression for probability is

$$P(x, y)\ dxdy = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \ \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/(2\sigma^2)} dxdy \tag{13}$$

We can "toss" the point with the same probability density using the polar coordinates r and θ. The relationship in polar coordinates between (x, y) and r and θ is

$$\begin{aligned} x &= r\ cos(\theta) \\ y &= r\ sin(\theta) \\ & and \\ dx\ dy &= rdr\ d\theta \end{aligned}$$

Using these equations to transform to polar coordinates yields

$$\begin{aligned} P(x, y)\ dxdy &= \frac{1}{\sigma^2 2\pi} e^{-r^2/(2\sigma^2)} rdr\ d\theta \\ &= \frac{1}{\sigma^2} e^{-r^2/(2\sigma^2)} rdr\ \frac{d\theta}{2\pi} \end{aligned}$$

So if we "throw" r with the probability density (in r) of $P(r) = (r/\sigma^2)e^{-r^2/(2\sigma^2)}$ and θ with a uniform probability density between 0 and 2π, then the point (r, θ) in the x-y plane will have an x-coordinate that has a Gaussian probability distribution. The y-coordinate will also have a Gaussian probability distribution as well. However, you can not use both the x and y points in the same simulation. One does not get two indendent Gaussian probability distributions from the "$r - \theta$ toss".

To obtain r, first throw r_1 with uniform probability between zero and 1. Then r is found by solving

$$\begin{aligned} r_1 &= \frac{1}{\sigma^2}\int_0^r e^{-r'^2/(2\sigma^2)} r \, dr \\ &= 1 - e^{-r^2/(2\sigma^2)} \\ &\mathit{or} \\ r &= \sigma\sqrt{-2 \, ln(1-r_1)} \end{aligned}$$

θ is obtained by throwing r_2 with uniform probability between zero and 1. Then $\theta = 2\pi r_2$. Now that you see the derivation, you can understand why the code:

```
r1 = random()/m;
r2 = random()/m;
r = s * sqrt(-2*ln(1-r1)) ;
theta = 2*pi*r2;
x = mu + r * sin(theta);
```

works. The mean μ is added to the Gaussian spread in the last line.

Random Numbers in ROOT

For those of you using ROOT, it is easy to "throw" a random number with a Gaussian distribution. There is a special function, rand.Gaus(mean,sigma). The following code should work:

```
TRandom3 rand(0); //set seed for random variable
x= rand.Gaus(mu, sigma);
```

Most likely the rand.Gauss function uses the method we described.

More on Scattering, Phase Shifts, and coding

In the last lecture we came up with the formula for the scattering amplitude in terms of the phase shifts. The result for **elastic scattering** is

$$f(\theta) = \sum_{l=0}^{\infty}(2l+1) f_l P_l(cos(\theta)) \tag{14}$$

where

$$f_l = \frac{e^{i\delta_l} sin(\delta_l)}{k} \tag{15}$$

where δ_l are the phase shifts, and $k = p/\hbar$. For the case of elastic scattering the δ_l are real. In our assignment, we only sum up to $l = 1$ since the momentum of the pion is small. In this case, the formula for $f(\theta)$ is

$$f(\theta) = \frac{1}{k}(e^{i\delta_0} sin(\delta_0) + 3e^{i\delta_1} sin(\delta_1) cos(\theta)) \tag{16}$$

Before we go over coding with complex numbers let me discuss where the name phase shift comes from. The δ_l are the shift in phase from the free particle solutions of the Schroedinger equation. In the absence of the potential $V(r)$ ($V(r) = 0$), the solutions to the Schroedinger equation for orbital angular momentum l are the spherical Bessel functions, $j_l(kr)$. In fact, a plane wave traveling in the z-direction expressed in spherical coordinates is given by:

$$e^{ikz} = \sum_{l=0}^{\infty}(2l+1) i^l j_l(kr) P_l(cos(\theta)) \tag{17}$$

where θ is the angle with respect to the z-axis.

For large r, the spherical Bessel function $j_l(kr) \to sin(kr - l\pi/2)/(kr)$. The effect of a real potential $V(r)$ is to cause a phase shift in this large r limit: $sin(kr - l\pi/2 + \delta_l)/(kr)$. To solve for the phase shifts δ_l, one just iterates the Schroedinger equation to large r, like we did for the bound state assignment. However for the scattering calculation, the energy is greater than zero, and the solution oscillates. One can obtain the phase shifts by examining the large r solution to the discrete Schroedinger equation. We will not solve the Schroedinger equation for the δ_l in our assignment 4. I'll give you δ_0 and δ_1 for a particular energy, and you will generate simulated data.

The sign of the phase shift for elastic scattering at low momentum depends on whether the interaction is attractive or repulsive. For an attractive interaction, the "wave function" curves more than in the absence of an interaction. The result is that the phase of the "wave function" ends up ahead of the "free particle" case. Thus, a **positive phase shift** means that the interaction (for the particular value of l) is **attractive**. For a repulsive interaction, the "wave function" curves less than the free particle case, and the phase shift lags. A **negative phase shift** indicates that the interaction is repulsive. I'll demonstrate this on the board for the $l = 0$ partial wave.

The amplitude is complex, so in your code you will need to use complex numbers where needed. In gcc, one need to include <complex.h>:

```
#include <complex.h>
#include <math.h>
```

You will need to declare any complex variables as complex:

```
complex f;
```

Some commands that you might need are cexp(), which is the complex exponential function, and cabs(), which is the complex absolute value squared. For example:

$$\begin{aligned} cexp(x) &\rightarrow e^x \\ cabs(f) &\rightarrow \sqrt{f^* f} \end{aligned}$$

Also, you will need to find the center of mass momentum, since $k = p_{cm}/\hbar$. Now you are ready to write your code for assignment 4. I'll go over any questions you might have during the Wednesday office hours. I'd recommend to first produce the data without any "Gaussian" scatter. Then throw the Gaussian dice to simulate the statistical error in a real experiment for each data point.