Monte Carlo Integration Notes

- The term "Monte Carlo" for simulations using random numbers goes back to late World War II, when von Neumann and Ulam were using random numbers to simulate the behavior of neutrons.
- The idea goes back to 1770's, when Buffon stated (Proc. Paris Acad. Soc., 1773) that π could be approximated in the following way:

  Take a ruled surface, with rules 1 unit apart. Let a needle have length $l < 1$. Throw the needle down numerous times. The value of π can be approximated by the probability that the needle intersects a line.

  To see this, let x be the center of the needle, and let ϕ be the angle that the needle takes with a perpendicular to the rules. Then the "sample" space for all possible positions of the needle is given by: $0 \leq x \leq \frac{1}{2}$, and $\frac{-\pi}{2} \leq \phi \leq \frac{\pi}{2}$, which is a rectangle in the ϕ, x plane with area $\pi/2$.

  What are all the points in this space that correspond to an intersection? If the needle intersects a rule, then a right triangle is formed, where the hypotenuse is the needle, and one leg measures the distance from the rule to the midpoint, x. If we let the hypotenuse (measured from the midpoint) have length r, then $r < \frac{l}{2}$ (Note: we are not considering the case where a needle lies totally on a line- the probability of that would be zero). This implies that $r\cos(\phi) < \frac{l}{2}\cos(\phi)$, and $x = r\cos(\phi)$. Thus we have shown that:

$$x < \frac{l}{2}\cos(\phi)$$

  Therefore, the probability of a needle crossing a line is given by:

$$\frac{\int_{-\pi/2}^{\pi/2} \frac{l}{2}\cos(\phi)\, d\phi}{\frac{\pi}{2}} = \frac{2l}{\pi}$$

  For example, if $l = \frac{1}{3}$, then the probability of the needle crossing the line is $\frac{2}{3\pi}$. Thus, if A is the empirically determined ratio, then $\pi \approx \frac{2}{3A}$
- This idea can be extended to determine area or volume. For example, imagine a dart board hanging on a rectangle. Start throwing darts at random, so that all darts hit the rectangle. The probability that we hit the dart board is given by:

$$\frac{\text{Area of the dart board}}{\text{Area of the rectangle}}$$

  If this quantity is empirically determined to be the value C, then:

$$\text{Area of the dart board} = C \cdot \text{Area of the rectangle}$$

- In each of these cases, it is critical that we have a procedure by which we obtain *random numbers*. We will not go into the details here- we will use Matlab's random number algorithms, and in particular, the generator for a uniform distribution, `rand`.

1. Monte Carlo Integration

There are numerous techniques that go by this name, but the most basic is "integration by rejection", which is a direct analogy to our dart board example. In

this case, if we have a (high dimensional) domain D, we estimate the integral in the following way:

$$\int_D f(\mathbf{x})d\mathbf{x} \approx \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

where the $\mathbf{x}_i$ are chosen at random (with a uniform distribution) in the domain D.

This will involve a very large loop:

```
Est=0;
for i=1:n
  Choose x at random from D
  Est=Est+f(x);
end
Est=Est/n
```

It can be shown[1] that, with $f : R \to R$, we can obtain an estimate of the error of the integral using statistics (we assume x_i has a uniform distribution):

$$\text{Prob}\left(\left|\frac{1}{n}\sum_{i=1}^{n} f(x_i) - \int_a^b f\right| < \frac{\lambda\sigma}{\sqrt{n}}\right) = \frac{1}{\sqrt{2\pi}}\int_{-\lambda}^{\lambda} \mathrm{e}^{-x^2/2}\,dx + O\left(\frac{1}{\sqrt{n}}\right)$$

where σ^2 is the variance, and λ is the confidence. For example, we will estimate the variance by taking the sample variance, and $\lambda = 1.96$ if we want the probability to be 95%. In general,

> The error varies directly with σ and inversely with $\sqrt{n}$. This is known as "the $n^{-1/2}$ law".

The error estimate means that we can do two things to improve the estimate of the integral: Change the number of samples, or decrease the variance.

As another note, it takes the "Law of Large Numbers" to prove the estimate, which means that these estimates hold true in the long run.

In general, one would not probably not use Monte Carlo integration for small (up to perhaps 6 or 7) dimensional integrals.

2. Homework

(1) Use the script file to do the following:
- Approximate the value of π by finding the area of a circle of unit radius:
$$\int_{\text{circle}} dx\,dy$$
- Consider the extra difficulty of this problem: We have to have a uniformly random selection of points within the unit circle.
- Do several trials given a fixed number of points (overall, this is an epoch). For each epoch, determine the mean and standard deviation of the estimates.
- Do this for an increasing number of points.
- Plot the means and standard deviations of your estimates using the `errorbar` command.

[1] Numerical Integration, Davis and Rabinowitz

(2) Write a Matlab script that uses Monte Carlo integration to determine the volume of the "ice cream cone" defined by:

$$\text{Above } z^2 = x^2 + y^2, \text{ and inside } x^2 + y^2 + (z-1)^2 = 1$$

Note that this volume is contained within the rectangular solid:

$$-1 \le x \le 1,\ -1 \le y \le 1,\ 0 \le z \le 2$$

- Before doing Monte Carlo, write down the integral that we're going to estimate using cylindrical coordinates. Have Maple integrate it symbolically.
- Consider when a randomly selected point is "good".
- Consider how you would get an idea of how good the estimate of the integral is.

```
%**********************************************************************
%Script file for Monte Carlo integration to obtain the area of a circle
%  using int_{circle} dx dy

k=5:15;
for n=1:length(k)
    fprintf('On epoch %d\n',n);
    numpts=2^k(n);
    numtrials=50;

    a=(2*rand(numpts,numtrials)-1)+i*(2*rand(numpts,numtrials)-1);
    b=abs(a);

    %Determine how many of these values lie within the unit circle
    G=sort(b);
    for j=1:numtrials
        V=find(G(:,j)<=1);
        Est(j)=length(V)/numpts;
    end
    Est=4*Est;  %Multiply by the area of the box
    H(n)=mean(Est);
    E(n)=std(Est);
    clear a b G Est V
end
errorbar(H,E);
```