

## Godzilla's Guide to Monte Carlo Estimation

Written for CSE 20212 by Prof. Flynn

Jan 9, 2007

Monte Carlo estimation is a numerical technique for computing a number by simulating a random process that yields the number we're interested in. Sometimes we're in a hurry, or lazy, and we don't want to work out the math. Sometimes the math cannot be worked out. The basic idea behind stochastic integration is to use more and more random numbers to accumulate better and better estimates of a quantity.

Let's use random numbers to estimate pi ( $\pi$ ), the ratio of the circumference of any circle to its diameter. We all learned that  $\pi = 3.1416\dots$ ; it's an irrational number. Recall that the area of a circle of radius  $r=1.0$  is  $\pi r^2 = \pi$ . Now a unit-radius circle fits nicely inside a 2x2 square, with a non-overlapping area of  $4-\pi$ . If we had the ability to throw a dart "randomly" at the 2x2 square, so that it is equally likely to land anywhere in the square, it would have a  $\pi/4 = 78.53\dots\%$  chance of landing inside the unit circle. So this suggests a nice estimator for  $\pi$ . Simply generate N random positions inside a 2x2 square in which a unit circle is inscribed. Count the number of times the random positions are **also** inside the circle; call it M. Then  $4M/N$  is an estimate of  $\pi$ . If you let N get larger and larger, you should be able to get arbitrarily close to the true value of  $\pi$ , provided a bunch of conditions (of no interest to us here) are true.

The "random" numbers to be used must be of the "uniform random" type, which means that all positions in the 2x2 square are equally likely. Random number generation is a topic beyond the scope of this handout. Here's a function that returns a random double-precision floating point number between -1 and 1 using the `rand()` function in `cstdlib`:

```
double my_randdouble() {  
    return (2.0 * rand()) / (double(RAND_MAX)+1) - 1.0;  
}
```

Check out p. 256 in Deitel & Deitel if you need a refresher in random numbers in C++.

A position  $(x,y)$  is inside the unit circle at the origin if  $x^2 + y^2 \leq 1$

You should be able to design a simple program that reads the value of N from the standard input stream, computes N random positions in the 2x2 square, determines the fraction of them that are also inside the unit circle, computes the ratio, multiplies it by 4, and reports the result to the standard output stream.

Think about how you might adapt this idea to the estimation of the area under a curve such as  $e^{-x}$  on the interval  $[0,1]$  or  $\sin(x)$  on the interval  $[0, \pi]$ .