

Lab V — The Law of Large Numbers and Monte Carlo Simulation*

Peter Brinkmann

The purpose of this lab is to introduce you to the law of large numbers and Monte Carlo simulation in a hands-on fashion. I am assuming that you have worked through Lab I as well as Project I, available at

<http://www.math.uiuc.edu/iprom/materials.html>.

In particular, you should be able to create, modify, and run simulations. When working on this lab, please work slowly and do one step at a time. If you skip ahead or read on before finishing the task at hand, it may spoil the fun!

1 The law of large numbers

Let's build a simple simulation that repeatedly simulates a random variable and displays the results in a 2-dimensional plot.

Action 1. Open the main IPROM library. Open the model `blank.mdl` and save it under a new name.

Pick a random variable and copy the corresponding block from the library to your blank model. Make sure to pick a random variable whose expectation is easy to compute and whose variance is rather large (for example, a Poisson

*Copyright © 2004, Peter Brinkmann (brinkman@math.uiuc.edu). Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.fsf.org/copyleft/fdl.html>.

random variable with parameter 10 will work nicely). Jot down the expected value μ of your random variable.

$$\mu =$$

Copy the block labeled **Plot** and copy it to your model. Leave some room between the random variable and the plotter. Connect the output of the random variable to the input of the plotter.

If you run the simulation a few times, you should see a rather ragged looking plot. You may want to double-click on the plotter and have it show regular updates.

Action 2. Copy an **Average** block to your model and reroute the output of your random variable through this block. You can accomplish this by dropping the **Average** block on the line connecting your random variable and the plotter.

Run the new simulation a few times. Do you observe any pattern?

Action 3. Increase the number of iterations, i.e., the stop time in the **Simulation parameter** menu, and run the simulation again. Do you observe anything now? If not, keep increasing the number of iterations until you do observe something. Increase the number of iterations again and see what happens. Jot down your observations:

Action 4. Repeat the same experiment with another random variable, i.e., delete your original random variable block and copy a different random variable to your model. Run the experiment again. What do you observe?

Try a few more different random variables and see what happens.

If you have carefully followed the instructions up to this point, you have seen experimental evidence of the strong law of large numbers. Take a moment to contemplate how the following theorem and your observations fit together.

Theorem 1.1 (The strong¹ law of large numbers [Ros02, Section 8.4]). *Let X_1, X_2, \dots be a sequence of independent and identically distributed random variables, each having a finite mean $\mu = E[X_i]$. Then, with probability 1,*

$$\frac{X_1 + X_2 + \dots + X_n}{n} \rightarrow \mu \quad \text{as } n \rightarrow \infty.$$

In other words, the average of X_1, \dots, X_n converges to μ as n tends to infinity.

2 IPROM and the law of large numbers

Much of the functionality of IPROM is based on the strong law of large numbers. Take another look at the very first IPROM model you've seen, `sample.mdl`. Run this model a few times to remind yourself how it works. Ostensibly, this model defines an event E (the value of a fair die is greater than four) and computes an approximation of the probability of E .

You may be surprised to learn that IPROM has no real concept of events or probability, unless you count the simulation of uniformly distributed random variables, which is not really a feature of IPROM but a function of Matlab. Instead, the block that defines the event E simulates the *indicator random variable* I of the event E , i.e.,

$$I = \begin{cases} 1 & \text{if } E \text{ occurs,} \\ 0 & \text{otherwise.} \end{cases}$$

The block that purports to compute the probability of E merely computes the average of repeated evaluations of E . Why does this give us an approximation of the probability of E ?

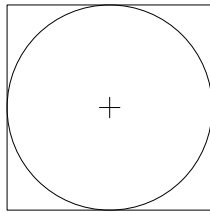
Note that $E[I] = 1 \cdot P\{I = 1\} + 0 \cdot P\{I = 0\} = P(E)$. Now the strong law of large numbers guarantees that the average of repeated evaluations of I will be close to the $E[I] = P(E)$, justifying the labels of the blocks. The block that computes the variance of a random variable works in an analogous fashion, i.e., it actually computes the *sample variance* of its input and relies on the fact that the expectation of the sample variance equals the variance [Ros02, Section 7.3, Example 3a].

¹In case you're wondering, there's also a weak law of large numbers [Ros02, Section 8.2]. It is instructive to ponder the subtle difference between the two theorems. We won't dwell on it in this class, though.

While this may seem like a strangely roundabout way of computing probabilities, this approach makes a lot of sense when dealing with problems that are hard to solve analytically. We will see further applications of this idea in the next section.

3 Monte Carlo simulation

Instead of discussing Monte Carlo methods in general, we will delve right into an example that illustrates some of the main ideas. To wit, we will create an IPROM model that will give us a numerical approximation of the value of π .



Action 5. Consider a disk of radius one inscribed in a square of side length two. Now, imagine that you randomly throw darts at the square, such that the horizontal and vertical coordinates of the points of impact are uniformly distributed. What is the probability of hitting the disk?

$$p =$$

Action 6. Identify a random variable block that is suitable for simulating the coordinates of the points of impact.

Jot down a Matlab expression in terms of the coordinates that defines the event of hitting the disk. Do *not* use the constant π in your expression. Hint: Matlab's `norm` function may be useful here.

Matlab expression:

Action 7. Create an IPROM model that simulates this random experiment and computes an approximation of the probability p . If you don't see how to set this up, take another look at the model `cond.mdl` that you studied in Project I.

Action 8. Modify your model such that the final result is an approximation of π . Run the model a few times in order to see how good the approximation is. You may need to increase the number of iterations to get a satisfactory result. Jot down the approximation provided by your model:

$$\pi \approx$$

You have created a so-called Monte Carlo simulation that finds an approximation of the value of π . Take a moment to appreciate the magic — a random simulation that doesn't know much about probability or geometry computes a probabilistic approximation of one of the fundamental constants of geometry! Moreover, computing the value of π doesn't even seem like a probabilistic problem to begin with. The strong law of large numbers makes it all happen.

Early examples of Monte Carlo methods include Fermi's study of the neutron as well as simulations required for the Manhattan project, although the actual term was only coined in 1949. Today, Monte Carlo simulations are an important tool in many branches of science and engineering, including

- nuclear reactor design,
- econometrics,
- Dow-Jones forecasting,
- VLSI design,
- and many more.

If you want to learn more about Monte Carlo methods, Wikipedia is a good place to start (http://en.wikipedia.org/wiki/Monte_Carlo_method).

Action 9 (Challenge problem). Find a Monte Carlo approximation of Euler's number e and build an IPROM model that implements this approximation. Hint: Remember the matching problem [Ros02, Section 2.5, Example 5m].

References

[Ros02] Sheldon Ross. *A first course in probability*. Prentice Hall, Upper Saddle River, NJ 07458, sixth edition, 2002.