An Exposition on Monte Carlo Integration: Approximating Integrals with Random Sampling

Owen Terry

April 2023

1 Introduction

Monte Carlo methods, loosely, are algorithms that use random sampling to give approximate results to mathematical problems. The problems which we use Monte Carlo on typically have exact solutions that can be found with deterministic methods, but any such method requires significantly more computation than a Monte Carlo approach does to get a similarly good approximation.

This paper will focus on the application of Monte Carlo methods to integration. We'll begin by introducing the necessary ideas from probability theory — we want to build both a formal and an intuitive understanding of what random variables are and how we work with them. We'll then give an idea of how Monte Carlo methods work in general, and apply this to integration of single-variable functions. We'll write a program that actually does Monte Carlo integration to demonstrate how effective it is, then we'll move on to integration in multiple dimensions. We'll conclude by showing why we would ever prefer to use Monte Carlo over deterministic integration techniques: when we start integrating in many dimensions, deterministic approaches become quite a bit more difficult, while Monte Carlo does not.

2 Basics of probability theory

The Monte Carlo methods we're going to look at involve defining random variables and then generating them over and over. Before we get into a discussion of Monte Carlo, we need to know three things:

- What is a random variable?
- What are some of their relevant properties?
- How do we generate them?

2.1 Random variables

A random variable can be either **discrete** or **continuous**. In either case, when we define a random variable, the idea is that we know the set of possible values that the variable can take on, and the distribution of probability among these possible values. Continuous random variables are more relevant to the Monte Carlo methods we'll be focusing on, but discrete random variables are more intuitive, so that's where I want to start. Once we have a good understanding of random variables as discrete, it'll be easier to talk about them as continuous.

Definition 1. A **discrete random variable** is a $2 \times n$ table of real numbers,

$$X = \begin{pmatrix} x_1 & x_2 & ... & x_n \\ p_1 & p_2 & ... & p_n \end{pmatrix},$$

that satisfy the following conditions:

1. $x_1 < \cdots < x_n$,
2. $p_i > 0$ for all p_i, and
3. $p_1 + \cdots + p_n = 1$.

The set $\{x_1, \cdots, x_n\}$ is the set of possible values X can take on. The first condition listed above ensures that each table denotes a unique random variable – the order in which we list the x_i has no impact on the nature of X, so we make sure that there's only one way we're allowed to order each set of x_i. This condition also ensures that no two elements of the set are the same.

Each p_i represents the probability of X evaluating to the corresponding x_i. Symbolically, $P(X = x_i) = p_i$. (The notation $P(a)$ represents the probability of a being true.) The second condition makes it so that we can't have any negative probabilities – what would a negative probability even mean? We also can't have any $p_i = 0$, because if this were the case, the corresponding x_i would not be a possible value of X, so we would simply exclude it from the table.

Finally, we want the probability of a tautology, i.e. something guaranteed to happen, to be 1. It is tautological that X will evaluate to an element in the set of x_i, so the third condition ensures that this has a probability of 1.

Example 1. We can model a die roll with a discrete random variable:

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}.$$

This table tells us that X can evaluate to any integer 1-6, and that it has an equally likely chance, 1 in 6, of evaluating to each. You can check for yourself that the table satisfies our three conditions, and that something would seem very strange if it didn't.

Definition 2. A **continuous random variable** is defined by a "probability density function" (pdf) on some real interval, $p : (a, b) \subset \mathbb{R} \to \mathbb{R}$, satisfying the following conditions:

1. $p(x) > 0$ for $a < x < b$, and
2. $\int_a^b p(x)dx = 1$.

The interval (a, b) represents the possible values X can take on, and the integral of the pdf on some subinterval of (a, b) represents the probability of X taking on a value in that subinterval. $P(a' \leq X \leq b') = \int_{a'}^{b'} p(x)dx$. We can think of the two conditions here as corresponding to conditions 2 and 3 of discrete random variables. For the first, we want each element of our set of possible values to be something that X can actually evaluate to. For the second, we want the probability of all possible events combined to equal 1.

Example 2. The graph below depicts a "normal distribution," a pdf commonly found in nature (for reasons which I encourage you to explore on your own). This particular normal distribution models the distribution of IQ among the world population. For any given range of IQs – 90 to 110, say – we can find the probability of a person's IQ landing in that range by calculating the area under the curve inside that range.

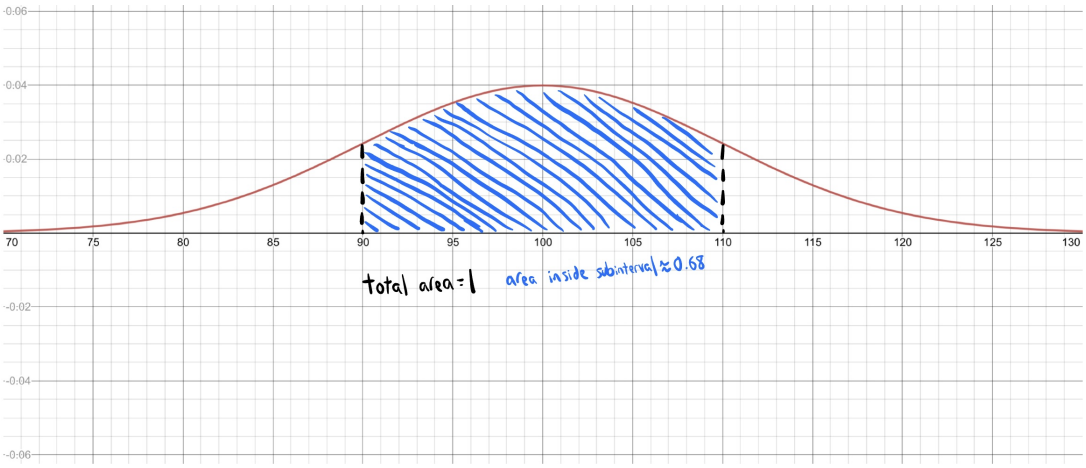

Figure 1: For this pdf, $P(90 \leq X \leq 110) = \int_{90}^{110} p(x)dx \approx 0.68$.

Remark 1. Notice that, because $P(X = a') = P(a' \leq X \leq a') = \int_{a'}^{a'} p(x)dx = 0$, the probability of X evaluating to any particular value in the interval is 0. This may be counterintuitive – one would think that a probability of 0 must represent something that is impossible. But think of the alternative: if each possible value, on a range of infinitely many possible values, had a positive probability, then we risk sum of the probabilities becoming infinite, when we want it to be 1. From another perspective, because there are infinite possible values X can take on, each singular value should be "infinitely unlikely," i.e. having a probability of 0.

We also might be tempted to erroneously think of $p(x_0)$ as the probability of X evaluating to x_0. A better way to think of this is as a measurement of "relative probability." If $p(x_0) = 3$ and $p(x_1) = 6$, then X is twice as likely to evaluate to x_1 as it is to evaluate to x_0. But still, $P(X = x_0) = P(X = x_1) = 0$.

Remark 2. When we refer to a variable with "uniform probability density," this means that the variable's probability density function is constant. For X defined over (a, b), we will have that $p(x) = \frac{1}{b-a}$; you can check that this is the only constant function such that $\int_a^b p(x)dx = 1$.

2.2 Independence, expected values, and variance

Independence is a concept that most readers will already be familiar with intuitively, if not formally. Informally, two variables are independent if one variable taking on a certain value does not change the probability distribution of the other variable. For example, two coin flips will be independent of each other — regardless of what we get when we flip the first coin, we still have a 50/50 chance of heads or tails when we flip the second.

Definition 3. Two discrete random variables X and Y are independent if

$$P(X = x, Y = y) = P(X = x)(Y = y)$$

for all x and y. Similarly, continuous random variables X and Y, defined over (a, b) and (c, d) respectively, are independent if

$$P(a' < X < b', c' < Y < d') = P(a' < X < b')P(c' < Y < d')$$

for all $a \leq a' \leq b' \leq b$ and $c \leq c' \leq d' \leq d$.

That is, we can find the probability of both X and Y taking on any given values or landing within two specified subintervals by multiplying the probability of each one taking on its given value or landing within its own subinterval. When we work with multiple variables in this paper,

assume they are independent unless it is stated otherwise.

We can think of the **expected value** of a random variable as the average result we'd get if we generated it over and over. The definition varies slightly between the discrete case and the continuous case, but the idea is the same.

Definition 4. The expected value of a discrete random variable X is

$$E(X) = \sum_{i=1}^{n} x_i p_i.$$

Make sure this makes sense – we are weighing each x_i by its probability. For example, if we had a random variable with a 75% chance of being 10 and a 25% chance of being 20, our expected value would be $E(X) = 10 \cdot 0.75 + 20 \cdot 0.25 = 12.5$. And naturally, if you were generating this random variable hundreds of times, you'd expect the average outcome to get pretty close to 12.5.

The continuous case is pretty much the same, but we have to change the formula so that we're integrating rather than adding.

Definition 5. The expected value of a continuous random variable X defined on (a, b) is

$$E(X) = \int_a^b xp(x)dx.$$

The integration makes this definition a little harder to grasp intuitively. It might be easier if we think about it through a comparison to the discrete case: we take each possible value in the interval, weigh it by its relative probability, and add them all up.

A third type of expected value we want to be comfortable with is the expected value of a function $f(X)$ of a continuous random variable X. But first, we need one more relevant definition.

Definition 6. The cumulative probability distribution, or cdf, of a continuous random variable X is given by $c(x) = P(X \leq x)$. That is, it represents the probability that X evaluates to a number less than or equal to some x.

From this, it quickly follows that $c(x)$ is equal to the area under the pdf curve to the left of x: $c(x) = \int_a^x p(t)dt$. This can be easily visualized. And from here, we can apply the Fundamental Theorem of Calculus to see that $p(x) = c'(x)$: the derivative of the cdf is the pdf.

Now we can state and prove what we call Law of the Unconscious Statistician, which tells us the expected value of a function of a random variable, $f(X)$. The idea here is that if X is a random variable, so is $f(X)$. As before, there are two formulas, one discrete and one continuous. This time, though, I'm going to jump straight into the continuous case.

Theorem 1. The Law of the Unconscious Statistician (LOTUS): If X is a continuous random variable defined on (a, b) with probability density function $p(x)$, then the random variable $f(X)$ has expected value

$$E(f(X)) = \int_a^b f(x)p(x)dx.$$

Proof. First, for reasons we will see later, we need to find a formula for the derivative of an inverse function. Suppose f is invertible, so that we have $y = f(x)$, and $x = f^{-1}(y)$. If we take the first

equation and differentiate both sides with respect to y, we get

$$\begin{aligned}
1 &= f'(x)\frac{dx}{dy} \\
\frac{dx}{dy} &= \frac{1}{f'(x)} \\
\frac{dx}{dy} &= \frac{1}{f'(x)} \\
\frac{dx}{dy} &= \frac{1}{f'(f^{-1}(y))} \\
\frac{d}{dy}f^{-1}(y) &= \frac{1}{f'(f^{-1}(y))} \\
dx &= \frac{1}{f'(f^{-1}(y))}dy,
\end{aligned}$$

the first line following from the chain rule, and the rest being basic algebraic manipulation. The last two lines both follow from the third-to-last; we write them both because we'll use both. Now, from the last line above it immediately follows that

$$\int_a^b f(x)p(x)dx = \int_{f(a)}^{f(b)} yp(f^{-1}(y))\frac{1}{f'(f^{-1}(y))}dy$$

by substitution. So we want to show that the right-hand expression equals the expected value.

Let's consider the cdf of y. We have

$$\begin{aligned}
c(y) &= P(f(X) \leq y) \\
&= P(X \leq f^{-1}(y)) \\
&= c(f^{-1}(y)),
\end{aligned}$$

which is the cumulative probability distribution of x. Now, recalling that the derivative of the cdf is the pdf, let's differentiate both sides.

$$\begin{aligned}
c'(y) = p(y) &= \frac{d}{dy}c(f^{-1}(y)) \\
&= p(f^{-1}(y))\frac{d}{dy}(f^{-1}(y)),
\end{aligned}$$

the last step coming from the chain rule. Notice that the last bit is exactly what we worked out a formula for! We can substitute to find

$$p(y) = \frac{p(f^{-1}(y))}{f'(f^{-1}(y))}.$$

Now, if we recall the following equation we found earlier, we can substitute $p(y)$ to show what we want to show.

$$\begin{aligned}
\int_a^b f(x)p(x)dx &= \int_{f(a)}^{f(b)} yp(f^{-1}(y))\frac{1}{f'(f^{-1}(y))}dy \\
\int_a^b f(x)p(x)dx &= \int_{f(a)}^{f(b)} yp(y)dy \\
\int_a^b f(x)p(x)dx &= E(Y) \\
\int_a^b f(x)p(x)dx &= E(f(X)).
\end{aligned}$$

□

Remark 3. The name gives us a hint as to how vital this law is – it's called what it is because statisticians often tend to think of it as the very definition of expected value, even though it isn't.

One important fact about expected values is that they are linear maps: they respect addition and scaling. It's hard to do much work with expected values without establishing this. As with the LOTUS, I will restrict the proof to the continuous case.

Lemma 1. Expected values respect addition. $E(X+Y) = E(X) + E(Y)$.

Proof. Suppose continuous random variables X and Y are defined on (a, b) and (c, d), respectively, and let the probability density function of $X + Y$ be $p(x, y)$. We have that

$$\begin{aligned} E(X+Y) &= \int_c^d \int_a^b (x+y)p(x,y)dxdy \\ &= \int_c^d \int_a^b xp(x,y)dxdy + \int_c^d \int_a^b yp(x,y)dxdy. \end{aligned}$$

We can now take x and y outside of the inner integral, rearranging the order of integration as necessary, to get

$$E(X+Y) = \int_a^b x \int_c^d p(x,y)dydx + \int_c^d y \int_a^b p(x,y)dxdy.$$

Finally, because $p(x)$ and $p(y)$ each have integrals equal to 1 over their full range, we get

$$\begin{aligned} E(X+Y) &= \int_a^b xp(x)dx + \int_c^d yp(y)dy \\ &= E(X) + E(Y). \end{aligned}$$

□

Lemma 2. Expected values respect scaling. For any real number c, $E(cX) = cE(X)$.

Proof. Using LOTUS, because cX is a function of X, we have that

$$\begin{aligned} E(cX) &= \int_a^b cxp(x)dx \\ &= c\int_a^b xp(x)dx \\ &= cE(X). \end{aligned}$$

□

Another property of expected values we want to be comfortable with is **variance**. When we work with a random variable, we need an idea of how good a job $E(X)$ is doing at predicting any given instance of X. If X could take on any value between zero and a million with uniform probability, for example, then $E(X)$ would be a rather poor predictor. But if X was only defined on $(0, 1)$, say, and its pdf had a steep spike around the expected value and fell off quickly on either side, then we would be much more confident that the next time we generate an instance of X, it will be pretty close to $E(X)$.

Definition 7. The variance of a random variable X is $\mathrm{Var}(X) = E([X - E(X)]^2)$.

In words, this is the expected value of the squared difference between X and $E(X)$. So if X tends to be close to $E(X)$, then the difference will tend to be small, so variance will be small. And the reverse is true when X tends to be far from $E(X)$. Variance thus gives an idea of how spread out a random variable is.

Lemma 3. Variance can alternatively be expressed as $E(X^2) - E(X)^2$. This is sometimes a more convenient formula.

Proof. This follows pretty easily from the definition of variance and linearity of the expected value. We have

$$\begin{aligned}\operatorname{Var}(X) &= E([X - E(X)]^2) \\ &= E(X^2 - 2XE(X) + E(X)^2) \\ &= E(X^2) - 2E(X)E(X) + E(X)^2 \\ &= E(X^2) - 2E(X)^2 + E(X)^2 \\ &= E(X^2) - E(X)^2.\end{aligned}$$

□

Like with expected values, we want to know what happens when we add or scale inside of a variance function. Variance respects addition the same way expected value does (lemma 4), but when we scale, we find something new (lemma 5).

Lemma 4. Variance respects addition. $\operatorname{Var}(X + Y) = \operatorname{Var}(X) + \operatorname{Var}(Y)$.

Proof. Here, we start from the second formulation of variance, which we just established.

$$\begin{aligned}\operatorname{Var}(X + Y) &= E((X + Y)^2) - (E(X + Y)^2) \\ &= E(X^2 + Y^2 + 2XY) - (E(X) + E(Y))^2 \\ &= (E(X^2) + E(Y^2) + 2E(XY)) - (E(X)^2 + E(Y)^2 + 2E(X)E(Y) \\ &= E(X^2) - E(X)^2 + E(Y^2) - E(Y)^2 + 2(E(XY) - E(X)E(Y)) \\ &= \operatorname{Var}(X) + \operatorname{Var}(Y) + 2(E(XY) - E(X)E(Y)) \\ &= \operatorname{Var}(X) + \operatorname{Var}(Y),\end{aligned}$$

the last step following from the fact that X and Y are independent. □

Lemma 5. $\operatorname{Var}(cX) = c^2 \operatorname{Var}(X)$.

Proof. This also follows quickly from the definition and linearity of expectation:

$$\begin{aligned}\operatorname{Var}(cX) &= E([cX - E(cX)]^2) \\ &= E([cX - cE(X)]^2) \\ &= E(c^2[X - E(X)]^2) \\ &= c^2 E([X - E(X)]^2) \\ &= c^2 \operatorname{Var}(X).\end{aligned}$$

□

This property of variance is going to be relevant when we talk about the law of large numbers.

2.3 The Law of Large Numbers (LLN)

This is a fundamental concept in probability in general, and is very important for Monte Carlo in particular because it allows us to grow more confident in our approximations as we add more data points to our random sample. Intuitively, the idea is that the more times we generate a random variable, the closer the average value will be to the expected value.

There are actually two variants of this law: strong and weak. We'll just be using the weak version. But before we formally state and prove it, we need to establish a definition and two lemmas.

Definition 8. Convergence in probability. A sequence of random variables $X_1, X_2, ..., X_n$ converges in probability to a random variable X if

$$\lim_{n\to\infty} P(|X_n - X| \geq \epsilon) = 0$$

for any $\epsilon > 0$. This is written as $X_n \xrightarrow{p} X$. What this means is that, if we pick an X_n with a big enough n, we can make it arbitrarily likely that X_n is within any arbitrarily small distance of X.

The Law of Large Numbers deals with convergence in probability. Now we're almost ready to look at it – we just need a few results that will make the proof easier.

Lemma 6. Markov's inequality. If X is a non-negative continuous random variable, and a is a positive real number, then

$$P(X \geq a) \leq \frac{E(X)}{a}.$$

Proof. First, because X is non-negative, we have

$$E(X) = \int_{-\infty}^{\infty} xp(x)dx = \int_{0}^{\infty} xp(x)dx.$$

Now, for $a > 0$, we have

$$\begin{aligned} E(X) &\geq \int_{a}^{\infty} xp(x)dx \\ &\geq \int_{a}^{\infty} ap(x)dx, \end{aligned}$$

the second step coming from the fact that $x \geq a$ on the region we're integrating. From this we find that

$$\begin{aligned} E(X) &\geq a\int_{a}^{\infty} p(x)dx \\ E(X) &\geq aP(X \geq a) \\ P(X \geq a) &\leq \frac{E(X)}{a}. \end{aligned}$$

□

Lemma 7. Chebyshev's inequality. For continuous random variable X and any $\epsilon > 0$,

$$P(|X - E(X)| \geq \epsilon) \leq \frac{\mathrm{Var}(X)}{\epsilon^2}.$$

Proof. We will now apply Markov's inequality to variance. Define $Y = (X - E(X))^2$. Notice that Y is non-negative, because it is a square, so Markov's inequality can be used. For any ϵ^2, we have

$$P(Y \geq \epsilon^2) \leq \frac{E(Y)}{\epsilon^2}.$$

Notice that $E(Y) = \mathrm{Var}(X)$. Substituting gives us

$$\begin{aligned} P((X - E(X))^2 \geq \epsilon^2) &\leq \frac{\mathrm{Var}(X)}{\epsilon^2} \\ P(|X - E(X)| \geq \epsilon) &\leq \frac{\mathrm{Var}(X)}{\epsilon^2} \end{aligned}$$

for any $\epsilon > 0$. □

The Law of Large Numbers is going to follow from this inequality.

Theorem 2. The Weak Law of Large Numbers. Let $Y_1, ..., Y_n$ be independent and identically distributed random variables, with $E(Y_i) = \mu$. Let X_n be the average of this sequence of Y_i:

$$X_n = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

When this is the case, X_n converges in probability to μ: $X_n \xrightarrow{p} \mu$.

Proof. First, we'll show that X_n has the same expected value as all the Y_i.

$$\begin{aligned} E(X_n) &= E(\frac{1}{n}\sum_{i=1}^{n} Y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} E(Y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mu \\ &= \frac{1}{n}\mu n \\ &= \mu. \end{aligned}$$

Next, we'll show that $\mathrm{Var}(X_n)$ is less than the variance of the Y_i by a factor of n.

$$\begin{aligned} \mathrm{Var}(X_n) &= \mathrm{Var}(\frac{1}{n}\sum_{i=1}^{n} Y_i) \\ &= \frac{1}{n^2}\,\mathrm{Var}(\sum_{i=1}^{n} Y_i) \\ &= \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}(Y_i) \\ &= \frac{1}{n^2}\,\mathrm{Var}(Y_i)n \\ &= \frac{\mathrm{Var}(Y_i)}{n}. \end{aligned}$$

Finally, we can plug these results into Chebyshev's inequality.

$$P(|X_n - \mu| \geq \epsilon) \leq \frac{\mathrm{Var}(Y_i)}{n\epsilon^2}.$$

When we take the limit of this equation as n goes to infinity, we get what we wanted:

$$\begin{aligned} \lim_{n\to\infty} P(|X_n - \mu| \geq \epsilon) &\leq \lim_{n\to\infty} \frac{\mathrm{Var}(Y_i)}{n\epsilon^2} \\ \lim_{n\to\infty} P(|X_n - \mu| \geq \epsilon) &\leq 0 \\ \lim_{n\to\infty} P(|X_n - \mu| \geq \epsilon) &= 0. \end{aligned}$$

Thus $X_n \xrightarrow{p} \mu$. □

Remark 4. This proof assumes that our Y_i have finite variance, which is not always the case. The law holds for infinite-variance variables too, but proving it is more involved. Establishing the finite variance case is sufficient for our purposes.

2.4 Generating random variables

We understand how to define random variables now, and we know about their important properties. But now, a clever reader might be wondering, how do we actually *do* anything with a random variable? We can say that X has 10 possible outcomes, each with a 10% chance of occurring, but how do we get instances of X and make sure that there's actually an equal probability of it evaluating to any of the possible outcomes every time we generate one? Right now, random variables are just a mathematical concept we've described – we have no method of putting them out into the real world.

- **Idea 1: Use physical objects.**

A naive approach would be to use physical objects, like coins or dice. If we have a discrete random variable with 6 equally probable outcomes, a fair die will generate instances of the variable just fine. But this approach doesn't generalize to the problems we want it to. For one thing, if our variable is sufficiently complicated — perhaps involving many possible outcomes, each of which has a different probability — it becomes quite hard to find a physical object that will do the trick. For another, even if we're working with something simple like a 50/50 variable, this method is inefficient. If we want to generate it 1000 times, we need to flip a coin 1000 times.

- **Idea 2: Draw slips.**

Another approach is to fill a hat with slips of paper, writing a number on each slip, then picking one out at random. We can then replace the slip, mix them around, and pick again. Each time we pick, we write down the number we drew, creating what we call a **table of random numbers**.

This is an improvement — we can fill a hat with 100 slips of paper much more easily than we can find something like a 100-sided die. This also allows us to give different probabilities to different outcomes, at least to some extent. For example, if we wanted to have a 41% chance of $X = 1$ and a 59% chance of $X = 2$, we could fill a hat with 41 1-slips and 59 2-slips.

But this method has its flaws as well. First, the more precision we want in our probabilities, the more slips we need. We'd need 1000 slips, for example, to have a 41.2% chance of 1 and a 58.5% chance of 2. Also, this approach is prone to physical error. You might not mix the slips well enough each time you draw, or some slips might stick to the side of the hat and be less likely to be drawn. In order to ensure that a table is reasonably close to truly random, we have to run statistical tests on it, which I won't describe here, and even then we know that there will be at least a little deviation from the ideal. Finally, and most irritatingly, tables of random numbers are ineffective for computer programs. If we want to store a table large enough to get a reasonable degree of precision, we end up substantially slowing down the speed of computation. Tables of random numbers are more effective for pencil-and-paper calculations, which is a rather big issue.

- **Idea 3: Use randomness in nature.**

So we want to be able to generate random numbers in such a way that we can work with them on a computer. In one such method, we have the computer observe something in the outside world that is unpredictable enough for us to consider it random. For example, if our computer can access the level of noise in a vacuum tube, we might be able to find some threshold and some time interval such that there is a 50/50 chance at any given time that the noise level exceeds the threshold within the time interval. We can define 0 as being below the threshold and 1 as being above. If we take n measurements, we give ourselves 2^n possible values for our random variable.

Like the previous method, tests must be carried out on the results we get here to ensure that it's actually random. In this case, though, we actually need to test regularly, because any small physical change to the system being observed could throw off the randomness. Also, it is sometimes useful to be able to generate the same sequence of random numbers multiple times, and this strategy does not let us do that.

- **Idea 4: Pseudorandom numbers.**

So we come to our final approach: pseudorandom numbers, which are in fact not random at all, but simply made to seem random by being the outputs of sufficiently difficult-to-predict algorithms. We start with a seed number γ_0, which can either be chosen using the physical method described above or just given. We plug γ_0 into an algorithm, giving us γ_1. We then plug γ_1 into the same algorithm, giving us γ_2, and so on. The list of γ_i is our list of random numbers. Again, we need to use statistical tests to check how random the generated list will appear. In addition to mostly avoiding the possible errors that relying on physical outcomes can create, this type of method allows us to repeat a sequence: we just start with the same seed number.

Example 3. An example of such an algorithm is the mid-square method. Note that this is not actually a very good algorithm at all – it tends to produce too many small numbers. I'm just choosing it because it illustrates our point nicely. We first fix γ_0 as (let's say) some 4-digit decimal. We then square it, getting an 8-digit decimal, and take the middle 4 digits. This is γ_1. We repeat this same process over and over.

So if $\gamma_0 = 0.5706$, for instance, then $\gamma_0^2 = 0.32558436$, so $\gamma_1 = 0.5584$. Now, $\gamma_1^2 = 0.31181056$, so $\gamma_2 = 0.1810$, and so on.

A problem we might encounter in any pseudorandom algorithm is that our sequences are necessarily periodic. Because any given γ_i will be followed by the same γ_{i+1}, once we loop back around and produce a value that we've already produced, we know exactly what will follow. Effective pseudorandom number generators create sequences with such long periods that we don't have to worry about looping back around.

So pseudorandom number generators have pitfalls just like the other methods we've described, but they're more manageable. As such, many random number generators that actually get used in real life apply pseudorandom techniques, including Java's Math.random() method, which we will employ later when we do our own Monte Carlo integration.

Now that we see that there are ways to effectively generate many instances of a random variable, we can get started with Monte Carlo, where we apply these methods.

3 Monte Carlo methods

Before we get into integration, I want to give an idea of what Monte Carlo itself is. It's not a unified method, so there isn't a formal definition of "how to do Monte Carlo," or any one algorithm to follow. But there's a general strategy that applications of the Monte Carlo technique tend to use, along these lines:

1. First, define a random variable. The set of possible values of this variable should be a domain relevant to the problem we're trying to solve.
2. Generate a bunch of instances of this random variable, using some method of random number generation. Record all these values.
3. Do some calculations using information about the values you got.

Example 4. Let's take a look at how we might estimate the value of π using a Monte Carlo method. Begin by defining a continuous random variable X on the square $[-1, 1] \times [-1, 1]$, with a uniform probability distribution. Next, generate a large number of instances of this variable, i.e. a bunch of random points in the square. For each point, compute its distance from the origin. If $d \leq 1$, then the point is inside a circle of area π.

We know that all points are in a square of area 4. So if we take the ratio of points inside the circle to total points, we should approximate $\pi/4$. If we multiply this approximation by 4, we approximate π.

Notice that the more precision you want, the more data points you need. For example, if we generated 100 random points, the best approximation we could get is $\pi \approx 31/100 = 3.1$. And it's not too unlikely that we end up with something like $\pi \approx 2.8$ — this would happen if just 3 more points fell outside the circle than expected. If we want an approximation that gives us more

significant figures than this, then, we need orders of magnitude more data points. So we can see why it's important to have an efficient method of generating random variables.

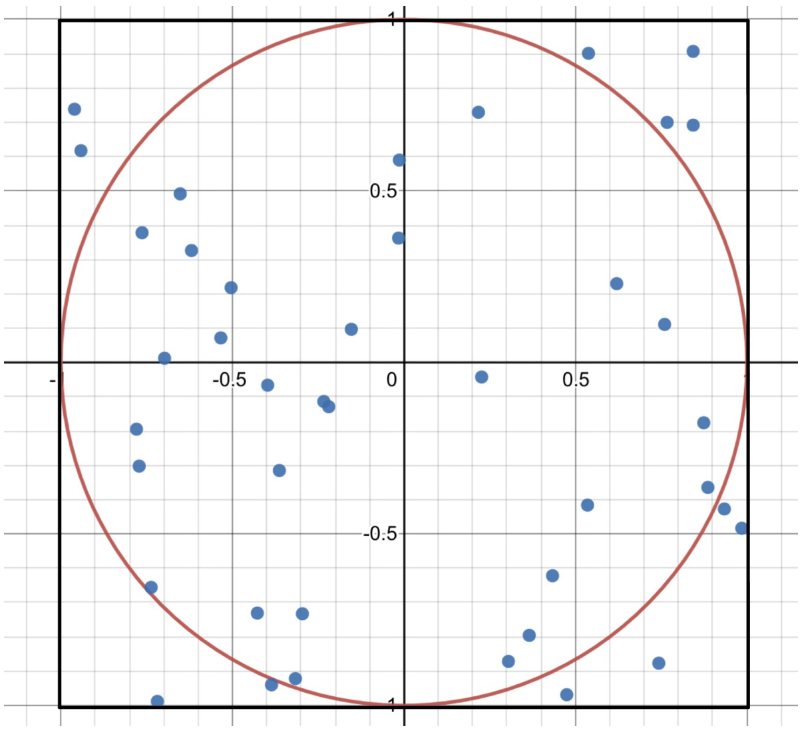

Figure 2: According to this Monte Carlo approximation, $\pi \approx 4 \cdot \frac{29}{41} = 2.2829$.

Also notice that this algorithm isn't unique to approximating π; we can use it to approximate the area of any shape. Simply pick a rectangle that fully surrounds the shape, find the rectangle's area, then generate a bunch of points and find the ratio of points in the shape to total points. This approximates the ratio of the shape's area to the rectangle's area.

4 Monte Carlo integration in 1 dimension

Let's take this general idea of using a random sample to approximate a real value and see how we can apply it to the integration of a single-variable real function, $f : \mathbb{R} \to \mathbb{R}$. We want to approximate

$$F = \int_a^b f(x)dx.$$

Here's the approach: we make a very poor approximation of this integral by taking the area of a rectangle whose width is the interval, $b - a$, and whose height is $f(x)$ at some random x in the interval. We then make many more very poor approximations, each one at a new random x, and average the results. This gives us what we call the **Monte Carlo estimator**.

Definition 9. If we have a function $f : (a, b) \to \mathbb{R}$ and a sequence of identical continuous random variables $X_1, ..., X_N$ with uniform probability density on (a, b), then the Monte Carlo estimator of $F = \int_a^b f(x)dx$ is

$$\langle F^N \rangle = \frac{b-a}{N} \sum_{i=1}^{N} f(X_i).$$

The fact that we're using this $\langle F^N \rangle$ to approximate F should indicate that its expected value is probably F itself. This can be proved pretty easily. But that's not all — $\langle F^N \rangle$ actually converges

in probability to F, meaning that we can be arbitrarily confident that we are arbitrarily close to F if we just take a big enough N.

Theorem 3. The Monte Carlo estimator converges in probability to the integral it's estimating. $\langle F^N \rangle \xrightarrow{p} F$.

Proof. This is a result of the Law of Large Numbers. Let's first rearrange our definition so that it lines up with the LLN:

$$\frac{\langle F^N \rangle}{b-a} = \frac{1}{N} \sum_{i=1}^{N} f(X_i).$$

Now, from the LLN it immediately follows that

$$\frac{\langle F^N \rangle}{b-a} \xrightarrow{p} E(f(X_i)).$$

We can multiply both sides by $(b-a)$ and use LOTUS to find

$$\langle F^N \rangle \xrightarrow{p} (b-a) \int_a^b f(x)p(x)dx.$$

Now, recall that $p(x) = \frac{1}{b-a}$ for variables with uniform probability density. This gives us

$$\langle F^N \rangle \xrightarrow{p} (b-a) \int_a^b \frac{f(x)}{b-a} dx$$

$$\langle F^N \rangle \xrightarrow{p} \int_a^b f(x)dx$$

$$\langle F^N \rangle \xrightarrow{p} F.$$

□

There are a few ways to approach this more intuitively to convince ourselves that $\langle F^N \rangle$ should converge in probability to the integral it's estimating. One thought is that, if we "flatten out" the function we're integrating, so that it's a constant function which evaluates to the average value of $f(x)$ on (a, b), the area under the curve will be the same as it was before we flattened it. The estimator will approach the area of this rectangle as we add more and more randomly chosen rectangles to the formula, so it should also approach the actual integral.

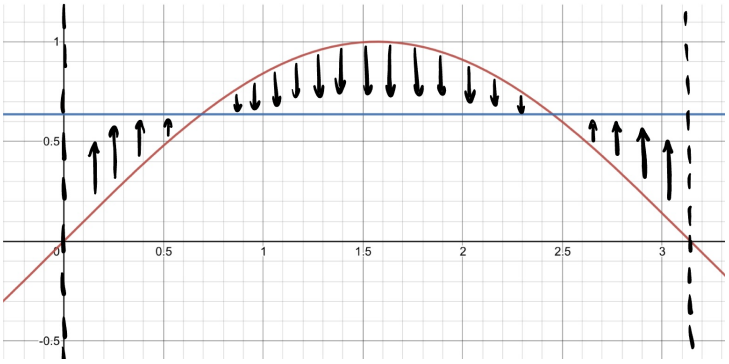

Figure 3: If we flatten out our curve, the area underneath shouldn't change.

Another way we can think about it is that, if we take the $(b-a)$ term outside the sum so we have

$$\frac{b-a}{N} \sum_{i=1}^{N} f(x_1) = \frac{b-a}{N} f(x_1) + \frac{b-a}{N} f(x_2) + ...,$$

this begins to resemble a Riemann sum. We're summing N rectangles, each with a width of the interval divided by N and a height of some point on the graph. Unlike the Riemann sum, our points are random, but if we take enough of them, they should be pretty evenly distributed. So if the Riemann sum approaches the integral as we add more points, so should our formula.

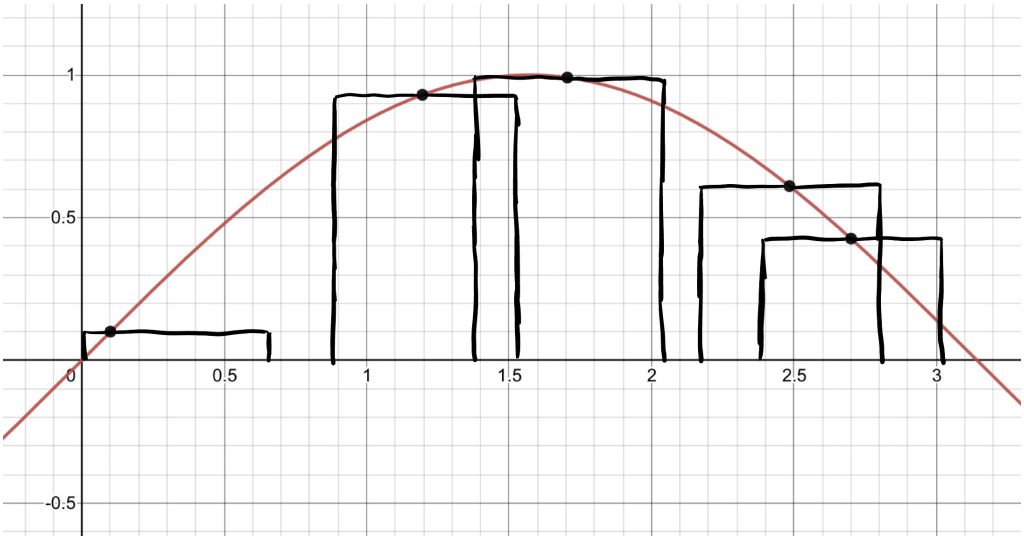

Figure 4: As we generate more random data points, our formula starts to look more like a Riemann sum.

At this point, I want to actually do some Monte Carlo integration. We've seen a description of how it's meant to work, but have yet to see it in action. Below is a Java program that lets us approximate $\int_0^\pi \sin(x)dx = 2$, with a sample size on whatever order of magnitude we want.

```java
import java.lang.Math;
import java.util.Scanner;

class MonteCarlo{

    public static void main(String[] args){

        //gets order of magnitude from user
        Scanner sampleSize = new Scanner(System.in);
        System.out.print("Order of magnitude: ");
        long n = sampleSize.nextLong();

        double sum = 0; //creates a sum variable which we will add each (b-a)/n * f(x) term to

        //the important stuff
        for(long i = 0; i < Math.pow(10,n); i++){
            double randomx = Math.PI * Math.random(); //gets a random number x between 0 and pi
            double randomf = Math.sin(randomx); //f(x)
            double width = Math.PI/Math.pow(10,n); //(b-a)/n
            double summand = width * randomf; //(b-a)/n * f(x)
            sum = sum + summand; //adds the above term to our total sum
        }

        //prints the total sum
        System.out.println(sum);

    }
}
```

Figure 5: A program that runs a Monte Carlo integration.

Some notes on reading this if you're not familiar with Java: first, the comments should give you an idea of what's going on even if you don't look at the code at all. Everything through line

12 is just setting up the code and asking the user to give a sample size, so you can ignore that. Ignore the words "long" and "double." The equals sign is used differently in Java than it is in math, so on line 20, we're adding the old value sum to summand, then assigning this as sum's new value.

I ran the program a few times for different sample sizes. Here's a summary of the results:

$$n = 10 : \int_0^\pi \sin(x)dx \approx 1.8375232432, 1.5440194162, 1.7501411125, 2.4743414293, 2.1003806730$$
$$n = 10^3 : \int_0^\pi \sin(x)dx \approx 1.9860773296, 1.9921060765, 2.0548162660, 2.0153501796, 2.0197456334$$
$$n = 10^6 : \int_0^\pi \sin(x)dx \approx 1.9986754065, 1.9991289485, 1.9990952128, 1.9993835580, 1.9998863311.$$

This should help give a sense of the degree of precision we can get from this technique. With a sample size of 10, we end up erring by nearly 25% in two of our five trials. When we move up to 1,000, each trial is within 3% of the true value, and by eyeballing the results we could probably guess that the true value is around 2. And once we get to 1,000,000, all five trials are correct to 3 significant figures.

A clever reader might have noticed that this is among the worst ways we could possibly approximate this integral. If we take a more standard approach, like a Riemann sum, our approximation approaches the exact value far more quickly, and it does so without random variation. Taking the left-hand Riemann approximation of $\int_0^\pi \sin(x)dx$ with n subintervals gives

$$n = 10 : \int_0^\pi \sin(x)dx \approx 1.98352$$
$$n = 50 : \int_0^\pi \sin(x)dx \approx 1.99934$$
$$n = 200 : \int_0^\pi \sin(x)dx \approx 1.99996.$$

At $n = 200$, we already have a closer Riemann approximation than we do with Monte Carlo at $n = 10^6$, and we get to deal in absolutes rather than probabilities. So if we're going to justify the use of Monte Carlo integration, we need some sort of problem where this is not the case.

5 Monte Carlo integration in multiple dimensions

We can generalize our estimator to multi-dimensional integrals, and show that this estimator also converges in probability, pretty easily.

Definition 10. If we have a function $f : \Omega \to \mathbb{R}$ and a sequence of identical continuous random variables $X_1, ..., X_N$ with uniform probability density on Ω, then the Monte Carlo estimator of $F = \int_\Omega f(\vec{x})d\vec{x}$ is

$$\langle F^N \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(\vec{x})}.$$

Remark 5. Notice that if we let $\Omega = (a, b) \subset \mathbb{R}$, then $p(\vec{x})$ becomes $\frac{1}{b-a}$, so this formula is exactly the formula we previously described when working with single-variable functions. This perspective should help clear up why we changed the formula here the way we did.

Theorem 4. The Monte Carlo estimator still converges to the integral in multiple dimensions. $\langle F^N \rangle \xrightarrow{p} F$.

Proof. We will essentially mirror the strategy we followed in the proof of the corresponding theorem for the single-variable estimator. First, we'll rearrange so that we can apply the LLN.

$$p(\vec{x})\langle F^N\rangle = \frac{1}{N}\sum_{i=1}^{N} f(X_i).$$

Now, the LLN tells us that

$$p(\vec{x})\langle F^N\rangle \xrightarrow{p} E(f(X_i)).$$

Again, isolating the estimator and applying LOTUS, we get

$$\begin{aligned}
\langle F^N\rangle &\xrightarrow{p} \frac{1}{p(\vec{x})}\int_\Omega f(\vec{x})p(\vec{x})d\vec{x} \\
\langle F^N\rangle &\xrightarrow{p} \int_\Omega f(\vec{x})d\vec{x} \\
\langle F^N\rangle &\xrightarrow{p} F.
\end{aligned}$$

□

When approximating integrals with deterministic methods, the work becomes exponentially harder as our domain's dimension grows. Consider our previous example of a left-hand Riemann sum, where we split the interval $[0, \pi]$ into n subintervals. If we were to do the analagous process of approximating an integral over $[0, \pi]^2$, we would need to split the domain into n subintervals in both directions, giving us n^2 sub-boxes to work with. In general, if we take n data points to approximate an integral over some 1-dimensional domain, we need n^k data points to get an analagous approximation over a k-dimensional domain. This is referred to as the **curse of dimensionality**.

The value of Monte Carlo integration is that it does not suffer from this curse. We've seen that single-variable Monte Carlo integrals approach the exact value of an integral much more slowly than deterministic methods, but when we add dimensions, this rate does not change, while the rate of deterministic approximations suffers exponentially.

Theorem 5. $\mathrm{Var}\langle F^N\rangle$ is proportional to $\frac{1}{N}$ for a Monte Carlo estimator of an integral of any dimension. $\mathrm{Var}\langle F^N\rangle = O(\frac{1}{N})$.

Proof. We will first find $\mathrm{Var}\langle F^N\rangle$ for single-variable estimators, then observe that nothing changes when we add dimensions.

$$\begin{aligned}
\mathrm{Var}\langle F^N\rangle &= \mathrm{Var}(\frac{b-a}{N}\sum_{i=1}^{N} f(X_i)) \\
&= \mathrm{Var}(\frac{1}{N}\sum_{i=1}^{N}(b-a)f(X_i)) \\
&= \frac{1}{N^2}\mathrm{Var}(\sum_{i=1}^{N}(b-a)f(X_i)) \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}((b-a)f(X_i)) \\
&= \frac{1}{N}\mathrm{Var}((b-a)f(X_i)).
\end{aligned}$$

The term $\mathrm{Var}((b-a)f(X_i))$ is constant for any given estimator, so $\mathrm{Var}\langle F^N\rangle$ is proportional to $\frac{1}{N}$ times some constant.

$$\mathrm{Var}\langle F^N\rangle = O(\frac{1}{N}).$$

Now, when we switch to our multivariable estimator, the same argument should easily pass.

$$\begin{aligned}\text{Var}\langle F^N\rangle &= \text{Var}(\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(\vec{x})}) \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}(\frac{f(X_i)}{p(\vec{x})}) \\ &= \frac{1}{N}\,\text{Var}(\frac{f(X_i)}{p(\vec{x})}).\end{aligned}$$

Again, because $\text{Var}(\frac{f(X_i)}{p(\vec{x})})$ is constant, we have

$$\text{Var}\langle F^N\rangle = O(\frac{1}{N}).$$

□

This means that no matter how many dimensions we're working with, $\text{Var}\langle F^N\rangle$ will only be a function of sample size – if we want less variance on our estimator, we just need to take a bigger N. In other words, the difference between $\langle F^N\rangle$ and F itself just depends on N. This holds for 1-dimensional integrals and for 100-dimensional integrals, whereas we can assert no such thing for deterministic integration techniques.

This is really quite a striking result, and stands in contrast to what we would probably expect on an intuitive level. With one-dimensional functions, we can technically integrate using random sampling, but it's just a terrible idea – recall how we found a better approximation of $\int_0^\pi \sin(x)dx$ with a Riemann sum of 200 points than with a Monte Carlo integration with of million points. But once we start looking at higher dimensions, this Monte Carlo approach, which approximates the integral very slowly, stays just as slow as it was, and eventually surpasses deterministic methods in efficiency.

6 References

1. Sobol' 1994: A Primer for the Monte Carlo Method
2. Monte Carlo Integration on Wikipedia (am I allowed to cite wikipedia?)
3. Monte Carlo Methods on Wikipedia
4. https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-in-practice/monte-carlo-integration.html
5. https://cs.dartmouth.edu/wjarosz/publications/dissertation/appendixA.pdf
6. https://statproofbook.github.io/P/mean-lotus.html
7. https://towardsdatascience.com/proof-of-the-law-of-large-numbers-part-1-the-weak-law-daf412178d3a
8. https://www.probabilitycourse.com/chapter6/6_2_2_markov_chebyshev_inequalities.php
9. https://dlsun.github.io/probability/linearity.html
10. https://www.probabilitycourse.com/chapter3/3_1_4_independent_random_var.php
11. https://www.probabilitycourse.com/chapter7/7_2_5_convergence_in_probability.php
12. https://towardsdatascience.com/the-basics-of-monte-carlo-integration-5fe16b40482d