PROFESSOR: He ended up last Thursday's lectures talking about Gaussian distributions. As he said, one of the interesting things about a Gaussian is it can be fully characterized by its mean and its standard deviation. And this concept of being able to take a curve and characterize it with a small number of parameters is something we'll continue to see as a very important way of looking at modeling physical systems.

And in fact, that is the part of the term that we've entered. And it's a part of the term we'll spend a lot of time on. And the whole issue is, how do we construct computational models that will help us understand the real world?

When we can, we love to model distributions as Gaussians, or normal, because they're so nicely characterized. We have nice rules of thumb that tell us how close things lie to the mean, et cetera. However, it's important to understand that if something is not actually normally distributed and we pretend it is, we can get very misleading results out of our model.

So let's think about the fact that not all distributions are normal. So consider rolling a single die. Each of the 6 outcomes is equally probable. So we would not expect to see a peak, say, at 3 or 4, and a trough at 1. A 3 or a 4 is the same probability as 1.

Similarly, if one thinks about the Massachusetts state lottery, or any fair lottery, the probability of each number coming up is the same. So it would be a flat line. If you had a million numbers, the probability of each number is 1 over a million. And so if you plotted the probability of each number, again, you'd get a flat line.

Such distributions are called uniform. Each result is equally probable. We can fully characterize a uniform distribution with a single parameter, its range. If I tell you it ranges over 1 to a million, that's all you need to know to know what the distribution looks like. So they're even simpler than normal distributions.

Uniform distributions occur quite often in games devised by humans, but almost never in nature. And typically they're not very useful for modeling complex systems. We have to work hard to invent something that's normal. Most things are not naturally that way. I'm sorry, to invent things that are uniform. Normal, as we saw last time, occurs all the time in nature.

The other thing that occurs quite frequently are exponential distributions. They're used in a lot of different ways. For example, people who are trying to plan things like highway systems use exponential distributions to model inter-arrival times, how much time there is between each car, say, entering the Mass Turnpike. We'll see many other examples of them.

The key thing about them is they have the property of being memoryless. They are in fact the only continuous distributions that are memoryless.

So let's look at an example with which some of you are more familiar than you want to be, the concentration of a drug in the human body. For those who are watching on OpenCourseWare, it's not because all the students are drug users. It's because they're working on a problem set that involves modeling drugs in the human body. I don't know how many of you are drug users, all right?

Assume that at each time step, each molecule has a probability p of being cleared by the body. The system is memoryless in the sense that at each step, the probability of a particular molecule being cleared is independent of what happened at the previous steps. So the fact that a molecule didn't get cleared at time t has no impact on whether or not it will be cleared at time t1. The probability doesn't go up as it doesn't get cleared. So it's independent of the previous steps.

So at time t equals 1, what's the probability of the molecule still being in the human body? If the probability of being cleared at each step is p, it's 1 minus p, right? So if the probability of being cleared was 0.5 at each time step, the time that it still exists after the first time step is 1 minus 0.5-- i.e., 0.5.

So what's the probability of it still being in the human body at time t equals 2? Well, it wasn't cleared at time 1. Since it's memoryless, whether or not it's cleared at time 2 is also 1/p. And so it existing still after two steps is going to be 1 minus p squared. We saw that with independent probabilities. And the nice thing about working with exponential distributions is we know that the probabilities are independent.

More generally, its still being in the body at time t is going to be 1 minus p to the t. So we have a nice closed-form solution that will give us the probability of each molecule surviving until time t. All right?

So now let's look at the question of, suppose that at time t equals 0, there are m0 molecules. Now we can ask, how many molecules are there likely to be at any time t? Well, let's write a little program to look at that.

So that's this program, clear. We'll start with n, the number of molecules, the probability of clearing it at each step, and the number of steps. And we'll keep track of the num remaining. So at the beginning, we have n molecules remaining. And then for t in range steps, we're just going to multiply n, the number we started with, times the probability of each molecule still existing. And then we'll plot it.

Does that make sense? So this is a tiny bit of code that basically implements that formula over on the board.

Let's run it. And we'll run it starting with a 1,000 molecules, a probability of each being cleared of 0.01, and we'll look at 500 time steps.

All right. This is kind of interesting. We're getting a straight line. That doesn't look like an exponential, does it? Or does it? Why do we have a straight line here? Somebody? Because I used a semilog axis. So let's look at it now without that. We now see something that really does look like exponential decay. It drops very quickly in the beginning, and then it asymptotes towards 0.

But of course it never quite gets there in a continuous model. If we had a discrete model, we would eventually have to get to 0, because that last molecule would either get cleared or not. But in a continuous world-- which is in this case probably not a good model, or not a perfect model I should say, because it allows us to have a quarter of a molecule there, which we kind of know is physiologically nonsense, biochemically nonsense.

But you can see we get this exponential decay. But as we saw previously, if we plot an exponential on a log axis, as the math would tell us, we get a straight line. And that, in fact, is a very simple and nice way to see whether you have an exponential distribution. Put it on a log axis, see if it's straight. It's a good trick, and one we use a lot.

OK. So there, I took the physical model I described and derived, through a little bit of math, what the result should be, and implemented some code to give us a plot of what that told us.

Let's look at a different way of doing it. I could've instead written a Monte Carlo simulation to do the same kind of thing. So here, instead of working out the probabilities, I just tried to write some code that exactly mimicked the physical process that I was talking about.

So instead of knowing that I could just look at 1 minus p to the t, at each step, I cleared some molecules. I just used random.random. If I came out with something less than the clear probability, I got rid of that molecule. And I did that for each molecule, deciding whether or not it should be cleared. For molecule m in range, looking at all the remaining molecules, I either clear one or I don't. And then I can plot that.

So let's look what happens if I compare the two results. So I'm going to do the original analytical model of clear, and then the simulation model of clearing, and see what I get.

Well, much to my relief, I get kind of the same curve. Not exactly. You'll notice that the blue curve, the analytical model, is a beautiful smooth curve, whereas the red curve has got a little bit of jaggies. It's clearly very similar to the blue curve, but not identical. It doesn't surprise me. There is some randomness in there. And in fact, I could have gotten unlucky and gotten something that didn't look like the blue curve. But given the sample size, that would have been quite surprising.

Which of these two models do you like better? So we've got two models. We've got one I'll call the analytic model, and one I'll call the simulation model. Both show exponential decay. That is to say the number of molecules declines exponentially, quite quickly. But they're not quite identical.

So which would we prefer? Or which would you prefer? There is no right answer for this. Just for fun, I'll ask for a poll. Who prefers the analytical model? Who prefer the simulation? All right. Somebody who prefers the analytical, tell me why.

AUDIENCE: It looks a lot nicer.

PROFESSOR: Well, all right. It looks a lot nicer. That's kind of human nature, to prefer something that looks prettier. On the other hand, what we're really interested in is the question of not aesthetics, but fidelity to the actual physical situation. A straight line might look even nicer, but it wouldn't be accurate.

So when we think about evaluating a model, what we really should be asking, I think, are two questions. One is fidelity. And another way to think about that is credibility. Typically, we're creating a model because we don't know the actual answer. And we're trying to see what might actually happen if we, say, ran a physical experiment.

And so we have to ask the question of, do we believe the results the model are giving us. And so that sort of boils down to not a question of mathematics, but a question of reasoning. Can we look at the model and convince ourselves that it is accurate?

And the other question is utility. And I can think about that as, in some sense, what questions are answerable with the model?

So the first one is pretty much a question of personal preference. And for this particular simulation, which is pretty simple, or this particular model, it's hard to argue that one is more believable than the other. I might argue the second is more believable, because it's a direct implementation of the physical system. I didn't rely on my math being right. But the math is pretty simple here.

What's, I think, more apparent is in this case there is some additional utility offered by the simulation model. And it's often true of that, simulation models, that we can ask what-if questions, because we can easily change the model to be slightly different in ways that is usually harder for an analytic model.

For example, suppose these drug molecules had this peculiar property that every 100 time steps, they could clone themselves. And so every molecule that was there became two molecules. Unlikely for the drug. Not so unlikely for, say, a bacterium or a virus, as you've seen.

Well, a little hard to figure out how to do the probabilities in the case that that happens, because we'll no longer get this beautiful, simple exponential decay. But quite easy to think about how we would change the simulation model, which is what I have done here.

So I said here, if time is not equal to 0 and time is evenly divisible by 100, then I'm just going to double the number of molecules. Every living molecule will clone itself. And now we'll see what we get.

Well, we get this rather peculiar-looking sawtooth distribution. We still have, overall, an exponential decay. But we see every once in while it jumps up, and then it comes down. It's not so easy to write a simple closed-form formula that describes this, but very easy to produce a simulation model that gives you some insight to what's happening here. And that's, I think, one of the great attractions of simulation modeling, is we get to do this sort of thing.

Many, many physical systems exhibit exponential decay or exponential growth. For example, people in Japan now are very interested in half-life of various radioactive particles. And when we talk about half-life, we mean that there is exponential decay in radioactivity. That's what half-life is. So people are looking at what is the half-life of iodine, say, versus other radioactive particles.

We also see exponential growth a lot. I used to have a swimming pool which I had to maintain, and I realized if I let the algae get out of control in the pool, it went from having very little algae to having a lot of algae very quickly, because the algae doubles every period. And so all of a sudden, it takes off.

So exponential growth is-- exponential decay are important things. We see them all the time. People use the word very carelessly when they mean quick growth. They say exponential. But of course, it has a very specific meaning.

OK. We've now, for the moment at least, finished our short venture into probability and distributions. We'll come back to it a little bit when we talk about how to lie with statistics. But before we do that, before we leave probability for a while, just for fun, I want to pose to you one of these probability problems that hurts people's heads. It's a very popular one.

How many people here have heard of the Monty Hall problem? OK, a lot of you. So as we play the game, those of you who know the answer, I'll ask your forbearance not to blurt it out. So it's a wonderful problem. It's so exciting that people have written books about it.

So here's how it works. This is from a game show called, I think, *Let's Make a Deal*, with the host, Monty Hall, who did it forever. So the way it works is you start with three doors. Behind one of the doors is a great prize-- for example, an automobile. Behind each of the other doors is a booby prize, typically a goat. I don't know why people don't like goats, but apparently they don't.

So the way it works is Monty invites someone from the audience, chosen on the basis of their outlandish costumes. And they come down and they're told what wonderful prize is behind one of the doors. And then they're asked to choose a door. So the person might choose a door and say, I'll choose door number one.

Monty then opens one of the other two doors. He knows which doors have the goats and which door has the car. He opens a door with the goat. So now there are two doors left. And he asks the contestant, do you want to switch. Do you want to stick with door one or would you like to switch to door two?

And the Monty Hall problem is, what should she do? And the audience will always shout out advice. So I do have a simulation of that. I'd like to run. I need three people to volunteer to be doors. Come on, three doors. It's not so hard. Come on down. And I need one person to volunteer to be the contest. Is anybody in a costume here? I don't know. Mitch is kind of in one, but-- all right. These are the contestants. All right, you're door number (2). You're door number (1). You are door number (3).

A contestant please. There's $1 in one of these. You can actually win something of value. All right, we have a contestant coming down. Oh, all right, we have two contestants coming down. All right. The aisle wins.

All right, choose a door. You choose door number (2). All right, let us open door number (1). And let's see what's in door number (1). Show it to the class. It is a goat. Now you have a choice. You can stick with your original decision, or you can switch to door number (3). Suggestions?

AUDIENCE: Switch.

AUDIENCE: Switch.

AUDIENCE: Switch.

AUDIENCE: Switch.

AUDIENCE: Lower one.

AUDIENCE: Don't switch it.

AUDIENCE: Switch.

PROFESSOR: All right. She is going to stick with door number (2). Let us open door number (2). She wins $1. It is yours. Don't spend it all at once. Thank you, everybody.

All right, now, was she lucky or was she smart, is the question? Does it matter? This was a subject of enormous debate in the mathematical community. In 1991, *Parade* magazine published a correct solution to the problem, and approximately 10,000 readers, including a 1,000 with PhDs in mathematics, wrote to *Parade* telling them they had published the wrong solution. And the debate roiled on.

So who thinks she was lucky and who thinks it actually matters whether you switch? Who thinks it matters, those who don't know the problem? Who thinks it doesn't matter? All right. The doesn't-matters win by a small margin. And in fact, that's what the readers of *Parade* thought. But they were wrong. It matters a lot whether you switch.

Let's do the analysis first, analytically, and then we'll do a simulation. So the player makes a choice. And this is some interesting ways to think about probability. And with the probability of 1/3, the player has chosen the correct door. All right? Now that means that with a probability of 2 out of 3, the car lies behind one of the other two doors.

Now here's the key step. Monty opens a door that he knows does not contain the prize. The key thing to notice here is the choice of doors is not independent of the choice of the player, because Monty will never choose the door that the player has initially picked.

Now since the probability of the prize being behind the two remaining doors is 2 out of 3, the probability of the prize being behind one of the doors that he did not open is 2 out of 3 -- in fact, behind the other door.

In fact, you were extraordinarily lucky to win the dollar, because switching doubles the odds of winning. Because remember, your odds of winning were 1 out of 3 when you first chose the door. That left two doors. The probability of the car being behind one of those two doors was 2/3.

Monty opened the one that didn't contain the car, because he knew it contained a goat. So that must mean the probability of the car being behind the remaining door is 2 out of 3. So you double your odds of winning.

The logic is kind of clear. It didn't stop people from aggressively debating it for the longest of times. And I kind of didn't believe it myself. So I did what I usually do, is I wrote some code. And let's look at two pieces of code here. And again, the theme here is how we can use simulation models to understand slightly complex, or more than slightly complex, situations.

So here's the way the game works. So I've got a simple simulation that counts the number of wins. And the way it's done is, for t in range number of trials, the contestant picks 1, 2, or 3 at random. I've tried to mimic exactly the game. So the car is behind one of the doors. The contestant guesses a door.

And then there's this 'choose' function to open one of the two. And we're going to have 2 ways of choosing which door gets opened.

So the Monty Hall way-- Monty chooses. He takes the guessed door and the prize door, and he opens the non-guess that contains the goat. So if (1) is the guessed door, and (1) is not the guessed door and it's not the prize door, then he opens (1). Same thing for (2). And if (1) or (2) is not the choice, he opens (3).

As opposed to the random choose function, which just chooses at random between the doors that weren't guessed. So it might open the car, at which point the contest is told, sorry, you lose, you don't even have a choice anymore.

We're then going to run the simulation with Monty choosing and random choice, and see what we get. So you've got the code on the handout to do this. I'm not going to go over the details. The thing to notice about it is it's yet another example of how we can use PyLab to do some interesting plots. This time I'm going to print a pie chart, just to show that we can do those. And let's see what happens.

So people understand what's going on? That I've got these two functions, montyChoose and randomChoose. I'm using those functions as parameters, a very convenient thing, and running the simulation each way. And let's see what happens.

All right. So what we see here is, when I run montyChoose, sure enough, it comes out to about 2/3 of the time, you win if you change, and only 1/3 of the time if you don't, pretty close to what the math predicts. In fact, sort of astonishingly close. On the other hand, if Monte had been just choosing at random, then we see it really doesn't matter whether you switch or not.

So again, from a probability point of view, we see how subtle these things can be based upon whether decisions are independent of previous decisions, or not independent. And we also see, in some sense, that we can write a very small piece of code that actually provides a simulation of a real, in this case, game, and we can have, I think, a lot of confidence. We can look at the code and say, is it really the way the game is described? Yes. And then we get nice results that tell us what to do.

And in this case it tells us that, if Monty is choosing based upon what he knows, then by all means, you should switch. And I'm sorry that it didn't work out that way when we played the game, but that's the way probabilities are, that you didn't switch and you lucked out. So now you're a rich lady.

All right. So that's one thing we can do. One more thing I want to talk about, before we leave the subject of Monte Carlo simulations-- it's pretty clear that these kind of simulations are very useful for tackling problems in which predictive non-determinism plays a role. And at first blush, you might think that, OK, that's the only time we should use a Monte Carlo simulation, when there's some inherent randomness in the problem, and therefore it's hard to model analytically, and therefore we'll use randomness in the code.

Interestingly enough, particularly in recent years, but for quite a while, people have understood the notion of using randomized algorithms to solve problems in which randomness plays no role. And that's a little surprising, but an incredibly useful concept to put in your bag of tricks, the ability to use randomization to solve problems that are not random.

So let me talk about an example. Consider the concept of pi. Pi has been around for a long time. For thousands of years, people have known that there's a constant-- called pi since about the 18th century-- associated with circles, such that the circumference of a circle is always going to be equal to pi times the diameter. The area of a circle is always going to be pi r-squared, et cetera.

So for thousands of years, people knew that there was such a constant. They just didn't know what it was. And there's a long and beautiful history of people attempting to estimate pi. About the earliest estimate I've found is from the Egyptians, in something called the Rhind Papyrus, from 1650 BC, or thereabouts. And it estimated pi to be 4 times-- let me get this right-- 8/9 squared, which is 3.16, more or less. That was pretty good.

About a 1,000 years later, an estimate of pi appears in the Bible. Or at least, it's implied by the Bible in a description of one of Solomon's construction projects. It says, "And he made a molten sea, 10 cubits from the one brim to the other. It was round all about, and his height was five cubits. And a line of 30 cubits did compass it round about."

So you can take that and solve for pi, because you've given the circumference, and other details, the diameter. You can solve for pi. And you see, if you do that, pi comes out to be exactly 3. Not quite as accurate as the Egyptians had 1,000 years earlier.

Now perhaps the Bible is wrong. I don't want to offend anybody. Or perhaps the molten sea wasn't perfectly circular. Or maybe the circumference was measured from the wall outside and the diameter from the inside. Or maybe it was just that was a good enough number to use in construction, because a cubit was something like the length of your forearm. Different people have different length forearms, and there was no reason to try and be more precise. Who knows?

The best estimate of pi in ancient times was from Archimedes of Syracuse. And he did something quite amazing for around 200 BC. He didn't give the value of pi. He said, I don't know what the value is, but I can give you an upper bound and a lower bound.

And he did this by carefully constructing a polygon with a huge number of tiny little straight lines that would approximate a circle, and then actually measuring things. So he built a polygon with 96 sides, and concluded that pi was somewhere between 223 divided by 71, and 22/7. Very sophisticated at the time, to be giving upper and lower bounds.

If we look at what the middle of that is, it's actually amazingly good. It's 3.1418. Not bad.

All right. What does this have to do with Monte Carlo simulations? Many years later-- in fact, in the 1700s, two French mathematicians invented another way of computing pi. Buffon and Laplace. Actually Buffon first proposed it. He got it wrong. Laplace corrected it. And they said, we can find pi using a stochastic simulation. They didn't use those words, but that's what they basically described. And they talked about it in terms of needle-dropping.

So think about having a square, and inscribing in the square a circle. And you'll excuse my lack of artistic ability. So they put one of those on the floor. And then they dropped needles, which would get carried around by the wind and land in some random place. And they counted the number of needles that landed in the circle, and the number of needles that landed in the square but not in the circle.

And then they did a little math. Let's assume for the sake of argument here that the radius of the circle is 1. They observed the following equation must hold, that the needles in the circle over the needles in the square should be equal to the area of the circle divided by the area of the square. It seems logical, if they're landing at random, that they would get distributed proportional to the area.

And then they solved for pi, knowing that the area of the circle is pi r-squared. They could then say that pi-- in fact, in this case, since we know the radius is 1, and 1 squared is 1, that tells us that the area of the circle should be pi, right? Pi times 1.

So they said pi is equal to the area of the circle, which is equal to the area of the square times the needles in the circle, divided by the needles in the square.

So they had that formula. Unfortunately they couldn't drop enough needles to get a very good estimate. So they described how to do it. But this was an experiment. They did the math, they had a nice formula. They did not have an experimental apparatus that would actually let them drop enough needles to get a very good estimate. It would take a lot of patience. And maybe they wouldn't land at random. Who knows what.

Fortunately, today, we have a much easier way to do that. So we can write some code that does the simulation. We're going to have a way to throw the needles or drop the needles. And then what we're going to do is we're going to have a simulation that we're going to run, that's going to-- I don't know how many needles to drop.

I'm going to keep dropping needles until I get a small enough standard deviation that I can be confident that I have a bound on pi with some confidence interval. In fact, I'm going to use 5% here, and use that rule of thumb that Mitch talked about last time, about standard deviations.

And I say, all right, I'm going to keep running the experiment until the standard deviation of trials is 5% or less. Two standard deviations is small enough that I get my answer within some precision. I'm going to ask here for a precision of 0.01. And so therefore my standard deviation should be that precision divided by 4, because I'm looking for 2 standard deviations on either side of the mean, which is why I'm dividing by 4 and not by 2. So I divide by 4 and I see what I get.

So let's run it. And this will take a little bit of time. It will take no time if I don't uncomment the code to actually run the experiment. Estimate pi. And we'll get some estimates.

So what we can see here is that my estimates change as I run experiments. Every time I run this-- or not every time, I often get a different number of needles I need.

But you can see that my first estimate is not very good. My estimates do get better, though not monotonically better. But what does get monotonically better is the standard deviation gets smaller and smaller, which is what you would expect.

So there's no guarantee that by running a bigger trial, I get a more accurate result. What there is a guarantee is that I can have more confidence in my result. I could have gotten lucky and run a small number of needles and gotten it exactly right by chance. But I would have been wrong to assume it was right, because let's pretend we didn't know what the value of pi was, a priori.

But what I can say here is since my standard deviation is now 0.002, and if we look at it, we'll see that these things are normally distributed, I can be pretty sure that the true value of pi is 3.1407 et cetera, plus or minus 0.00023, et cetera, with a 95% confidence.

So this is using the stuff that you saw in the last lecture to now combine that statistical background with this simulation to compute a pretty darn good estimate of pi. And if I ran more needles, if I wanted to get more precise, I can get as precise as I want to be, as many digits of precision as I want.

So again, what we see here is that we've been able to solve a problem that had nothing to do with randomness. The value of pi is not a random number. And yet we used randomness to solve it. A very common technique. And we use some very simple statistics to know whether or not we should believe our solution. And so those are the two lessons I want you to take home.

Now if you look at your handout, you'll see that at the bottom, I've used the same technique to do integration. If you think about what integration means, when you ask, what is the integral of some formula, what you learned when you first looked at calculus was that that was the area under some curve, right? That's what the integral is. And you learned all sorts of complicated mathematics to solve complicated integrals.

Well, you can pose an integration problem exactly analogous to this. You draw your curve. You drops some needles. You count how many fall under the curve, how many don't fall under the curve in some larger area. And you can solve the integration. And that's exactly what I've done here, where f is the function being integrated. I won't go through the details.

Now in fact, this kind of simulation is not a good way to solve single integrals. It's much better to use something like Simpson's rule, whatever that is. But in fact, it is frequently used in practice for more complicated things, a double or triple integration, where the mathematics gets fairly complicated. People will often solve those problems using a Monte Carlo simulation. It's a practical method for tackling it. And again, in your handout, you'll see a very simple piece of code that does a double integral.

All right. That's all for today.