

Physics through Computational Thinking
Dr. Auditya Sharma and Dr. Ambar Jain
Department of Physics
Indian Institute of Science Education and Research, Bhopal
The Monte Carlo Method

Hi, guys. So, we have been looking at the Monte Carlo method. We have seen a few examples of how this works out in practice. So, these discussions have been sort of very strongly example based and implementation based. We have not really gone into the theory of the Monte Carlo method or why it works and so on. The theoretical aspects would require a substantially more involved discussion and would involve more time and so on. Anyway, that is not really the goal of this course.

So, in the same spirit we will continue with some more examples. So, one more example which I will discuss in this particular module is that of computing π by an alternate method. We already saw one method of computing π using an integral and using an important sampling. Now, so this the method that I am going to describe here is related to something called the Buffon's Needle drop experiment.

So, so this is a centuries old problem which was considered by this person called Buffon. And so, who imagined dropping a lot of pins. And so, you imagine a sheet of paper which has lots of parallel lines. It could be you know your whole notebook, for example. And then you drop pins of a certain, certain length and then you ask, what is the probability that, you know, this pin would cross a line? And what is the probability that it does not cross a line. So, one has to take care of you know the various angles present and so on.

And it is possible to do a probability calculation of this and find out the answer. And it turns out that this answer is also actually related to π and in fact there is way to compute π using, you know, you doing this kind of a needle drop experiment repeatedly. You can drop like a million pins on top of a piece of paper with lots of lines.

And I mean, the calculation is somewhat involved. And you have to consider cases where whether the you know, the distance between the lines is greater or lesser than the, the length of

the pin involved and so on. So, the method I am going to describe today to compute π is rather simple and it illustrates the philosophy of the Monte Carlo method. So, here we go.

(Refer Time Slide: 3:04)

• Strategy for calculating π using random number is straightforward. Consider a circle embedded in a square, such that radius of circle is 1 and sides of the square are 2.

• If we randomly and uniformly generate points in the square and count the points that fall inside the circle what fraction of points will lie inside the circle? In other words if I throw darts randomly inside the square bounding the dartboard, what is the probability of landing inside the dartboard?

• The probability is given by

$$P_{\text{circle}} = \frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi}{4}$$

Slide 2 of 2

Okay. So, the idea is to consider a region which is you know a circle of radius 1 and there is a square that encloses this circle, and so you just imagine you know dropping this kind of doing a needle drop experiment of Buffon or somebody else. Imagine that there is a flat surface which, you know, which has this kind of a circle of unit radius. And there is a square around it which will have length 2 because the diameter of the circle is 2.

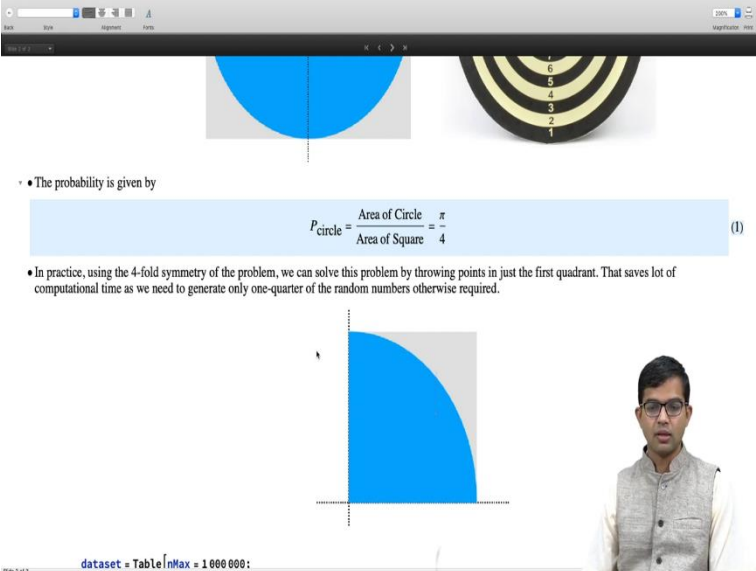
And then you imagine dropping lots of darts or you know if it would be darts if it is vertical. But I guess it makes more sense to think of this as a plane on the ground. And then you are dropping lots of needles from above. And then you just simply count the number of needles which fall inside the circle and evaluate the ratio of the number of needles which fall inside the circle to the total number of needles dropped. And of course, the key requirement here is that it is going to be a uniform distribution.

So, every point inside this square of unit of side 2 is equally likely. So, there is no bias for these needles. They can be dropped; they are dropped everywhere with the same probability. And if this happens, then indeed it is of Monte Carlo theory tells us that the ratio of the number of points which are dropped inside the circle, divided by the total number of pins which are

dropped, it is going to be basically the area of the circle divided by the total area, which we will see immediately will give us an estimate of π .

So, how does this go about? So, probability of this falling inside the circle is $\pi * (\text{radius})^2$ which is just $1/(\text{Area of square})$, which is 4, right? Because you have a square whose each side is 2 in whatever units. So, it is going to be $\pi/4$.

(Refer Slide Time: 5:37)



• The probability is given by

$$P_{\text{circle}} = \frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi}{4} \quad (1)$$

• In practice, using the 4-fold symmetry of the problem, we can solve this problem by throwing points in just the first quadrant. That saves lot of computational time as we need to generate only one-quarter of the random numbers otherwise required.

dataset = Table[nMax = 1000000;

Now in practice using the 4-fold symmetry of the problem, we can actually solve this problem by considering just the first quadrant. You do not have to, so, what we are planning to do is of course do this computationally. It is just impractical to actually drop so many pins Buffon experiment style.

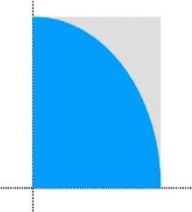
But we have a pseudo random number generator, right. Because we have Mathematica. And then we can use this to generate lots of random number as long as they are drawn from a uniform distribution in the appropriate manner. We can actually get a very good estimate of π by this method. So, since you can get $\pi/4$, you can also get π .

And so, to do this, we will just consider one quadrant of this circle. And then perform an experiment only inside this and and ask what will be the... so it is going to be just. So, the area of the small square now is what? It is going to be, one squared. So, then the area of this circular

arc, this one fourth of the circle is just $\pi/4$ right. So, it is exactly the same. So you still have to multiply by 4 in the end.

(Refer Time Slide: 6:51)

• In practice, using the 4-fold symmetry of the problem, we can solve this problem by throwing points in just the first quadrant. That saves lot of computational time as we need to generate only one-quarter of the random numbers otherwise required.




```
In[4]:=
RandomReal[{0, 1}]

Out[4]=
0.96741
```

```
dataset = Table[nMax = 1000000;

$$\frac{1}{nMax} \left( \text{Table}[\text{If}[\left[ \frac{\text{RandomReal}[\{0, 1\}]^2}{\text{Total}} \leq 1, 1, 0 \right], \{nMax\}] // \text{Total} \right) 4.0, \{100\}];$$

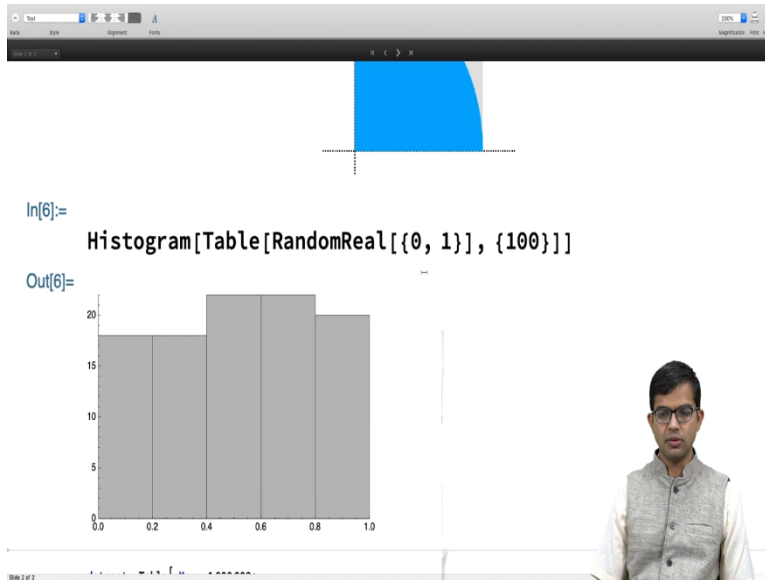
```



So, here is the code which does this. There is a compact way of doing this whole thing and that is. So, as always with Mathematica, you just try to create a table of tables. So, like that is what we do here. So, so you say that let us go inwards out. So, if I generate RandomReal. I told you that RandomReal for me will get me a random number between, between 0 and 1.

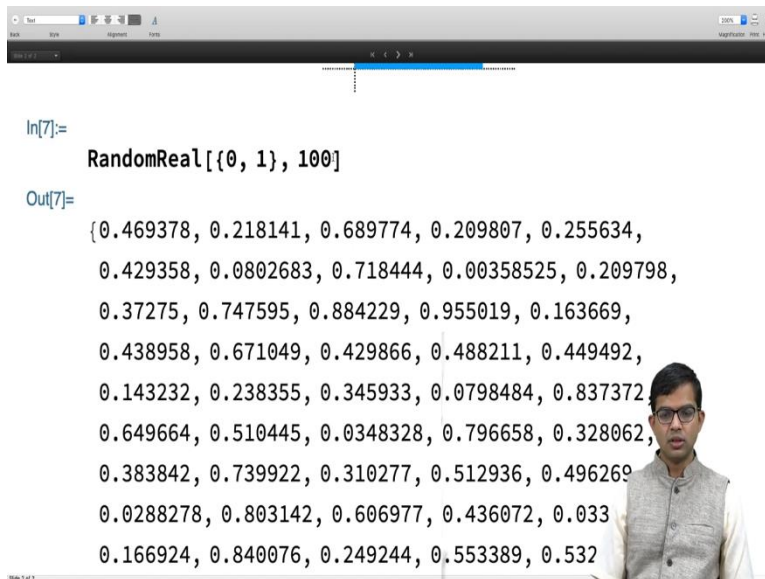
So, let me look at input here and then I have RandomReal. If I do it once, it gives me a random number between 0 and 1. So, I can. I mean, to be. So, I can do this in any interval. So, I can specify it like here.

(Refer Time Slide: 8:04)



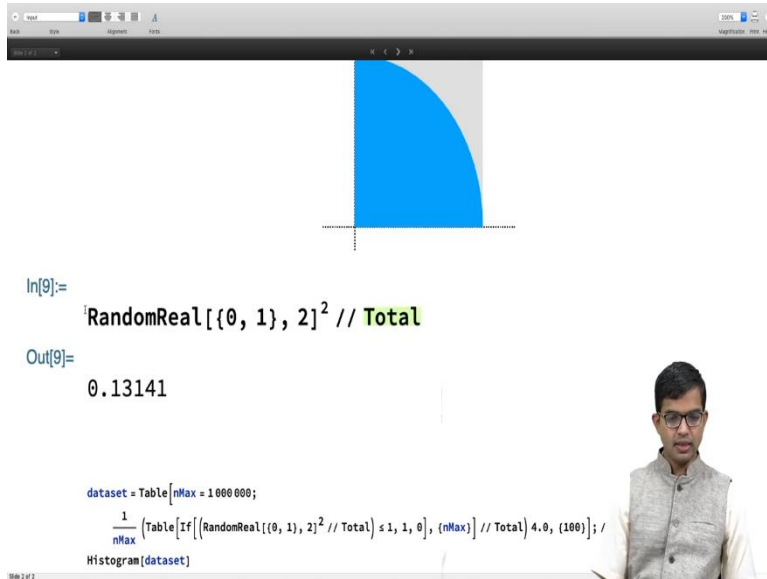
So, I can look at what this does if I make a table of this. So, a table of RandomReal. And let say I generate a 100 of these and let me make a histogram of this. So, it is giving me a uniform distribution between 0 and 1 in this interval.

(Refer Time Slide: 8:40)



So, actually what I want to do is generate, 2 such numbers and instead of, ya, so equivalently, I could have also done it like this. So, I can actually do. So, it gives me a hundred numbers.

(Refer Time Slide: 9:06)



The screenshot shows a Mathematica window with a blue quarter-circle plot in the top right. Below the plot, the following code is entered in the input field:

```
In[9]:= RandomReal[{0, 1}, 2]^2 // Total
```

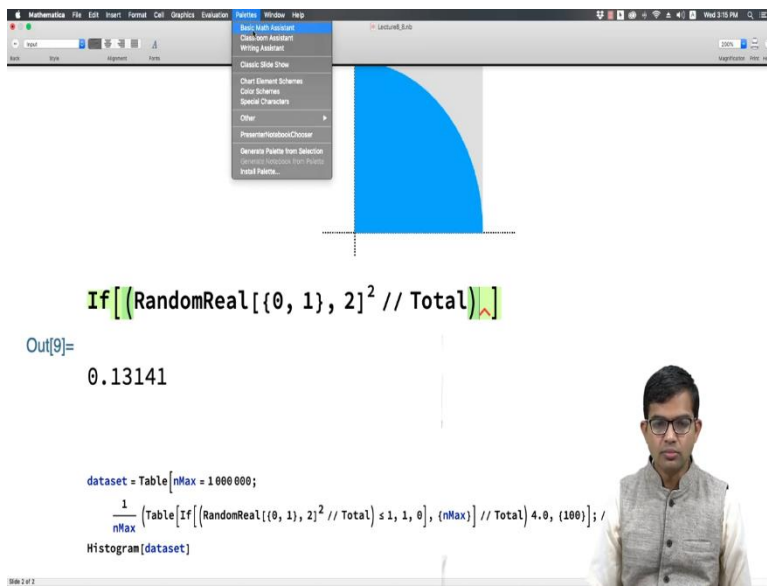
The output is:

```
Out[9]= 0.13141
```

Below the output, the following code is entered:

```
dataset = Table[nMax = 1000000;  
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total) <= 1, 1, 0], {nMax}] // Total) 4.0, {100}]; /  
Histogram[dataset]
```

A small inset video of a man is visible in the bottom right corner of the Mathematica window.



The screenshot shows a Mathematica window with a blue quarter-circle plot in the top right. A palette menu is open over the plot, showing options like 'Basic Math Assistant', 'Click and Drag', 'Writing Assistant', 'Classic Style Sheet', 'Chart Element Schemes', 'Color Schemes', 'Sentence Characters', 'Other', 'Primitives/OpenGL/Chooser', 'Generate Palettes from Selection', 'Generate Palettes from Palettes', and 'Install Palettes...'. Below the plot, the following code is entered in the input field:

```
If[(RandomReal[{0, 1}, 2]^2 // Total) < 1]
```

The output is:

```
Out[9]= 0.13141
```

Below the output, the following code is entered:

```
dataset = Table[nMax = 1000000;  
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total) <= 1, 1, 0], {nMax}] // Total) 4.0, {100}]; /  
Histogram[dataset]
```

A small inset video of a man is visible in the bottom right corner of the Mathematica window.

```

If[(RandomReal[{0, 1}, 2]^2 // Total) < 1]
Out[9]=
0.13141

dataset = Table[nMax = 1000000;
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total) < 1, 1, 0], {nMax}] // Total) 4.0, {100}]
Histogram[dataset]

```

```

In[13]=
If[(RandomReal[{0, 1}, 2]^2 // Total) < 1, 1, 0]
Out[13]=
0

dataset = Table[nMax = 1000000;
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total) < 1, 1, 0], {nMax}] // Total) 4.0, {100}]
Histogram[dataset]

```

So instead what I do is I get 2 such numbers. So, why do I need to do 2 and square it? If I take two such numbers, it is along the X and along the Y. So, remember I am in the first quadrant. So the X value will go from 0 to 1 and the Y value will go from 0 to 1. So, I will generate two such random numbers and then I will just square it.

If I square it, what does it mean? It means I am finding $X^2 + Y^2$ right? I am finding X^2 and Y^2 and then I will total it. I can always do this. Then I can total it. Total will just give me a single number. So basically that is, if I find a point which is sum for X is somewhere here and Y

somewhere here, it is a point like this and the distance from the origin is distance squared is just given by $X^2 + Y^2$ and that is what I am doing.

So, this number clearly can go all the way up to 1 plus 1. That is the maximum. So, then you would be somewhere here. So, then you ask how many of these points that I generate will lie inside this quadrant of the circle. And clearly that will happen whenever this distance is less than or equal to 1. Correct? So, what I can do is I can say if RandomReal total this distance is, this entire thing, I should put it, enclose it in these kind of brackets.

If it is less than or equal to 1, if it is less than or equal to. So, that is so you can get the syntax for this using Palettes and Basic Math Assistant and you can go to, and you can go to these commands here and search around. So, there you go. So, you have less than or equal to. There are, of course, shortcuts as well, but you can also do this. Less than or equal to 1.

Less than or equal to. If this is less than or equal to 1, then you want the answer 1. And if it is not less than or equal to 1, then you want to get a 0. So, let us see. So it gives me a zero once. This random number was most of the times it is going to give me one because it is more likely. But, of course, from time to time we get a 0. And so that is the whole point.

(Refer Time Slide: 11:51)

```
In[13]:= If[(RandomReal[{0, 1}, 2]^2 // Total) <= 1, 1, 0]
Out[13]= 0

dataset = Table[nMax = 1000000;
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total) <= 1, 1, 0], {nMax}] // Total) 4.0, {100}]; //
Histogram[dataset]
```

So, you can keep on doing this many, many times and that is what is done here. So, you make a table of this and do it nMax times and then you count the total. If you sum all this this array, it tells you the total number of 1s there. And that is all you want, really. And times 4, because ultimately what do we want. We have to multiply by 4. We are interested in π . So, I will multiply by 4. And then I have to divide by total number of attempts which is just nMax, and that is it. So, that will create for me a another table. I want to generate a large number of these and I will call that data set.

(Refer Time Slide: 12:28)

The screenshot shows a Mathematica notebook interface. The input cell contains the following code:

```

In[14]:=
dataset = Table[nMax = 1000000;
  1/nMax (Table[If[(RandomReal[{0, 1}, 2]^2 // Total] <= 1, 1, 0), {nMax}] // Total] 4.0, {100}); // Timing
Histogram[dataset]

```

The output cell shows the result of the code execution:

```

Out[14]=
{99.8467, Null}

```

The next output cell shows a histogram of the data generated by the code:

```

Out[15]=

```

The histogram displays the distribution of the estimated values of π . The x-axis represents the value of π , with major ticks at 3.138, 3.140, 3.142, 3.144, and 3.146. The y-axis represents the frequency, with major ticks at 0, 5, 10, 15, 20, and 25. The distribution is unimodal and centered around 3.14, with the highest frequency occurring between 3.140 and 3.142.

Below the histogram, there is a caption:

• The Histogram above tells us how accurately we have determined the value of π using the Monte Carlo Method.

So, let me do this. So, let me get rid of this Math Assistant. So, now I have this whole thing. Timing is just done to keep track of how much time it takes. Ya, so it is evaluating. May take a minute. Yes, there you go. So, the answer is, is out there and so you see that it is a histogram. And I have a dataset with, with lots of numbers which are you know crowded around this 3.14. So, not only have I just got an estimate of this. But it also gives me a histogram in the end. So, it tells us how accurately we have determined the value of π using the Monte Carlo method.

(Refer Time Slide: 13:41)

```
In[18]:= TableForm[dataset]

Out[18]//TableForm=
  3.32
  2.88
  3.32
  3.
  3.32
  3.28
  3.28
  2.88
  3.24
  3.16
  3.
  3.4
  3.24
```

```
Out[13]=
  0

In[16]:= dataset = Table[nMax = 100;
   $\frac{1}{nMax} \text{Table}[\text{If}[(\text{RandomReal}[\{0, 1\}, 2]^2 // \text{Total}] \leq 1, 1, 0], \{nMax\}] // \text{Total} 4.0, \{100\}]; // \text{Timing}$ 
  Histogram[dataset]

Out[16]=
  {0.021099, Null}

In[19]:= Mean[dataset]

Out[19]=
  3.1412
```

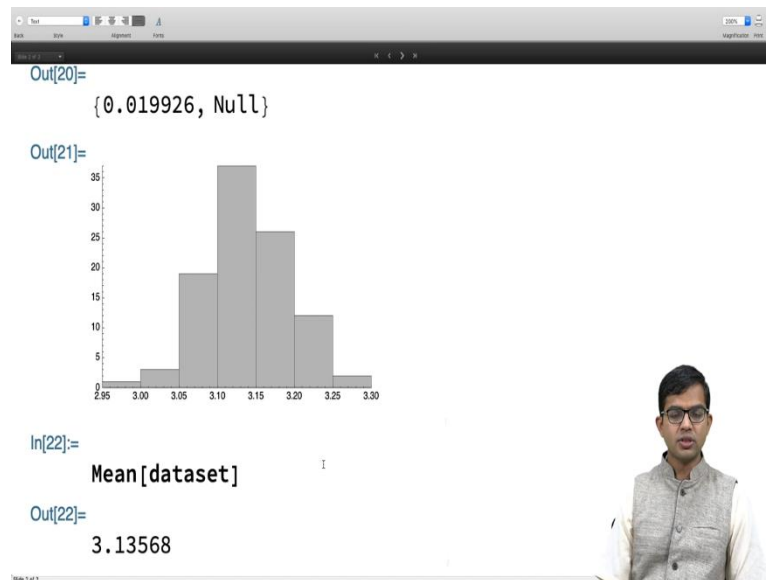
So, let us actually take a look at what this dataset looks like. It has a large number of quantities. Let me just reduce this number. Then you want to get a feeling for what is going on here. We can actually start with let us say just a hundred such numbers. And let me evaluate this.

If I do it, then, of course, it is a little bit more skewed and so it is giving me. So, let us say table form. If I do table form, I can go ahead and make this input and I will say table form of dataset. So, there you see you got a whole bunch of numbers which are all in the ballpark of you know 3.14.

So, I can just simply take an average of all this. Let me just take a mean of this instead of table form. Let me take the mean of this. And mean is indeed very close to the value of π that we all already know.

So, I want you to go back and take this piece of code and work this out for yourself. So, so timing is also put in here and you can play this game and see how, how good this value is. For example, you can make a plot of you know the mean of this quantity as a function of nMax.

(Refer Time Slide: 15:12)



So, with just 100, I am already getting a very good value. Let me see what happens if I have a 1000. If I have a 1000 and then if I do mean of this data set, then it is around the same ballpark. So, this is a game that you can play and you can actually do very sophisticated things. You can find error bars. You can find out how the convergence is. You can play with you know there are 2 quantities involved here. One is nMax.

And the other one is, so there is, so many so many pins are getting dropped. So, and you can find error bars using spread around a mean value, using in the histogram and so on. You can do more sophisticated approaches. And another useful task would be to compare the results of this method versus that obtained from the Importance sampling method using the integral.

And ultimately, they are all sort of similar in flavour. And so you might expect that the computational labor involved and the accuracy that comes out will be sort of comparable. That it is a worthy exercise to try out. Okay, so, that is what this small module is about. It is yet another example of the Monte Carlo method to compute π . Thank you.