

# Physics through Computational Thinking

Dr. Auditya Sharma & Dr. Ambar Jain

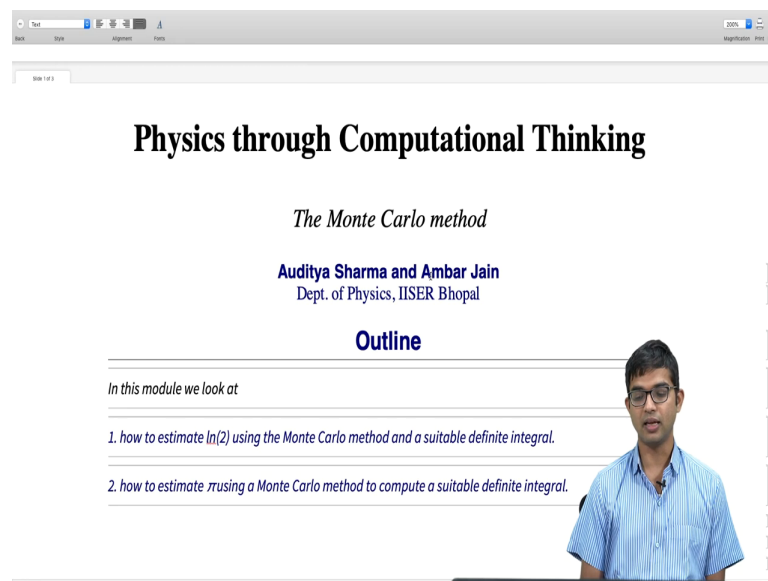
Department of Physics

Indian Institute of Science Education and Research, Bhopal

Lecture 40

The Monte Carlo Method - 2

(Refer Slide Time: 0:40)



The screenshot shows a presentation slide with the following content:

- Slide 1 of 1
- Physics through Computational Thinking
- The Monte Carlo method
- Auditya Sharma and Ambar Jain  
Dept. of Physics, IISER Bhopal
- Outline
- In this module we look at
- 1. how to estimate  $\ln(2)$  using the Monte Carlo method and a suitable definite integral.
- 2. how to estimate  $\pi$  using a Monte Carlo method to compute a suitable definite integral.

A video inset in the bottom right corner shows a man in a blue shirt speaking.

Hi guys, so this is the next module on The Monte Carlo Method, so in this one we will do something very fun I think and that is to use the Monte Carlo Sampling Method to evaluate a definite integral and using this get an estimate for  $\log 2$  and then for  $\pi$ . So, that is the agenda for this module.

So, let me start by just giving you a brief introduction to this method of computing definite integrals using The Monte Carlo Method and so we will just directly, quickly give you some examples and then you see how the method operates and only then later on we can think about why it works at all.

(Refer Slide Time: 1:14)

An elementary integral

Consider the integral

$$I = \int_0^1 \frac{1}{1+x} dx. \quad (1)$$

It is straightforward for us to find this integral. It is simply given by:

$$I = \ln(2). \quad (2)$$

In fact, we can get *Mathematica* to evaluate it for us:

```
In[54]:= Integrate[1/(1+x), {x, 0, 1}]
Out[54]= Log[2]
```

So, suppose I want to integrate this function  $\frac{1}{1+x}$  and the limits are set to 0 and 1, it is very straight forward, I am very sure you can use your pen and paper or even you will be able to write it down instantaneously and the answer is just the log 2, very easy. And so in fact we can also tell mathematica do this for us and tell us what is the answer and I have already done this and you can see that using the integrate function, I have the answer log 2.

So, what I want to do is, show you how with The Monte Carlo Sampling we can get an estimate for this number what does it really to log 2.

So, the method is very simple, all you do is to get a random number generator. So, I did not explicitly say it out last time, but basically the point is that these programs give you what are called pseudo random numbers.

So, since the program is able to do some operations, some complicated operation and give out a number, there is a deterministic method involved underlying it, because there is a deterministic method involved, you cannot really say that it is random. So, in that sense it is actually, there is a deterministic aspect to it so it is not random.

But on the other hand, it appears to be random, it satisfies all the properties of a random numbers. How do you know it satisfies all the properties? You can look at its distributions, so

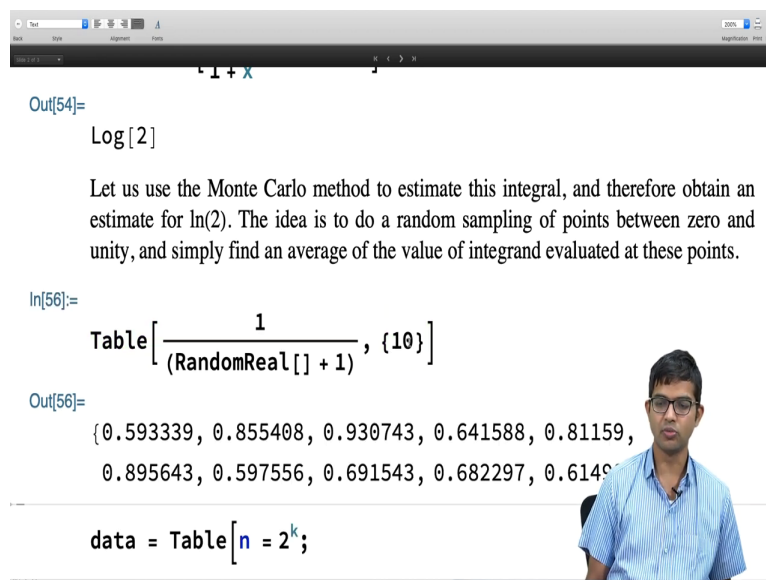
we saw for example histograms, and you can look at its first moment, second moment, third moment, so you can do a range of tests on the pseudo random numbers for generator.

In fact this is the way to benchmark whether a pseudo random number generator is reliable one or not. You need to study its various moments and since it is able to satisfy the properties of the distribution it is suppose to generates, it is indeed, it mimics the full original distribution as closely as you want, so it is a pseudo random number.

So, that is what is used in all these numerical simulations in this Monte Carlo Methods of all kinds. So, now what I will do is call my RandomReal function so which gives me lots of random numbers between 0 and 1 and it will give me a uniform distribution as we have already checked by using the histogram function in previous module.

And then I will simply evaluate this function  $1/(1+x)$  at all of these different random numbers. X between 0 and 1 and take the average, and that is it. So, the formulae will give me an estimate for this integral and you will see that it agrees actually really pretty well in fact for this kind of the method involving the random numbers.

(Refer Slide Time: 4:12)



Out[54]=  
Log[2]

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.

In[56]=  
Table $\left[\frac{1}{(\text{RandomReal}[] + 1)}, \{10\}\right]$

Out[56]=  
{0.593339, 0.855408, 0.930743, 0.641588, 0.81159,  
0.895643, 0.597556, 0.691543, 0.682297, 0.6149}

data = Table[n = 2<sup>k</sup>;

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.


```
Table[ $\frac{1}{(\text{RandomReal}[] + 1)}$ , {n}]
```

Out[56]=

```
{0.593339, 0.855408, 0.930743, 0.641588, 0.81159,
0.895643, 0.597556, 0.691543, 0.682297, 0.614987}
```

```
data = Table[n = 2k;
```

```
{n, Mean[Table[Mean[Table[ $\frac{1}{(\text{RandomReal}[] + 1)}$ 
```



### An elementary integral

Consider the integral

$$I = \int_0^1 \frac{1}{1+x} dx. \quad (1)$$

It is straightforward for us to find this integral. It is simply given by:

$$I = \ln(2). \quad (2)$$


In fact, we can get *Mathematica* to evaluate it for us:

```
In[54]= Integrate[ $\frac{1}{1+x}$ , {x, 0, 1}]
```

Out[54]=

```
Log[2]
```

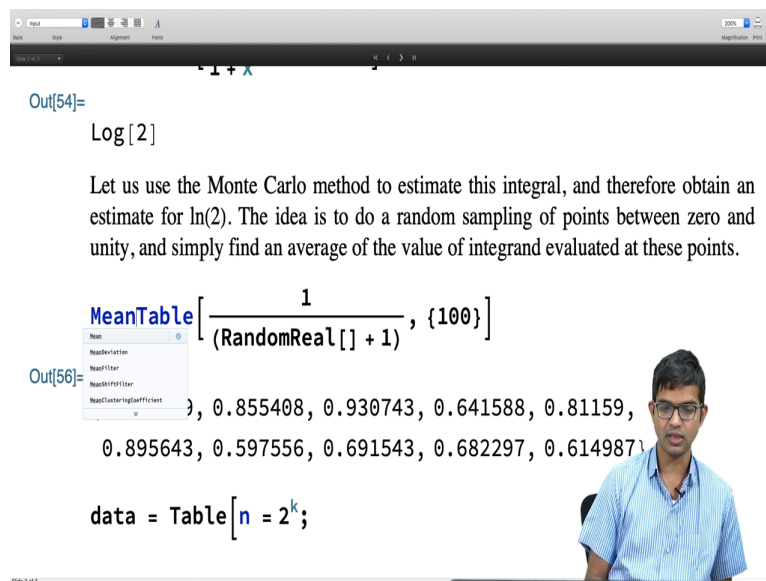
Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ .



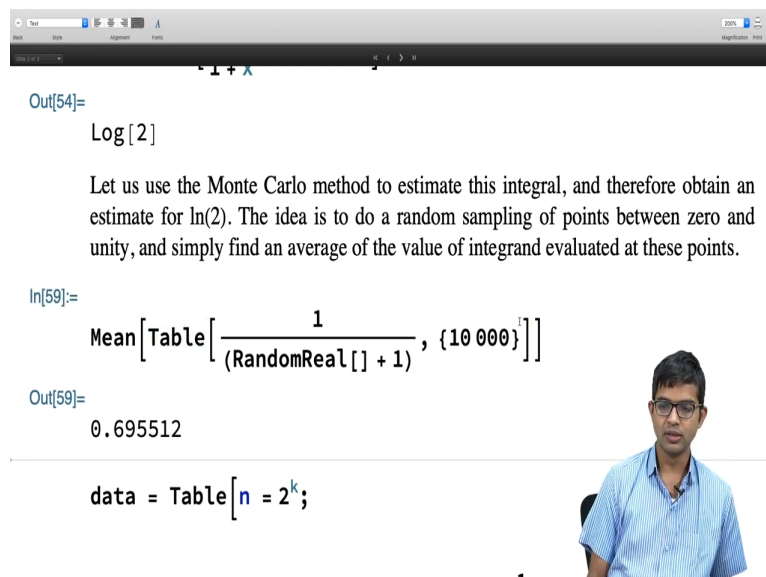
So how am I going to do it? So, let me actually open this up for you. I have a small piece of code, but so what I am doing is really first of all, I am using, so this is random real, so let me copy this out and paste it here. And convert the cell to input. So, if I just do `RandomReal, 1/(RandomReal + 1)` that you understand because it is just generating a random number between 0 and 1 and I am evaluating this function  $1/(1+x)$ .

But I do not want to do it just once, I want to do it many many times. So, am going to create, I am going to table this up, write it as table of this function, I do not know, I can do it 10 times for example, so it will give me a table like this. But I do not want it to be just any number like this, I want to keep this as a variable `n`, and why is that? I will tell you in a moment.

(Refer Slide Time: 5:23)



```
Out[54]=  
Log[2]  
  
Let us use the Monte Carlo method to estimate this integral, and therefore obtain an  
estimate for ln(2). The idea is to do a random sampling of points between zero and  
unity, and simply find an average of the value of integrand evaluated at these points.  
  
MeanTable[ $\frac{1}{(\text{RandomReal}[] + 1)}$ , {100}]  
Out[56]=  
{0.855408, 0.930743, 0.641588, 0.81159, 0.895643, 0.597556, 0.691543, 0.682297, 0.614987, ...}  
  
data = Table[n = 2k;
```



```
Out[54]=  
Log[2]  
  
Let us use the Monte Carlo method to estimate this integral, and therefore obtain an  
estimate for ln(2). The idea is to do a random sampling of points between zero and  
unity, and simply find an average of the value of integrand evaluated at these points.  
  
In[59]:=  
Mean[Table[ $\frac{1}{(\text{RandomReal}[] + 1)}$ , {10000}]]  
Out[59]=  
0.695512  
  
data = Table[n = 2k;
```

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.

```

Table[n = 2^k;
{n, Mean[Table[1/(RandomReal[] + 1), {n}]]]


```

Out[59]=  
0.695512

```

data = Table[n = 2^k;
{n, Mean[Table[Mean[Table[1/(RandomReal[] + 1), {n}]]], {n}]]

```



So, once I have computed this, I told you that; so let me illustrate what I want to do. I will generate 100 of these and then find the mean of this. OK. So, that is all, that is the Monte Carlo Method. So, that should already give me a pretty good estimate, and if I make this 100 to a 1000, it should become better and better and it should go to  $\log 2$ .

But I want to also study this systematically, and that is why I want to keep this as  $n$ , so what I will do is that I will make this  $n$  and then I will store both  $n$  and mean of this and actually make a table out of this itself. I want  $n$  itself to vary as a power of 2 so I will first of all define  $n$  as  $2^k$ , that is what I have done here, and then I want to store my value of  $n$ , comma this mean and then.

(Refer Slide Time: 6:28)

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.


$$\text{Mean}\left[\text{Table}\left[\frac{1}{(\text{RandomReal}[] + 1)}, \{100\}\right]\right]$$

... Table: Iterator {n} does not have appropriate bounds.

Out[60]=

$$\text{Mean}\left[\text{Table}\left[\frac{1}{\text{RandomReal}[] + 1}, \{n\}\right]\right]$$

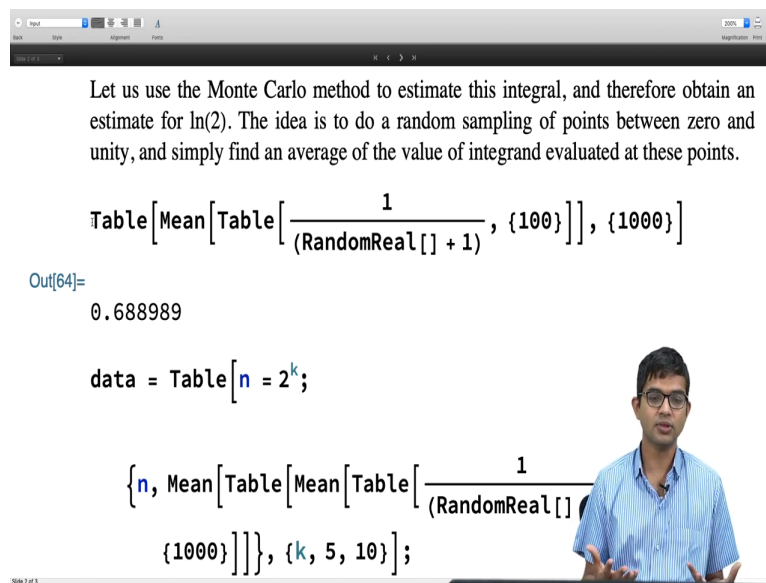
$$\text{data} = \text{Table}[n = 2^k;$$

$$\{n, \text{Mean}\left[\text{Table}\left[\text{Mean}\left[\text{Table}\left[\frac{1}{(\text{RandomReal}[] + 1)}, \{n\}\right]\right], \{n\}\right]\right]$$


So, there is 1 more step involved which is that I can ask myself so before I go to this, let me actually open this up. So, I can ask myself, so I have computed this mean, so it does not understand what n is.

So, let us say 100, so if I have 100, it has computed this mean for me, if I do it again it will; but I can ask myself what would happen or what kind of numbers do I get if I do this many times? So, I want to create a table of this itself, a table of these means.

(Refer Slide Time: 6:56)



Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.

$$\text{Table}\left[\text{Mean}\left[\text{Table}\left[\frac{1}{(\text{RandomReal}[] + 1)}, \{100\}\right]\right], \{1000\}\right]$$

Out[64]=  
0.688989

$$\text{data} = \text{Table}\left[n = 2^k;$$
$$\left\{n, \text{Mean}\left[\text{Table}\left[\text{Mean}\left[\text{Table}\left[\frac{1}{(\text{RandomReal}[] + 1)}, \{1000\}\right]\right]\right], \{k, 5, 10\}\right];$$

So, I will create a table of these means running this Monte Carlo simulation many many times if you want. Let me fix this number to be, I do not know 1000, I have fixed there, but it is something that you can play with. So, having got this, I want to find the mean of this itself, so this is another mean finally, and this is the number I will store.

So, there are two kinds of means here, one is the mean which is computed in my full Monte Carlo simulation, I just do a Monte Carlo simulation over many steps, I have sampled the region between 0 and 1 uniformly many many times, found the average of this and I am claiming that is a measure of this. But, I do this whole exercise itself a 1000 times and find its mean, that is what this is.




(Refer Slide Time: 7:55)

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\ln(2)$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points.

```
data = Table[n = 2k;  
{n, Mean[Table[Mean[Table[  
1  
(RandomReal[] + 1), {n}]]],  
{1000}]]], {k, 5, 8}]
```

Out[64]=  
0.688989

```
data = Table[n = 2k;
```




And finally, I want to store this number as I vary  $n$ . So, now I will replace this 100 by  $n$  and this  $n$  itself I like to use powers of 2. So, data is equal to table of  $n = 2^k$  and then I will store up these two values  $n$  and mean and then I will allow this  $k$  to go from 5 to 8 let us say and then I close this table. So, that is all, so this is a very compact code.

(Refer Slide Time: 8:47)

```
data = Table[n = 2k;  
{n, Mean[Table[Mean[Table[  
1  
(RandomReal[] + 1), {n}]]],  
{1000}]]], {k, 5, 10}];
```

TableForm[data]

Out[68]/TableForm=  
32 0.693576  
64 0.692928  
128 0.692816  
256 0.692935  
512 0.693065  
1024 0.693004



And then maybe I will allow it to go from 5 to 10, and then I will, instead of leaving the data as it is, I will give it a nicer form to this, table form of this data. So, this the font was a bit messed up, so I have  $n$  equal to  $2^k$ , yeah now it should work. There you go.

So, finally I have an answer where I get, if I chose the number of samples within my interval 0 to 1 to be 32, I get you know this 0.693576 then and if I make it to 64, I get another number, or you can see that for larger and larger values of n it is becoming closer and closer to log 2. So, in fact I can even go ahead and study how the error varies.

(Refer Slide Time: 9:47)

512 0.692981  
1024 0.693385

In[69]:= ListPlot[data, Joined -> True]

Out[69]=

errdata = Table[n = 2<sup>k</sup>;

But before I do that, let me just plot this, get a list plot of this, and you see that indeed it is converging to a number which we already know it should be log 2. Log 2 is around 0.693.

(Refer Slide Time: 9:58)

errdata = Table[n = 2<sup>k</sup>;

{n,

Mean[

Table[

Abs[Mean[Table[ $\frac{1}{(\text{RandomReal}[] + 1)}$ , {n}], - Log[2]]],

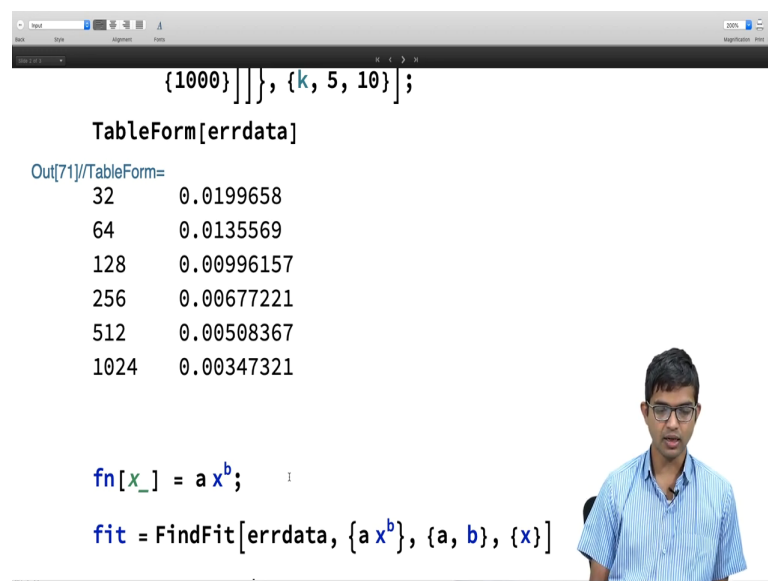
{1000}]]], {k, 5, 10};

TableForm[errdata]

So, in order to find the error data, so what I do is, I just directly subtract this log 2. I subtract this log 2, and then take the absolute value of this. I take the deviation of this Monte Carlo result from log 2 and take the absolute, sometimes it will be higher, sometimes lower than this, so I will just take the absolute deviation, absolute value of the deviation and then only table of this, where I am doing the tabling up with 1000 times, that part I am doing only after I have taken the absolute.

So, this is a, you can go back and check this code and unravel it, like I did for the first one, you should, this is always the standard rule with mathamatica is when you read the mathamatica code is, to take the inner most part and then sort of go outwards and take break it into small pieces, understand each of them and then go inward out and then basically you will know what is going on otherwise it can lo very messy sometimes.

(Refer Slide Time: 11:06)



```

{1000}]]], {k, 5, 10}];

TableForm[errdata]

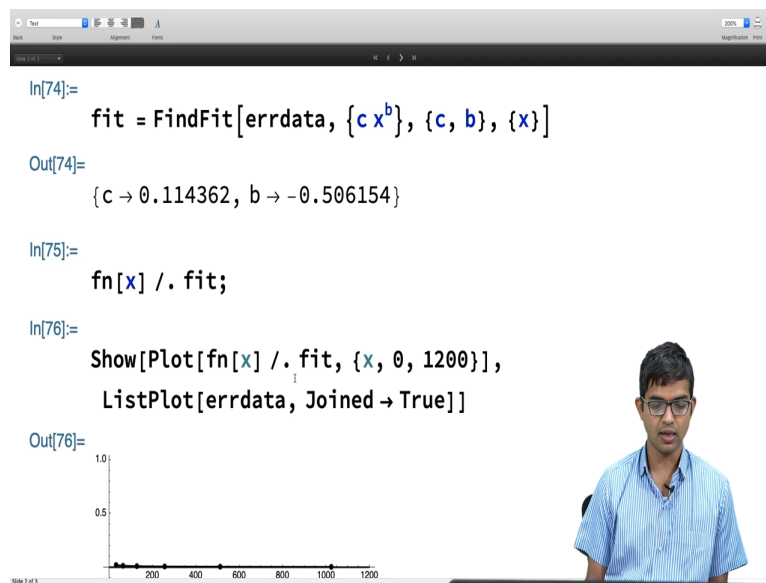
Out[71]/TableForm=
32      0.0199658
64      0.0135569
128     0.00996157
256     0.00677221
512     0.00508367
1024    0.00347321

fn[x_] = a x^b;
fit = FindFit[errdata, {a x^b}, {a, b}, {x}]

```

Alright so there you go, if I do this, I have, then I do the table form of err data, and I get, you know, you see that the error is falling as a function of this n. And then I am interested in understanding in what way does it fall? Alright, so you go back to many modules ago when we described how to get fits of these kinds of data. So, I can actually do a study of how the errors are falling with the sampling number and then I am going to fit it to this form  $ax^b$ .

(Refer Slide Time: 11:43)



The screenshot shows a Mathematica notebook interface. The code input cells are:

```
In[74]:= fit = FindFit[errdata, {c x^b}, {c, b}, {x}]
```

The output for In[74] is:

```
Out[74]:= {c -> 0.114362, b -> -0.506154}
```

The next code input is:

```
In[75]:= fn[x] /. fit;
```

The next code input is:

```
In[76]:= Show[Plot[fn[x] /. fit, {x, 0, 1200}],  
ListPlot[errdata, Joined -> True]]
```

The output for In[76] is a plot showing a function fit and error data. The plot has a y-axis from 0 to 1.0 and an x-axis from 0 to 1200. The function fit is a curve that starts at (0, 1) and decays towards zero. The error data is shown as a series of points connected by lines, which are scattered around the function fit.

So, the problem is, I have used up a for something else, so let me call it by some other; instead of a let me call it c and there you go, so c is this and then find fit, so function has to be, has to use the parameters from in here, and then if I show my plot and this together on the same plot, so by the way before I fix this, so notice that b is a value which is very close to minus 0.5, so that is the point.

So, basically it goes as 1 over into the half, c is not very important but the fact that b is very close to minus a half, that is something instructive here. So, in fact the errors fall off as  $1/\sqrt{n}$ . So, why is this plot not showing up.

(Refer Slide Time: 12:40)

```

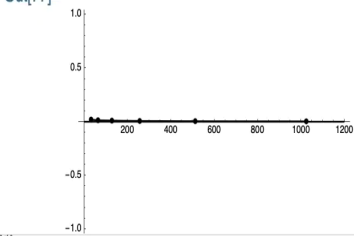
(c → 0.114362, b → -0.506154)

In[75]:=
fn[x] /. fit;

In[77]:=
Show[Plot[fn[x] /. fit, {x, 0, 1200}],
ListPlot[errdata, Joined → True]]

Out[77]=

```



Slide 2 of 3

```

256    0.00677221
512    0.00508367
1024   0.00347321

ListPlot[errdata, Joined → True]

fn[x_] = c x^b;

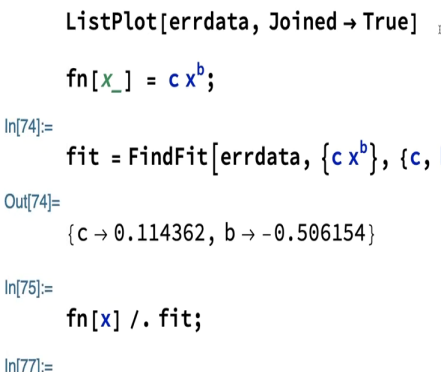
In[74]:=
fit = FindFit[errdata, {c x^b}, {c, b}, {x}]

Out[74]=
(c → 0.114362, b → -0.506154)

In[75]:=
fn[x] /. fit;

In[77]:=

```



Slide 2 of 3

```

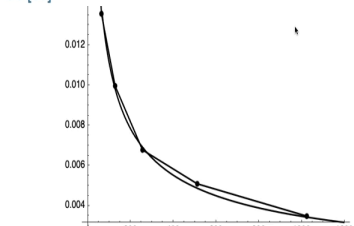
fn[x] /. fit

Out[81]=
0.114362
x^-0.506154

In[82]:=
Show[Plot[fn[x] /. fit, {x, 0, 1200}],
ListPlot[errdata, Joined → True]]

Out[82]=

```



Slide 2 of 3

So, let me plot only this. Suppose I plot only; I think this point is, this should work out. I got to make this into input, so there you go, so this part is working out. So, there you go, it all works out. So, indeed the fit is very good and it goes as  $1/\sqrt{n}$ , so that is the main message from this, in this exercise.

(Refer Slide Time: 13:39)

**Estimating  $\pi$**

Now, let us consider the integral

$$I = \int_0^1 \frac{4}{1+x^2} dx. \quad (3)$$

It is straightforward for us to find this integral. It is simply given by:

$$I = \pi. \quad (4)$$

We can get *Mathematica* to check it for us:

```
In[98]:= Integrate[ $\frac{4}{1+x^2}$ , {x, 0, 1}]
```

```
Out[98]=  $\pi$ 
```

Slide 1 of 3

So, let us quickly look at another example, in this we want to estimate  $\pi$  using the same technique, with Monte Carlo sampling. So, once again I have conveniently chosen this interval from 0 to 1 and this function  $\frac{4}{1+x^2}$ , this integral is also should be very familiar, so it is  $4 \tan^{-1}(x)$ , and it is going to give you  $\tan^{-1}(\pi)$ ,  $\tan^{-1}(1) = \pi/4$  so  $4 \pi/4$  is just  $\pi$ . So, if you do this integral numerically with Monte Carlo integration, it should give you an estimation of  $\pi$ . And, we can of course quickly get mathematica to check this for us, indeed it is  $\pi$ .

(Refer Slide Time: 14:18)

Let us use the Monte Carlo method to estimate this integral, and therefore obtain an estimate for  $\pi$ . The idea is to do a random sampling of points between zero and unity, and simply find an average of the value of integrand evaluated at these points, as before.

```
In[84]:= data = Table[n = 2k;
```

$$\left\{ n, \text{Mean}\left[ \text{Table}\left[ \text{Mean}\left[ \text{Table}\left[ \frac{4}{(\text{RandomReal}[])^2 + 1}, \{n\}\right], \{1000\}\right] \right], \{k, 5, 10\}\right];$$

```
TableForm[data]
```

```
Out[85]/TableForm=
```

Slide 1 of 3

```

data = Table[n = 2^k;

{n, Mean[Table[Mean[Table[

$$\frac{4}{(\text{RandomReal}[])^2 + 1}$$


, {n}]]],
{1000}]]], {k, 5, 10}];

TableForm[data]

```

Out[100]/TableForm=

|      |         |
|------|---------|
| 32   | 3.14153 |
| 64   | 3.14725 |
| 128  | 3.14073 |
| 256  | 3.14092 |
| 512  | 3.14318 |
| 1024 | 3.1414  |



```

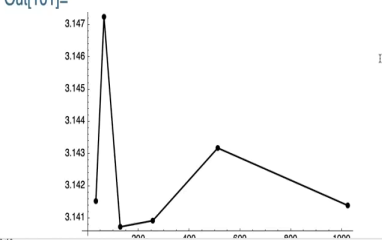

64 3.14725
128 3.14073
256 3.14092
512 3.14318
1024 3.1414

```

In[101]:=

```
ListPlot[data, Joined -> True]
```

Out[101]=

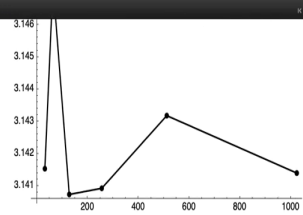



And we will use basically everything from the previous example. So, I will directly use the code from there and if you want to understand all the nitty-gritty in here, you can actually go back to that part of this video and check what this code is really doing.


So, only thing that has changed here is instead of function  $\frac{1}{1+x}$  I had, in the previous example, now  $\frac{4}{1+x^2}$ . RandomlyReal Square comes in and everything else is the same. So, if I run this you will see that it is nicely converging to  $\pi$ . So, 3.1415 already from 32 onwards, and to once again I can do a ListPlot of this, and indeed it looks very nice.



(Refer Slide Time: 15:06)

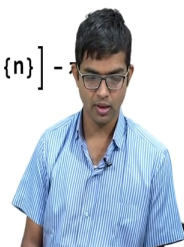


```
In[87]:=  
errdata = Table[n = 2^k;  
  
  {n,  
   Mean[  
     Table[
```



Slide 3 of 3

```
In[87]:=  
errdata = Table[n = 2^k;  
  
  {n,  
   Mean[  
     Table[  
       Abs[Mean[Table[  
          $\frac{4}{(\text{RandomReal}[]^2 + 1)}$ , {n}] -  $\frac{4}{5}$   
         {100}]]], {k, 5, 10}];  
   TableForm[errdata];
```



Out[88]//TableForm=

Slide 3 of 3

```

Abs[Mean[Table[ $\frac{1}{(\text{RandomReal}[])^2 + 1}$ , {n}], - $\pi$ ],
{1000}]]], {k, 5, 10}];

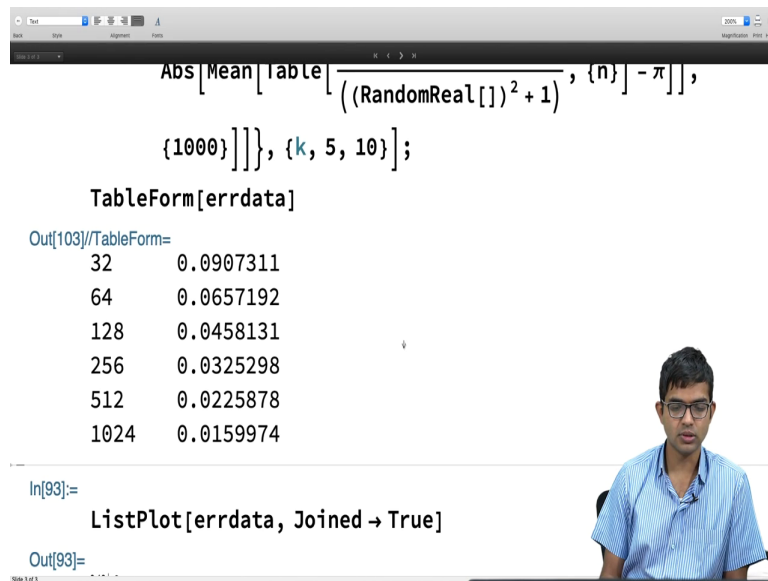
TableForm[errdata]

Out[103]/TableForm=
32    0.0907311
64    0.0657192
128   0.0458131
256   0.0325298
512   0.0225878
1024  0.0159974

In[93]:=
ListPlot[errdata, Joined -> True]

Out[93]=

```



So, if I do error data, so again I am using the same type of method as before, to compute the errors I am using the absolute function and subtracting from  $\pi$  here instead of  $\log 2$ .  $\pi$  is the exact value, and what is the deviation from  $\pi$ ? It could be positive or negative, so I am using the absolute function.

And if I go ahead and so once again I can use 1000 here, 100 is also, but let us see what happens if I do 1000 and indeed you see that the error falls off with the number of Monte Carlo steps. As the number of Monte Carlo steps is increasing, we can ask in what precise way does it fall.

(Refer Slide Time: 15:56)

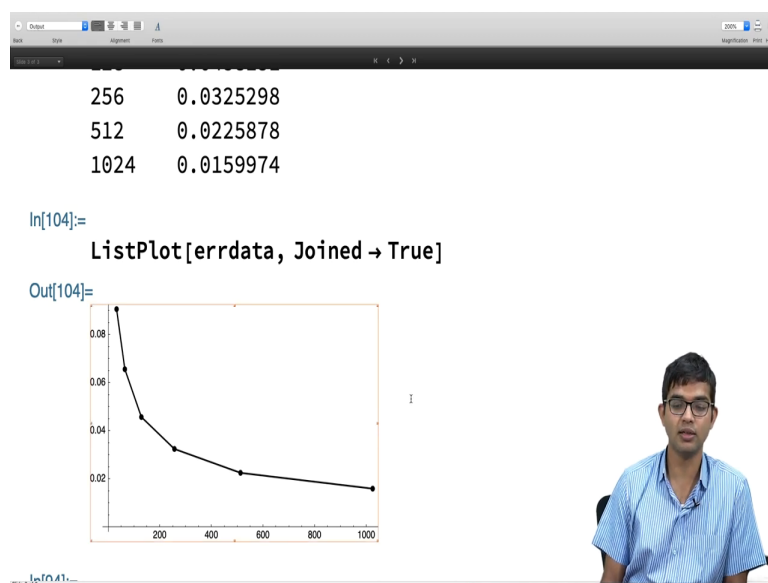
```

256    0.0325298
512    0.0225878
1024   0.0159974

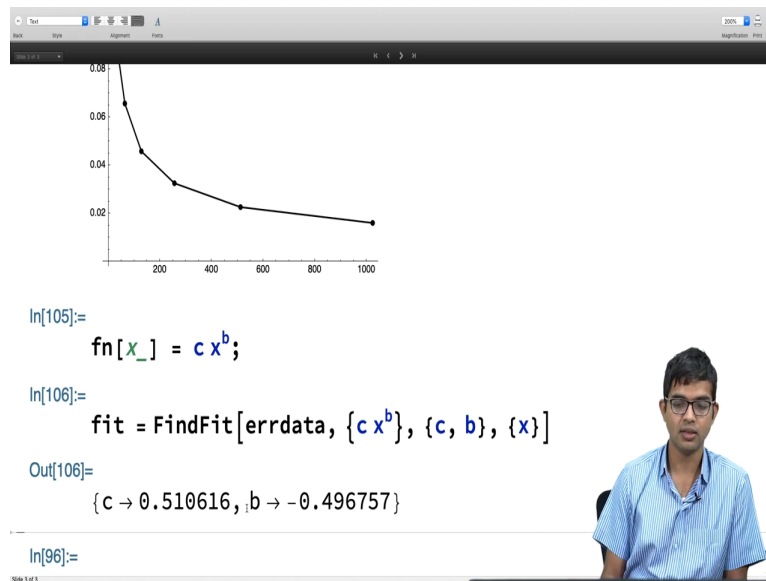
In[104]:=
ListPlot[errdata, Joined -> True]

Out[104]=

```

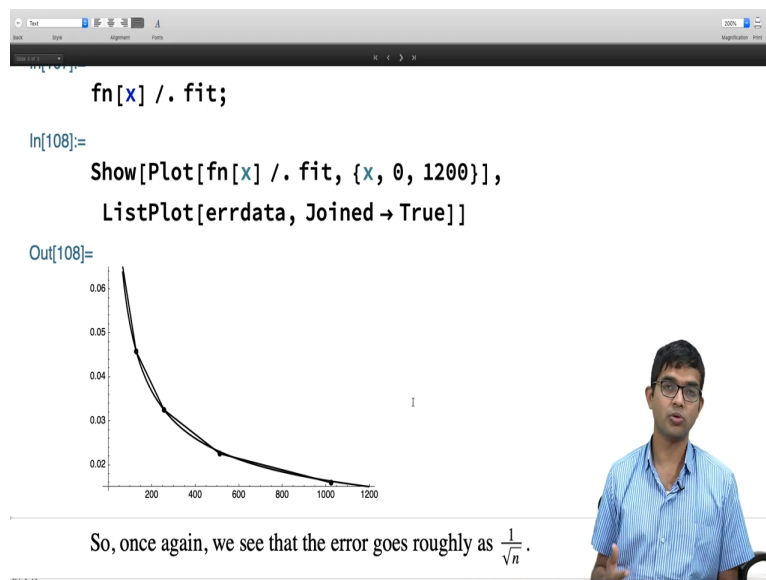


| Number of Monte Carlo Steps (k) | Error (Mean Absolute Deviation) |
|---------------------------------|---------------------------------|
| 32                              | 0.0907311                       |
| 64                              | 0.0657192                       |
| 128                             | 0.0458131                       |
| 256                             | 0.0325298                       |
| 512                             | 0.0225878                       |
| 1024                            | 0.0159974                       |



So, let us quickly plot the error data itself, you see that it is falling off, we saw that it went as  $1/\sqrt{n}$  of n earlier, and maybe the same will hold here. So, we can quickly check this functional form of, which is the power log form. Then I do this fit using the fine fit, indeed I find that the exponent is very close to -1/2.

(Refer Slide Time: 16:24)



So, I can use this to super impose the data set and this fitting function and you see the agreement is excellent. And once again we see that the error goes roughly goes as 1 over square root n. So, there is a theory behind why this is  $1/\sqrt{n}$ , you can work it out and so on, what are, for our purposes it suffices to just see this as an illustration of the Monte Carlo Method.

(Refer Slide Time: 16:51)

**An elementary integral**

Consider the integral

$$I = \int_0^1 \frac{1}{1+x} dx. \quad (1)$$

It is straightforward for us to find this integral. It is simply given by:

$$I = \ln(2). \quad (2)$$

In fact, we can get *Mathematica* to evaluate it for us:

```
In[54]:= Integrate[1/(1+x), {x, 0, 1}]
Out[54]= Log[2]
```

So, what did we do in this module? We saw that if you want to evaluate some definite integrals between some limits  $a$  and  $b$ , you can simply sample in this entire interval, take a uniform distribution and then sample these points, the more points that you sample, the better will be the final value.

So, you sample these points uniformly, and then simply compute the average of the value of the function evaluated at all these points. So, there is a lot of theory you can study the details, maybe if you do it as a like a full fledge course on probability and stochastic processes and all. But for our purposes we just want to see how this works out. This is yet another simple illustration of a very powerful tool.

So, we hope to give you some more examples and get you started off on this Monte Carlo method. So, there are lot of very very sophisticated techniques which are built on this. So, the power of this method should not be diluted by thinking why bother doing these kind of integral which you can do anyway by hand or get mathamatica to do it, or use some other numerical tool.

But the power of this method comes in other context, where other methods will fail and Monte Carlo turns out to be an excellent tool. So, that is it for this module. Thank you.