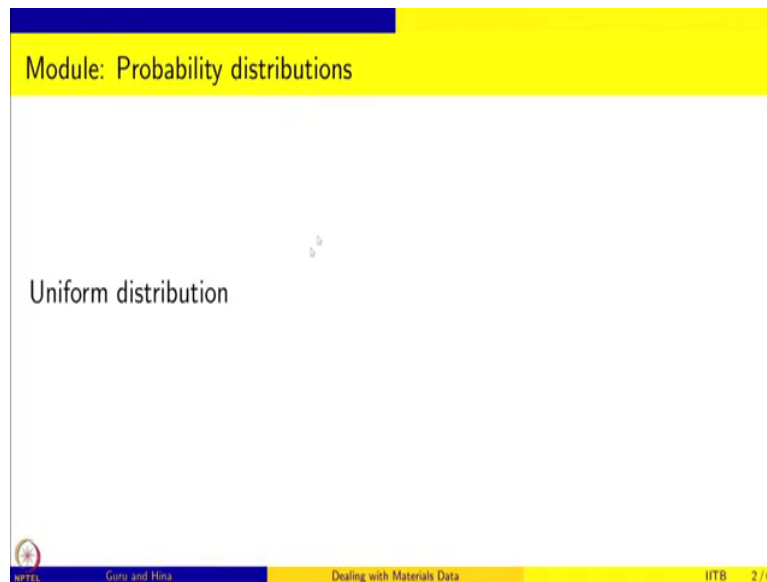


Dealing with Materials Data
Professor M P Gururajan
Professor Hina A Gokhale
Department of Metallurgical Engineering and Materials Science
Indian Institute of Technology, Bombay
Lecture 59
Uniform distribution and
Summary of probability distributions

Welcome to Dealing with Materials Data. In this course, we are looking at the collection, analysis and interpretation of data from Material Science and Engineering. We are in the module on Probability Distributions.

(Refer Slide Time: 0:26)



The slide features a yellow header bar with the text "Module: Probability distributions". The main content area is white and contains the text "Uniform distribution". At the bottom, there is a footer bar with a blue section on the left containing the NPTEL logo and the text "NPTEL Guru and Hina", a yellow section in the middle containing "Dealing with Materials Data", and a blue section on the right containing "IITB 2 / 6".

Value of π simulation

- Uniform: can be discrete or continuous
- `unif` in R
- `extraDistr` library to get `dunif`
- Can be used for simulations

NPTEL Gurus and Hina Dealing with Materials Data IITB 3 / 6

And we are going to talk about uniform distribution. Uniform distribution can either be discrete or continuous. And `unif` is the function call in R for getting the continuous uniform distribution.

And if you include the `extra distr` library, you can also get the discrete uniform function. And these distributions are important because they can be used for simulations and we are going to look at couple of simulations to calculate the value of pi for example.

(Refer Slide Time: 01:06)

Monte Carlo simulation for integration

The diagram shows a light blue square with side length 'a' and a red circle inside it. The area of the square is labeled 'A' and the area of the circle is labeled 'A'. The side length 'a' is indicated by a horizontal line at the bottom of the square.

So, what is the idea? So, let us say that you have square region A and B are the sides. So, rectangular region A and B are the sides of this rectangle. So, you know the area of this region and let us say that you have circular region in the middle with some area A. Now, if you throw random darts at this. Then the number of darts that lie in this region compared to the ones that

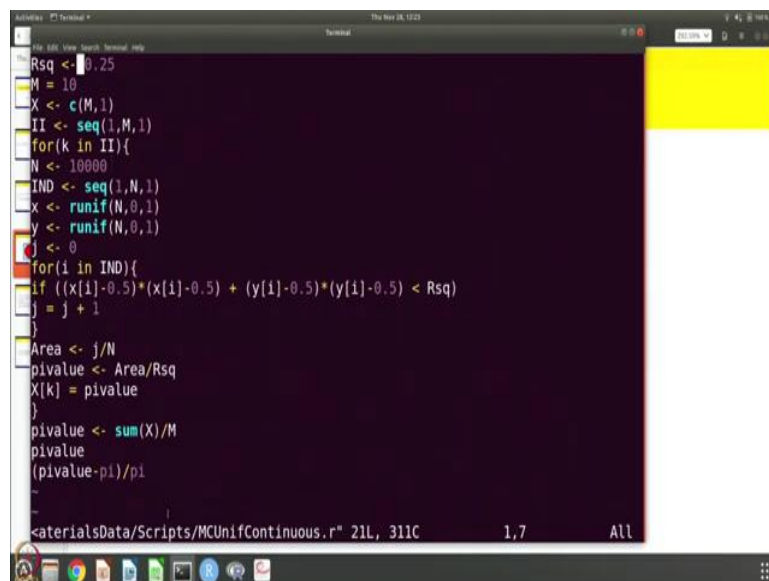
are thrown at this rectangular region, the ratio will give basically the ratio of this shaded area, shaded in red.

So, if you can use the uniform distribution and generate points and from them look at how many lie here and how many you generated, the ratio will basically give the area. And if you know this area you can actually calculate the area.

So, if even if it is some complex shape, for example, you can take the image, you can pixelate it and you can generate and find out whether the generated point has the pixel values let us say 1 or 0, then the ratio will actually give you the area of that region. So, that is a way to do the integration to get the area that is one aspect.

But suppose this is a nice regular shape like a circle and we know what the area is then you can use that to actually calculate the value of pi and that is the simulation that we are going to do. So, let us do those same simulations and come back.

(Refer Slide Time: 03:04)



```
Rsq <- 0.25
M = 10
X <- c(M,1)
II <- seq(1,M,1)
for(k in II){
  N <- 10000
  IND <- seq(1,N,1)
  x <- runif(N,0,1)
  y <- runif(N,0,1)
  j <- 0
  for(i in IND){
    if ((x[i]-0.5)*(x[i]-0.5) + (y[i]-0.5)*(y[i]-0.5) < Rsq)
      j = j + 1
  }
  Area <- j/N
  pivalue <- Area/Rsq
  X[k] = pivalue
}
pivalue <- sum(X)/M
pivalue
(pivalue-pi)/pi
```

And for doing those simulations, I am going to use this to R scripts, so the R squared is basically the square of the radius of the circle that I am considering. So that is 0.25, which means 0.5 is the radius of the circle. So, square of that radius is 0.25. And, we will come back to this M later. And, so let us start from here and we will come back and look at what it is.

So, I am going to say capital N is 10000, okay. So and so we are going to generate 1 to 10000, a sequence of numbers and we are going to generate random variates, uniform variates between

0 and 1, 10000 of them. So, that is why this is there. So every x, y pair then gives you a point in a square of side 1, right.

So, it gives you a points that are lying in a square of area 1. And if you then find out how many of the random points that you generated, were inside this region. I am assuming that at the center of this square is where my circle is centered. So, for every pair that you have generated, you can find out how many of them lie within this region. And so the area fraction is basically the number of points that lie within this to the total number of points that you threw that you generated.

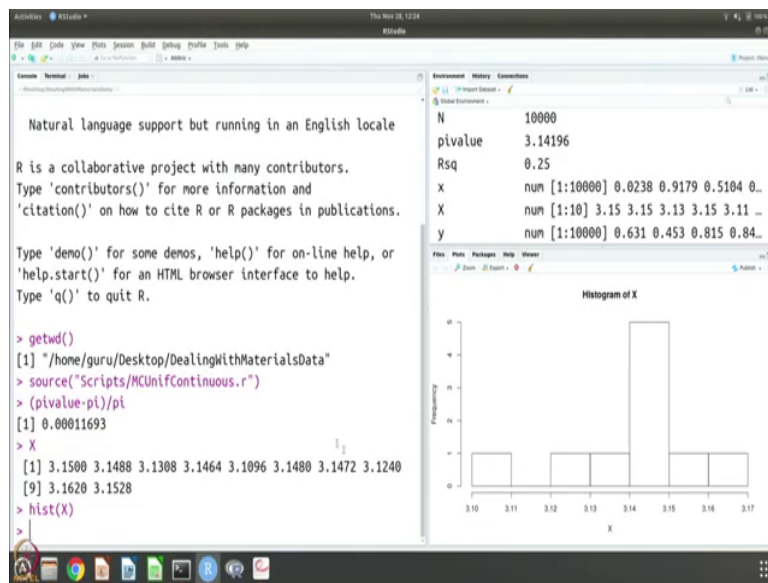
So, the π value is nothing but this area divided by the R^2 because πR^2 is basically this area. So, if you want to get the π value, you can divide by R^2 . And so, what we are doing here is that so, we do 10000 of these pairs and calculate π and we are going to do 10 such simulation. So, this M is basically an outer loop which generates many such π values from simulations and x is a vector.

And so, we are running another loop. So, see how we are running loops, so you have an index, so that is 1 to N . Then for i in index means it will go from 1, 2, 3, etc. Similarly, here it is a sequence that goes from 1 to M . So, for k in i means it will go 1, 2, 3, etc. So, it will take each k value and each k is one simulation.

So, k equal to 1 for example is the first simulation and it will show the π value in x_1 and k equal to 2 is the second simulation and it will store the π value in the second simulation in x_2 etc. So, finally, to get the π value, we sum all these π values that you get from different simulations and divide by the total number of simulations you have done. So, we can then get the π value.

You can also get the relative error π value minus π and divided by π will give you the relative error that we are getting by the doing this simulations.

(Refer Slide Time: 06:35)



The screenshot shows the RStudio interface. The main editor window contains the following text:

Natural language support but running in an English locale

R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.

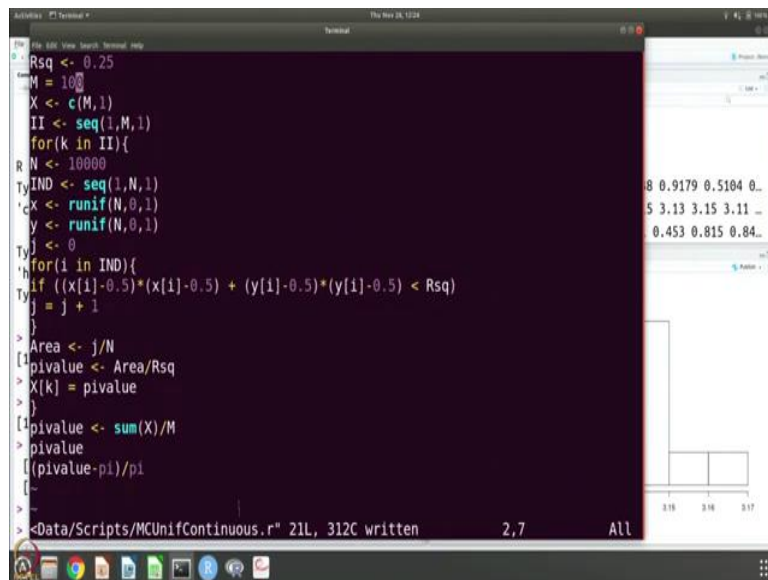
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help. Type 'q()' to quit R.

```
> getwd()
[1] "/home/guru/Desktop/DealingWithMaterialsData"
> source("Scripts/MCUnifContinuous.r")
> (pvalue-pi)/pi
[1] 0.00011693
> X
[1] 3.1500 3.1488 3.1308 3.1464 3.1096 3.1480 3.1472 3.1240
[9] 3.1620 3.1528
> hist(X)
>
```

The Environment pane on the right shows the following variables:

N	10000
pvalue	3.14196
Rsqr	0.25
X	num [1:10000] 0.0238 0.9179 0.5104 0...
X	num [1:10] 3.15 3.15 3.13 3.15 3.11 ...
y	num [1:10000] 0.631 0.453 0.815 0.84...

The Histogram of X plot shows the frequency distribution of the variable X. The x-axis is labeled 'X' and ranges from 3.10 to 3.17. The y-axis is labeled 'Frequency' and ranges from 0 to 10. The histogram consists of several bars, with the tallest bar centered around 3.14.



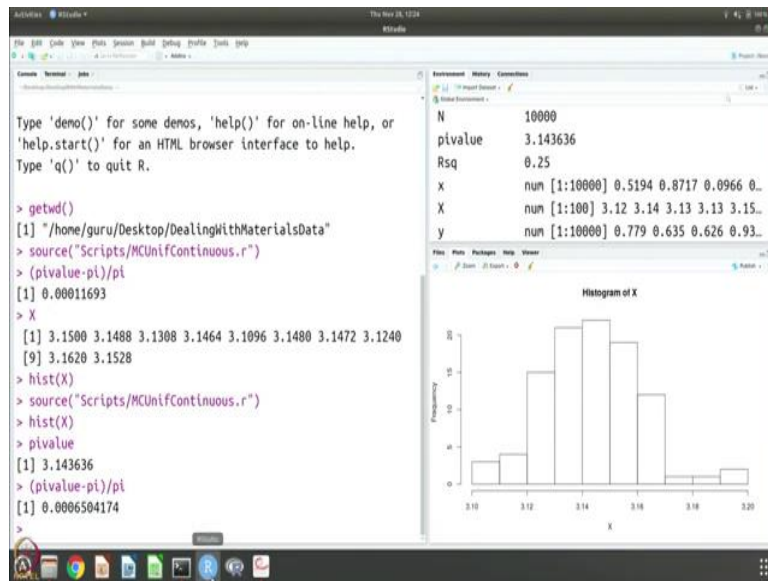
The screenshot shows the RStudio interface with a terminal window displaying R code for a Monte Carlo simulation. The code is as follows:

```
Rsqr <- 0.25
M = 1000
X <- c(M,1)
II <- seq(1,M,1)
for(k in II){
R N <- 10000
Ty IND <- seq(1,N,1)
x <- runif(N,0,1)
y <- runif(N,0,1)
Tyj <- 0
for(i in IND){
if ((x[i]-0.5)*(x[i]-0.5) + (y[i]-0.5)*(y[i]-0.5) < Rsqr)
Tyj = j + 1
}
Area <- j/N
pvalue <- Area/Rsqr
X[k] = pvalue
}
pvalue <- sum(X)/M
pvalue
((pvalue-pi)/pi)
[-]
>
```

The terminal output shows the following results:

```
Rsqr <- 0.25
M = 1000
X <- c(M,1)
II <- seq(1,M,1)
for(k in II){
R N <- 10000
Ty IND <- seq(1,N,1)
x <- runif(N,0,1)
y <- runif(N,0,1)
Tyj <- 0
for(i in IND){
if ((x[i]-0.5)*(x[i]-0.5) + (y[i]-0.5)*(y[i]-0.5) < Rsqr)
Tyj = j + 1
}
Area <- j/N
pvalue <- Area/Rsqr
X[k] = pvalue
}
pvalue <- sum(X)/M
pvalue
((pvalue-pi)/pi)
[-]
>
```

The histogram plot on the right shows the frequency distribution of the variable X. The x-axis is labeled 'X' and ranges from 3.10 to 3.17. The y-axis is labeled 'Frequency' and ranges from 0 to 10. The histogram consists of several bars, with the tallest bar centered around 3.14.



So, let us first start and run this simulation. And, so let us find out which directory we are in. So, we are in the right directory. So, we have to source and it is in the scripts directory. And we have uniform continuous.org. So we have source it and the pi value is 3.14196 is what it is giving you.

And if you calculate, so you get the error to be point 0.0001693. So, you can also look at the x. So, these are the 10 simulations that we did and it gave 15, 14, 130, 814, 64 etc. And you can look at the, so it gives you that it is peaking somewhere around 3.14. Of course, you can do more simulations.

Let us do that. Let us generate some 100 points and run the simulation. So, now let us get the histogram of x. So, you can see that it peaks around 14. And because we have done more simulations, this time, pi should be much better, what is a pi value? So, we got this and let us calculate the error in the pi value. So, it is 0.00065, right?

(Refer Slide Time: 08:18)

```

Rsq <- 0.25
M = 1000
X <- c(M,1)
TyII <- seq(1,M,1)
for(k in TyII){
  TyN <- 10000
  IND <- seq(1,N,1)
  x <- runif(N,0,1)
  y <- runif(N,0,1)
  } <- 0
  for(i in IND){
    if ((x[i]-0.5)*(x[i]-0.5) + (y[i]-0.5)*(y[i]-0.5) < Rsq)
    {
      j = j + 1
    }
  }
  Area <- j/N
  pvalue <- Area/Rsq
  X[k] = pvalue
}
pvalue <- sum(X)/M
pvalue
1*(pvalue-pi)/pi

```

```

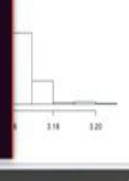
[1] "/home/guru/Desktop/DealingWithMaterialsData"
> source("Scripts/MCUnifContinuous.r")
> (pvalue-pi)/pi
[1] 0.00011693
> X
[1] 3.1500 3.1488 3.1308 3.1464 3.1096 3.1480 3.1472 3.1240
[9] 3.1620 3.1528
> hist(X)
> source("Scripts/MCUnifContinuous.r")
> hist(X)
> pvalue
[1] 3.143636
> (pvalue-pi)/pi
[1] 0.0006504174
> source("Scripts/MCUnifContinuous.r")
> pvalue
[1] 3.142121
> (pvalue-pi)/pi
[1] 0.0001681142
> hist(X)

```

And so, you can similarly go on doing more and more simulations, so maybe 1000 of them and so, it takes some time because it is going to do 1000 simulations and in each it is going to generate 10000 pairs of points. So, you get 142121. And the error is, you can see it is 3 decimal, so it is gone to the fourth decimal place. And you can generate of course histogram of x and you can see that it is now peaking exactly at 3.14.

(Refer Slide Time: 09:02)

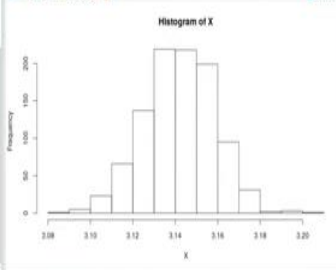
```
Rsq <- 0.25
M = 10000
[X <- c(M,1)
> II <- seq(1,M,1)
> for(k in II){
[N <- 10000
> IND <- seq(1,N,1)
> x <- runif(N,0,1)
> y <- runif(N,0,1)
} <- 0
> for(i in IND){
> if ((x[i]-0.5)*(x[i]-0.5) + (y[i]-0.5)*(y[i]-0.5) < Rsq)
> } = j + 1
}
[1] Area <- j/N
[1] pvalue <- Area/Rsq
[1] X[k] = pvalue
}
}
[1] pvalue <- sum(X)/M
[1] pvalue
[1] (pvalue-pi)/pi
```



A small histogram showing the distribution of X values. The x-axis ranges from approximately 3.14 to 3.18, and the y-axis represents frequency. The distribution is roughly bell-shaped and centered around 3.14.

```
> source("Scripts/MCUnifContinuous.r")
> (pvalue-pi)/pi
[1] 0.00011693
> X
[1] 3.1500 3.1488 3.1308 3.1464 3.1096 3.1480 3.1472 3.1240
[9] 3.1620 3.1528
> hist(X)
> source("Scripts/MCUnifContinuous.r")
> hist(X)
> pvalue
[1] 3.143636
> (pvalue-pi)/pi
[1] 0.0006504174
> source("Scripts/MCUnifContinuous.r")
> pvalue
[1] 3.142121
> (pvalue-pi)/pi
[1] 0.0001681142
> hist(X)
> source("Scripts/MCUnifContinuous.r")
```

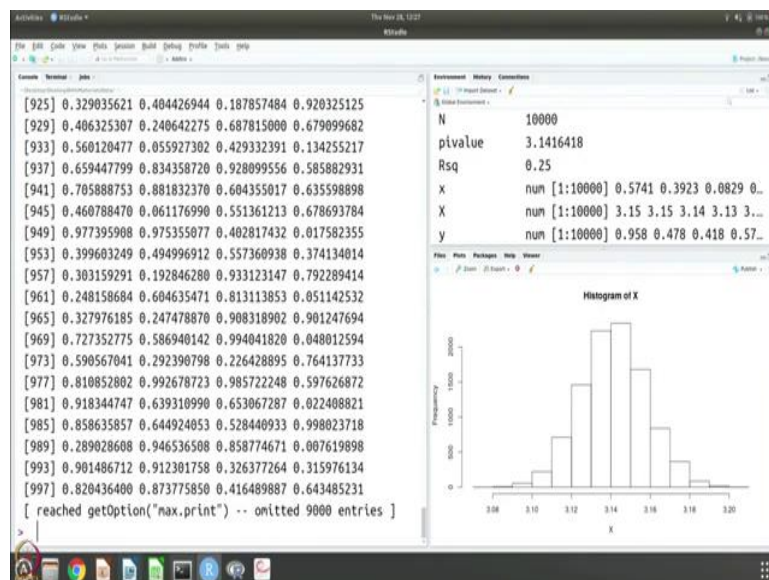
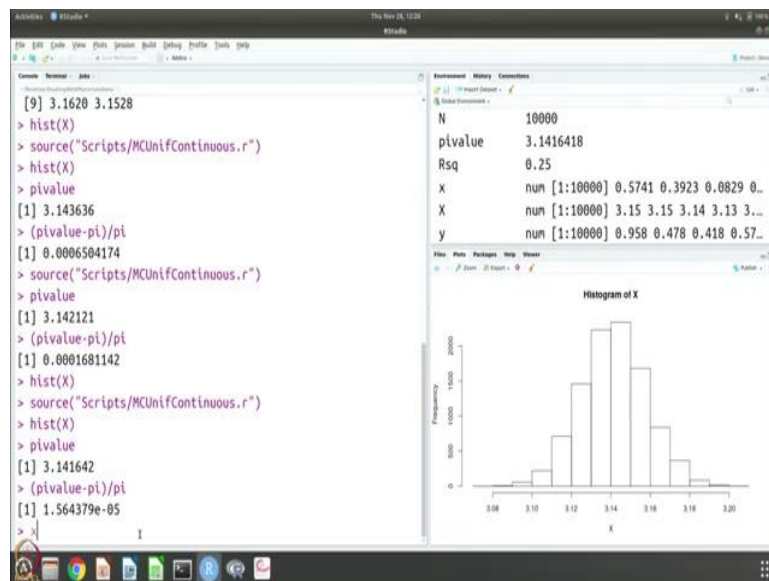
Environment	History	Connections
N	10000	
pvalue	3.1416418	
Rsq	0.25	
X	num [1:10000] 0.5741 0.3923 0.0829 0...	
X	num [1:10000] 3.15 3.15 3.14 3.13 3...	
y	num [1:10000] 0.958 0.478 0.418 0.57...	



A larger histogram titled "Histogram of X". The x-axis is labeled "X" and ranges from 3.08 to 3.20. The y-axis is labeled "Frequency" and ranges from 0 to 2000. The histogram shows a distribution of X values centered around 3.14, with a peak frequency of approximately 2000.

And okay so, let us do one more last simulation, it is going to be very time consuming simulation. But it will generate hopefully. So, error here is 0.017 percent or something. So, let us see if we can get error smaller than this, using these large number of simulations.

(Refer Slide Time: 09:38)



So, let us do the histogram of x. So, it is here and what is the pi value? We get this. And let us calculate the error. So, error has gone to 10 power minus 5, see it was 10 power minus 4 here and it has become 1.5 10 per minus 5, so by doing larger and larger number of stimulations and averaging out, you can have more accurate calculation of pi.

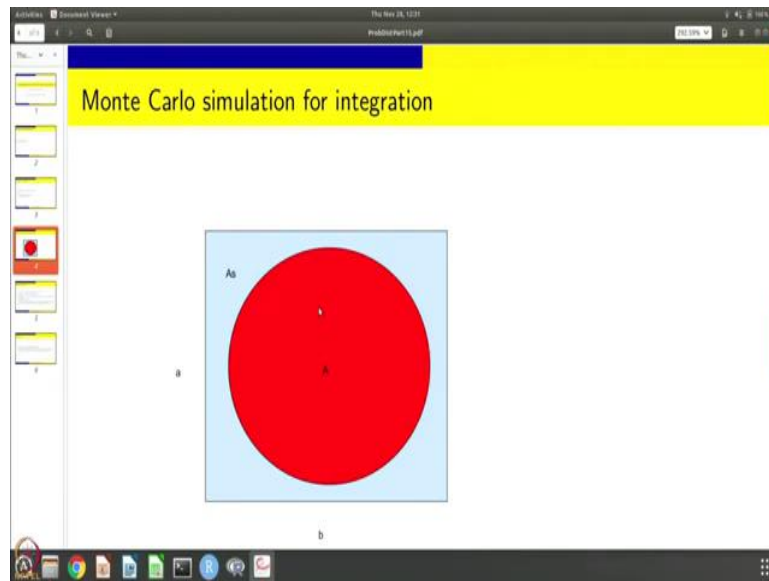
Now, in this case, we actually took an area of 1 by 1 and generated random numbers. So, if you look at the random numbers that are generated like x for example. So, these are all continuous numbers, so it is between 0 and 1, it is generating any lots of numbers, right. That is what you see here.

(Refer Slide Time: 10:31)

```
library(extraDistr)
Rsq = 2500
M = 1000
l <- 100
lby2 <- 0.5*l
X <- c(M,1)
II <- seq(1,M,1)
for(k in II){
  N <- 10000
  IND <- seq(1,N,1)
  x <- rdunif(N,1,l)
  y <- rdunif(N,1,l)
  j <- 0
  for(i in IND){
    if ((x[i]-lby2)*(x[i]-lby2) + (y[i]-lby2)*(y[i]-lby2) <= Rsq)
      j = j + 1
  }
  Area <- j*l*l/N
  pvalue <- (Area/Rsq)
  X[k] = pvalue
}
pvalue <- sum(X)/M
<lsData/Scripts/MCUnifDiscrete.r" 25L, 371C written 3,8 Top
```

```
[973] 0.590567041 0.292390798 0.226428895 0.764137733
[977] 0.010852802 0.992678723 0.985722248 0.597626872
[981] 0.918344747 0.639310990 0.653067287 0.022408821
[985] 0.858635857 0.644924053 0.528440933 0.998023718
[989] 0.289028608 0.946536508 0.858774671 0.007619898
[993] 0.901486712 0.912301758 0.326377264 0.315976134
[997] 0.820436400 0.873775850 0.416489887 0.643485231
[ reached getOption("max.print") -- omitted 9000 entries ]
> source("Scripts/MCUnifDiscrete.r")
> hist(X)
> pvalue
[1] 3.159955
> (pvalue-pi)/pi
[1] 0.005845056
> source("Scripts/MCUnifDiscrete.r")
> hist(X)
> source("Scripts/MCUnifDiscrete.r")
> hist(X)
> (pvalue-pi)/pi
[1] -0.001323613
>
```

Variable	Value
N	10000
pvalue	3.1374344
Rsq	2500
X	num [1:10000] 10 70 58 23 78 35 58 2...
X	num [1:1000] 3.14 3.17 3.16 3.1 3.16...
y	num [1:10000] 65 10 18 80 67 47 48 8...



Now, you can also generate discrete random numbers. So to do that, we are going to look at the, so here again the structure of the code is very same. And but, what we are going to do is that we are going to use this library extra distr, so that it will allow us to generate these discrete random numbers.

And, we are going to do the same thing. So, we are going to have an R square which is 25 and we are going to have M of 10000, it is going to take obviously long time and in length L is 10. So, that is the size of my simulation box 10 by 10 and so 0.5 L 5 comma 5 is where the center of my circle is. So, the circle itself is of radius r. So, that is why it is 25.

But, now the random numbers are generated are discrete and that they are between 1 and L okay. And so we are going to calculate the area but now because the area fraction should also have L squared. So, j by n is basically the area fraction multiplied by the area which we know is L squared. So, that will be the area and if you divide the area by the R square, you will get the pi value and we are going to generate some 10000 pi values and then we are going to use it to evaluate the pi.

So, let us see how the simulation works. So, this is uniform, but this is not discrete at all. Of course, there are 10000 simulation so it is going to take a long time to do. And it is also slightly relatively slower than the continuous random uniform distribution variates but the idea is the same. And, so instead of having circle of 0.5 radius on a 1 by 1 square, we have a 10 by 10 square on which we have a circle of radius 5.

So, relative areas are the same. So, we should have the, so let us now look at how the histogram of the x looks in this case. So, it is also, as you can see in this case, start peaking at 3.1, it is

peaking at 3.16 and you can see the pi value is also 3.16. And so, you calculate the error, the error is about 0.6 percent. So, the discrete distribution is not performing very well. It is it could be several reasons, one is the way the uniform variates are generated that can have a say.

And we are also trying to approximate a circle using this 1 by 1 kind of thing. So the circular region, unlike in the continuous distribution, so there will be points that will either fall or for inside or outside but it is only an approximation to the circle that you will make with this kind of units, because we have taken an unit of 1. So, that could be another reason why, in other words if you go back to the figure, so continuous is this, but if you take graph sheet kind of thing, then you will have the step kind of thing.

So, that also does not give you a very good approximation for the circle. So, if you want to get a better approximation for the circle, you should go to much larger system sizes, so, that will probably give you a better calculation, but it will also take longer to run. So, I do not know how long, how much longer it is going to take, if you try to run a larger simulation.

Let us say that, we get some 10, 2500 and L is 100 and so let us source and see okay, so we are somewhere around 3.14. So, that could be one of the reasons why we are having and you need to do much a larger number of simulations to get better values, right. So, it is going to take time because now, it is, okay now, you can see that it peaks at 3.137 is what it gives and you can calculate the error.

And there is now 0.13 percent unlike the 0.6 percent that you saw in this case. So, it is very important depending on the approximation you make and so you will see better results. And, we are going to use this idea later to look at microstructure. So, as you can see, in this case, because we knew that it was a circle, we tried to calculate the pi value, but if it was some random shape and if you generate these uniform variates, and all you need to do is to just find out whether it fell in a region which was red or fell in the region which was blue.

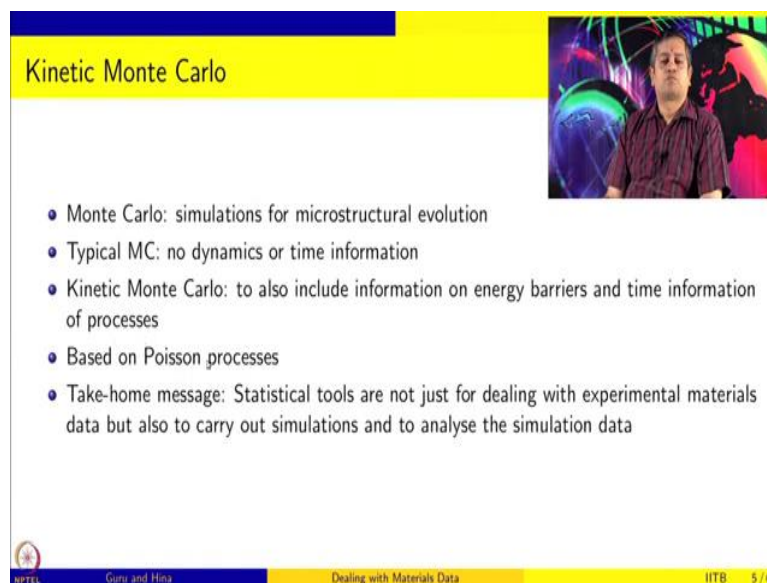
Typically, you take microstructures binarised them 1, 0 and you find out whether it fell in region 1 or region 0. And, so you can pixelate the microstructure and for every pixel, you take random numbers, see which pixel you reach and find out whether that pixel is 0 or 1. And you can keep counting how many zeros to the total, for example, will give you the area fraction of the 0 phase and one will give you the 1 phase.

So it is an idea that you can use to do the analysis, typical analysis of micro structure of for sizes of particles, for example, involves the other methodology. So, typically one draws lots of

lines and finds how many times it crosses. And there are stereology techniques using which you can get some idea about the sizes and the areas and things like that.

What we are giving here is another method. It is a computer based method, which will also give you same information, for example, estimating the area fraction of 2 phases, you can use a similar idea. But as a simple example, we just took a known shape for which the area is known. So, we use that to actually evaluate the pi value. So, now let us go back to our presentation.

(Refer Slide Time: 17:52)



The slide is titled "Kinetic Monte Carlo" and features a yellow header bar. Below the title, there is a list of five bullet points. To the right of the text, there is a small video inset showing a man in a maroon shirt speaking. At the bottom of the slide, there is a footer with the IITB logo, the text "Guru and Hina", "Dealing with Materials Data", and "IITB 5/6".

- Monte Carlo: simulations for microstructural evolution
- Typical MC: no dynamics or time information
- Kinetic Monte Carlo: to also include information on energy barriers and time information of processes
- Based on Poisson processes
- Take-home message: Statistical tools are not just for dealing with experimental materials data but also to carry out simulations and to analyse the simulation data

Okay, so this kind of Monte Carlo simulation, we have just used it for integration, but there is something known as Kinetic Monte Carlo. So, Monte Carlo can also be used for microstructural evolution simulations. But typical Monte Carlo simulation does not include any dynamics or time information.

Kinetic Monte Carlo is a type of Monte Carlo simulation in which one can include information on energy barriers on the energy surface of the system and hence also the time information that is associated with the processes. And the kinetic Monte Carlo is actually based on Poisson processes, okay. So, what is the purpose of this session?

So, in this session, so we have been looking at probability distributions and till now for many different problems we showed where they are relevant for material science and engineering. But the Monte Carlo simulations for integrations or their extensions to actually look at microstructure evolutions and the extension of Monte Carlo method itself to kinetic Monte

Carlo to also estimate the time or dynamics associated with the processes of micro structures are chemical reactions and things like that.

So, they also have their own statistical underpinnings. So, probability distributions play a key role in also understanding simulation data and the results are setting up simulations and so on and so forth. So, the take home message from this session is that statistical tools are not just for dealing with experimental materials data but also for simulation materials.

So, in other words, it does not matter what is the source of your data, statistical methods are valid for all data. So, that is the, so I have kept a uniform to the last because it is one of those rare distributions which can be either discrete or continuous. I have also kept it to the last because it ties in with simulations and kinetic Monte Carlo which is related to the Poisson processes and the Poisson distribution is something that we have discussed also has some relevance.

And if possible later during case studies, we will look at some of these simulation methodologies also and how to do statistical analysis on this data. So, that we will come to at a later point. So, this brings us to the end of the probability distribution module, we started with some discrete probability distributions.

So, we looked at Bernoulli trials and binomial, negative binomial, hyper geometric and so on. And, we also found Poisson, so on and we also found where they are important and then we move to continuous distributions of which a normal is very-very important, we spend some time understanding normal distribution.

And then there are several other continuous distributions like not log normal, for example, and Weibull and exponential and so on. And they have their own relevance. But there are whole bunch of them, which we did not discuss okay, so but as and when they are needed. For example, gamma distributions we have not discussed, but they might become important, beta distributions we did not discuss, it might also become important.

So, we will discuss some of these probability distributions later, is very difficult to be exhaustible about all probability distributions. And the idea is that once you know how to deal with few of them, you will be able to deal with rest of them on your own. And then, we looked at uniform distribution and in all cases, we have looked at the practical importance or significance of these distributions.

Then we also looked at some distributions like chi squared, t and f which are relevant for confidence intervals and regression modeling and things like that. So, we will discuss those distributions, when we look at the analysis.

(Refer Slide Time: 22:20)

Summary

- Discrete and continuous probability distributions
- Some important probability distributions for analysing experimental and simulation data
- Estimating confidence intervals and testing hypotheses: probability distributions

NPTEL Guru and Hina Dealing with Materials Data IITB 6 / 6

And so, this is the summary slide for the probability distribution. And we are going to look at how to analyze the experimental data and present and understand and so on. And when we come there again we will continue with probability distributions, specifically chi square t and f will find lots of use here.

And in any case, we will keep talking about other probability distributions and use of probability distributions for the rest of this course. So, probability distributions is probably the core of this course. So, we have taken lots of time to understand several of them because we are going to continuously use them to do more and more detailed and complex analysis in the sessions to come. Thank you.