

# Physics through Computational Thinking

Dr. Auditya Sharma & Dr. Ambar Jain

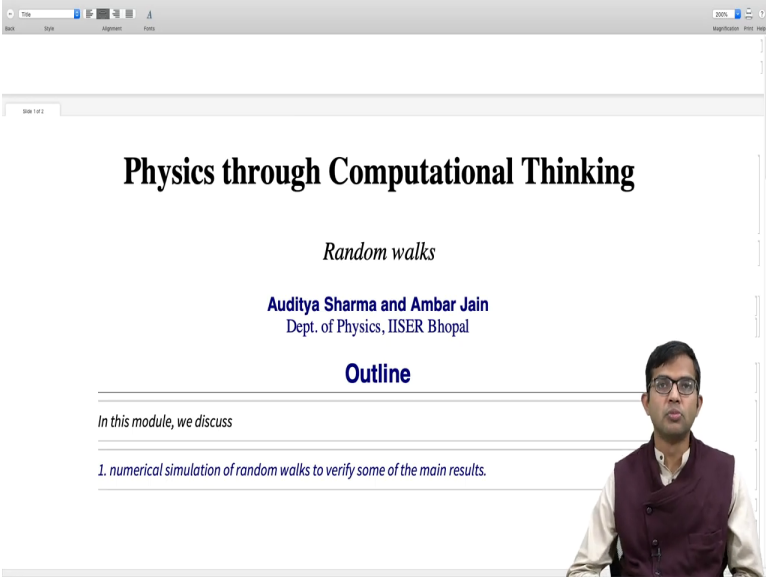
Department of Physics

Indian Institute of Science Education and Research, Bhopal

## Lecture 46

### Random Walks 3

(Refer Slide Time: 0:29)



The image shows a presentation slide with a video inset of a speaker. The slide content is as follows:

Slide 1 of 2

# Physics through Computational Thinking

*Random walks*

**Auditya Sharma and Ambar Jain**  
Dept. of Physics, IISER Bhopal

## Outline

---

*In this module, we discuss*

---

*1. numerical simulation of random walks to verify some of the main results.*

The video inset shows a man with glasses and a maroon vest over a white shirt, speaking.

Hi Guys. So, the agenda for this module is to make contact with the philosophy of this whole course, Computational Thinking. So, now that we have developed the theory of random walks, and in fact solved the random walk problem for all practical purposes for the discrete version in one week. And then we have seen how generalization of this the 3D is possible. So, now it is time to implement this on Mathematica. So, this is the agenda for this.

We will do a numerical simulation and verify the main results. What is the main result which we got from the random walk discussion so far? It is that the average distance covered by a random

walker, walking in one week is the typical distance, is actually like  $\sqrt{N}$ . So, we will verify this systematically, numerically now.

(Refer Slide Time: 1:20)

**Numerical Simulation of a random walk**  
**Special Case: The unbiased random walk**

Let us numerically generate a few sample random walks and visualize them.

```
In[1]:=  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True];
```

Out[1]=  
{0}

Such structures are commonly encountered in the study of polymers, stock price time series data, and trajectories of only a few. We recall the key result

$\langle m^2 \rangle = N$

So, in order to this, so let us first consider the unbiased random walk problem and then we can generalize it. So, there is this very nice simple approach to writing a code for this. You can actually first generate a random walk, visualize it. So, let us say I create an array of size 1. I have just one element in it.

(Refer Slide Time: 1:48)

**Numerical Simulation of a random walk**  
**Special Case: The unbiased random walk**


Let us numerically generate a few sample random walks and visualize them.

```
In[4]:=
a = {0};
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],
  {n, 2, 500}]
ListPlot[a, Joined -> True];
```

Such structures are commonly encountered in the study of polymers, stock price time series data, and trajectories of Brownian particles. We recall the key result

$$\langle m^2 \rangle = N.$$

Can we verify this with our numerics?



And now I can use this Do command, along with Append to generate a random walk. So, let me actually run this first and then I will explain to you, what is going on.

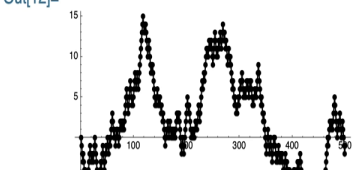

(Refer Slide Time: 1:57)

**Numerical Simulation of a random walk**  
**Special Case: The unbiased random walk**

Let us numerically generate a few sample random walks and visualize them.

```
In[10]:=
a = {0};
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],
  {n, 2, 500}]
ListPlot[a, Joined -> True]
```

Out[12]=

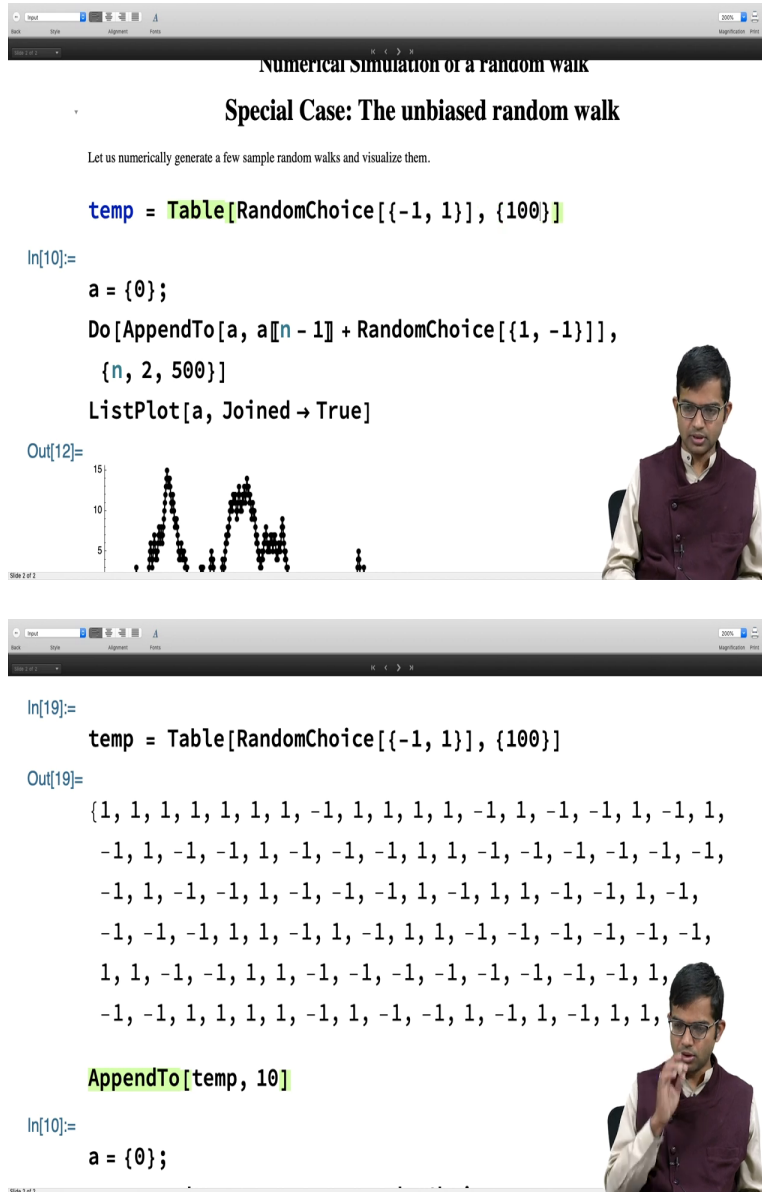







you accept that indeed this random choice is a good pseudo random number generator. And it gives for you, either +1 or -1.

(Refer Slide Time: 3:43)



**Numerical Simulation of a random walk**

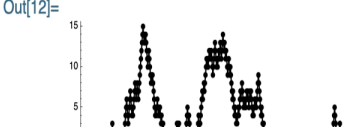
**Special Case: The unbiased random walk**

Let us numerically generate a few sample random walks and visualize them.

```
temp = Table[RandomChoice[{-1, 1}], {100}]
```

```
In[10]:=  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

Out[12]=



Slide 2 of 2

```
In[19]:=  
temp = Table[RandomChoice[{-1, 1}], {100}]
```

```
Out[19]=  
{1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1,  
-1, 1, -1, -1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1,  
-1, 1, -1, -1, 1, -1, -1, -1, 1, -1, 1, 1, -1, -1, 1, -1,  
-1, -1, -1, 1, 1, -1, 1, -1, 1, 1, -1, -1, -1, -1, -1,  
1, 1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1,  
-1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1,
```

```
AppendTo[temp, 10]
```

```
In[10]:=  
a = {0};
```

Slide 2 of 2

So, now let us look at what my code is doing. I am claiming that I can start an array with just one element. And that is 0. And then AppendTo. What does AppendTo do? So it can, so if I do Append. So, let us take this AppendTo. Let may define some array. So, if I have A equal to, let me now just, so let me say temp is equal to. Actually let me just use this.

Table of, since we have already learnt this. So, I will do RandomChoice. And the options are -1 and +1. And how many of this? Let us say I enter 100 of this. I have created an array, and I have called it temp. Now, I will use this Append. I should make this also input. AppendTo of temp. Then let us say, I will add 10. Or I will, I want to incorporate one more element into this array. So, that is what I am saying it does.

(Refer Slide Time: 5:20)

```

1, 1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, -1,
-1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1}

In[21]:=
AppendTo[temp, 100]

Out[21]:=
{1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, 1, -1, -1,
1, -1, 1, -1, 1, -1, -1, 1, -1, -1, -1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, 1, -1, -1, 1, -1, -1, -1, 1, -1,
1, 1, -1, -1, 1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1,
1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, 1, 1,
1, -1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, 10, 100}

```



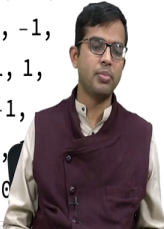
```

1, 1, -1, -1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, -1,
-1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1}

In[22]:=
AppendTo[temp, 5]

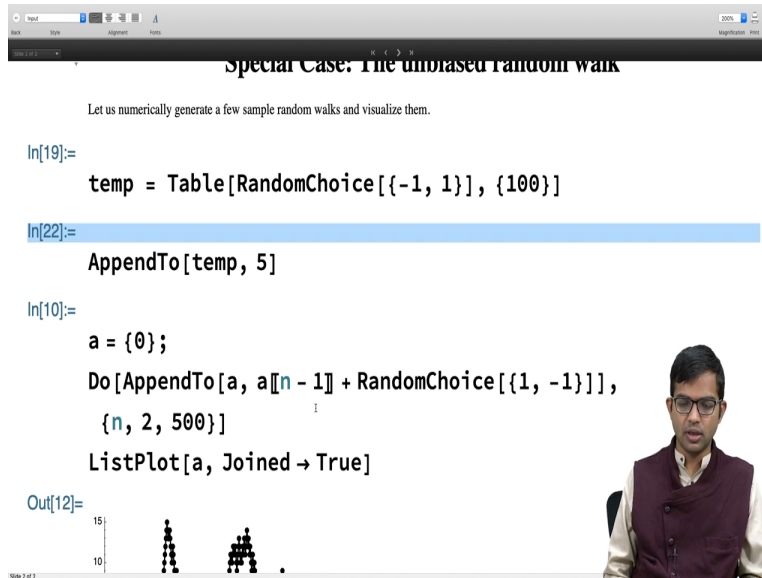
Out[22]:=
{1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, -1, 1, -1, -1,
1, -1, 1, -1, 1, -1, -1, 1, -1, -1, -1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, 1, -1, -1, 1, -1, -1, -1, 1, -1,
1, 1, -1, -1, 1, -1, -1, -1, -1, 1, 1, -1, 1, -1, 1,
1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, 1, 1,
1, -1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, 10, 100}

```



Now, you look at this array. It has all these 1 and 1 and minus 1 and so on. And now you see, one more 10 has come in. So, if I do it again, so you have the 10 is there and also 100 has come in. Then I can keep on adding any number I want. You know that. I do not know. 5 for example. So, you see that that is what this Append does. It takes an array and then it includes one more element into it. So, having familiarized ourselves with AppendTo array.

(Refer Slide Time: 5:51)



The screenshot shows a Mathematica notebook window titled "Special Case: The unbiased random walk". The text below the title reads: "Let us numerically generate a few sample random walks and visualize them." The notebook contains the following code:

```
In[19]:=  
temp = Table[RandomChoice[{-1, 1}], {100}]  
  
In[22]:=  
AppendTo[temp, 5]  
  
In[10]:=  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

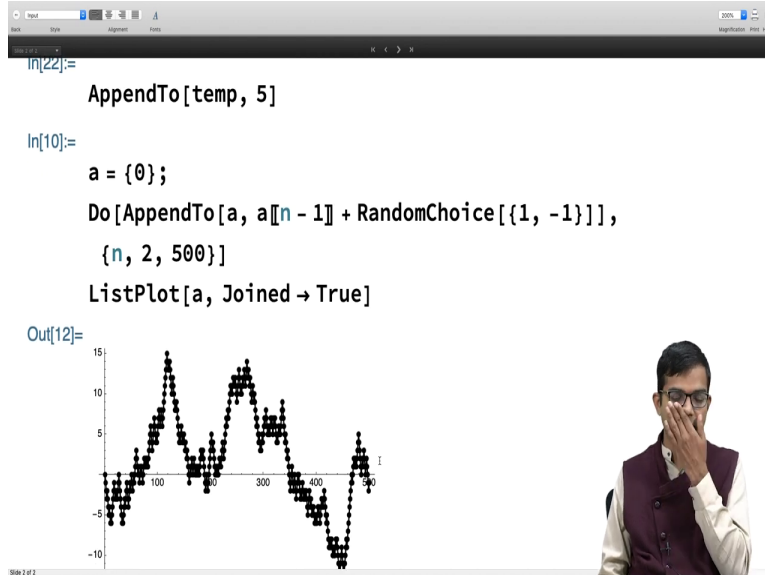
The output of the code is a plot showing two random walks. The first walk starts at 0 and ends at 5. The second walk starts at 0 and ends at 10. The plot shows the path of each walk as a series of points connected by lines.

So, now let us look at the logic of this code. What is it doing? It is saying, initially I start an array with just 0 in it, nothing else. And then after. Then there is a Do command. You can look up the syntax for the Do command. I think it is intuitive. So, basically n goes from 2 to all the way up to 500. So, it will take the (n - 1)th element.

So, in the initial case, first value of n is 2. So, n - 1 is 1. And that is just 0. It takes the previous element and adds either +1 or -1. That is what a random walk is. So, and then it stores all this information. So, initially you are going to have 0. So, then in fact we can look at A, what it looks like if you want.



(Refer Slide Time: 6:35)



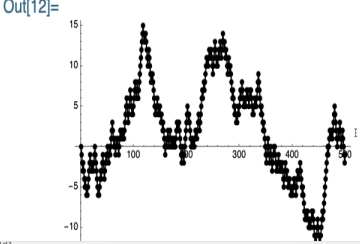
The screenshot shows a Mathematica notebook interface. The input cell contains the following code:

```
In[22]:= AppendTo[temp, 5]
```

```
In[10]:= a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```


The output cell shows a plot of a random walk:

```
Out[12]=
```



The plot shows a jagged line representing the random walk. The x-axis is labeled from 0 to 500, and the y-axis is labeled from -10 to 15. The line starts at (0,0) and ends at (500,5). The path is highly irregular, with many peaks and troughs.

Slide 2 of 2



I mean, but that is it is even better to just plot it. That is what we have done here.

(Refer Slide Time: 6:37)



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

```
In[22]:= AppendTo[temp, 5]
```

```
In[23]:= a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

The output cell shows a plot of a different random walk:

```
Out[25]=
```



The plot shows a jagged line representing a different random walk. The x-axis is labeled from 0 to 500, and the y-axis is labeled from -10 to 30. The line starts at (0,0) and ends at (500,5). The path is highly irregular, with many peaks and troughs, and it reaches a higher peak of approximately 30 around x=450.

Slide 1 of 1



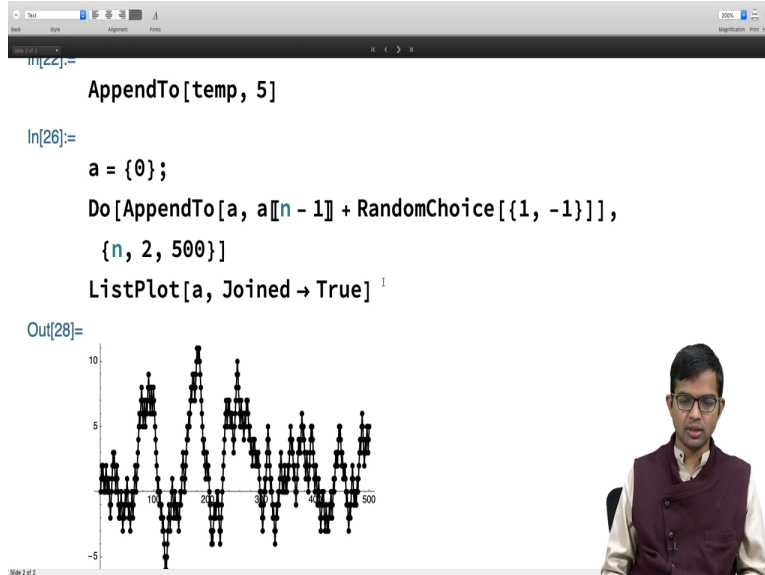
Let me generate it again. So, you get a different random walk.

(Refer Slide Time: 6:40)

```
In[22]:= AppendTo[temp, 5]
```

```
In[26]:= a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

```
Out[28]=
```



Slide 2 of 2

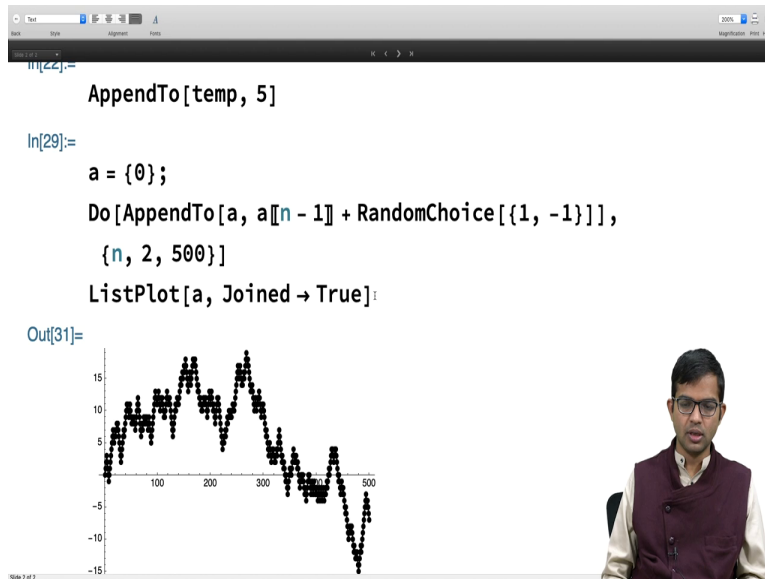
A third time we will do, you get another random walk.

(Refer Slide Time: 6:42)

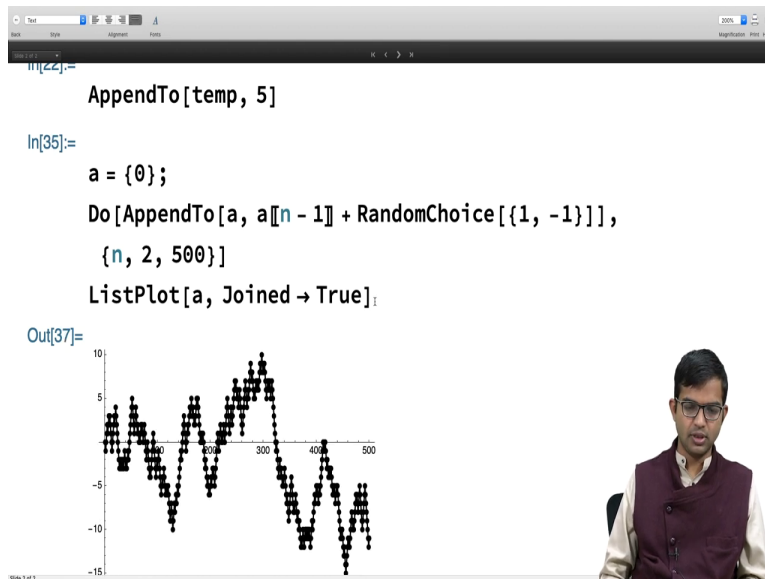
```
In[27]:= AppendTo[temp, 5]
```

```
In[29]:= a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

```
Out[31]=
```



Slide 2 of 2



This is something that you can play with. And check for yourself that indeed it is giving, it is a reasonable random walk generator. OK So, having shown you a simple one line code with using a Do loop to generate a random walk.

(Refer Slide Time: 7:03)

**Numerical Simulation of a random walk**  
**Special Case: The unbiased random walk**

Let us numerically generate a few sample random walks and visualize them.

```
In[19]:= temp = Table[RandomChoice[{-1, 1}], {100}]  
  
In[22]:= AppendTo[temp, 5]  
  
In[35]:= a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

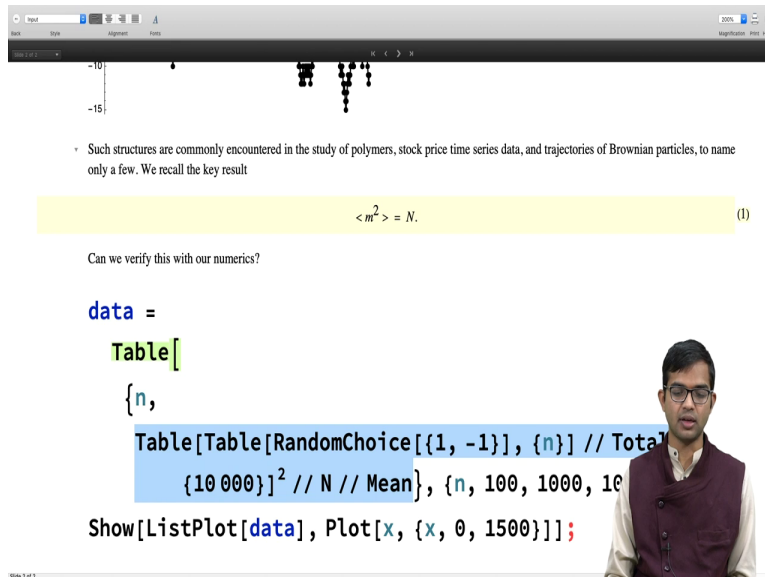
Out[37]=

Such structures are commonly encountered in the study of polymers, stock price time series data, and trajectories of Brownian particles. We recall the key result

$$\langle m^2 \rangle = N.$$

Now, what we want to see is, our goal is to test for this  $\langle m^2 \rangle = N$  result. So, such structures are commonly encountered in the study of polymers, stock prices, the time series data, trajectories of Brownian particles very important result, which connected the statistical mechanics to these kind of random walk models. Diffusive motions of. So, our goal here is to verify that average of  $\langle m^2 \rangle = N$  is a reasonable result, using numerics.

(Refer Slide Time: 7:36)

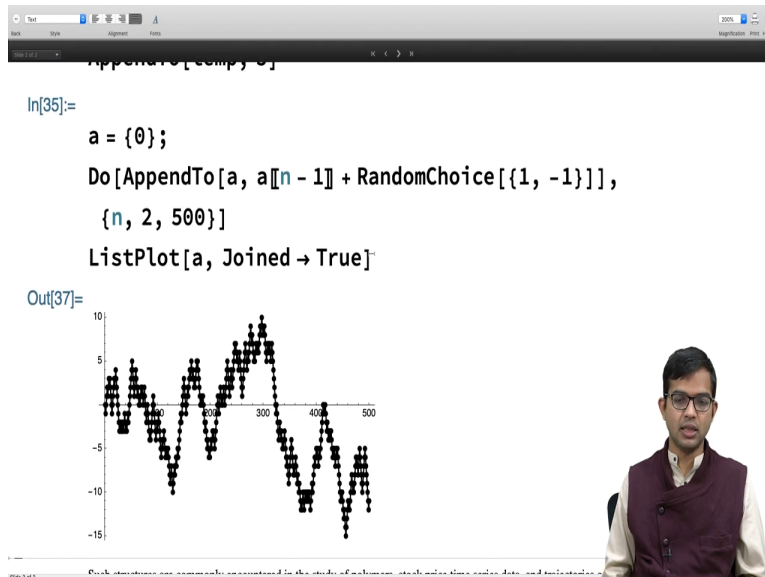


Such structures are commonly encountered in the study of polymers, stock price time series data, and trajectories of Brownian particles, to name only a few. We recall the key result

$$\langle m^2 \rangle = N. \quad (1)$$

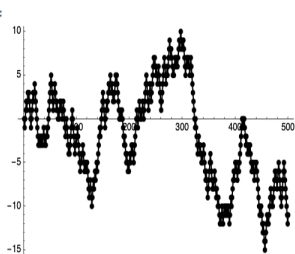
Can we verify this with our numerics?

```
data =  
Table[  
  {n,  
    Table[Table[RandomChoice[{1, -1}], {n}] // Total  
      {10 000}]^2 // N // Mean], {n, 100, 1000, 10000}]  
Show[ListPlot[data], Plot[x, {x, 0, 1500}]];
```



```
In[35]:=  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + RandomChoice[{1, -1}]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

Out[37]=



So, in order to this, I let me just show you the code first and then I will explain what is going on. So, I generate. So, this time I am going to generate a table of 1. {1, -1}, {n}. If I just sum. So, either I can do it like here which is you know I can keep on Appending to it, the sum of the previous, you know the latest point I am, I can add +1 or -1 to it.

And give you this kind of data. Which is actually a storing information about where the particles is after the nth step. This is one kind of data, which will be generating data which is useful. But

here let me generate just a table with information about whether I went to the right or left. So, that is what this central thought of this code is. I like always in mathematica, you must go inwards out, to understand a piece of code.

So, this one is generating for you a series of 1 and -1. Having got it, I can just sum it no. That is what this is doing. The total, it is a post processing operation. So, the sum of this is indeed the position of your particle. You have a bunch of 1 and -1. You just add all of them, it tells you where you are. So, I could have of course used this method to generate this as well. But this is just another way of doing it.

Now, having done this, what I want to do is, I want to make 10000 such tables. Because ultimately I want average over, you know,  $n_i^2$ . I create 10000 such m. So, this is already m. The sum is basically where I am. After, you know, if I take n going from 100 to 1000 in steps of 100. For every value of n. And then I have to do this mean. Evaluate this and then I compute this average.

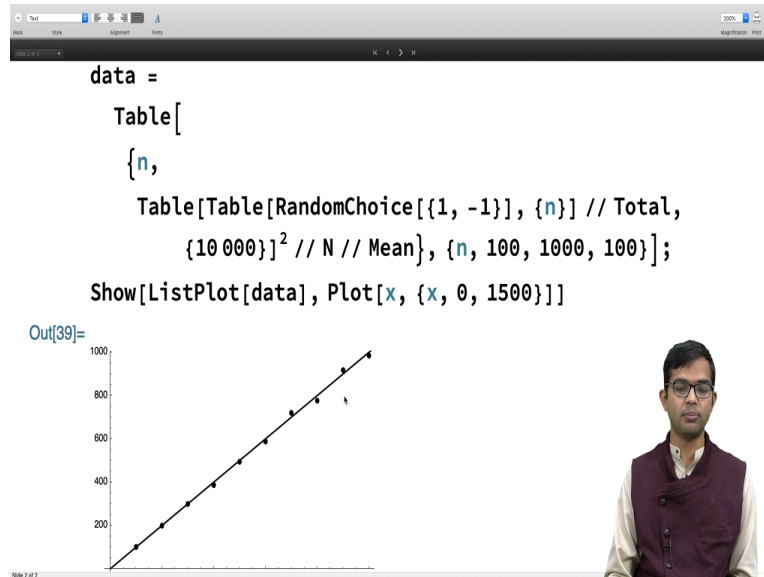
So, this average is computed over such 10000 such samples. Basically I am generating 10000 random walks and for each random walk, I am just computing  $n_i^2$ . That is done here. You see basically I can generate a whole you know, I can take the square of a whole array. That is what I am using here. And then I compute it numerically and then I am also doing a mean of that quantity.

All of this I want you to, you know, take away each part of this code and understand for yourself that you actually do really do understand. And then finally I am making another table, there is a whole bunch of tables. But I want you to individually understand each table and then see that how it all makes sense. So, finally what is this table doing? This table is storing information about n and  $\langle n_i^2 \rangle$  basically. That is all.

This entire thing, starting from here, all the way up to here, is you must think of it as just one number. So, this outer most table is storing n. And this big number, given that n goes from 100 to 1000 in steps of 100. And finally I can just go ahead and plot this. So, let me just plot this. So, it

is going to take some a minute I think, because this is somewhat of a large run. Since I am using 10000.

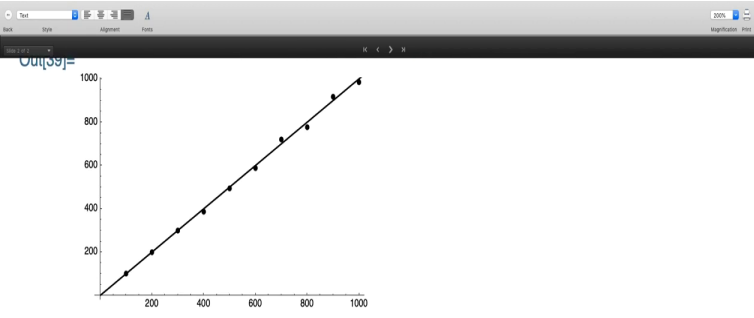
(Refer Slide Time: 11:10)



So, there you go. I have a perfect straight line. What am I plotting? I am plotting average of  $m$  square. And I see that it goes indeed it seems to go linearly with  $n$  and it sits on top of this in fact. It is not just linear, but it has the correct slope as well. It is exactly equal to you know  $n$ . I have plotted this function  $x$  going from 0 to 1500.

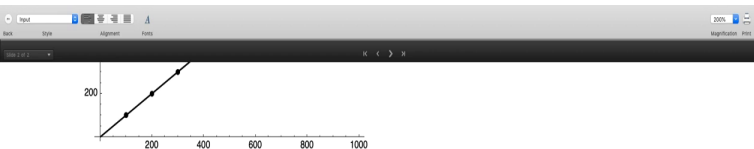
And it seems to agree excellently with the, with the analytical result. So, in fact, we can do better. We can actually work out the numerical histogram associated with this. So, we know, not only the that the  $\langle m^2 \rangle = n$ . But we also know, in fact know the distribution  $P_N(m)$ .

(Refer Slide Time: 12:01)



```
nMax = 1000;  
binsize = 10;  
histdata =  
Histogram[  
Table[Table[RandomChoice[{1, -1}], {nMax}] /  
{10000}], {binsize}, "PDF"];
```

Slide 2 of 2



```
nMax = 1000;  
binsize = 10;  
histdata =  
Histogram[  
Table[Table[RandomChoice[{1, -1}], {nMax}] // Tot  
{10000}], {binsize}, "PDF"];  
Show[histdata, Plot[ $\sqrt{\frac{1}{2 \pi nMax}} e^{-x^2/(2 nMax)}$ ,  
{x, -5  $\sqrt{nMax}$ , 5  $\sqrt{nMax}$ }]];
```

Slide 2 of 2

So, let us use this histogram and use it in PDF mode. Probability distribution function mode. So, this is something that you should look up. Histogram can work in many different modes. You can check the documentation on this. So, here we have probability distribution function.

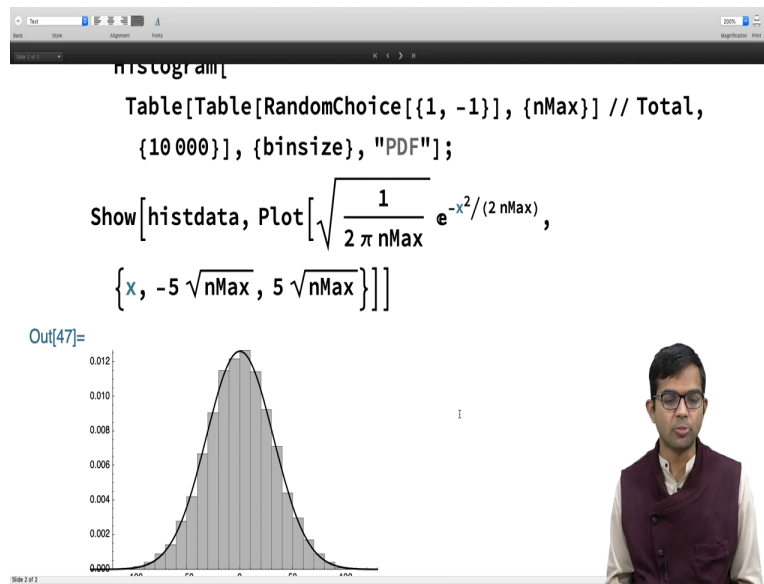
And I have already computed, this quantity, earlier in the previous module, you can check. I have shown how this probability distribution is given by this Gaussian function and then I can just go ahead and. So, you have to introduce this bin size. This is also sometimes a cause for confusion.



Bin size is gives you freedom on how you, you know you take all the data that you have and you have to put them into various bins and create a numerical probability distribution.

So, you have some freedom there. That also is something that you can play with. I have just chosen my nMax and n binsize like here. You can pause the video and try to mimic this and also understand what is going on here. So, I have a histogram. So, this part is the same. It is just simply the table of 1 and -1. This is the random work. And then I generate this data. So, let me plot this.

(Refer Slide Time: 13:25)



There you go. So, it is right on top of the analytical curve. And its extra justification. That both our analytical work is correct. And the numerical approach also seems to be reasonable.

(Refer Slide Time: 13:43)

**The biased random walk**

What about the general biased random walk, for which  $p$  maybe different from  $\frac{1}{2}$ ?

```
p = 0.55;  
a = {0};  
Do [AppendTo[a, a[[n - 1]] + 1 - 2 UnitStep[RandomReal[] - p],  
    {n, 2, 500}];  
ListPlot[a, Joined -> True];
```

**Homework**

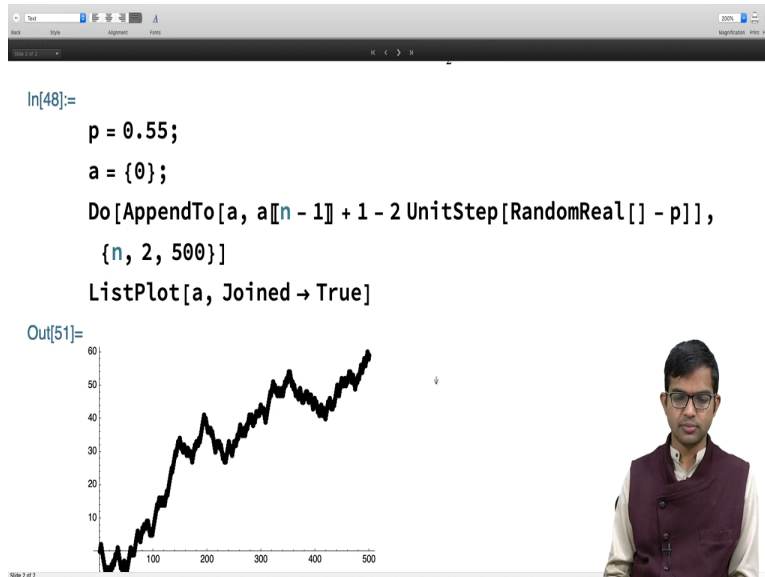
So, in fact, you can now extend this. You can take  $p$  to be different from  $\frac{1}{2}$ . You can consider a biased random walk. So, I have chosen  $p$  to be 0.55. You can choose it to be whatever. Now, instead of just using RandomChoice, I am using you know this, this is my way of generating a random number which with probability  $p$  will take you to the right and with probability  $1 - p$  will take you to the left.

So, it is  $1 - 2 \text{UnitStep}[\text{RandomReal}]$ . RandomReal will just give you a random number between 0 and 1, from, drawn from uniform distribution, between 0 and 1. And then you have to subtract  $p$  and then take a unit step. And then multiply by 2,  $1 - 2$  times this. You should go back and you should convince yourself that indeed this is the right expression. I have checked this and I can tell you that indeed it will give you a biased random walk.

It will give you, you can think of it as just this part. Take this part apart and run it many many times. And you will say, you can do a do a numerical experiment to check that. If you take you know 100 such random numbers, probably 55 of them will be +1 and only 45 will be -1. That is what is expected. Or you can do even more. You can do a Monte Carlo stimulation on this to check that it is indeed giving you the right statistics.

So, that is the way you know pseudo random number generators are checked. Distributions whether they are true to the properties of distribution involved. So, I will allow you to play with this. So, and so this is something that I have checked.

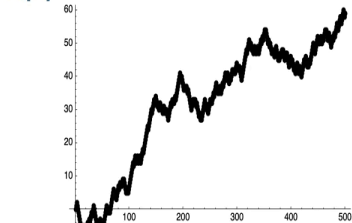
(Refer Slide Time: 15:24)



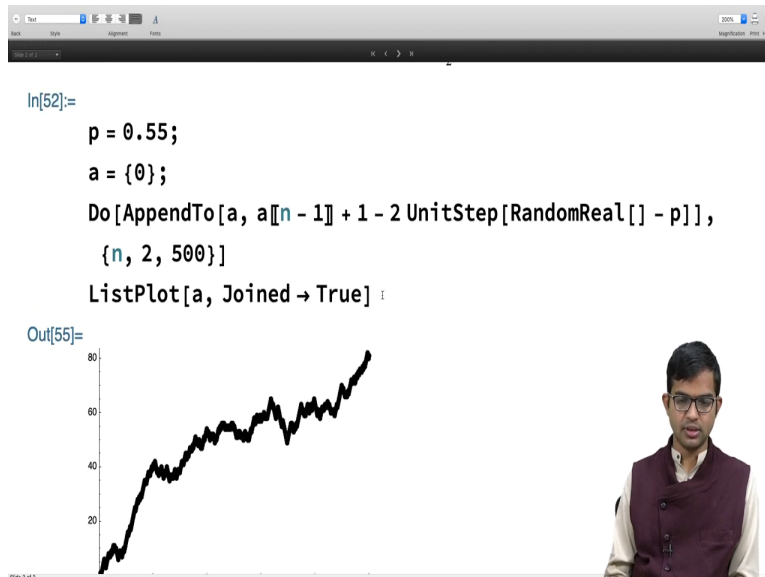

In[48]:=

```
p = 0.55;  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + 1 - 2 UnitStep[RandomReal[] - p]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

Out[51]=



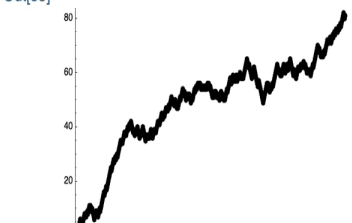
Slide 2 of 2




In[52]:=

```
p = 0.55;  
a = {0};  
Do[AppendTo[a, a[[n - 1]] + 1 - 2 UnitStep[RandomReal[] - p]],  
  {n, 2, 500}]  
ListPlot[a, Joined -> True]
```

Out[55]=



Slide 2 of 2



```
In[56]:=
p = 0.55;
a = {0};
Do[AppendTo[a, a[[n - 1]] + 1 - 2 UnitStep[RandomReal[] - p]],
{n, 2, 500}]
ListPlot[a, Joined -> True]
```

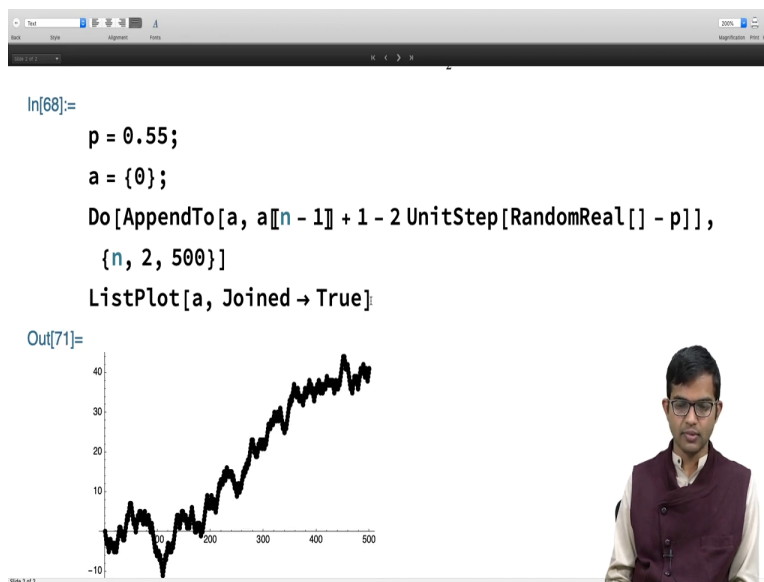
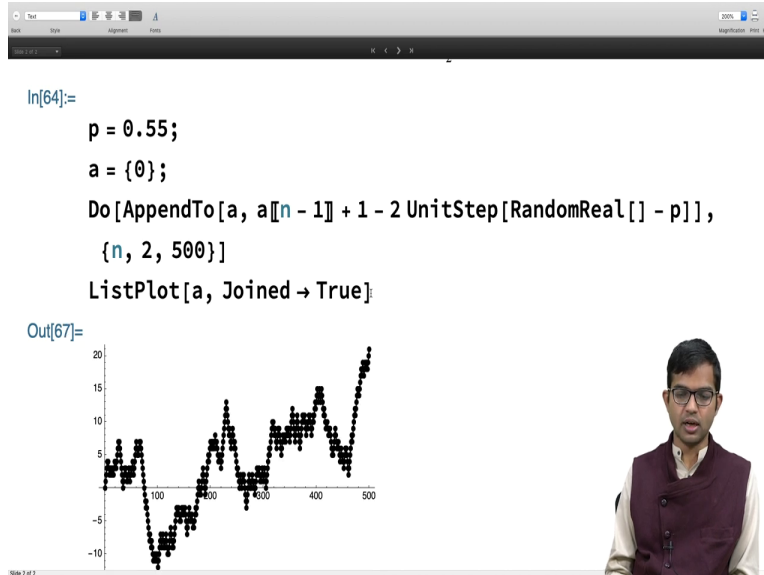
Out[59]=

Slide 2 of 2

```
In[60]:=
p = 0.55;
a = {0};
Do[AppendTo[a, a[[n - 1]] + 1 - 2 UnitStep[RandomReal[] - p]],
{n, 2, 500}]
ListPlot[a, Joined -> True]
```

Out[63]=

Slide 2 of 2

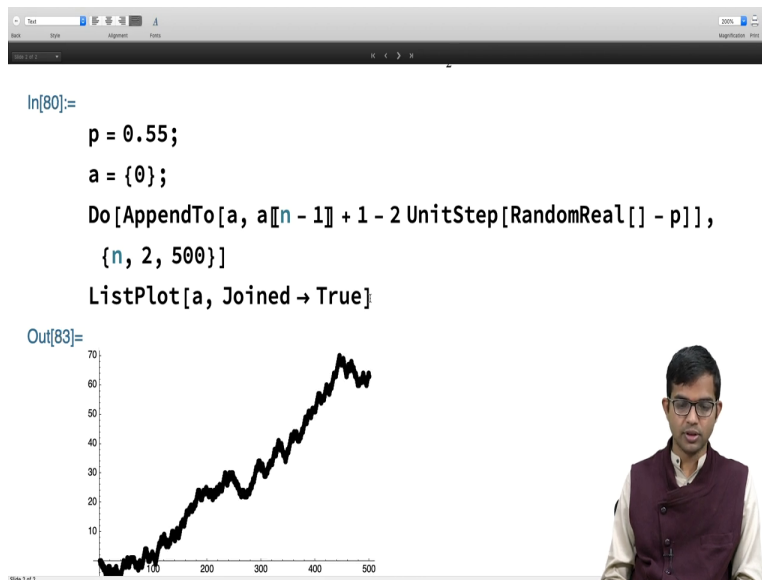
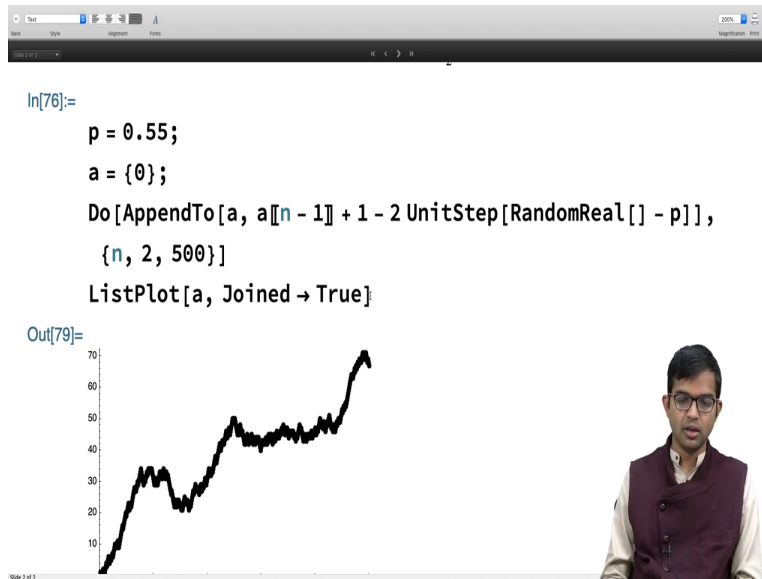


So, let me generate. Okay, so there you go. This is generating the random walk. Now, you see that there is a drift involved. Your particle is you know very resistant to return to the origin now. It is going to run away. But slowly, because it is only 0.55. So, in fact, on average it is it has a speed now. That speed of going away from the origin is  $p - q$ .

So, if  $p = q$ , then you can say that there is no drift velocity as such. It is equally likely to go to the right or to the left. And therefore you can actually show that if you wait long enough, it will for sure come back to the origin. That is another very interesting question in itself.

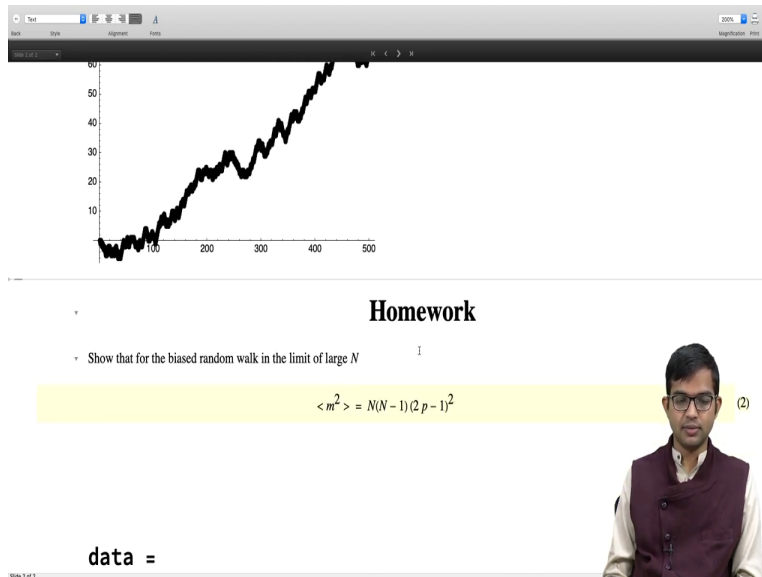
So, this is what people call first passage. But that is not the discussion right now. So, right now we are just looking at what happens to the biased random walk.

(Refer Slide Time: 16:17)



The biased random walk will make it go further and further away from the origin. That is what will happen. See, you see that it is hardly ever going below the X axis. That is because it is more likely to move to the right.

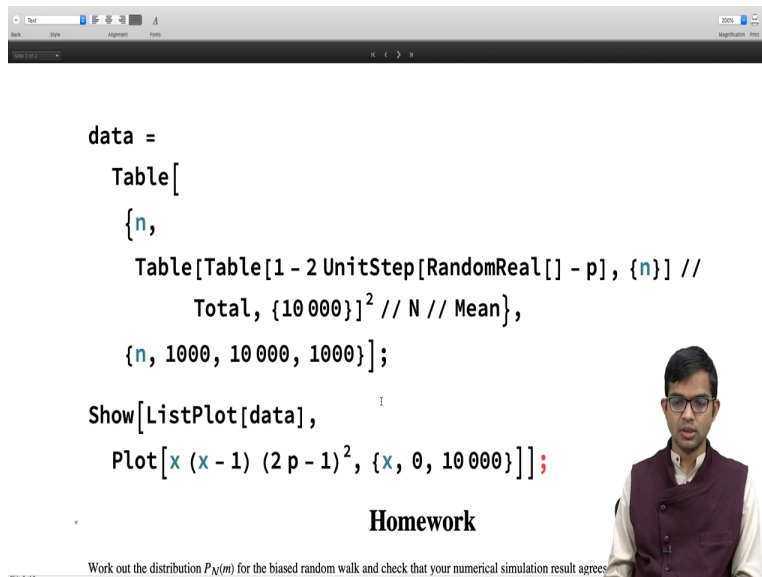
(Refer Slide Time: 16:31)



The slide displays a plot of a biased random walk with a clear upward trend. Below the plot, the word "Homework" is centered. A bullet point asks to show that for a biased random walk in the limit of large  $N$ , the mean squared displacement is given by the equation  $\langle m^2 \rangle = N(N-1)(2p-1)^2$ . The equation is highlighted in yellow. Below the equation, the text "data =" is visible. A small inset video of a man in a maroon vest is in the bottom right corner.

Now, the homework is to check that this  $\langle m^2 \rangle = N(N-1)(2p-1)^2$ . I have worked this out. And now, I want you to use you know the same type of code as here. But you do some small modifications and check that the biased random walk corresponds to this equation.

(Refer Slide Time: 17:00)



The slide shows Mathematica code for simulating a biased random walk. The code defines a list of data points for different sample sizes  $n$  and calculates the mean squared displacement. The code is as follows:

```
data =  
  Table[  
    {n,  
      Table[Table[1 - 2 UnitStep[RandomReal[] - p], {n}] //  
        Total, {10 000}]^2 // N // Mean},  
    {n, 1000, 10 000, 1000}];  
Show[ListPlot[data],  
  Plot[x (x - 1) (2 p - 1)^2, {x, 0, 10 000}];
```


Below the code, the word "Homework" is centered. A small inset video of a man in a maroon vest is in the bottom right corner.

And then I also have the data for the histogram. That is also, that is also homework. So, I have data for, let us check this.

(Refer Slide Time: 17:24)


```
In[85]:=
data =
Table[
{n,
Table[Table[1 - 2 UnitStep[RandomReal[] - p], {n}] //
Total, {10 000}]^2 // N // Mean},
{n, 1000, 10 000, 1000}];

In[84]:=
Show[ListPlot[data],
Plot[x (x - 1) (2 p - 1)^2, {x, 0, 10 000}]]
```



```
In[85]:=
data =
Table[
{n,
Table[Table[1 - 2 UnitStep[RandomReal[] - p], {n}] //
Total, {10 000}]^2 // N // Mean},
{n, 1000, 10 000, 1000}];

In[84]:=
Show[ListPlot[data],
Plot[x (x - 1) (2 p - 1)^2, {x, 0, 10 000}]]
```

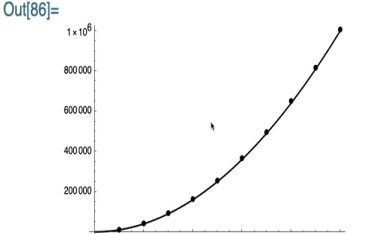


Yes, so this has to be run afresh. While I mean, I said homework, but I also have the code here, and I have already shown it to you. So, maybe you should find your own way of doing it as well, in addition to what I have here. Yeah, so it is still running. So, give it another 30 seconds I think, it should complete it. There you go.



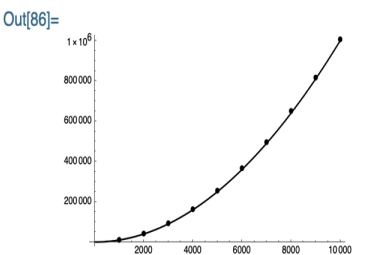
(Refer Slide Time: 18:00)

```
Table[Table[1 - 2 UnitStep[RandomReal[] - p], {n}] //  
Total, {10 000}]^2 // N // Mean},  
{n, 1000, 10 000, 1000}];  
  
In[86]:= Show[ListPlot[data],  
Plot[x (x - 1) (2 p - 1)^2, {x, 0, 10 000}]]  
  
Out[86]=
```



Slide 2 of 2

```
In[86]:= Show[ListPlot[data],  
Plot[x (x - 1) (2 p - 1)^2, {x, 0, 10 000}]]  
  
Out[86]=
```



**Homework**

Work out the distribution  $P_N(m)$  for the biased random walk and check that your numerical simulation result agrees

Slide 2 of 2

So, now if I plot this. So, earlier when I was plotting it, it was getting data from the older one. And therefore it is not agreeing. But in deed now it is very beautifully quadratic in nature. It is  $x(x-1)(2p-1)^2$ . So, I would urge you to actually not look at my solutions, but work out your own solution and check it yourself. And then only cross check against my solution. And then I do not give you the solution now for  $P_N(m)$ . You can work out something very similar to this problem. And do the biased problem to write a piece of code which will give you generate numerically  $P_N(m)$  for the biased random walk.

That is going to be very illuminating. It is going to be a useful exercise. And I urge you all to do this. So, what have we done in this module? So, the main thing is that once again I emphasize. The key result to take home is that  $\langle n_i^2 \rangle = N$  in fact for the discrete random walk. And we have managed to verify this numerically as well. Then the data sits right on top of the analytical expectation. You also managed to do it for the biased random walk. And as homework, you will verify that the probability distribution also works out. Both analytically and numerically. Thank you.