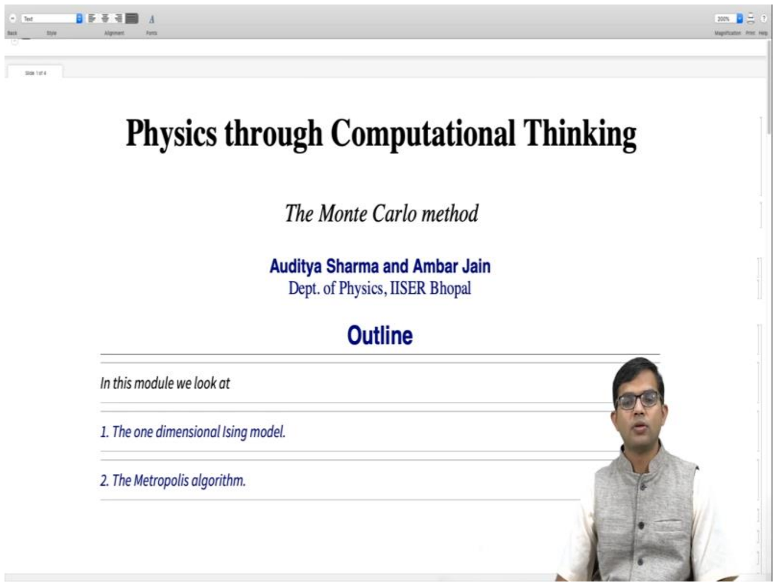
Physics through Computational Thinking
The Monte Carlo Method

By Dr. Auditya Sharma and Dr. Ambar Jain
Department of Physics
Indian Institute of Science Education and Research, Bhopal

(Refer Slide Time: 00:26)



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. The one dimensional Ising model.

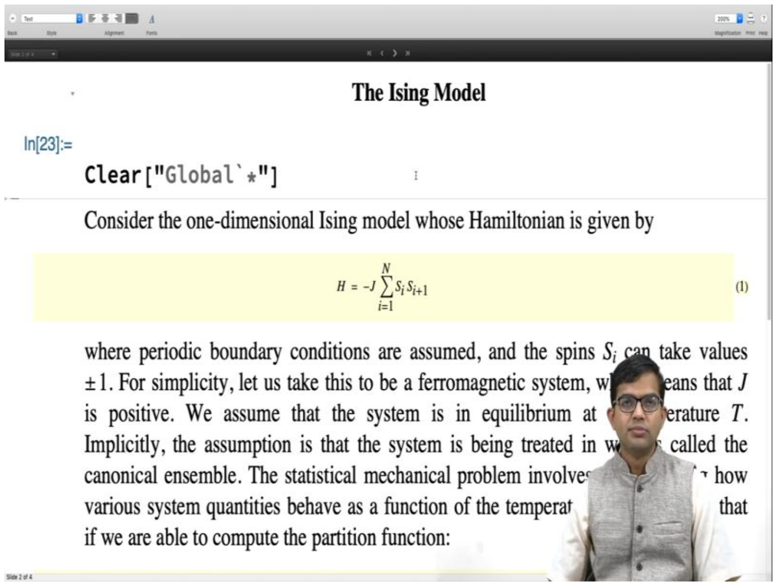
2. The Metropolis algorithm.

Hello, everyone. So, we have been looking at a few examples of the Monte-Carlo method. All of the examples we have looked at so far have been rather elementary in nature. Which illustrates the method. And, you know with the help of very simple code running between just the few lines in a most cases. So, now I want to bring in a slightly advanced problem, which one can work out using this method.

And this involves some statistical mechanics knowledge, which I guess I am not supposed to assume. However, I will try to give you like a one slide primer into the statistical mechanics so the one-dimensional ising model. And so even if you do not get all the details of all the physics involved here, I just want you to appreciate the power of the Monte-Carlo method. So, it comes out very beautifully with in statistical mechanics and it is used in very complicated simulations are written in this field.

And oftentimes this is the best understanding of this system is actually come from Monte Carlo simulation, very sophisticated one using lots of you know technology in terms of algorithms at various levels you know to speed up the processes, and to use lots of computational power. And it gets used up in the at the forefront of research even today, in current research it is used heavily. So, but what I want to do is just give you a glimpse of how this kind of a Monte-Carlo method will play out in this context.

(Refer Slide Time: 02:06)



The Ising Model

In[23]:=

```
Clear["Global`*"]
```

Consider the one-dimensional Ising model whose Hamiltonian is given by

$$H = -J \sum_{i=1}^{N} S_i S_{i+1} \quad (1)$$

where periodic boundary conditions are assumed, and the spins S_i can take values ± 1. For simplicity, let us take this to be a ferromagnetic system, w eans that J is positive. We assume that the system is in equilibrium at erature T. Implicitly, the assumption is that the system is being treated in w called the canonical ensemble. The statistical mechanical problem involves how various system quantities behave as a function of the temperat that if we are able to compute the partition function:

Okay so, I will start by clearing this. So, the ising model is like the standard model for magnetism, Right. So we will consider the one-dimensional ising model. And of course, there are various more realistic models were decided on higher dimensions probably in three dimensions and so on. So, you can think of you know classical spins, classical spins are magnetic moments. You know, you can think of every atomic site in some lattice which is occupied by a magnetic moment.

And you ask the question whether if there are some internal forces in your system which tend to align all of these local magnetic moments along the same direction. And if they do, then the, the overall system has its own magnetism and it becomes a what is called a ferromagnetic. If you have a bar magnet it can lift iron filing and so on. So, really what is happening at a microscopic level is it has lots of local magnetic moment which are all aligned along the same direction.

And together they will generate a magnetic field which can act as a tangible force that we can measure, right. So the microscopic model, one very commonly used and successful model of this kind of magnetism is the ising model. And in 1D it turns out that this problem can be solved exactly, analytically and simply. It is not a very hard problem. It can be taught to a starting undergraduate-students so perhaps you will see this in a basic stat-mech course when you do it.

But for now, just believe me, when I am makes a certain assumptions which perhaps you will only encounter later on. And then you will be able to verify it. So, now so what is the stat mech problem? So, you have a certain Hamiltonian. And in this case, you have a bunch of spins. Take think of these spins residing on a 1D line and there are n of those spins. Each spin can either be up or down.

So, we will represent this by values plus 1 and minus 1. Each spin can take either a plus 1 or a minus 1. And so, for simplicity, we take this Hamiltonian to be a ferromagnetic Hamiltonian. Which means that your J is positive in this case. When all the spin aligned in the same direction, then the energy is minimal. And so the tendency is for all the spin to align the same direction.

So, as opposed to ferromagnetism, ferromagnetic order one can also have anti ferromagnetic part. Where if J were negative, then the lowest energy state would correspond to a case where, alternate spins are up and down. So, the lowest energy state would be somewhere up-down, up-down, up-down and so on. So, that is called anti ferromagnetic order which does not havenet magnetic moment associated with. But let us not go there.

So, we are thinking of a classical ising ferromagnet in one-dimension. Now statistical mechanics tells us that, you know, there is a competition between the system tending to low make its energy as low as possible. And there are entropic effects. So, entropy is you know you might have think of encountered entropy as a form of a tendency to create more disorder right systems generally involved in the direction of creating more disorder. And this comes from at a microscopic level, the possibility of your system to exist in lot of so-called accessible micro states.

So, that is not, that is not something it is clear. Do not worry. You will probably encounter it in a stat mech course. So, the idea is that you may have. Your system has a certain microscopic state associated with a certain energy. For example, if your system is in a state given by certain energy. But actually, there will be like tons of different ways the system can be in microscopically and still have the same energy. The fact that the system has so much choice; is actually at the heart of where all the entropy comes from.

So, if there is more choice available for your system, to be microscopically to wiggle around and still maintain its overall parameter as the same energy or whatever than it has there is greater entropy. And so, the tendency for your system to be in a low energy state and entropy are often forces which go against each other. And this competition is what gives you a delicate equilibrium. So, there is a statistical mechanical equilibrium, which comes about at the temperature T. And so, there are many ensembles, one can construct something called a micro canonical ensemble, the canonical ensemble.

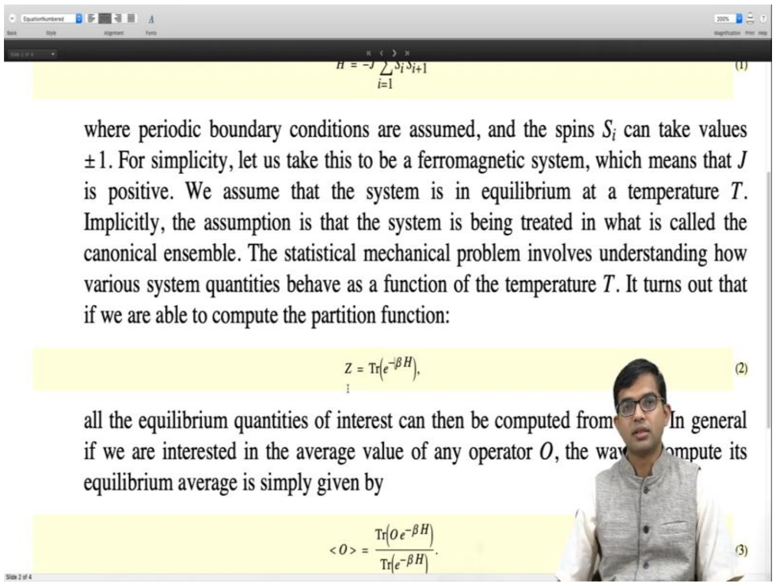
So, in this canonical ensemble, it turns out that you assign weights for each configuration. If there are n spins, each of which can be either plus or minus 1 there are total, the total number of configuration is 2^n , each spin can be either up or down. So, you take a product from all of these possibilities, you get 2^n . So, which of these configurations is more likely?

So, it turns out stat mech tells us that actually every configuration has a certain weight, certain probability of a certain of appearing at any given temperature. So, the states which have energies close to the average energy at a certain temperature are more likely. So, this is where the idea of important sampling comes in. So, you know, you have a partition function which is defined like here, which is actually sum of all these weights.

So, Boltzmann's weights are associated with these probabilities. So, you just compute exponential of $-\beta$ times, the energy of a certain configuration and then you have to divide by this partition function that is the probability of a certain configuration. So, it turns out that, you know, there is quite some detail involved here. So, you have to sum over all every configuration state and so on to get this partition function. But believe me when I say that if you can somehow compute this partition function.

Which is a trace of this, which means that you have to sum over all the 2^n , states find its energies and do the sum. And if we can somehow come up with a closed form expression for this, there is a way to get to the free energy of the system. The free energy turns out is simply given by log of this with some, some factor associated with it, that's all. And from which you can extract all the quantity of interest, you can get the average energy, you can get you know magnetization, you can get other susceptibilities and co-relation function, everything comes out if you can compute this and. But this is usually a very hard problem.

(Refer Slide Time: 09:18)



$$H = -J \sum_{i=1} S_i S_{i+1} \quad (1)$$

where periodic boundary conditions are assumed, and the spins S_i can take values ± 1. For simplicity, let us take this to be a ferromagnetic system, which means that J is positive. We assume that the system is in equilibrium at a temperature T. Implicitly, the assumption is that the system is being treated in what is called the canonical ensemble. The statistical mechanical problem involves understanding how various system quantities behave as a function of the temperature T. It turns out that if we are able to compute the partition function:

$$Z = \mathrm{Tr}\left(e^{-\beta H}\right), \quad (2)$$

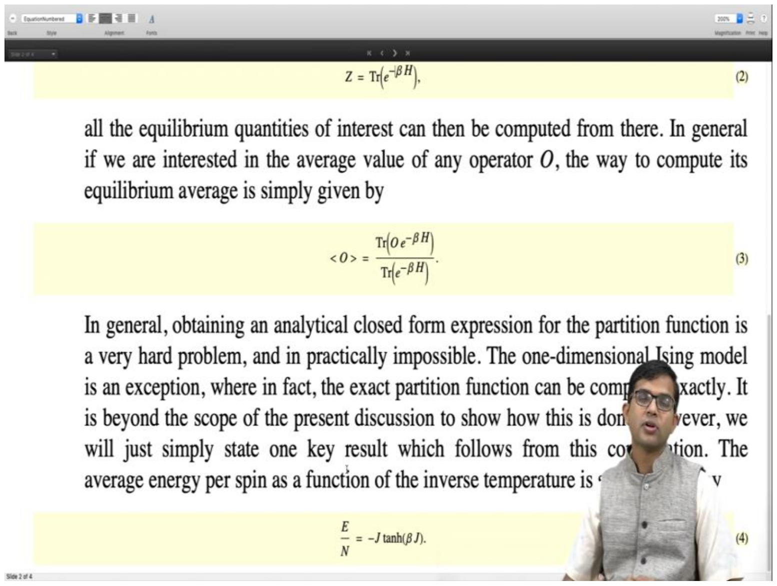
all the equilibrium quantities of interest can then be computed from In general if we are interested in the average value of any operator O, the way ompute its equilibrium average is simply given by

$$<O> = \frac{\mathrm{Tr}\left(O e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}. \quad (3)$$

And in fact, the 2D ising model is one of the grand achievements of theoretical physics of the last century of the 20th century, where it was shown that it, in fact was possible to solve this model exactly, analytically. So, an exact solution is available and in 3D onwards or higher or, you know, a more complicated lattices. Basically, it is an impossible problem for sure. There is no analytical solution, but even computational calculation of these kind of sums involving such a large number of states, itself becomes and you know prohibitively difficult calculation.

So, there are lot of results on this kind of; we are not going anywhere, right. Our goal is to just point out here that this is a hard problem in general, and that is where Monte-Carlo comes in. So, what Monte-Carlo does is it says, oh, there is a sum involving lots of terms we will just take only those terms which matter. That is where the important sampling philosophy comes in.

(Refer Slide Time: 10:21)



$$Z = \mathrm{Tr}\left(e^{-\beta H}\right), \quad (2)$$

all the equilibrium quantities of interest can then be computed from there. In general if we are interested in the average value of any operator O, the way to compute its equilibrium average is simply given by

$$< O > = \frac{\mathrm{Tr}\left(O\, e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}. \quad (3)$$

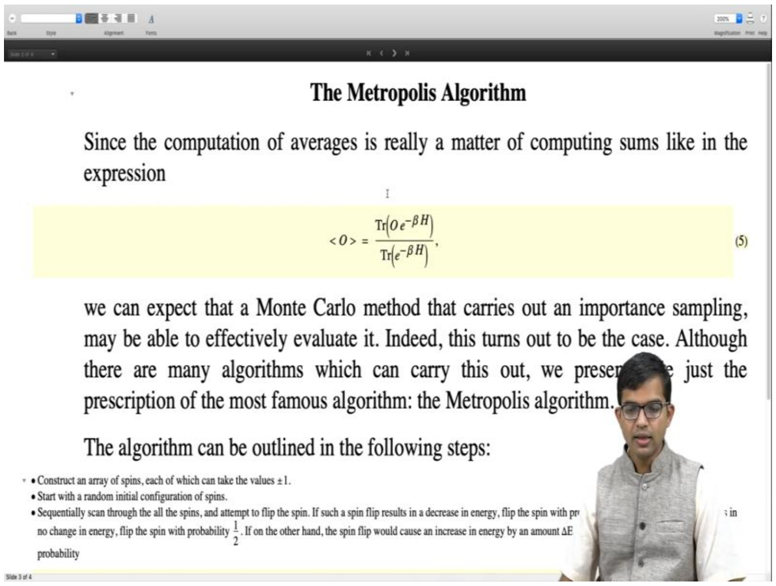
In general, obtaining an analytical closed form expression for the partition function is a very hard problem, and in practically impossible. The one-dimensional Ising model is an exception, where in fact, the exact partition function can be comp xactly. It is beyond the scope of the present discussion to show how this is don vever, we will just simply state one key result which follows from this co tion. The average energy per spin as a function of the inverse temperature is v

$$\frac{E}{N} = -J \tanh(\beta J). \quad (4)$$

And so, if you can evaluate the sum if you want to compute the average value of any quantity is simply given by average of o some operator o like energy. for example, is this trace of this operator times the weight and that weight I told you is just $e^{-\beta H}$ divided by the partition function, which is sitting in the denominator. So, in general, I like I said, it is it is a hard problem.

But for the 1D model, in fact you can very easily write down the partition function exactly, but I am not going to do that for you, but I will just simply borrow one result. It is available in textbooks for the 1D model. It turns out that the energy, the average energy per spin in equilibrium is simply given by $-J\tanh(\beta J)$. $\beta = 1/(k_b T)$ and all this kind of calculation. You are taking Boltzmann constant to be 1. So, it is just the inverse temperature.

(Refer Slide Time: 11:02)



The Metropolis Algorithm

Since the computation of averages is really a matter of computing sums like in the expression

$$< O > = \frac{\mathrm{Tr}\left(O\, e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}, \qquad (5)$$

we can expect that a Monte Carlo method that carries out an importance sampling, may be able to effectively evaluate it. Indeed, this turns out to be the case. Although there are many algorithms which can carry this out, we presen e just the prescription of the most famous algorithm: the Metropolis algorithm.

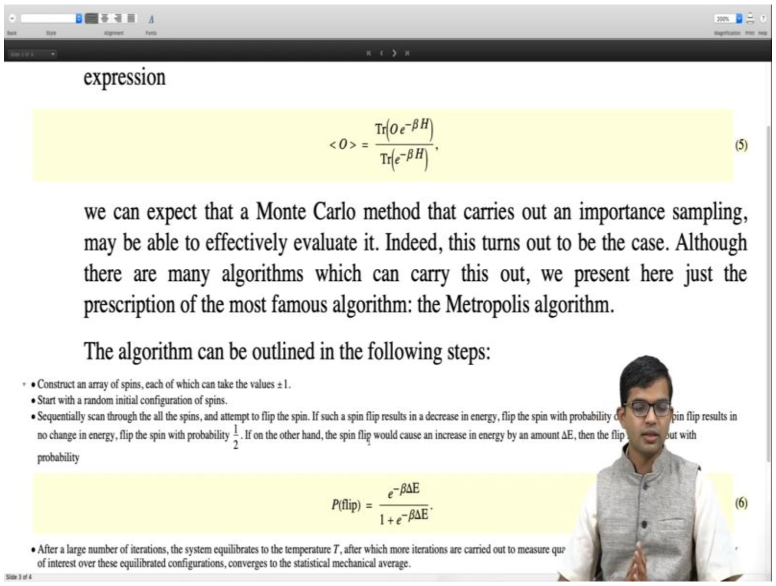
The algorithm can be outlined in the following steps:

- Construct an array of spins, each of which can take the values ± 1.
- Start with a random initial configuration of spins.
- Sequentially scan through the all the spins, and attempt to flip the spin. If such a spin flip results in a decrease in energy, flip the spin with pr s in no change in energy, flip the spin with probability $\frac{1}{2}$. If on the other hand, the spin flip would cause an increase in energy by an amount ΔE probability

Slide 3 of 4

So, our goal here is to use something called the Metropolis algorithm to verify, to get this result. So, let me first describe the Metropolis algorithm, which basically exploits importance sampling. So, since there is this kind of large sum involved one, would get the hunch that there is a, there must be a way to do this with importance sampling. And one of the early algorithms which you know came up with a method to do this is the Metropolis algorithm which is a Monte Carlo method. There are other algorithm which are available, some which are very sophisticated, some which are, you know very fine tuned for certain kinds of system which on, but we are not going there.

(Refer Slide Time: 11:36)



expression

$$< O > = \frac{\mathrm{Tr}\left(O e^{-\beta H}\right)}{\mathrm{Tr}\left(e^{-\beta H}\right)}, \quad (5)$$

we can expect that a Monte Carlo method that carries out an importance sampling, may be able to effectively evaluate it. Indeed, this turns out to be the case. Although there are many algorithms which can carry this out, we present here just the prescription of the most famous algorithm: the Metropolis algorithm.

The algorithm can be outlined in the following steps:

- Construct an array of spins, each of which can take the values ± 1.
- Start with a random initial configuration of spins.
- Sequentially scan through the all the spins, and attempt to flip the spin. If such a spin flip results in a decrease in energy, flip the spin with probability [illegible] pin flip results in no change in energy, flip the spin with probability $\frac{1}{2}$. If on the other hand, the spin flip would cause an increase in energy by an amount ΔE, then the flip [illegible] out with probability

$$P(\mathrm{flip}) = \frac{e^{-\beta \Delta E}}{1 + e^{-\beta \Delta E}}. \quad (6)$$

- After a large number of iterations, the system equilibrates to the temperature T, after which more iterations are carried out to measure qua [illegible] of interest over these equilibrated configurations, converges to the statistical mechanical average.

Slide 3 of 4

Let describe the most famous of this kind of algorithm which is a metropolis algorithm, which is easily described. So, what you do is you construct an array of spins, each of which can take values plus or minus one. That gives you a configuration. Put your put your system in some configuration and then start with some random initial configuration of spins.

Now, sequentially scan through all the spin, start to the first spin and then attempt to flip the spin. There is an energy associated with every stage of your spin. And if you try to flip the first spin so you ask whether this flip is going to reduce energy of the system. If it is, then you do the flip, right. Going downhill in energy is, is a desirable thing so you just go. It does not matter what temperature you want to equilibrate your system to.

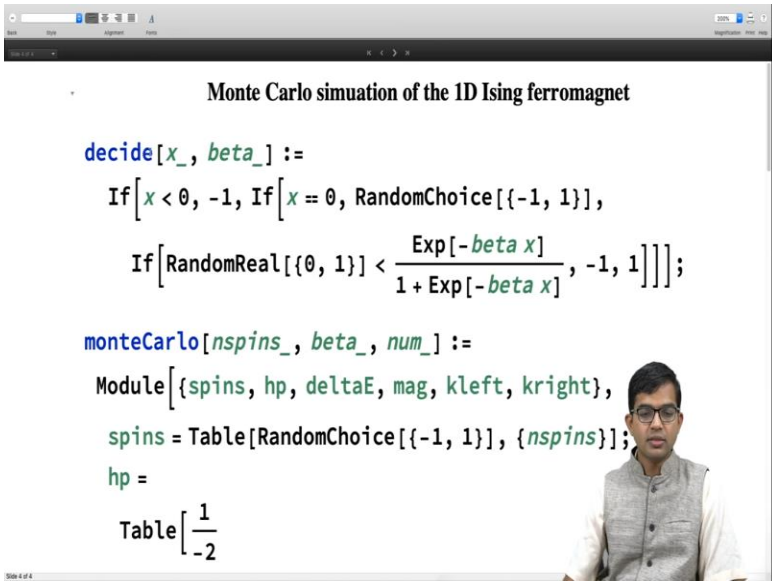
But on the other hand, if you want if this flipping of the spin is going to cause an increase in your energy, then this is… So, does the system have enough temperature? If the temperature is sufficiently high, then perhaps it is able to go up to cross this barrier. But otherwise probably no. So, what we do is, you know, we use this method where you flip your spin only with this probability, which is given by $\frac{e^{-\beta \Delta E}}{1+e^{-\beta \Delta E}}$.

And on the other hand, if the energy is going to remain unchanged, both configurations are equally good. So, you just flip it with probability half, you just so mix the system around. You try sometimes you just keep it as it is or you flip it and then you sequentially scan all spins. This whole thing comes to as one-one type of a spin and then you may have to do this kind of Monte-Carlo spin many, many times before your system basically equilibrates to that temperature. After a large number of such iterations, the system is garneted to equilibrate.

So, there is a you know whole bunch of arguments have establish this. A lot of papers have been written, lots of books have been written. This is, you know, this method has been tested to death. So, there is no doubt that this will work. So, you just accept this algorithm as a prescription and then we will try to implement it. But I mean, underlying this principle is something called a detailed balance. So, there a detailed balance principle which helps your system to go to equilibrium. So, there is a you know, the theory behind why we should actually give you equilibration or why you know this method will you drive your system to equilibrium, this is definitely out of the scope of the syllabus of the current attempt. But believe me that it works.

So, we will show how to implement this. And we will see in the case of the 1D ising model that indeed it the energy will agree with or with what you obtain numerically. So, let us go right there. So, in order to do this, I will first define this function called decide. So, I said that if my energy ΔE, I compute an energy, there is a change in energy ΔE. And if that ΔE is going to be positive or negative or zero, I have to make a decision and decide is here.

(Refer Slide Time: 14:43)



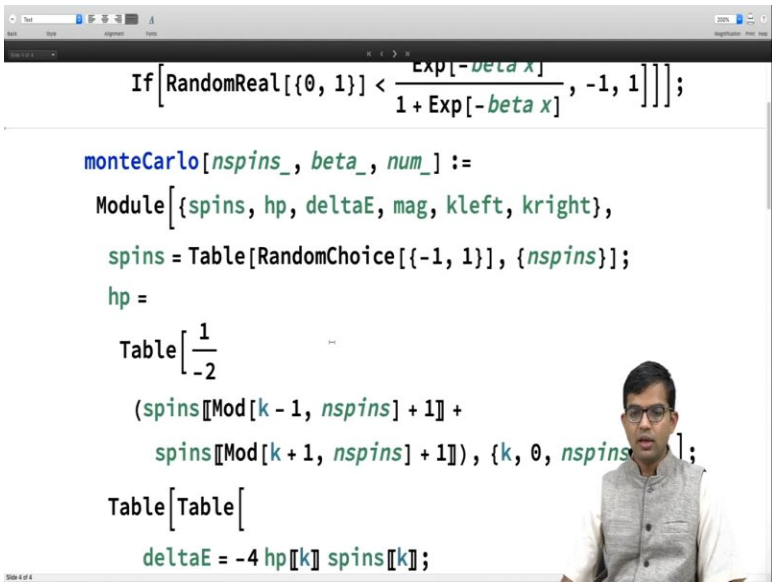
Monte Carlo simuation of the 1D Ising ferromagnet

```
decide[x_, beta_] :=
  If[x < 0, -1, If[x == 0, RandomChoice[{-1, 1}],
    If[RandomReal[{0, 1}] < Exp[-beta x]/(1 + Exp[-beta x]), -1, 1]]];

monteCarlo[nspins_, beta_, num_] :=
 Module[{spins, hp, deltaE, mag, kleft, kright},
  spins = Table[RandomChoice[{-1, 1}], {nspins}];
  hp =
   Table[1/-2
```

So, here I have got an X, which is my ΔE, β is inverse of temperature. If x < 0, then I will return -1. If X = 0, then I will give -1 or +1. So, you will see in a moment why I want to use -1 here. And if it is a random real else otherwise if it is positive, then I will generate a random real. And then if this quantity is less than this probability that I have already said, you know, that is the metropolis probability. Then I will give you -1. You flip it with, with this probability. And if the number is greater than this, then I will not flip. So, that is correspond to just giving 1. So, I will use this function, I will need to use this function. And I have a module.

(Refer Slide Time: 15:33)



```
      If[RandomReal[{0, 1}] < Exp[-beta x] / (1 + Exp[-beta x]), -1, 1]]]];

monteCarlo[nspins_, beta_, num_] :=
 Module[{spins, hp, deltaE, mag, kleft, kright},
  spins = Table[RandomChoice[{-1, 1}], {nspins}];
  hp =
   Table[1/-2
     (spins[[Mod[k - 1, nspins] + 1]] +
       spins[[Mod[k + 1, nspins] + 1]]), {k, 0, nspins ];
  Table[Table[
    deltaE = -4 hp[[k]] spins[[k]];
```

So, I will try to explain this module for you, but maybe what you should do is stare at this a little bit. You have to spend some time to be actually to understand what is going on here. And I do not want to go into all the details of how this works, so I have this module. So, when I define a module, I have to give it all this input possibilities. So, there is nspins these are number of spins, β is inverse temperature and num is this Monte Carlo iteration, how many times do I want to do? You will see inside in a moment, I will describe what exactly num is. And then I have these local variables spins hp, ΔE magnetization, kleft, k ight.

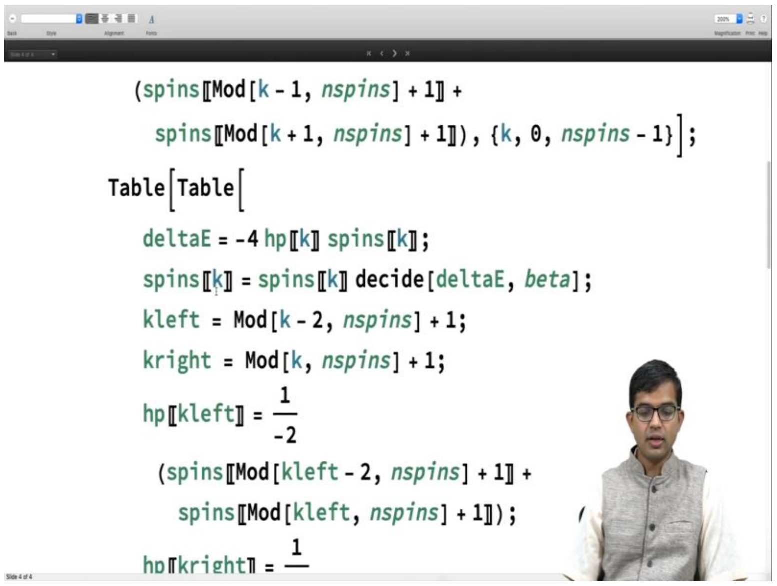
Actually, I did not use magnetization, but it is okay. Maybe you can leave it as it is. And if you, if you are interested in computing magnetization, you can go ahead and do it as well. I want to keep this as simple as possible; I will be computing only the energy. So, I will start with you know all these spins. There is a local array, I will create a table of random choice between -1 and +1. So, some of them are down and some of them are up.

So, this is a random initial configuration. First step easy to do. Now hp is going to generate for me the local field that your spin feels. So, I forgot to mention that I have assumed periodic boundary conditions here. So, I have $-J\sum_{i=1}^{N}.$ So, if you put i equal to N you see that s_n and s_{n+1}. But there is no s_{n+1} but s_{n+1} is understood to be s_1. In order to take care of this periodic boundary conditions I have to do some work here.

So, that why I use this thing called mod so $Mod[k-1, nspins]+1$ I have to do this. So, I am not going to try to explain this to you because it will probably consume a lot of time. So, but I want you to play with this and see that indeed if I just generate this table. So, this hp is what? hp is simply the local field that the ith spin feels. So, you see so I can right this Hamiltonian H as nothing but $\sum_i H_i S_i$. Now H_i is going to come from just the neighbour s_{i+1} and s_{i-1}. But since each of them is going to be repeated two times, I will have to divide it by 2 so I should actually write $H_i S_i$ where H_i is nothing but $(s_{i+1}+s_{i-1})/2$, subject to the condition that you know you have to understand is i-1 as sort of wrapped around. If you are at the edge, you have to wrap it around. All of that can be taken care of with this mod function. That is what I have done here. I am computing an array of the local fields. So, it is just -1/2. So, like I just said, minus comes because I have a -J. And in this whole calculation I have put J = 1, so there is no need for me to carry this J around and then spins instead of just doing k-1. I have to take care of this do a mod and then add and then to go to the spin on the right. I will have to do (k+1, nspins). And then also I have to add this.

And notice that I am allowing my k to go from 0 to nspins - 1. That is why I need to add this because nspins can take labels only from 1 to n or 1 to nspins. So, you need to open this up and convince yourself that this works.

(Refer Slide Time: 19:17)



```
    (spins⟦Mod[k - 1, nspins] + 1⟧ +
      spins⟦Mod[k + 1, nspins] + 1⟧), {k, 0, nspins - 1}];
Table[Table[
   deltaE = -4 hp⟦k⟧ spins⟦k⟧;
   spins⟦k⟧ = spins⟦k⟧ decide[deltaE, beta];
   kleft = Mod[k - 2, nspins] + 1;
   kright = Mod[k, nspins] + 1;
   hp⟦kleft⟧ = 1/-2
     (spins⟦Mod[kleft - 2, nspins] + 1⟧ +
       spins⟦Mod[kleft, nspins] + 1⟧);
   hp⟦kright⟧ = 1
```

Now then what do I do? I create a table. So, look at the inside table, this table generates for me. First, what it does is it finds delta E; suppose I go to the kth spin. So, this table has k going from 1 to n spins. So, this is what I said sequentially scan through all those things. You go to the first spin and ask what will be the cost of my spin, spin flip. So, that I am claiming is just delta E equal to minus four times local field times that spin, particular spin kth spin.

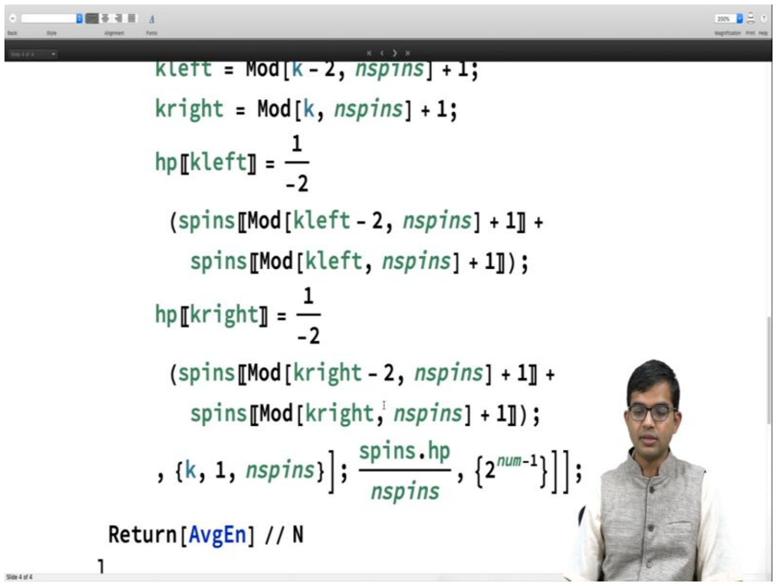
So, this you have to verify and it is not difficult because I told you that the whole system can be written a summation over Hi Si. So, you just work this out and you see that there is also a factor of 4 which comes in here according to the definition that I have, I have here. And then you will have spins of k is just spins of k into decide. So now, comes the point why I decided, I use this to give me either +1 or -1 so it makes a decision. I will flip the spin if this decide. Will I, will give me, I got a -1 or +1? If it gives me minus 1 it means that spins has become minus spins if it has given me a +1, it remains as it is. So, with a certain probability. This is going to get flipped or not.

And then I compute kleft. What is a spin to the left of this k? It is actually k mod. I have to do (K-2, nspins+1). Yeah this I want you to figure out why this works, by playing. You take nspins to be 10 or 20 or whatever and you try out various values of k and you take a mod of this and wrap it around. You find the correct answer for the spin to the left and the spin to the right. And then I have kright is equal to mod of (k, nspins+1). So, believe me, this works. And actually, do not believe me. You should test this and convince yourself that this indeed works.

So then what I do is I just update my hp. So, if I updated spins of k, I was update hp of left to this quantity 1/(-2) times the same expression as I have here. So, first you must convince yourself of this and then you will be able to quickly convince yourself of this as well. So, I am just using kleft here and then for hp of k ight? I will be using you know kright-2 and kright. The logic is very similar to here. So, first I have to find kleft and then I have to update the hp corresponding to that spin. So then again, with respect to that, I have to go to its left and to its right.

So, that is why there were two steps involved here. And then I will allow k to go from 1 to nspins. Now I want to be able to, I saw this entire thing constitutes 1 Monte-Carlo sweep but I want to do this entire Monte-Carlo sweep many, many times because this is how I will get my system to equilibrate. And I find that it is useful to do this in power of 2. So, what I will do is if I have given been given this num as an input, I will run it only to $2^n - 1$. That will be my equilibration. All of this constitutes equilibration.

(Refer Slide Time: 22:54)



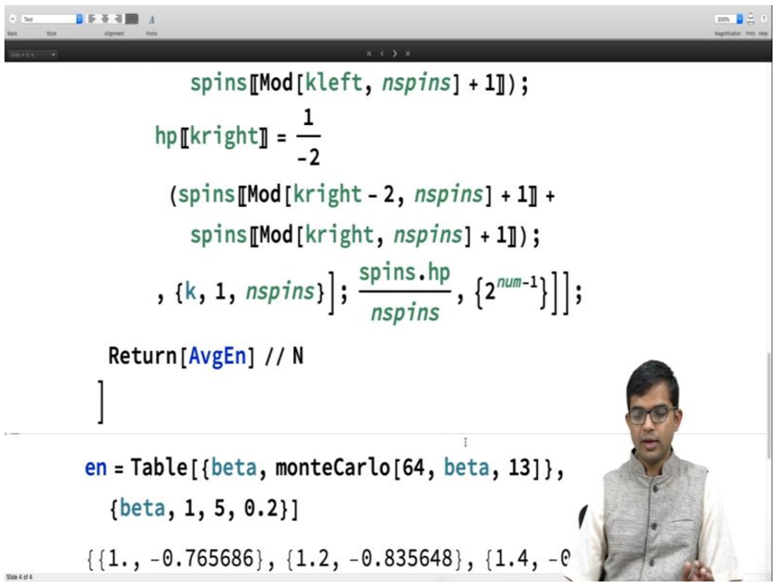
```
kleft = Mod[k - 2, nspins] + 1;
kright = Mod[k, nspins] + 1;
hp[[kleft]] = 1/-2
  (spins[[Mod[kleft - 2, nspins] + 1]] +
    spins[[Mod[kleft, nspins] + 1]]);
hp[[kright]] = 1/-2
  (spins[[Mod[kright - 2, nspins] + 1]] +
    spins[[Mod[kright, nspins] + 1]]);
, {k, 1, nspins}]; spins.hp/nspins , {2^(num-1)}]];
Return[AvgEn] // N
]
```

And then I will actually run the whole thing again, exactly the same thing I have copied here, and then I will be running again $2^n - 1$. But this time I will also compute this $spins.hp/nspins$. Now this spins up to this point it's the same, the previous co-renders are the same. But here I am asking you to compute this quantity spins.hp. So, if you see what is spins? Spins is just the configuration your system is in and hp is the local field that it feels. If I take the dot product of 2 arrays, it is just going to give me H1 S1 plus H2 S2 plus H3 S3 so on.

And what is that quantity is nothing but energy. Convince yourself of this. You have to play with this explicitly check that indeed it is actually the energy. And then I want energy/nspins. So, I want energy per unit spin. So, I will divide it by n spins and then do this many, many times. And so, I will also take a mean of this name because I am interested in an average. Once the data has got equilibrated after doing it so many times. Basically, it has got an equilibrium. Then I want to compute the mean or whatever quantity is of interest for me.

And here I am saying I am only interested in energy per number of spins. You can of course compute other quantities that so later and that will be exercised for you, what kind of quantities you want to work out. So, average energy is just given by this and then I will just simply return this average energy that is all, this module will give me average energy. So, finally let me show you how this works out.

(Refer Slide Time: 24:14)

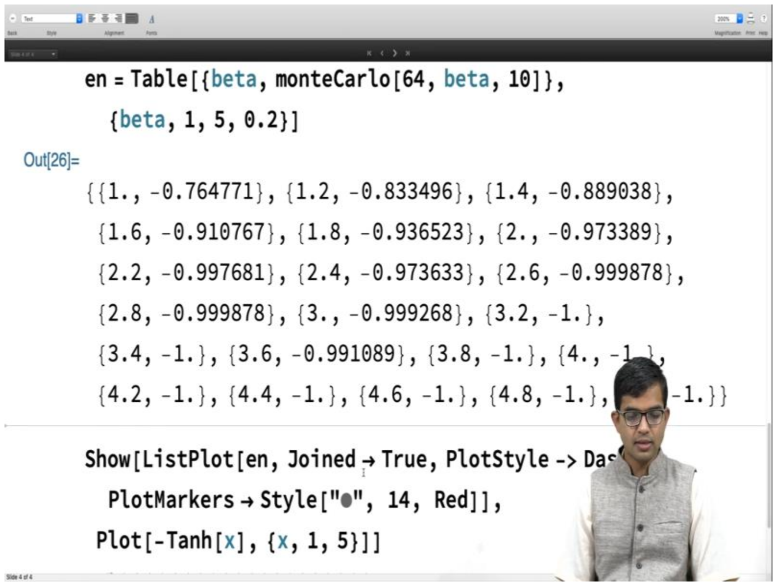


```
            spins[[Mod[kleft, nspins] + 1]]);
                       1
         hp[[kright]] = ---
                       -2
           (spins[[Mod[kright - 2, nspins] + 1]] +
            spins[[Mod[kright, nspins] + 1]]);
                                spins.hp
         , {k, 1, nspins}]; --------, {2^(num-1)}]];
                                 nspins
   Return[AvgEn] // N
  ]

  en = Table[{beta, monteCarlo[64, beta, 13]},
    {beta, 1, 5, 0.2}]
  {{1., -0.765686}, {1.2, -0.835648}, {1.4, -0
```

So, I will generate a table of this. I want to consider not just one energy for a particular value of β, but I will do it for a range of β going from 1 to 5 in steps of 0.2 let me take this num to be maybe 10. I will take this to be ten, let us generate this data. And then I know what. This is supposed to give me. I know that this is supposed to compare against this tanh function hyperbolic tan. So, I have taken my number of spins to be 64. You can play with other values of so it is still running that where you go it has completed it.

(Refer Slide Time: 24:59)

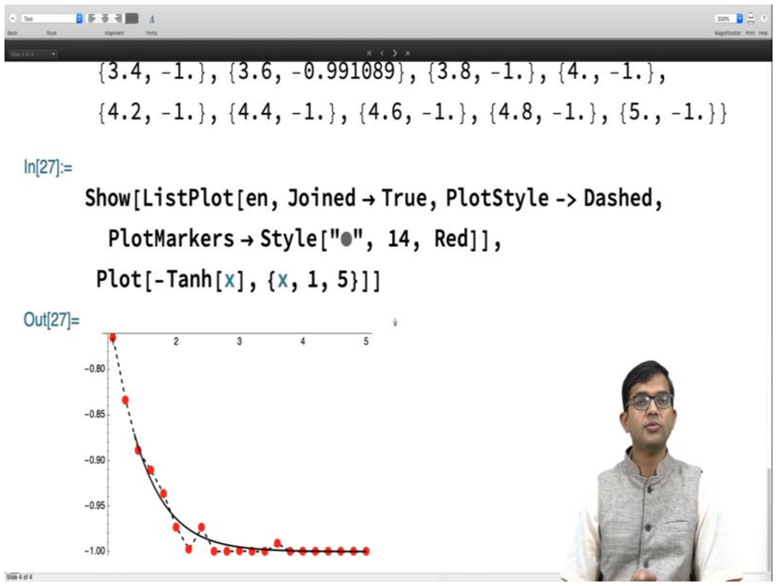


```
en = Table[{beta, monteCarlo[64, beta, 10]},
  {beta, 1, 5, 0.2}]
Out[26]=
{{1., -0.764771}, {1.2, -0.833496}, {1.4, -0.889038},
 {1.6, -0.910767}, {1.8, -0.936523}, {2., -0.973389},
 {2.2, -0.997681}, {2.4, -0.973633}, {2.6, -0.999878},
 {2.8, -0.999878}, {3., -0.999268}, {3.2, -1.},
 {3.4, -1.}, {3.6, -0.991089}, {3.8, -1.}, {4., -1.},
 {4.2, -1.}, {4.4, -1.}, {4.6, -1.}, {4.8, -1.}, [illegible] -1.}}

Show[ListPlot[en, Joined → True, PlotStyle -> Das
  PlotMarkers → Style["●", 14, Red]],
 Plot[-Tanh[x], {x, 1, 5}]]
```

Now I am going to go ahead and compare this against -tanh(x). I have taken J to be 1 so it is just simply minus hyperbolic tan. So, let us see, how these two functions compare and there you go. So, you see that indeed the Monte-Carlo data, the red guys are the Monte-Carlo data agrees very well with, with the analytical expectation. So, in fact, you should go ahead and play with this. You know you can change the system size larger the system size, the better will it become. And the larger of this value here, it is going to give you better results, but it is also going to consume more time. So, to keep it simple here I have chosen it to be 64 and only 10 here. You can play with different numbers and convince yourself that indeed it works.

(Refer Slide Time: 25:42)



```
{3.4, -1.}, {3.6, -0.991089}, {3.8, -1.}, {4., -1.},
{4.2, -1.}, {4.4, -1.}, {4.6, -1.}, {4.8, -1.}, {5., -1.}}
```

In[27]:=

```
Show[ListPlot[en, Joined → True, PlotStyle -> Dashed,
  PlotMarkers → Style["●", 14, Red]],
 Plot[-Tanh[x], {x, 1, 5}]]
```

Out[27]=

So, what have we learned here? In this module we have looked at the ising model. I have sort of quickly given you a crash course in the statistical mechanics of the 1-dimensional ising model. Not really going into the details at all, but just telling you that it is possible to compute the partition function from which all interesting quantities can be derived. And I have just stated the result of, you know what is the average energy as a function of temperature. And then I told you what the Metropolis algorithm is. And then I have told you that it is if you do it right, it is guaranteed to equilibrate.

And it works particularly well for spins system, like the ising model in 1D. And we check it by running a Monte-Carlo simulation, using a Metropolis algorithm for the 1-dimensional ising model. I showed you how to write the code for this and I ran the code and I check that my energy per number of spins agrees very well with the Monte-Carlo data, agrees very well with the analytical expectation. So, you can go ahead and play with other quantities, look up the partition function, look up functions or you know susceptibility or other quantity and verify that indeed it works.

And maybe you can also tweak the code to make it more efficient and also play games with other lattices. Once you have it for 1D, there is no stopping you from playing 2D ising model, which is a very famous problem, very hard, difficult problem, but analytically solvable. Where expressions are available. You can cross-check. Then you can go to 3D where there is no analytical solution available, and it is a very hard problem. And you can do Monte Carlo simulation, then extract a lot of information in higher dimensional or other kinds of latency and so on.

So, it is a whole art form. Monte-Carlo of the spin systems. You know, people spend their entire lives, some people you know they build their career based on Monte-Carlo simulations and it is a state-of-the-art technique. You know oftentimes there are so many sophisticated tools and you know details which are you know very specific to the kind of system being studied that you know many years of coding are involved in actually mastering this skill. So, but I think we have got a flavour of the Monte-Carlo method and in particular the metropolis algorithm based on this discussion here. Thank you.