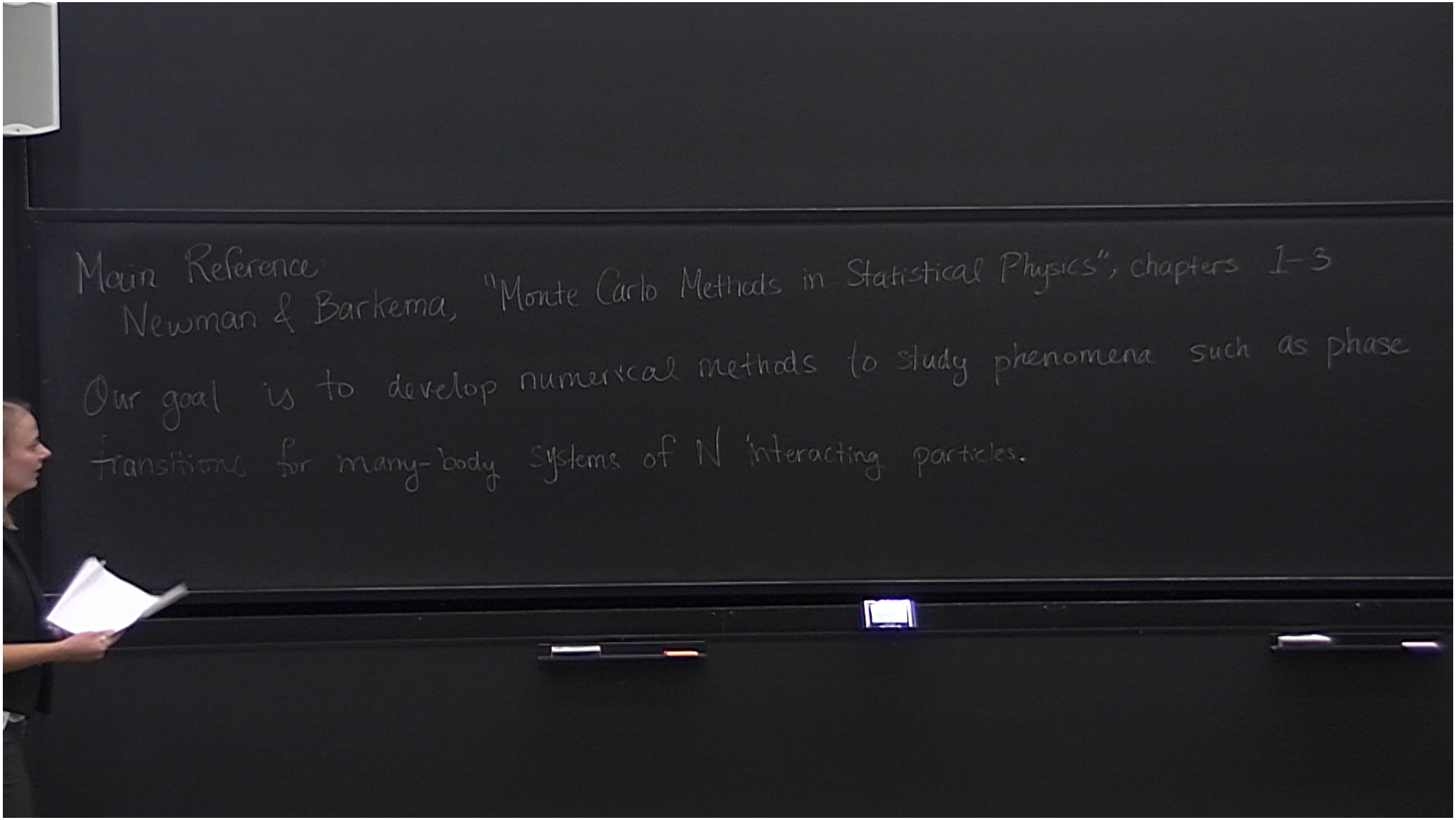
Title: PSI 17/18 - Statistical Mechanics - Lecture 12

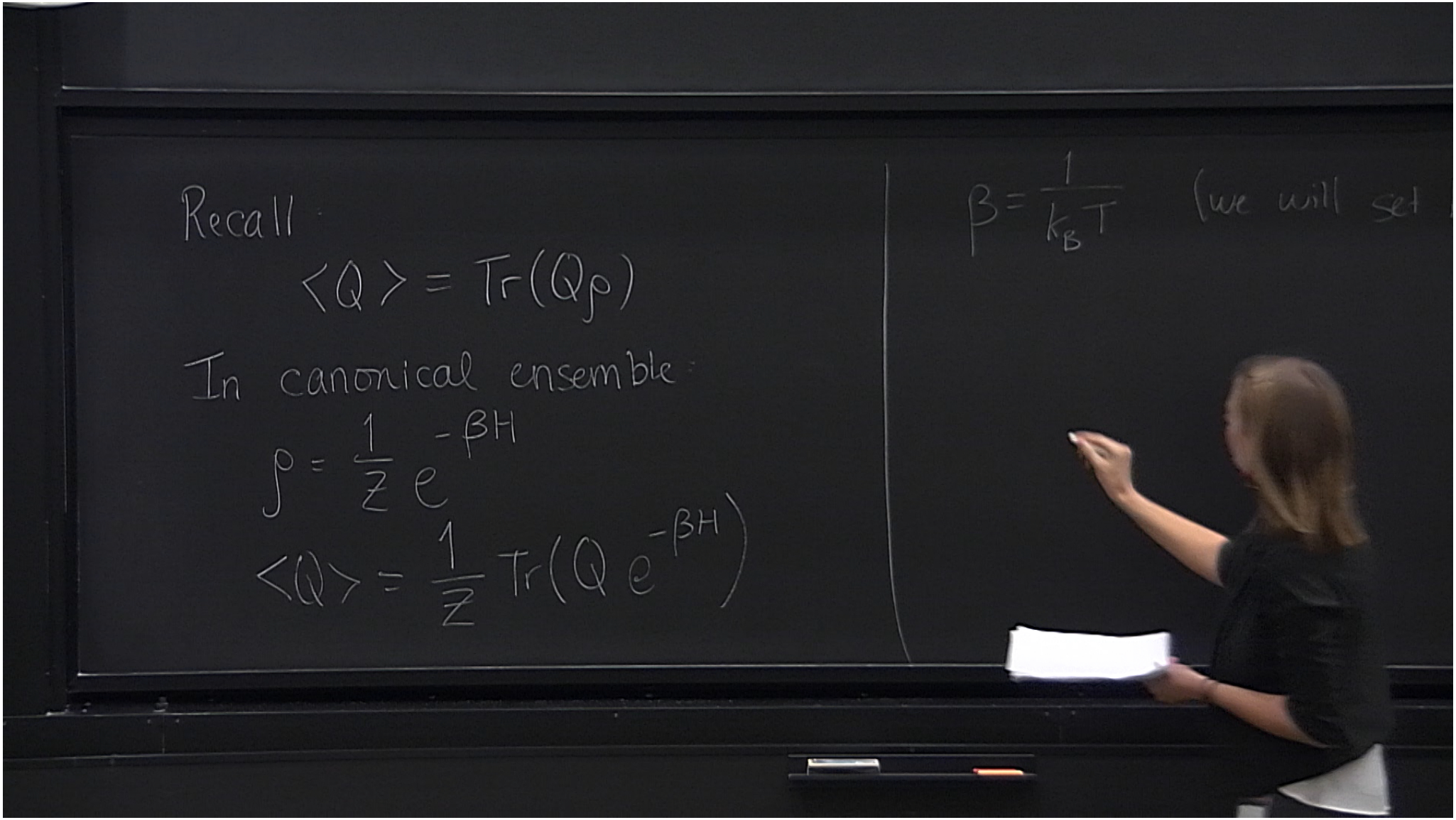
Date: Oct 25, 2017 10:45 AM

URL: http://pirsa.org/17100045

Abstract:

Main Reference:
Newman & Barkema, "Monte Carlo Methods in Statistical Physics", chapters 1-3


Main Reference
Newman & Barkema, "Monte Carlo Methods in Statistical Physics", chapters 1-3
Our goal is to develop numerical methods to study phenomena such as phase transitions for many-body systems of N interacting particles.


Recall
$\langle Q \rangle = Tr(Q\rho)$
In canonical ensemble
$\rho = \frac{1}{Z} e^{-\beta H}$
$\langle Q \rangle = \frac{1}{Z} Tr(Q e^{-\beta H})$
$\beta = \frac{1}{k_B T}$ (we will set

$\beta = \frac{1}{k_B T}$ (we will set $k_B = 1$)

$Z = \mathrm{Tr}(e^{-\beta H})$ is the partition function

$\beta = \frac{1}{k_B T}$ (we will set $k_B = 1$)

$Z = \mathrm{Tr}(e^{-\beta H})$ is the partition function

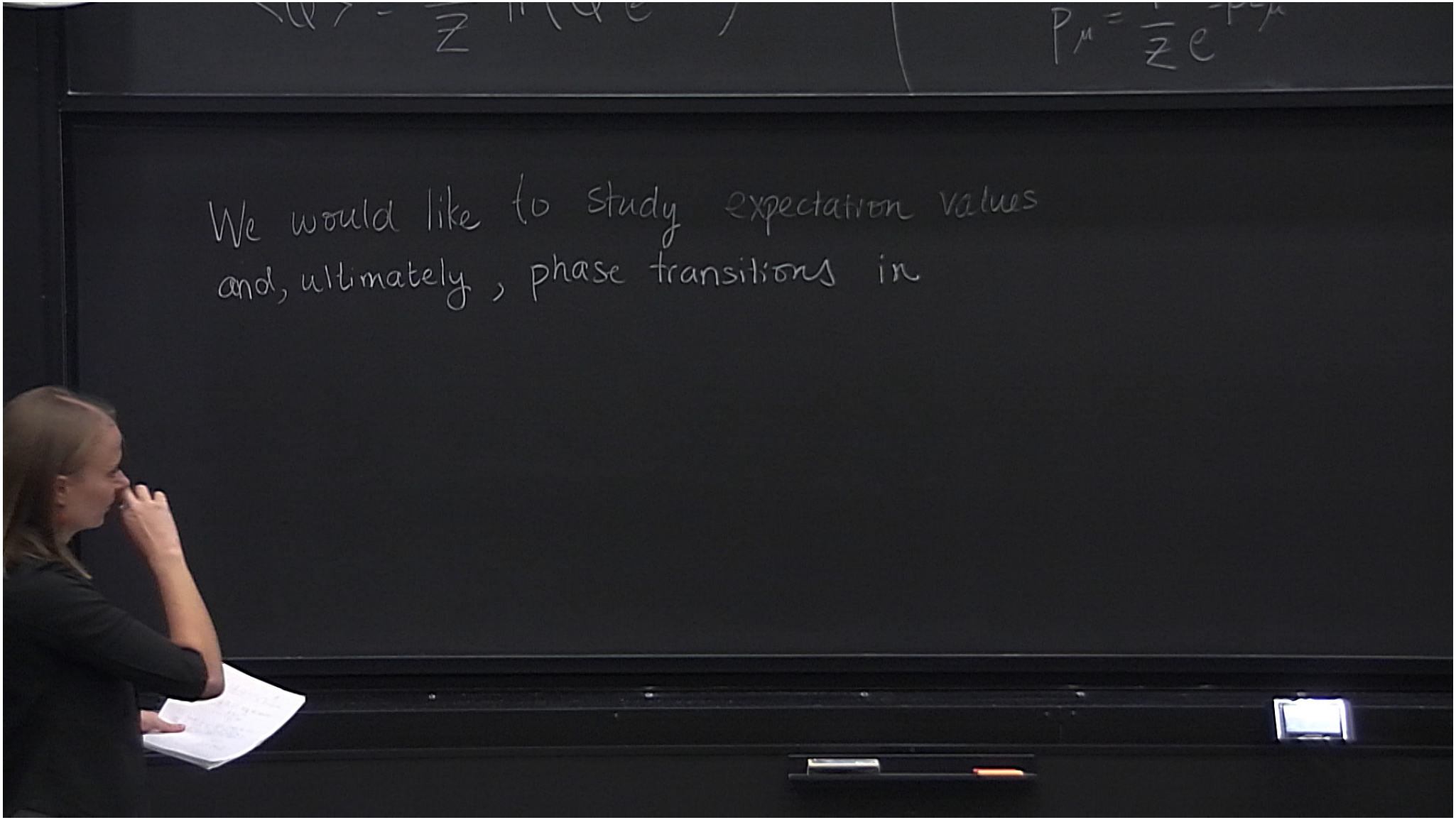
For classical systems:

$\langle Q \rangle = \sum_\mu Q_\mu P_\mu$ $\quad Z = \sum_\mu e^{-\beta E_\mu}$

$P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$

μ represents a state of the system

E_μ is the energy of state μ


We would like to study expectation values
and, ultimately, phase transitions in

$P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$

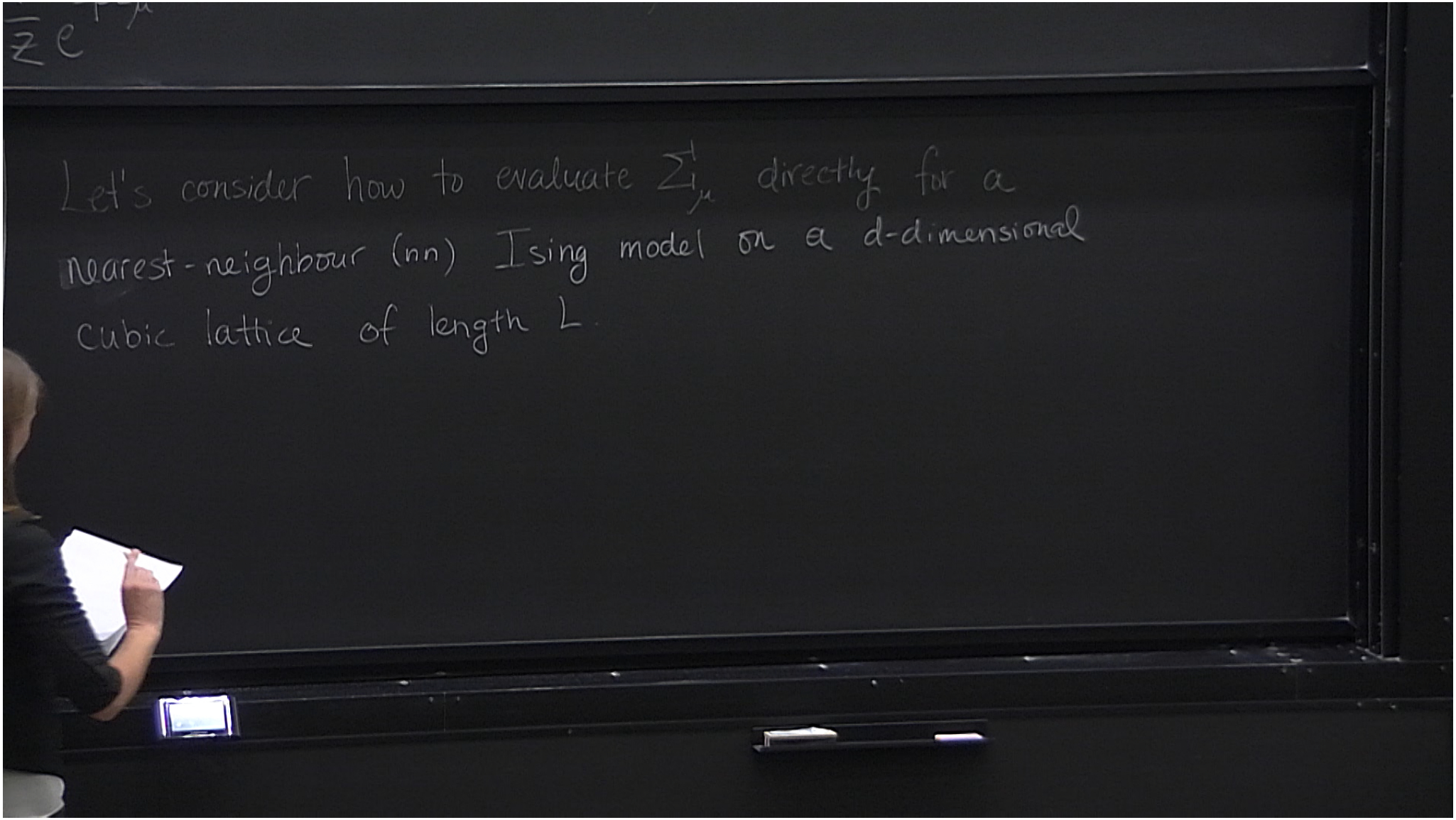
We would like to study expectation values and, ultimately, phase transitions in the thermodynamic limit ($N \to \infty$).

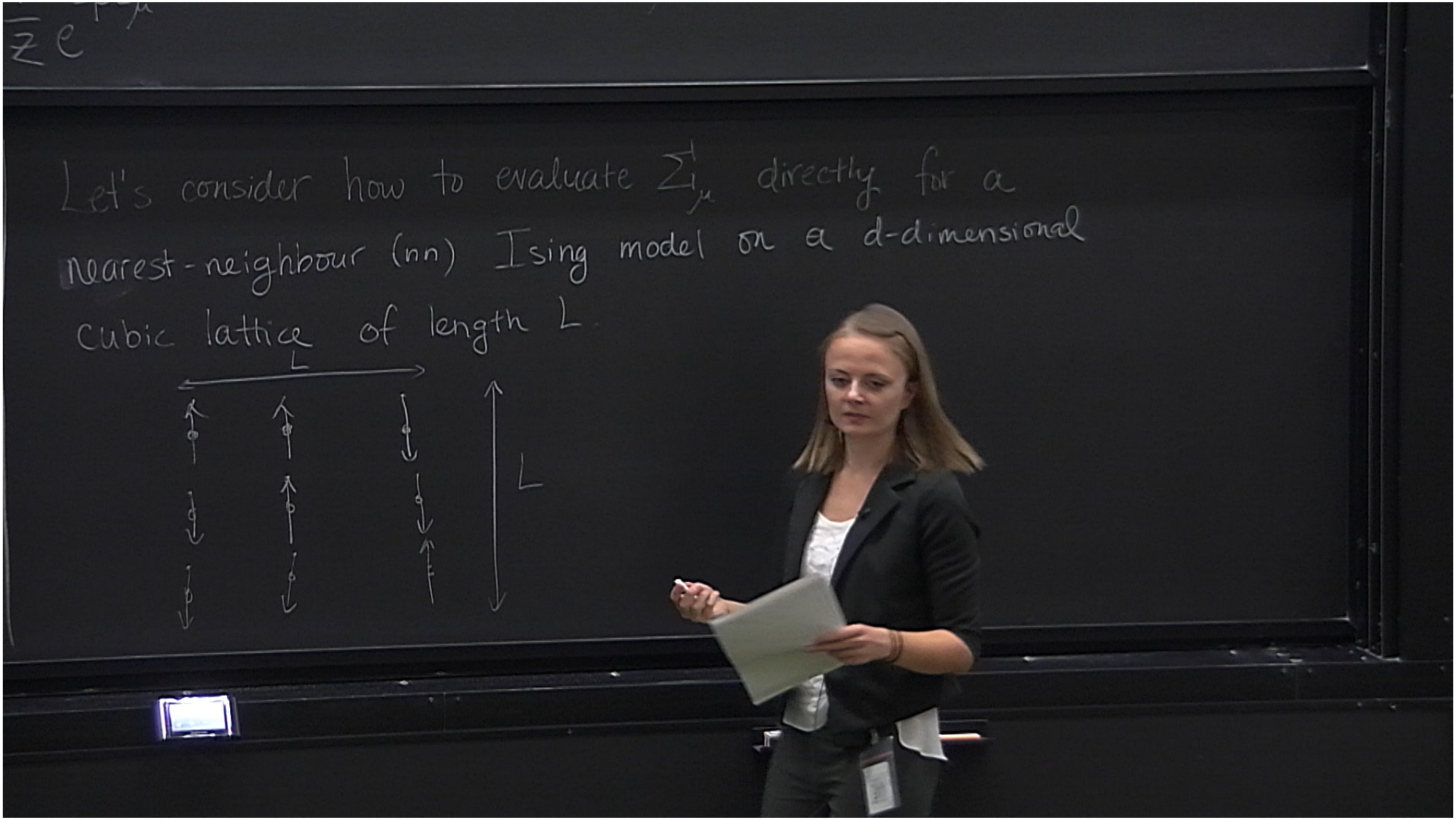
For example:

$$\langle E \rangle = \frac{\sum_\mu E_\mu e^{-\beta E_\mu}}{\sum_\mu e^{-\beta E_\mu}}$$

$Z e^{\beta \mu}$

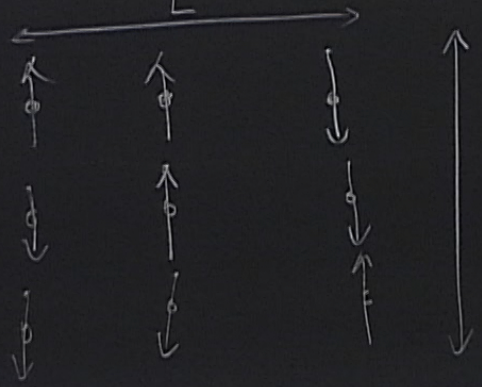
Let's consider how to evaluate $\Sigma_{i,\mu}^{\uparrow}$ directly for a nearest-neighbour (nn) Ising model on a d-dimensional


Let's consider how to evaluate directly for a
nearest-neighbour (nn) Ising model on a d-dimensional
cubic lattice of length L.


Let's consider how to evaluate directly for a
nearest-neighbour (nn) Ising model on a d-dimensional
cubic lattice of length L
L
L

$Z e^{\beta \mu}$

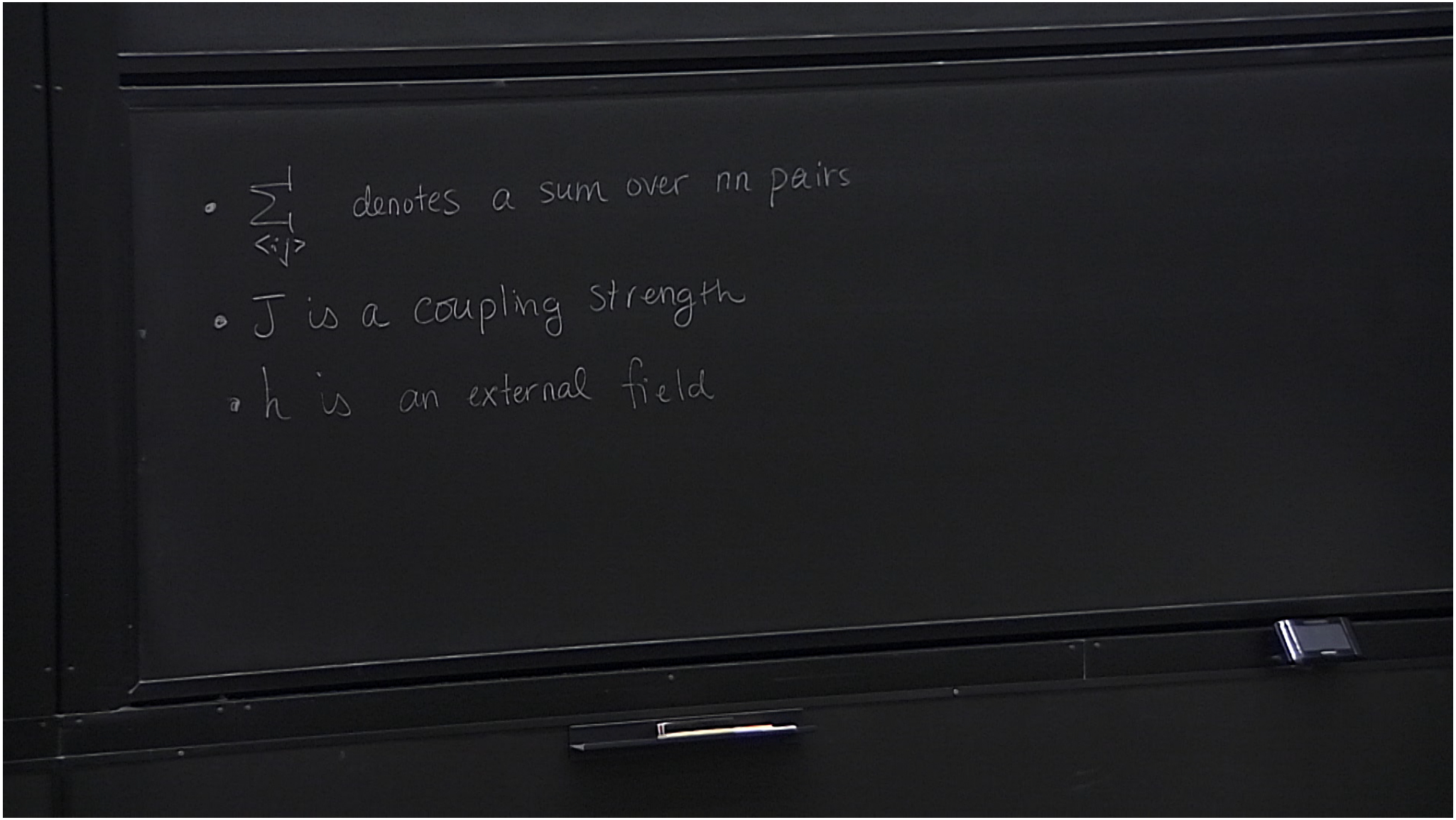
Let's consider how to evaluate $\sum_{i,\mu}^{\prime}$ directly for a nearest-neighbour (nn) Ising model on a d-dimensional cubic lattice of length L.

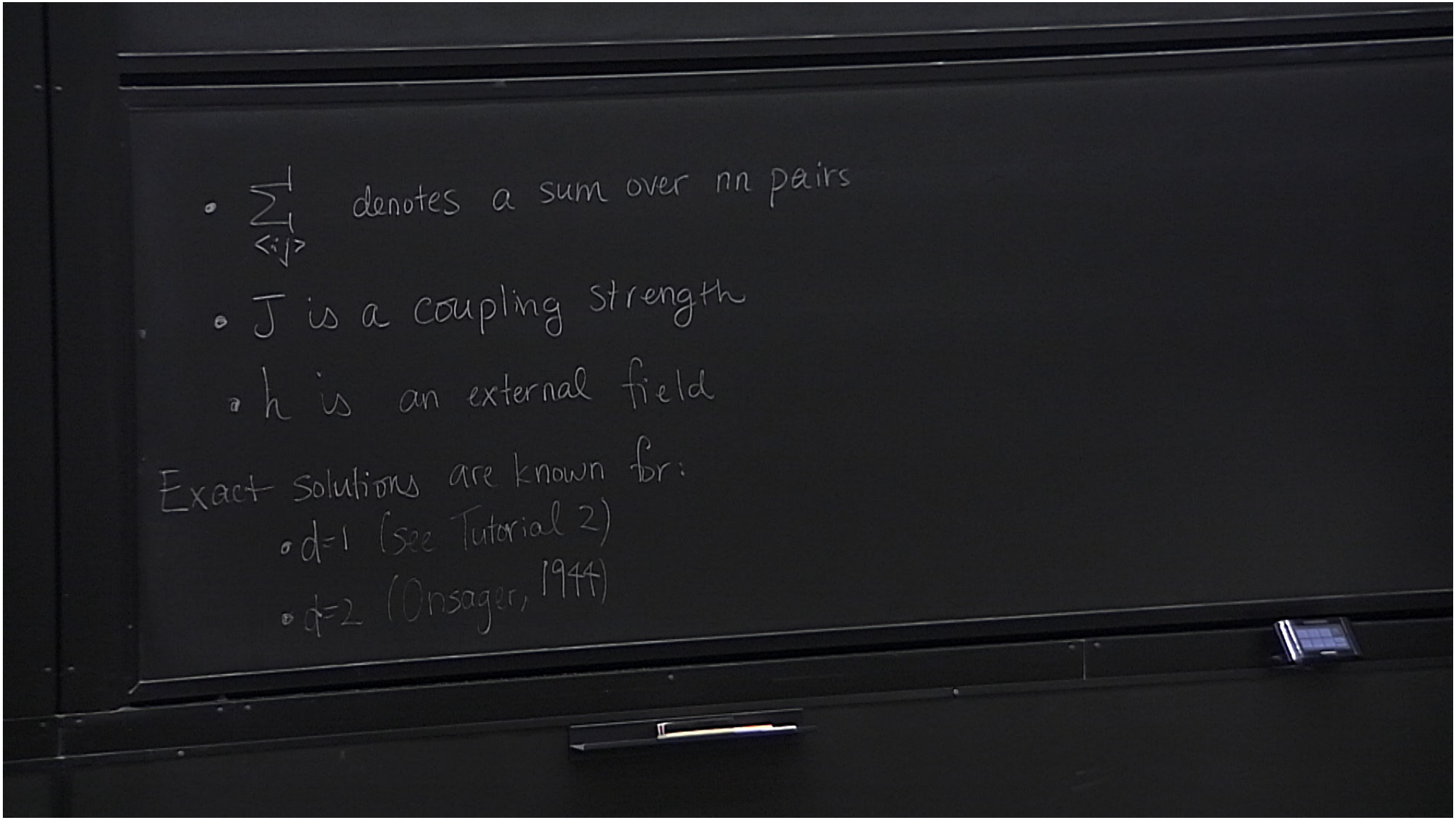


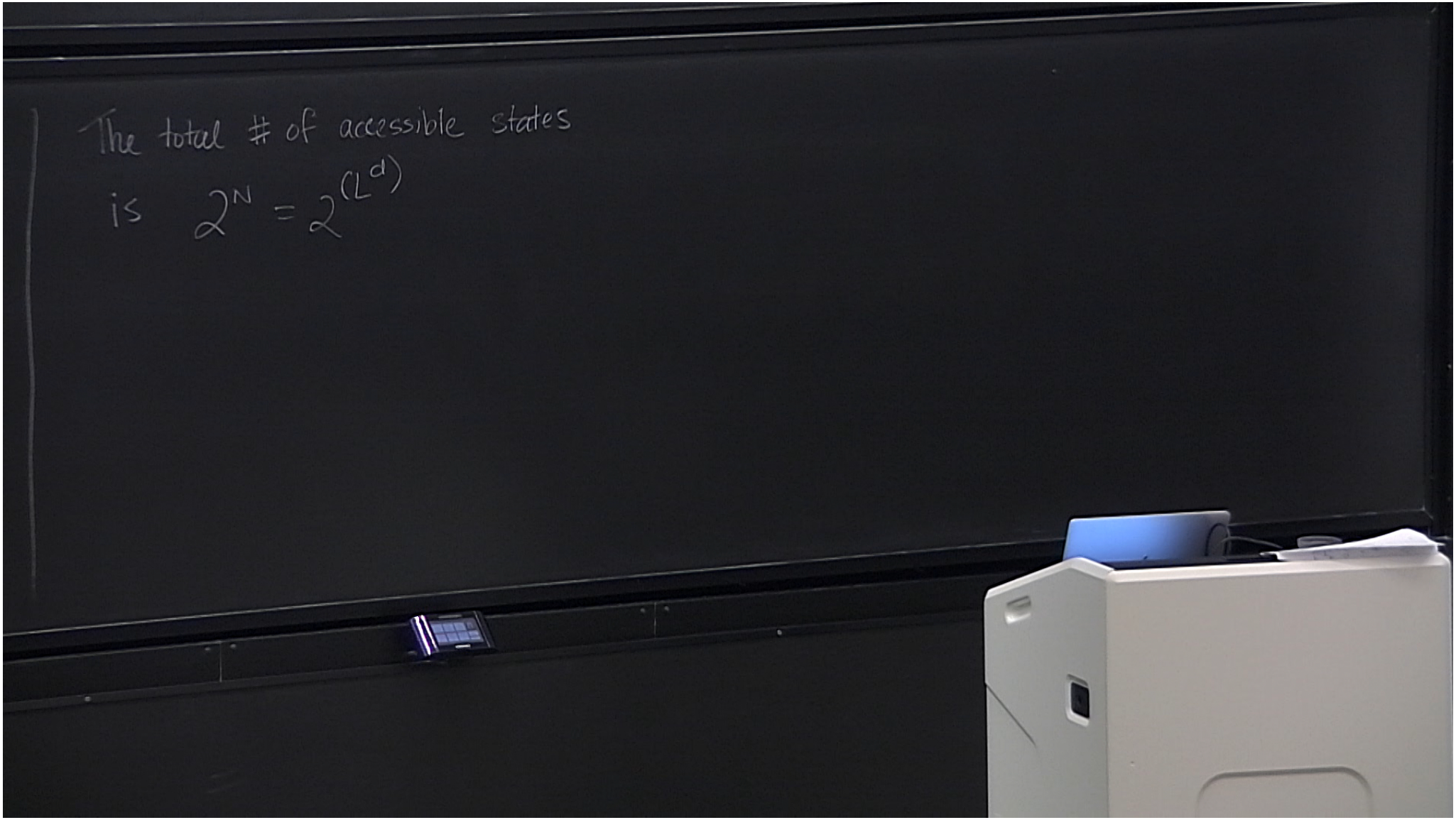
$N = L^d$

On each site i, there is a binary variable $\sigma_i = \pm 1$

$$H = -J \sum_{\langle ij \rangle}^{\prime} \sigma_i \sigma_j - h \sum_i^{\prime} \sigma_i$$

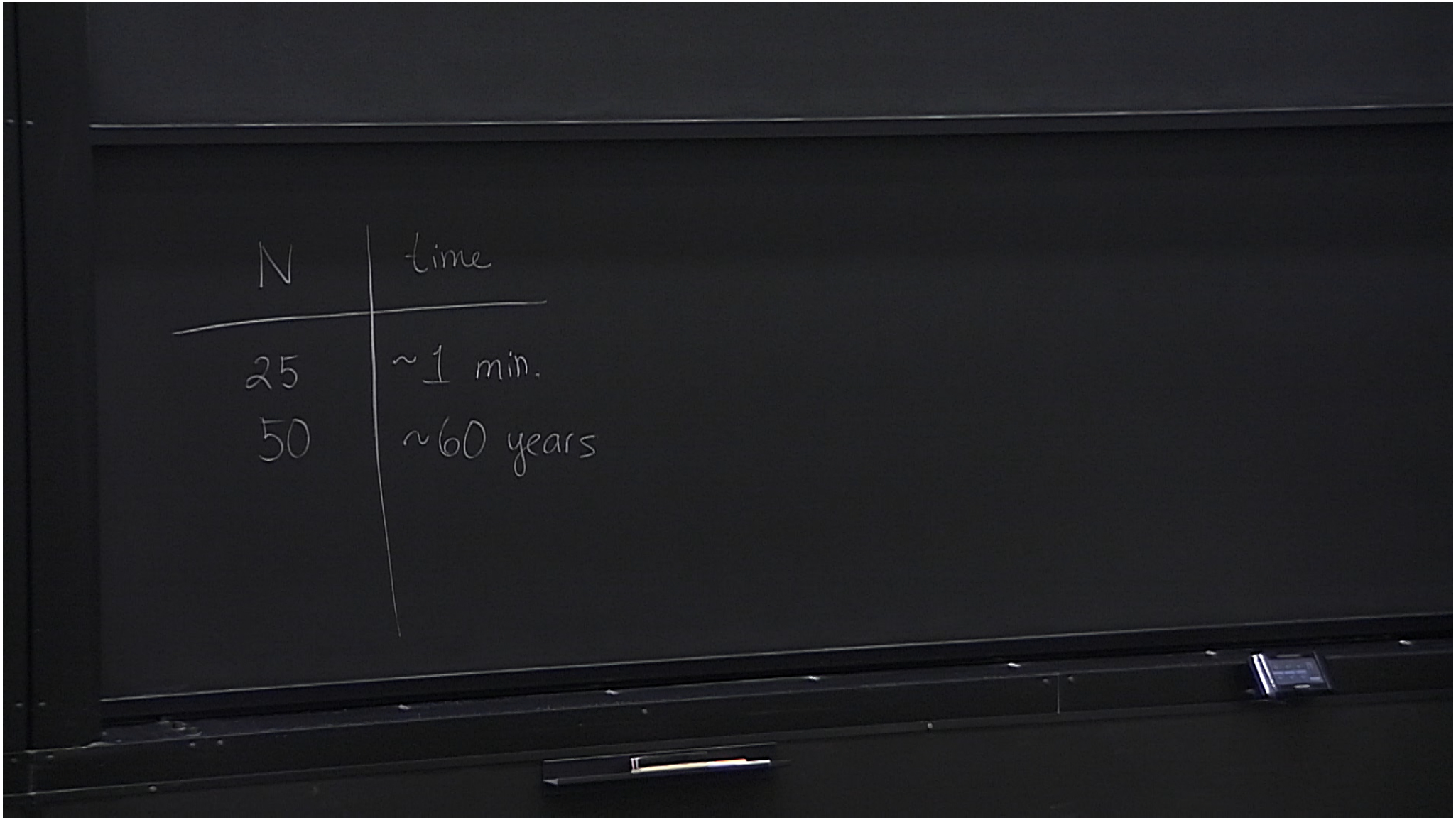

• $\sum_{<i,j>}$ denotes a sum over nn pairs
• J is a coupling strength
• h is an external field

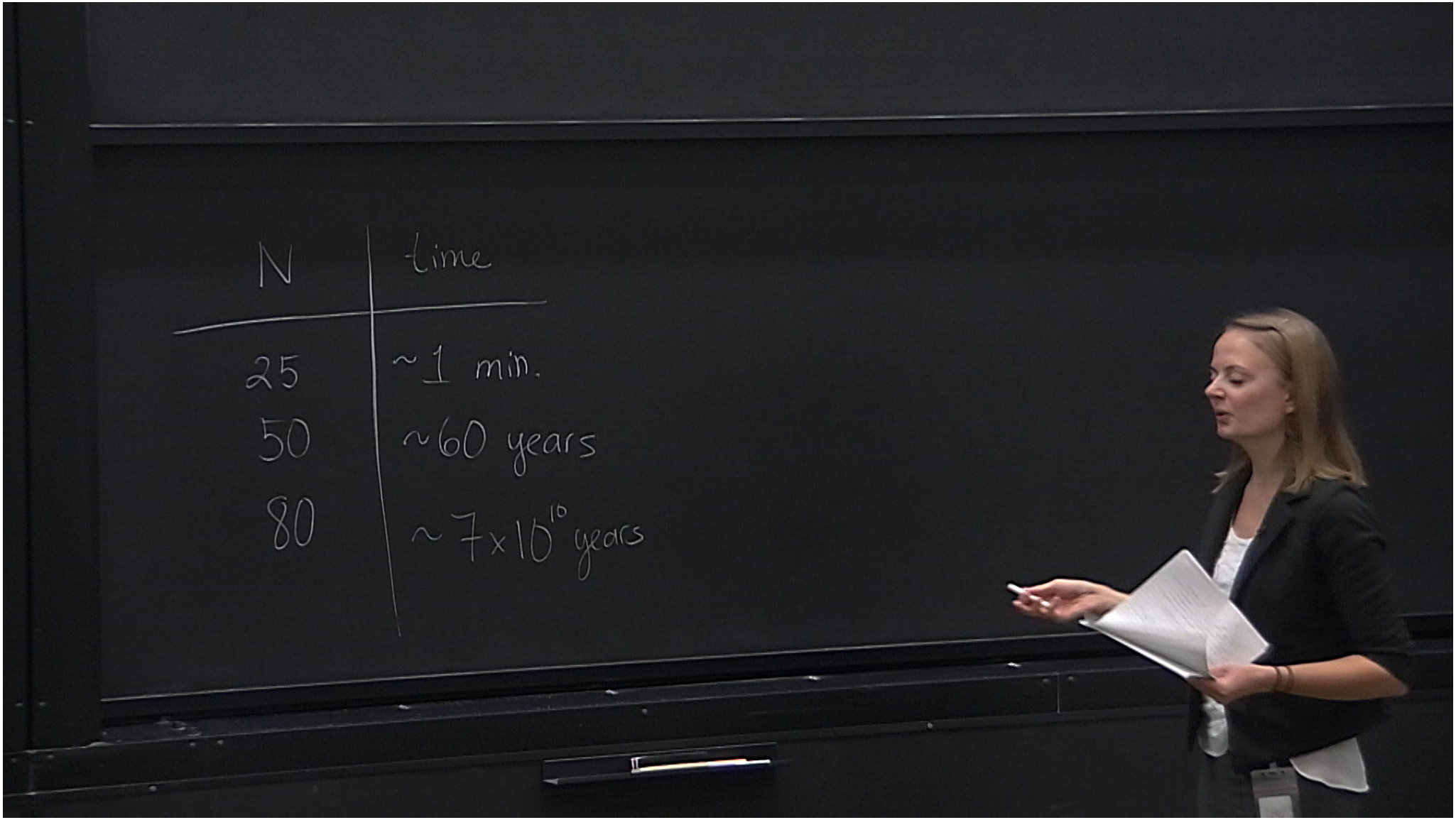

• $\sum_{\langle i,j \rangle}$ denotes a sum over nn pairs
• J is a coupling strength
• h is an external field
Exact solutions are known for:
• d=1 (see Tutorial 2)
• d=2 (Onsager, 1944)

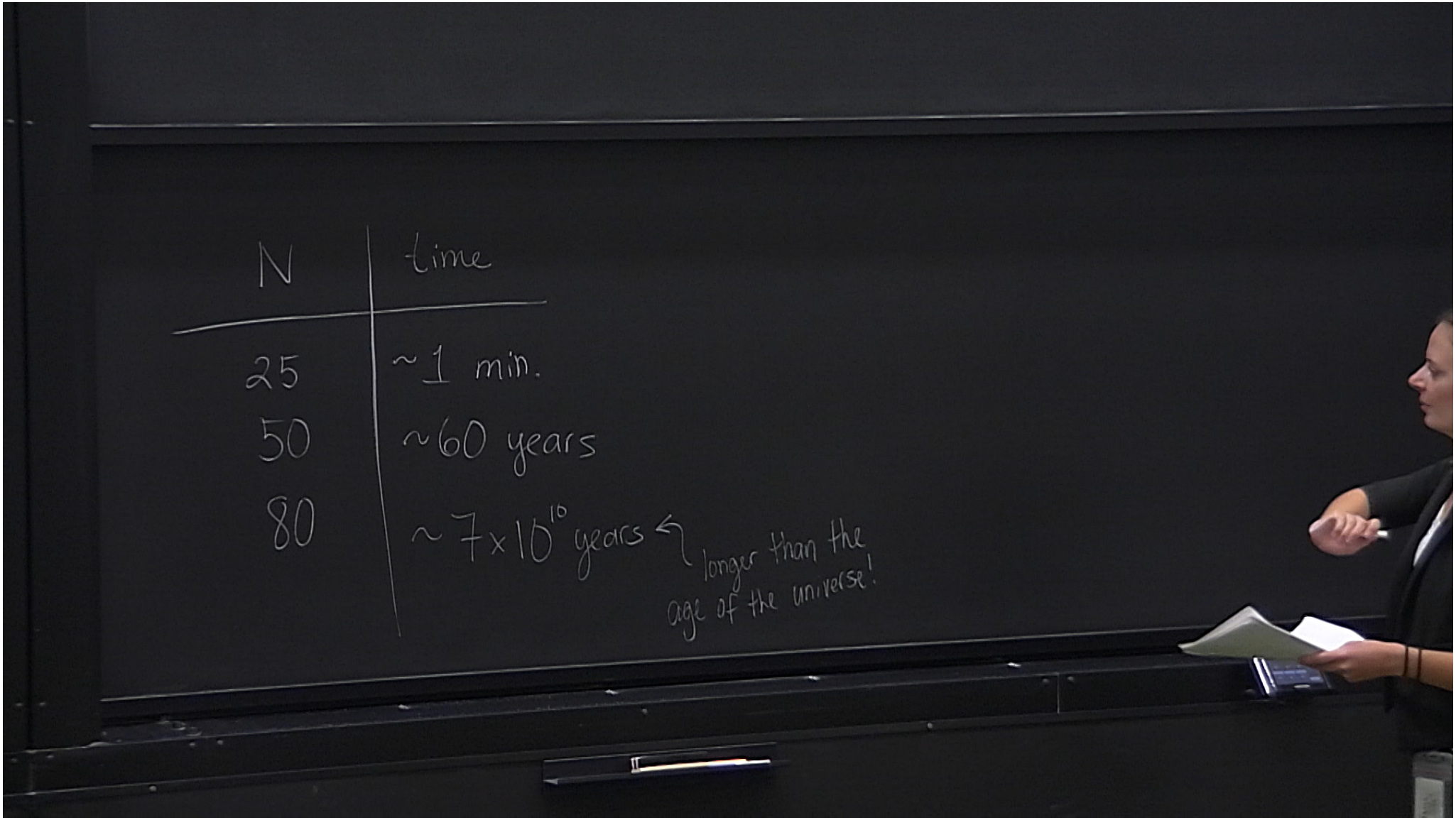

The total # of accessible states
is $2^N = 2^{(L^d)}$

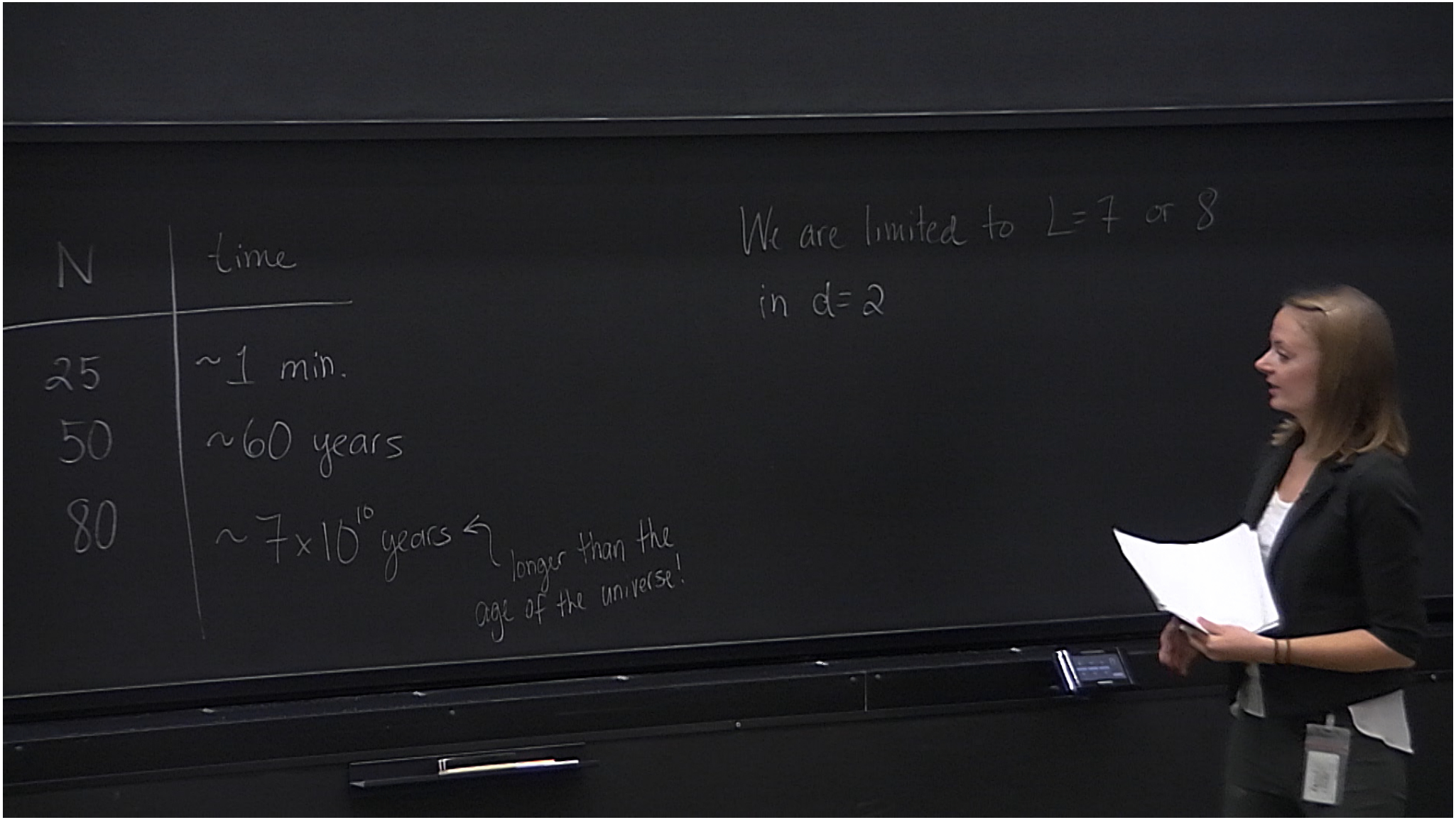
The total # of accessible states is $2^N = 2^{(L^d)}$

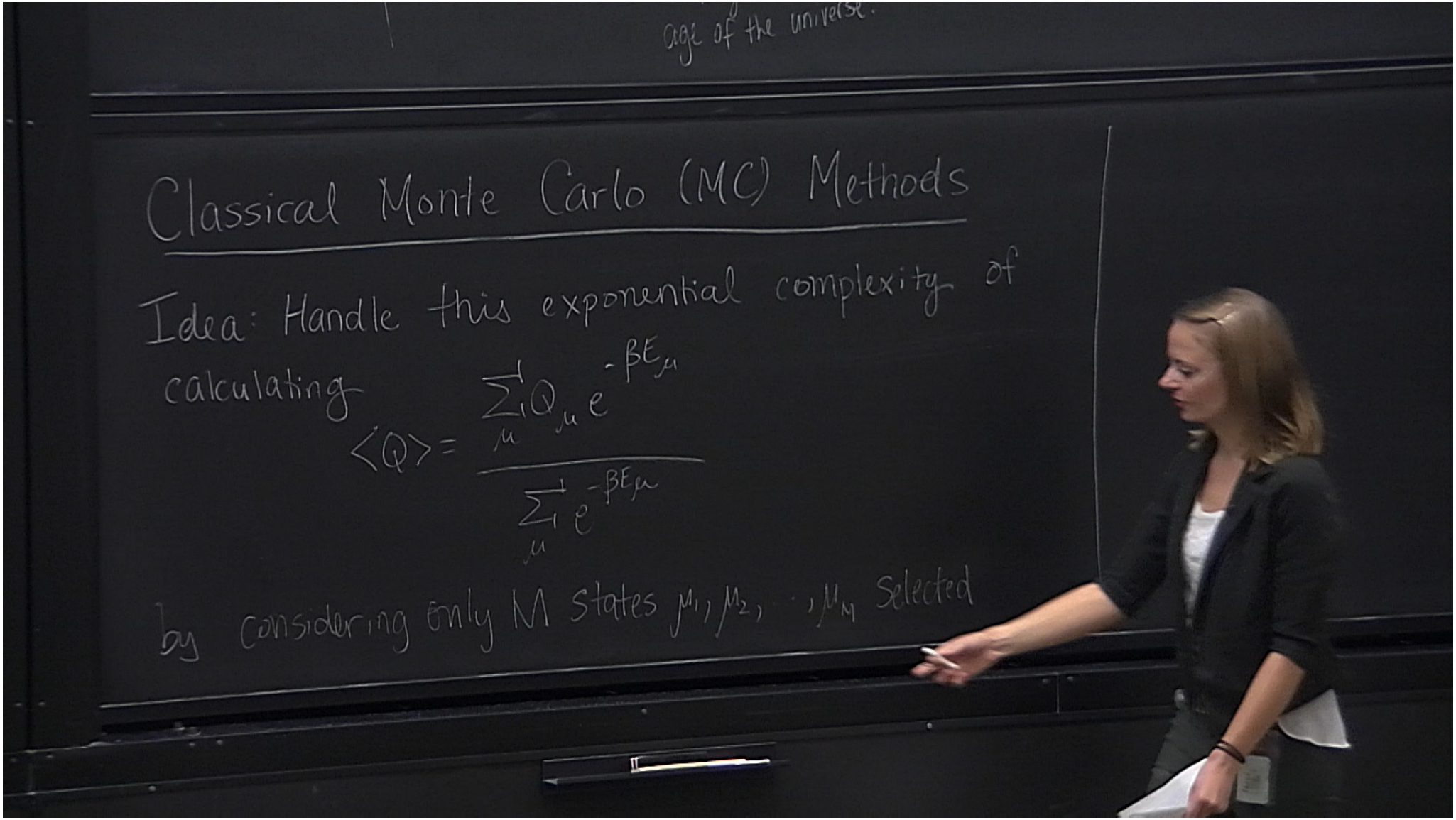
For $N=25$, evaluating $\sum_{\mu}^{!}$ requires at least 1 minute

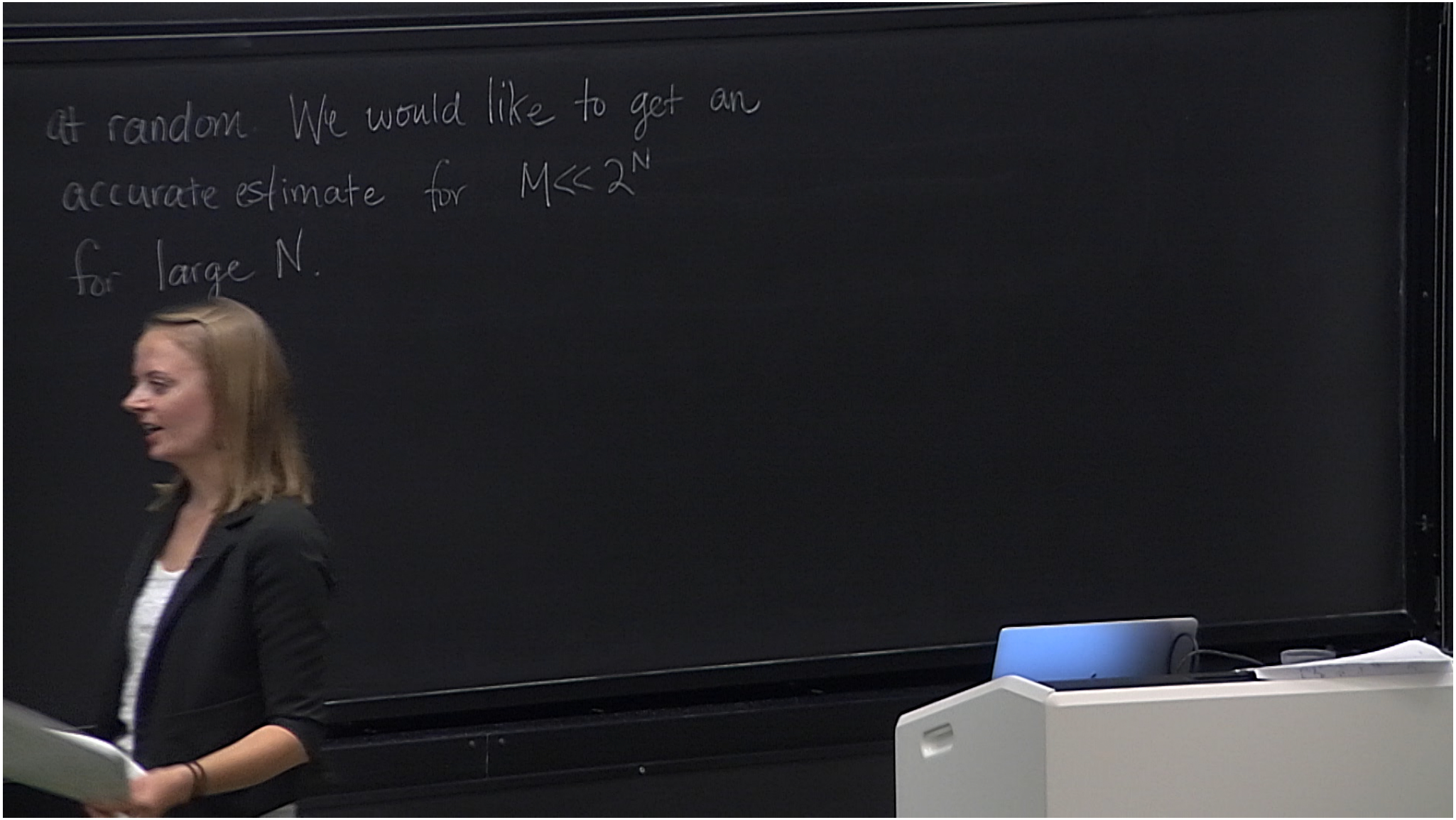

N
time
25
~1 min.
50
~60 years

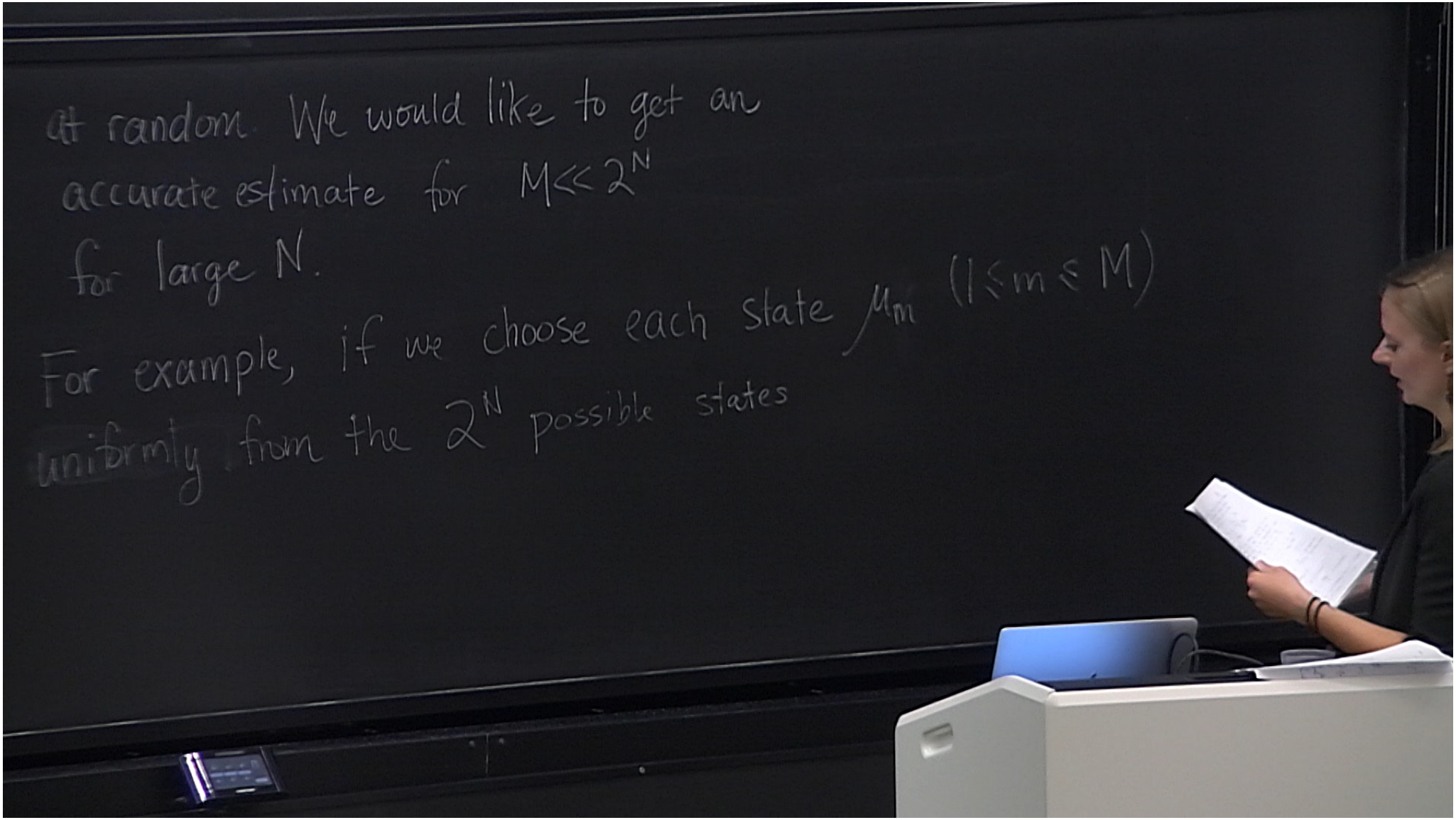

N
time
25
~1 min.
50
~60 years
80
~7×10^10 years

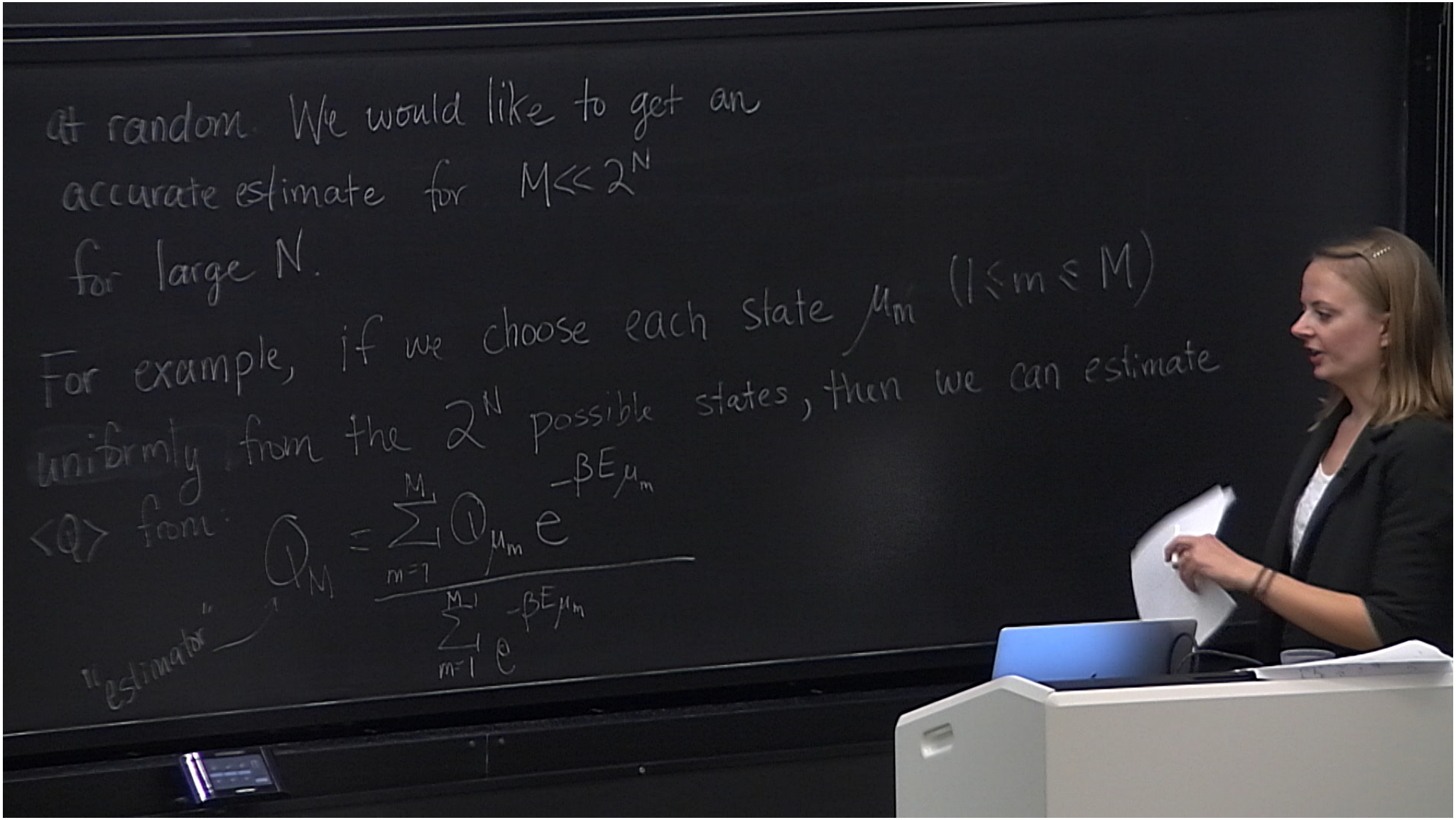

N
time
25
~1 min.
50
~60 years
80
~ 7 × 10^10 years
longer than the age of the universe!


N
time
25
~1 min.
50
~60 years
80
~ 7×10^10 years
longer than the age of the universe!
We are limited to L=7 or 8
in d=2


age of the universe.
Classical Monte Carlo (MC) Methods
Idea: Handle this exponential complexity of calculating
$\langle Q \rangle = \frac{\sum_{\mu}^{'} Q_\mu e^{-\beta E_\mu}}{\sum_{\mu}^{'} e^{-\beta E_\mu}}$
by considering only M states $\mu_1, \mu_2, \ldots, \mu_M$ selected


at random. We would like to get an
accurate estimate for $M \ll 2^N$
for large N.


at random. We would like to get an
accurate estimate for $M \ll 2^N$
for large N.
For example, if we choose each state μ_m $(1 \leq m \leq M)$
uniformly from the 2^N possible states


at random. We would like to get an
accurate estimate for $M \ll 2^N$
for large N.
For example, if we choose each state μ_m $(1 \le m \le M)$
uniformly from the 2^N possible states, then we can estimate
$\langle Q \rangle$ from:
$Q_M = \frac{\sum_{m=1}^{M} Q_{\mu_m} e^{-\beta E_{\mu_m}}}{\sum_{m=1}^{M} e^{-\beta E_{\mu_m}}}$
"estimator"

$$\langle Q \rangle = \frac{1}{Z} \mathrm{Tr}\left(Q e^{-\beta H}\right)$$

$$P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$$

Let's consider E_M for the Ising model in $d=1$.



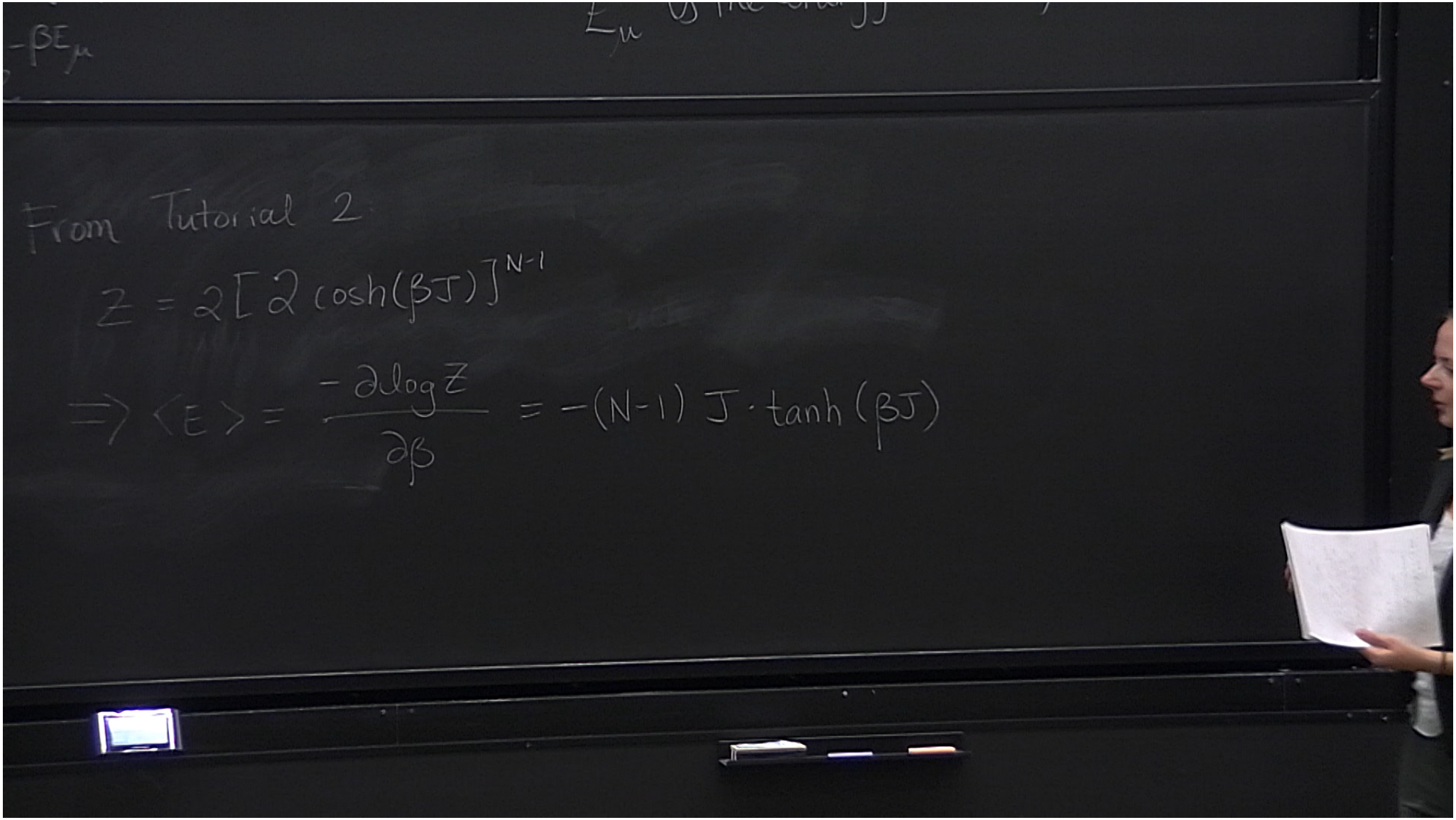
$$H = -J \sum_{i=1}^{N-1} S_i S_{i+1}$$

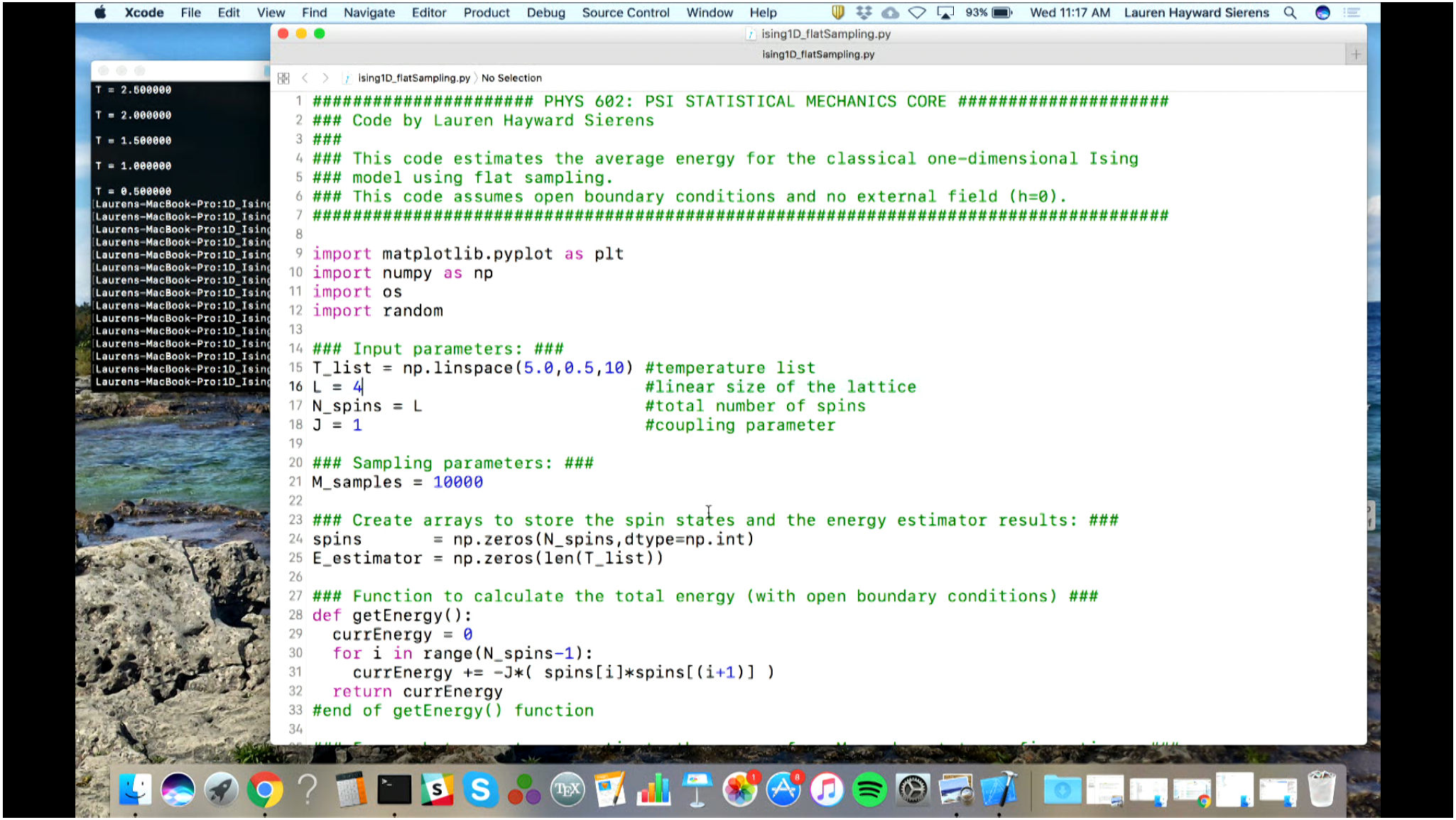
($h=0$, open boundary conditions)

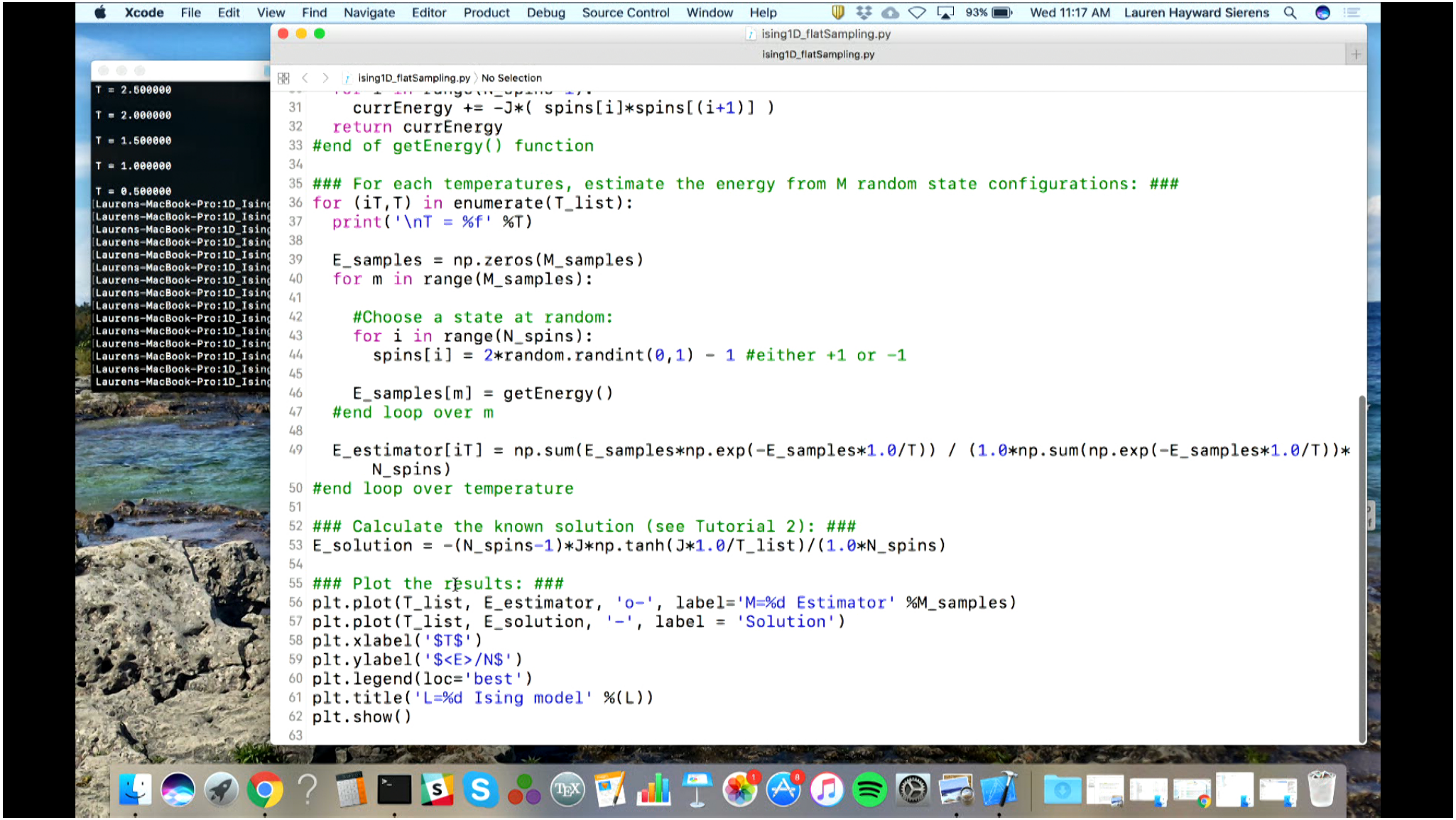
From Tutoria

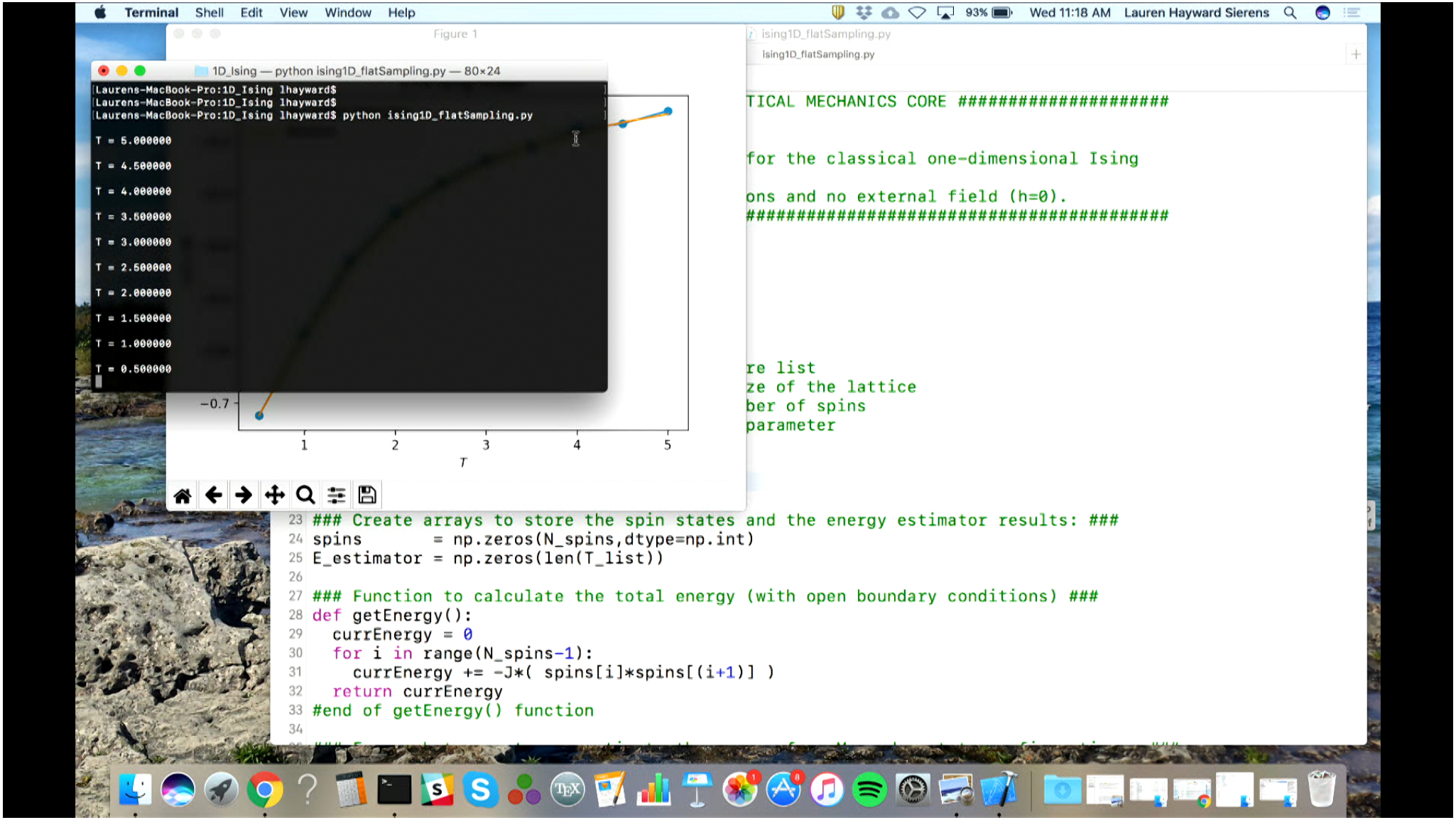
$$Z = 2[$$

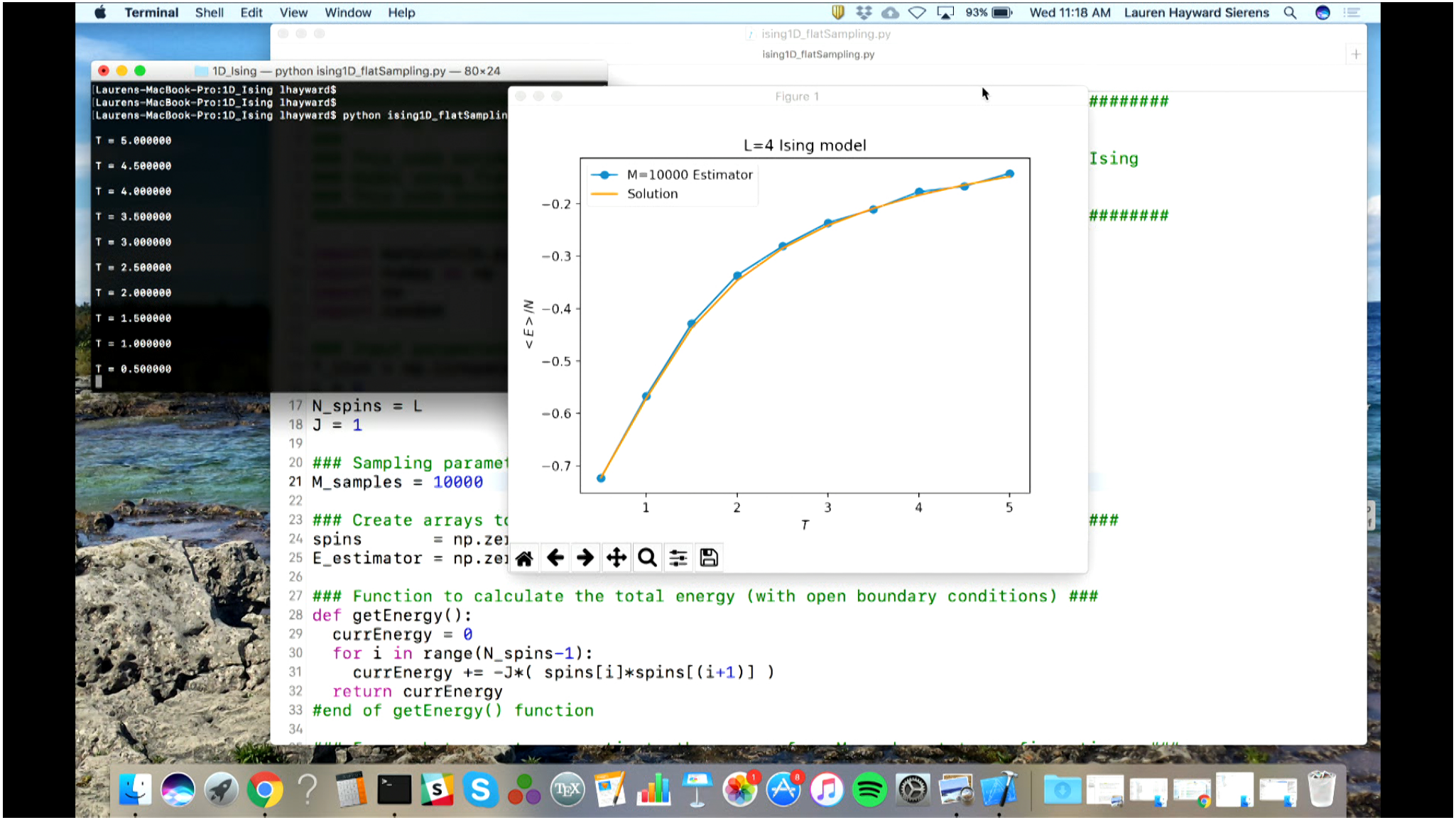
$$\Rightarrow \langle E \rangle =$$

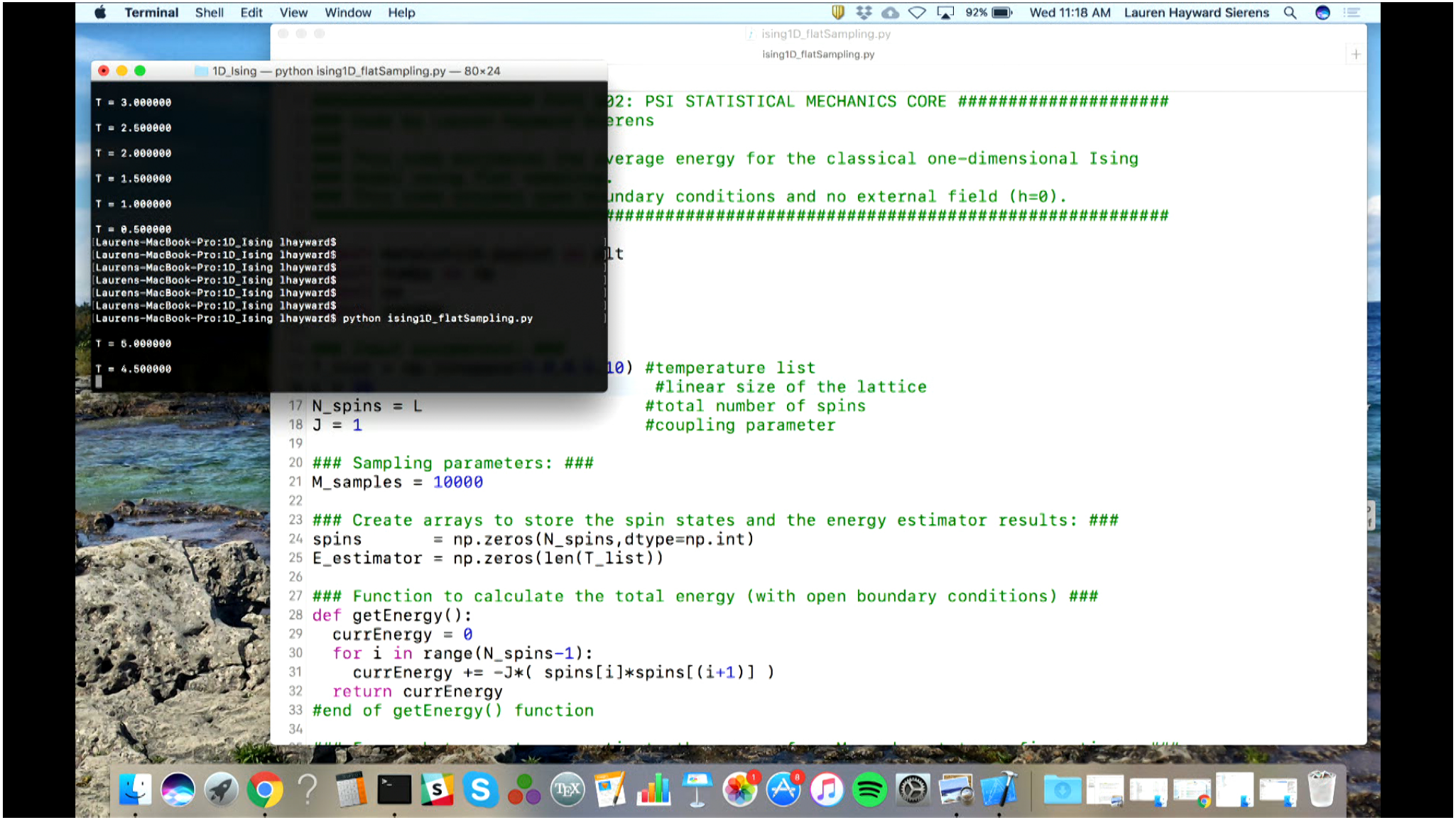

$-\beta E_\mu$
E_μ
From Tutorial 2:
$Z = 2[2\cosh(\beta J)]^{N-1}$
$\Rightarrow \langle E \rangle = \frac{-\partial \log Z}{\partial \beta} = -(N-1)\, J \cdot \tanh(\beta J)$

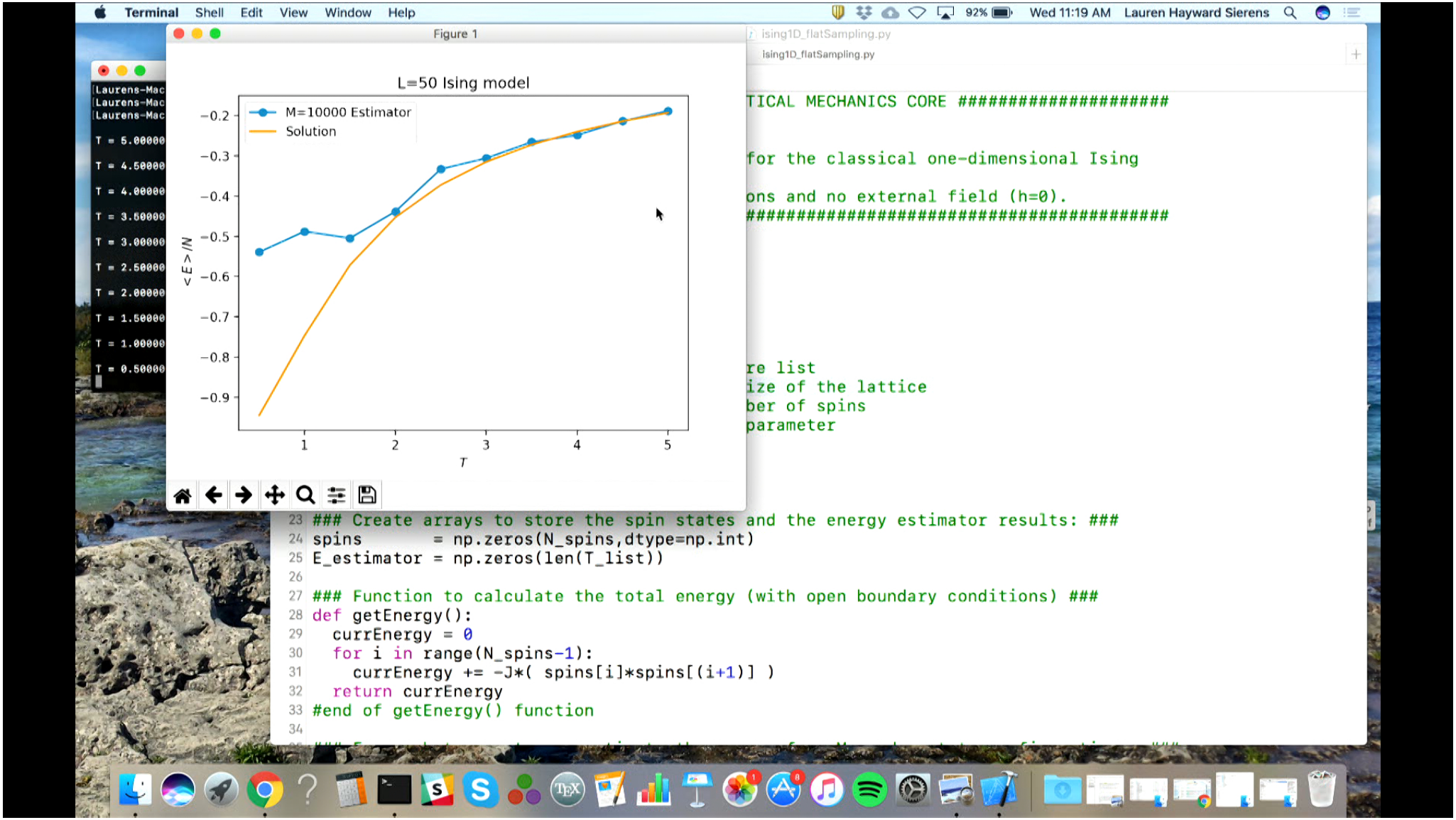

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
93% Wed 11:17 AM Lauren Hayward Sierens
ising1D_flatSampling.py
ising1D_flatSampling.py
ising1D_flatSampling.py No Selection
T = 2.500000
T = 2.000000
T = 1.500000
T = 1.000000
T = 0.500000
[Laurens-MacBook-Pro:1D_Ising
1 ###################### PHYS 602: PSI STATISTICAL MECHANICS CORE ######################
2 ### Code by Lauren Hayward Sierens
3 ###
4 ### This code estimates the average energy for the classical one-dimensional Ising
5 ### model using flat sampling.
6 ### This code assumes open boundary conditions and no external field (h=0).
7 ##
8
9 import matplotlib.pyplot as plt
10 import numpy as np
11 import os
12 import random
13
14 ### Input parameters: ###
15 T_list = np.linspace(5.0,0.5,10) #temperature list
16 L = 4 #linear size of the lattice
17 N_spins = L #total number of spins
18 J = 1 #coupling parameter
19
20 ### Sampling parameters: ###
21 M_samples = 10000
22
23 ### Create arrays to store the spin states and the energy estimator results: ###
24 spins = np.zeros(N_spins,dtype=np.int)
25 E_estimator = np.zeros(len(T_list))
26
27 ### Function to calculate the total energy (with open boundary conditions) ###
28 def getEnergy():
29 currEnergy = 0
30 for i in range(N_spins-1):
31 currEnergy += -J*(spins[i]*spins[(i+1)])
32 return currEnergy
33 #end of getEnergy() function
34

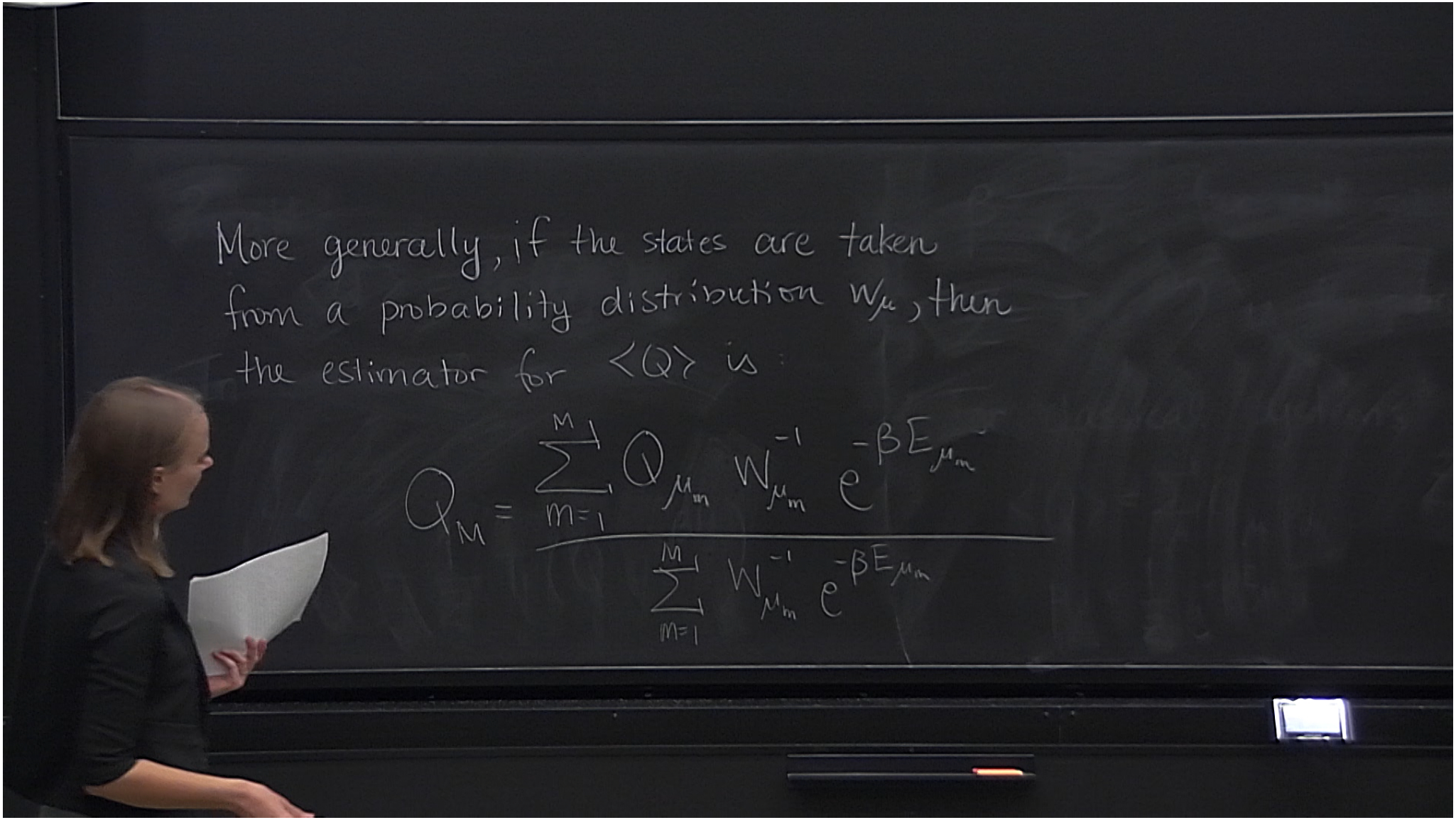

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
93% Wed 11:17 AM Lauren Hayward Sierens
ising1D_flatSampling.py
ising1D_flatSampling.py
ising1D_flatSampling.py No Selection
T = 2.500000
T = 2.000000
T = 1.500000
T = 1.000000
T = 0.500000
Laurens-MacBook-Pro:1D_Ising
31 currEnergy += -J*(spins[i]*spins[(i+1)])
32 return currEnergy
33 #end of getEnergy() function
34
35 ### For each temperatures, estimate the energy from M random state configurations: ###
36 for (iT,T) in enumerate(T_list):
37 print('\nT = %f' %T)
38
39 E_samples = np.zeros(M_samples)
40 for m in range(M_samples):
41
42 #Choose a state at random:
43 for i in range(N_spins):
44 spins[i] = 2*random.randint(0,1) - 1 #either +1 or -1
45
46 E_samples[m] = getEnergy()
47 #end loop over m
48
49 E_estimator[iT] = np.sum(E_samples*np.exp(-E_samples*1.0/T)) / (1.0*np.sum(np.exp(-E_samples*1.0/T))*
N_spins)
50 #end loop over temperature
51
52 ### Calculate the known solution (see Tutorial 2): ###
53 E_solution = -(N_spins-1)*J*np.tanh(J*1.0/T_list)/(1.0*N_spins)
54
55 ### Plot the results: ###
56 plt.plot(T_list, E_estimator, 'o-', label='M=%d Estimator' %M_samples)
57 plt.plot(T_list, E_solution, '-', label = 'Solution')
58 plt.xlabel('T')
59 plt.ylabel('$<E>/N$')
60 plt.legend(loc='best')
61 plt.title('L=%d Ising model' %(L))
62 plt.show()
63

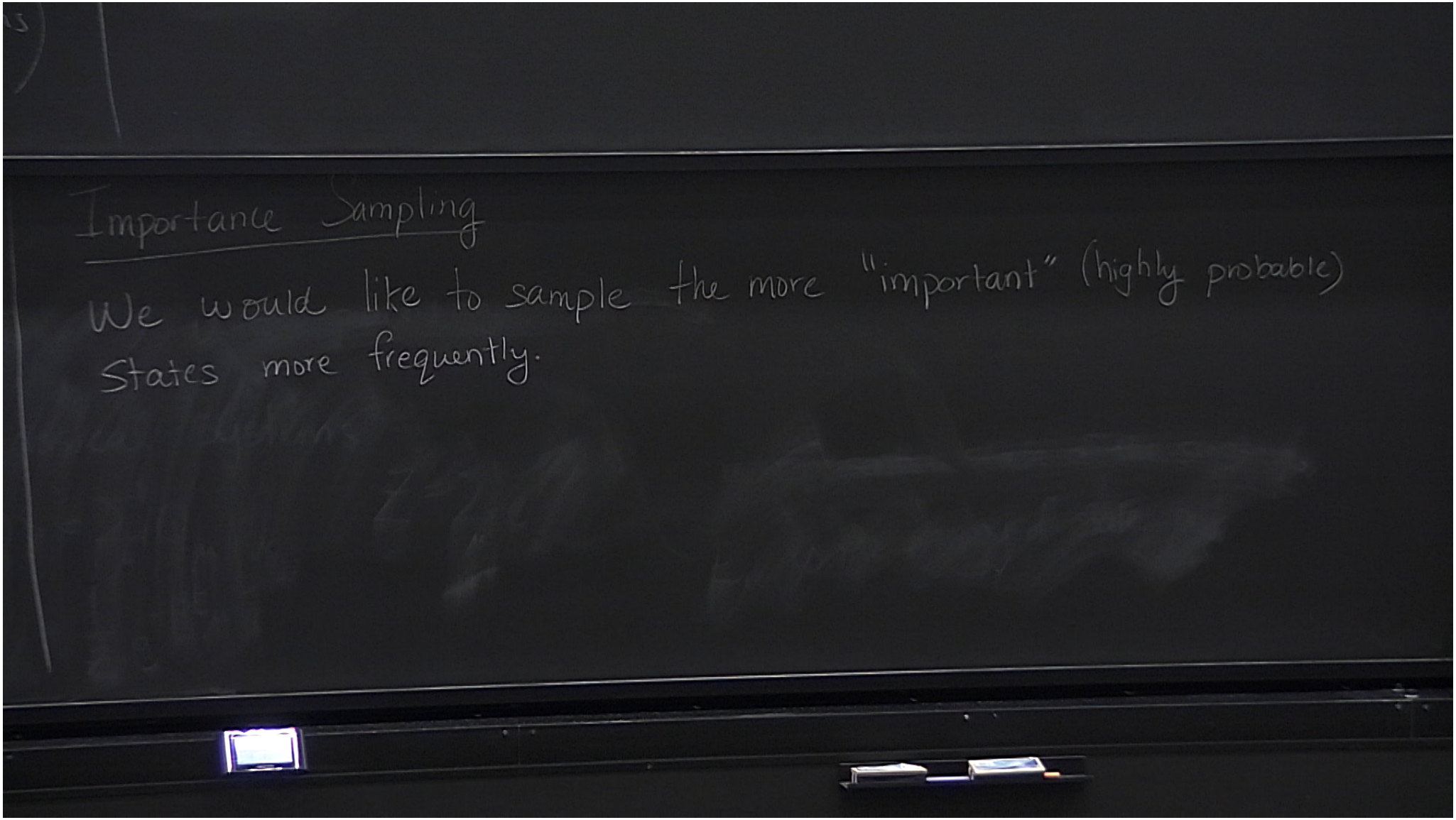

Terminal Shell Edit View Window Help
93% Wed 11:18 AM Lauren Hayward Sierens
Figure 1
1D_Ising — python ising1D_flatSampling.py — 80×24
[Laurens-MacBook-Pro:1D_Ising lhayward$
[Laurens-MacBook-Pro:1D_Ising lhayward$
[Laurens-MacBook-Pro:1D_Ising lhayward$ python ising1D_flatSampling.py
T = 5.000000
T = 4.500000
T = 4.000000
T = 3.500000
T = 3.000000
T = 2.500000
T = 2.000000
T = 1.500000
T = 1.000000
T = 0.500000
−0.7
1
2
3
4
5
T
ising1D_flatSampling.py
ising1D_flatSampling.py
TICAL MECHANICS CORE ####################
for the classical one-dimensional Ising
ons and no external field (h=0).
##
re list
ze of the lattice
ber of spins
parameter
Create arrays to store the spin states and the energy estimator results:
spins = np.zeros(N_spins,dtype=np.int)
E_estimator = np.zeros(len(T_list))
Function to calculate the total energy (with open boundary conditions)
def getEnergy():
currEnergy = 0
for i in range(N_spins-1):
currEnergy += -J*(spins[i]*spins[(i+1)])
return currEnergy
#end of getEnergy() function

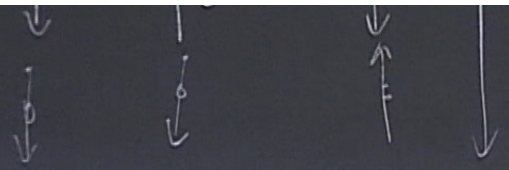

Terminal Shell Edit View Window Help
93% Wed 11:18 AM Lauren Hayward Sierens
1D_Ising — python ising1D_flatSampling.py — 80×24
Laurens-MacBook-Pro:1D_Ising lhayward$
Laurens-MacBook-Pro:1D_Ising lhayward$
Laurens-MacBook-Pro:1D_Ising lhayward$ python ising1D_flatSamplin
T = 5.000000
T = 4.500000
T = 4.000000
T = 3.500000
T = 3.000000
T = 2.500000
T = 2.000000
T = 1.500000
T = 1.000000
T = 0.500000
Figure 1
L=4 Ising model
M=10000 Estimator
Solution
<E>/N
T
N_spins = L
J = 1
Sampling paramet
M_samples = 10000
Create arrays to
spins = np.zer
E_estimator = np.zer
Function to calculate the total energy (with open boundary conditions)
def getEnergy():
currEnergy = 0
for i in range(N_spins-1):
currEnergy += -J*(spins[i]*spins[(i+1)])
return currEnergy
#end of getEnergy() function
Ising
########
########
###


Terminal Shell Edit View Window Help
92% Wed 11:18 AM Lauren Hayward Sierens
1D_Ising — python ising1D_flatSampling.py — 80×24
T = 3.000000
T = 2.500000
T = 2.000000
T = 1.500000
T = 1.000000
T = 0.500000
Laurens-MacBook-Pro:1D_Ising lhayward$
Laurens-MacBook-Pro:1D_Ising lhayward$ python ising1D_flatSampling.py
T = 5.000000
T = 4.500000
ising1D_flatSampling.py
02: PSI STATISTICAL MECHANICS CORE ####################
erens
verage energy for the classical one-dimensional Ising
ndary conditions and no external field (h=0).
##
lt
10) #temperature list
#linear size of the lattice
N_spins = L #total number of spins
J = 1 #coupling parameter
Sampling parameters:
M_samples = 10000
Create arrays to store the spin states and the energy estimator results:
spins = np.zeros(N_spins,dtype=np.int)
E_estimator = np.zeros(len(T_list))
Function to calculate the total energy (with open boundary conditions)
def getEnergy():
currEnergy = 0
for i in range(N_spins-1):
currEnergy += -J*(spins[i]*spins[(i+1)])
return currEnergy
#end of getEnergy() function


Terminal Shell Edit View Window Help
92% Wed 11:19 AM Lauren Hayward Sierens
Figure 1
L=50 Ising model
M=10000 Estimator
Solution
<E>/N
T
ising1D_flatSampling.py
TICAL MECHANICS CORE ####################
for the classical one-dimensional Ising
ons and no external field (h=0).
re list
ize of the lattice
ber of spins
parameter
Create arrays to store the spin states and the energy estimator results:
spins = np.zeros(N_spins,dtype=np.int)
E_estimator = np.zeros(len(T_list))
Function to calculate the total energy (with open boundary conditions)
def getEnergy():
currEnergy = 0
for i in range(N_spins-1):
currEnergy += -J*(spins[i]*spins[(i+1)])
return currEnergy
#end of getEnergy() function


More generally, if the states are taken from a probability distribution W_μ, then the estimator for $\langle Q \rangle$ is:
$$Q_M = \frac{\sum_{m=1}^{M} Q_{\mu_m} W_{\mu_m}^{-1} e^{-\beta E_{\mu_m}}}{\sum_{m=1}^{M} W_{\mu_m}^{-1} e^{-\beta E_{\mu_m}}}$$


Importance Sampling
We would like to sample the more "important" (highly probable)
states more frequently.



there is a binary variable $\sigma_i = \pm 1$

Importance Sampling

We would like to sample the more "important" (highly probable) states more frequently.

In particular, if $W_\mu = p_\mu = e^{-\beta E_\mu} / Z$, then

$$Q_M = \frac{1}{M} \sum_{m=1}^{M} Q_{\mu_m}$$

$$\sum_{m=1}^{M} W_{\mu_m}^{-1} e^{-\beta E_{\mu_m}}$$

But how do we choose states from this distribution when we don't know Z?

Markov Chain Monte Carlo

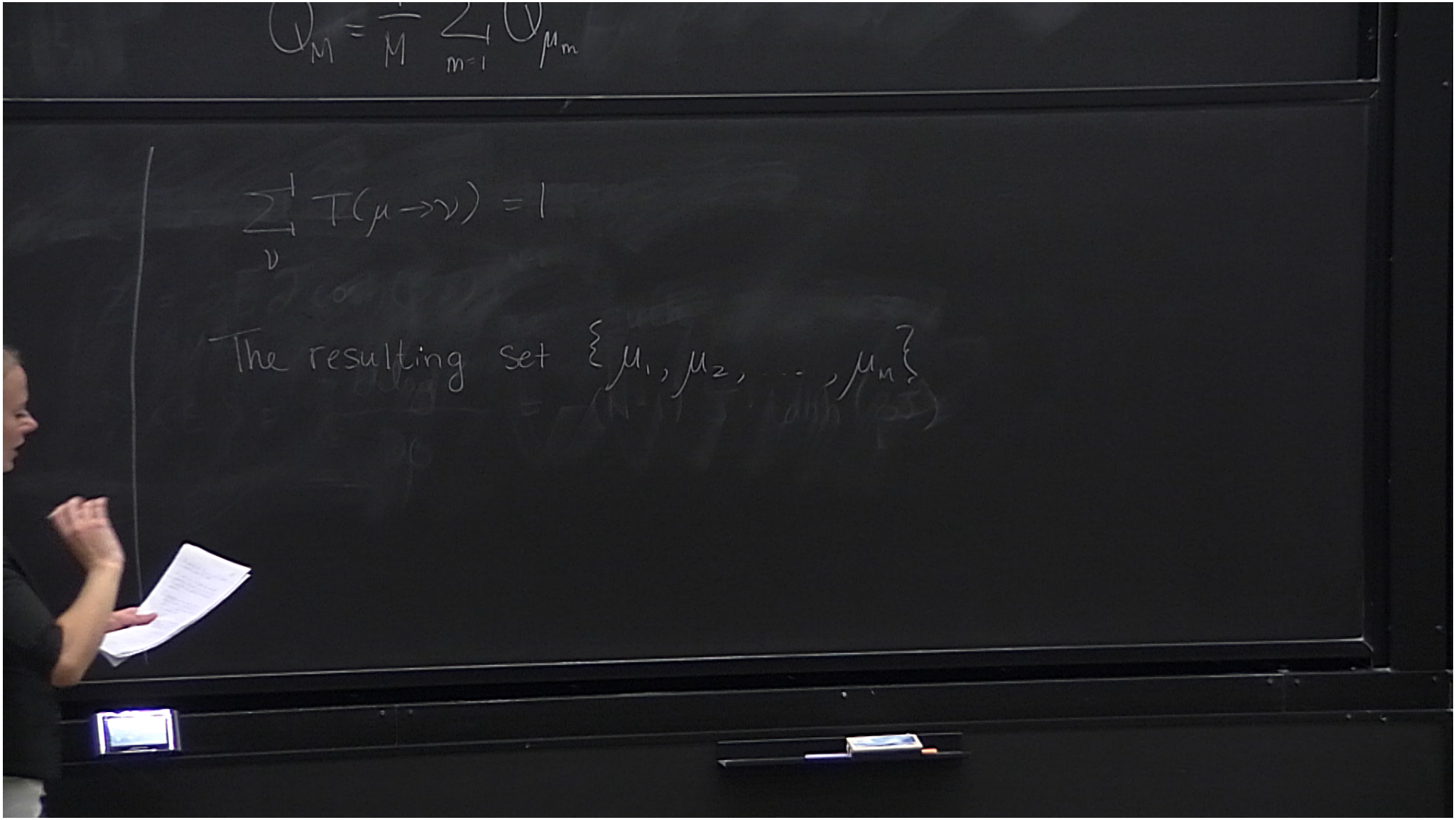
$$\sum_{m=1}^{M} W_{\mu_m}^{-1} e^{-\beta E_{\mu_m}}$$

But how do we choose states from this distribution when we don't know Z?

Markov Chain Monte Carlo

Idea: Given state μ_m, move to the next state μ_{m+1} according to transition prob $T(\mu_m \to \mu_{m+1})$

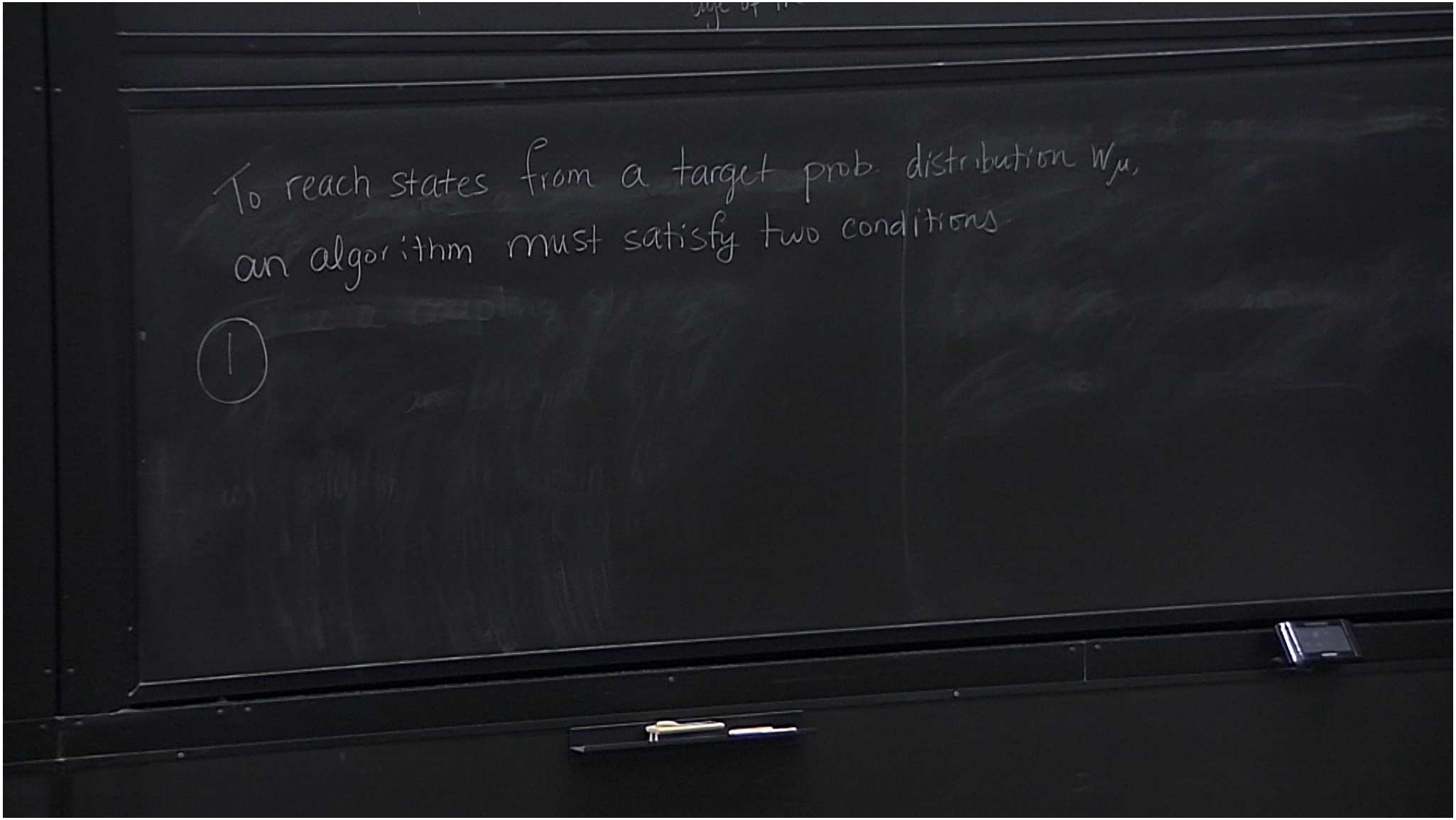
doesn't depend on $\mu_1, \mu_2, \ldots, \mu_{m-1}$

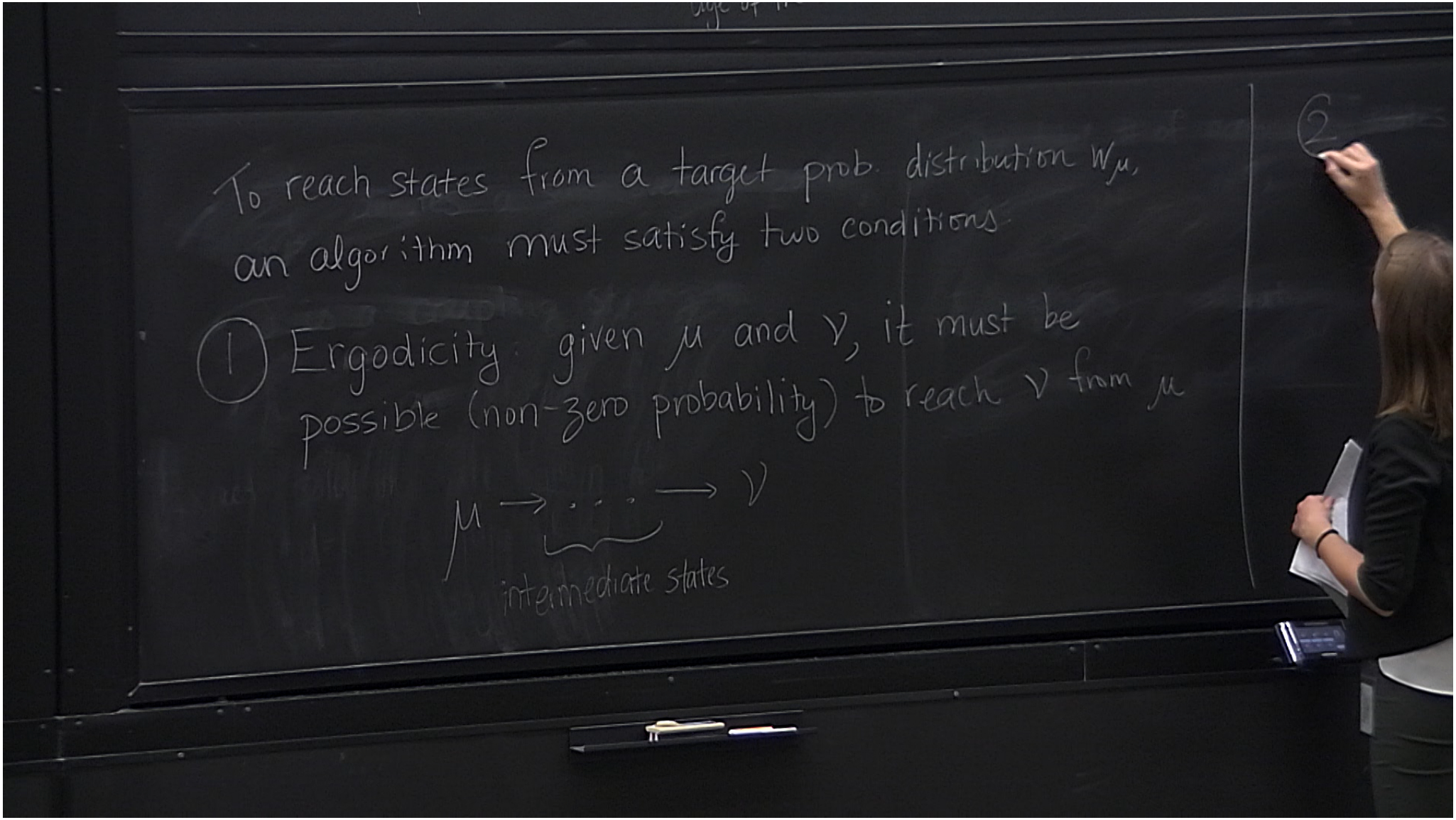

$Q_M = \frac{1}{M} \sum_{m=1} Q_{\mu_m}$
$\sum_{\nu} T(\mu \to \nu) = 1$
The resulting set $\{\mu_1, \mu_2, \ldots, \mu_n\}$

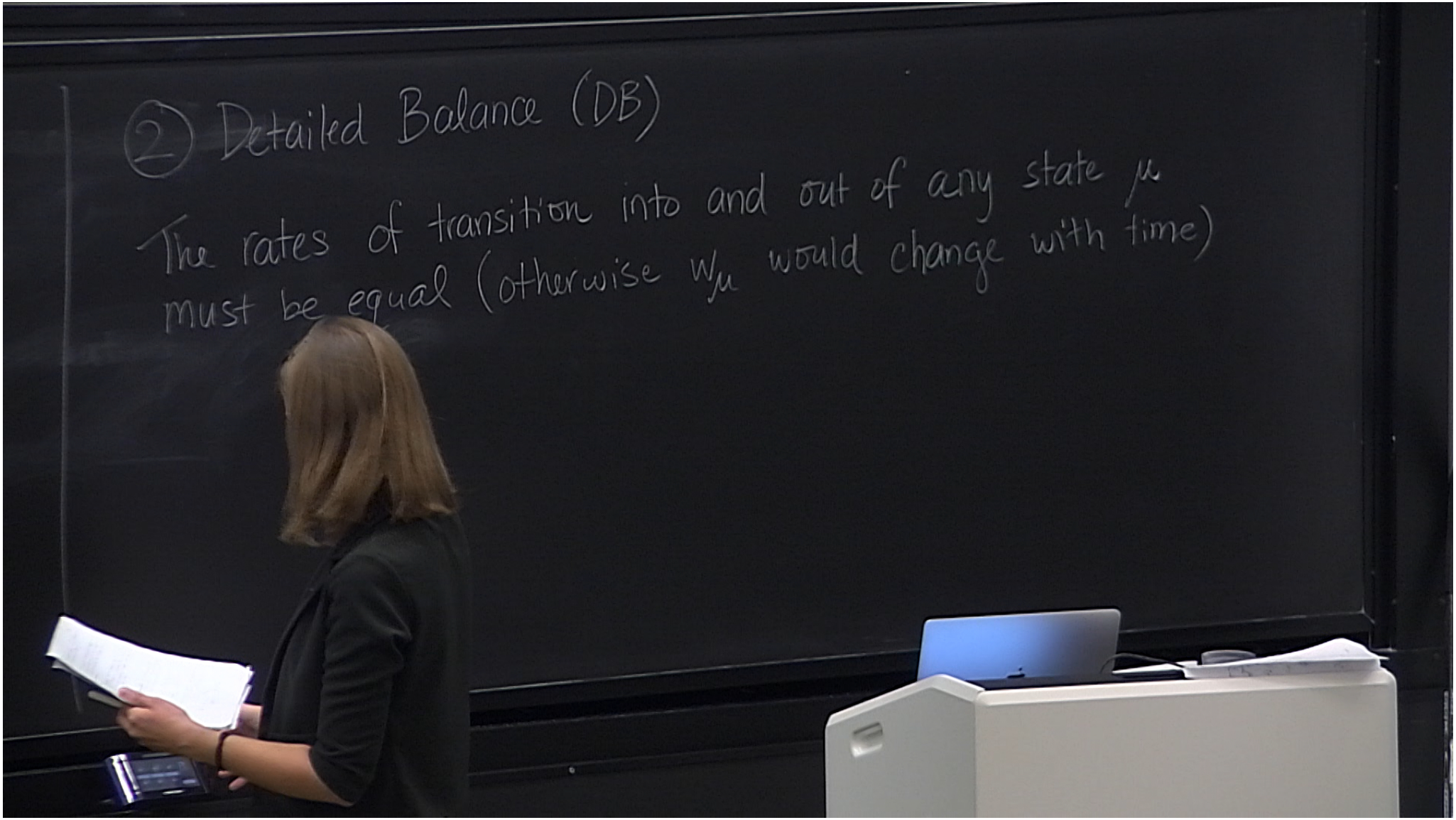
$$Q_M = \frac{1}{M} \sum_{m=1}^{M} Q_{\mu_m}$$

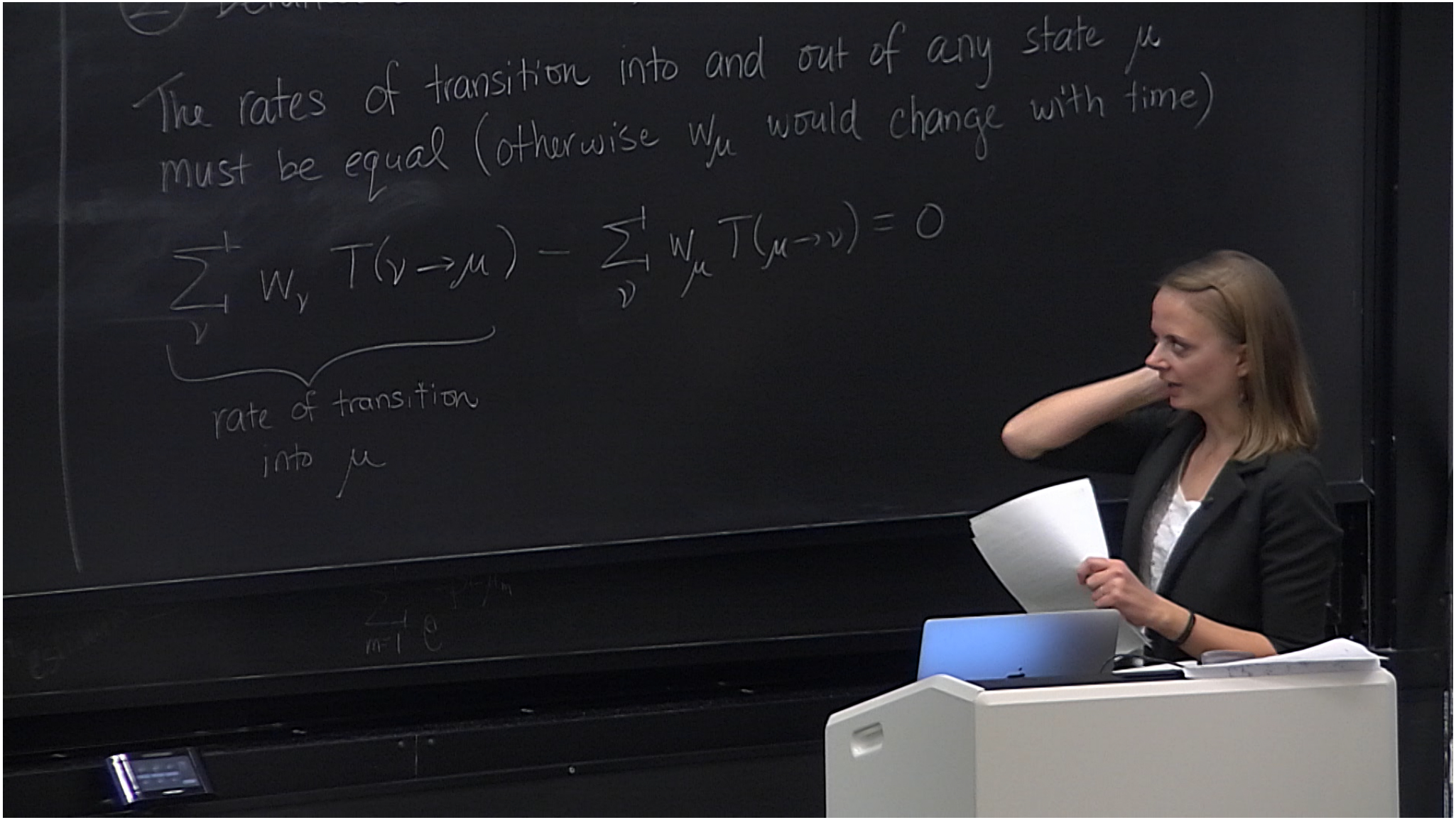
$$\sum_{\nu}^{1} T(\mu \to \nu) = 1$$

The resulting set $\{\mu_1, \mu_2, \ldots, \mu_n\}$ is called a Markov chain of states.


To reach states from a target prob. distribution W_u,
an algorithm must satisfy two conditions.
(1)


To reach states from a target prob. distribution π_μ, an algorithm must satisfy two conditions
(1) Ergodicity: given μ and ν, it must be possible (non-zero probability) to reach ν from μ
$\mu \rightarrow \cdots \rightarrow \nu$
intermediate states
(2


② Detailed Balance (DB)
The rates of transition into and out of any state μ
must be equal (otherwise W_μ would change with time)


The rates of transition into and out of any state μ must be equal (otherwise W_μ would change with time)
$\sum_\nu W_\nu T(\nu \to \mu) - \sum_\nu W_\mu T(\mu \to \nu) = 0$
rate of transition into μ

(2) Detailed Balance (DB)

The rates of transition into and out of any state μ must be equal (otherwise W_μ would change with time)

$$\sum_\nu W_\nu T(\nu \to \mu) - \sum_\nu W_\mu T(\mu \to \nu) = 0$$

rate of transition into μ

rate of transition out of μ

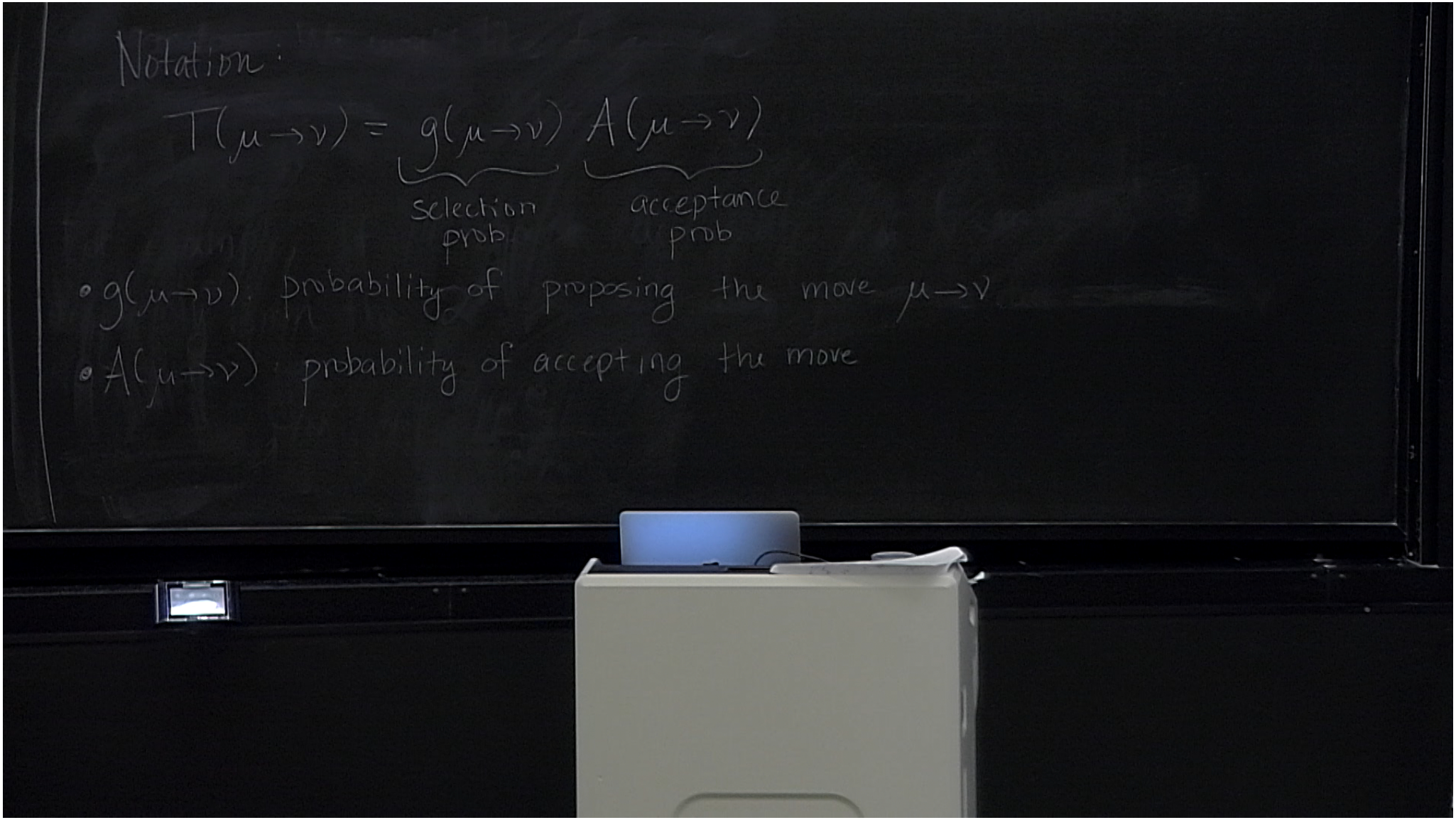
One possible way to satisfy this condition:

$$W_\mu T(\mu \to \nu) = W_\nu T(\nu \to \mu)$$ DB

One possible way to satisfy this condition:

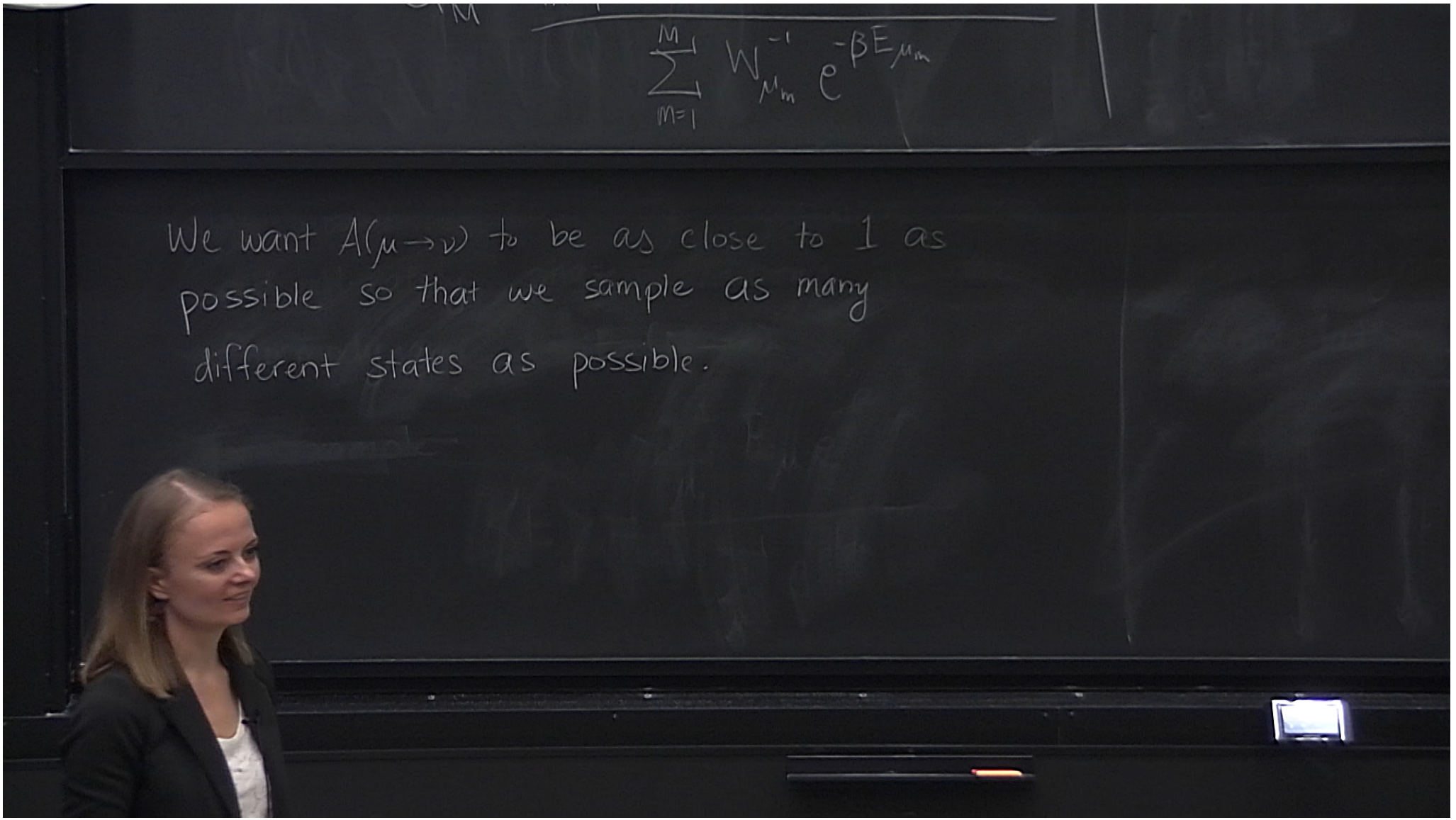
$$W_\mu T(\mu \to \nu) = W_\nu T(\nu \to \mu) \quad \text{DB}$$

So when $W_\mu = e^{-\beta E_\mu} / Z$: $\dfrac{T(\mu \to \nu)}{T(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)}$


Notation:
$T(\mu \to \nu) = g(\mu \to \nu)\, A(\mu \to \nu)$
selection prob
acceptance prob
• $g(\mu \to \nu)$ probability of proposing the move $\mu \to \nu$
• $A(\mu \to \nu)$ probability of accepting the move

$$\sum_{m=1}^{M} W_{\mu_m}^{-1} e^{-\beta E_{\mu_m}}$$

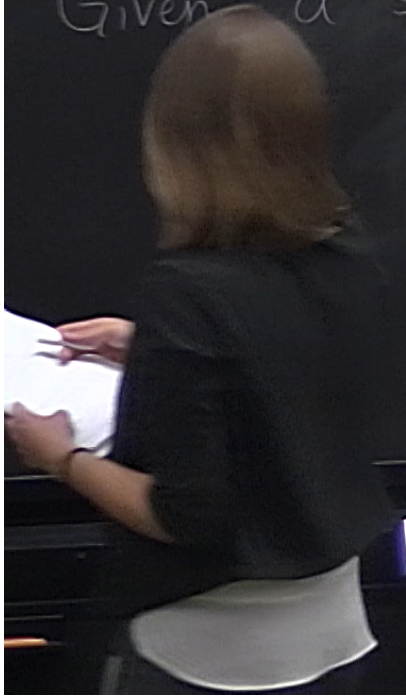
We want $A(\mu \to \nu)$ to be as close to 1 as possible so that we sample as many different states as possible.



$$Q_M = \frac{1}{M} \sum_{m=1}^{M} Q_{\mu_m}$$

Single-Spin-Flip Algorithms for the Ising Model

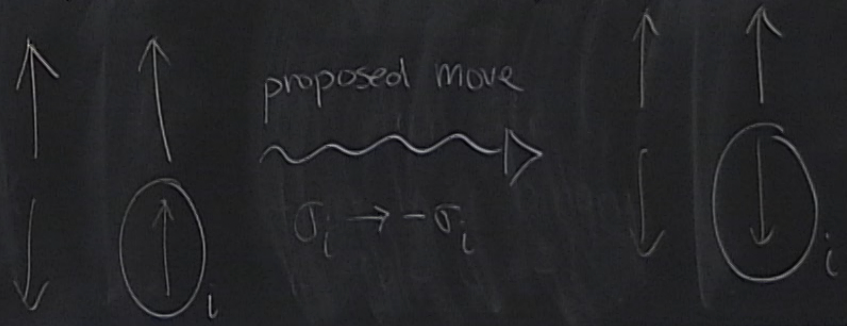
Given a state μ_m of N spins at step m of the Markov chain

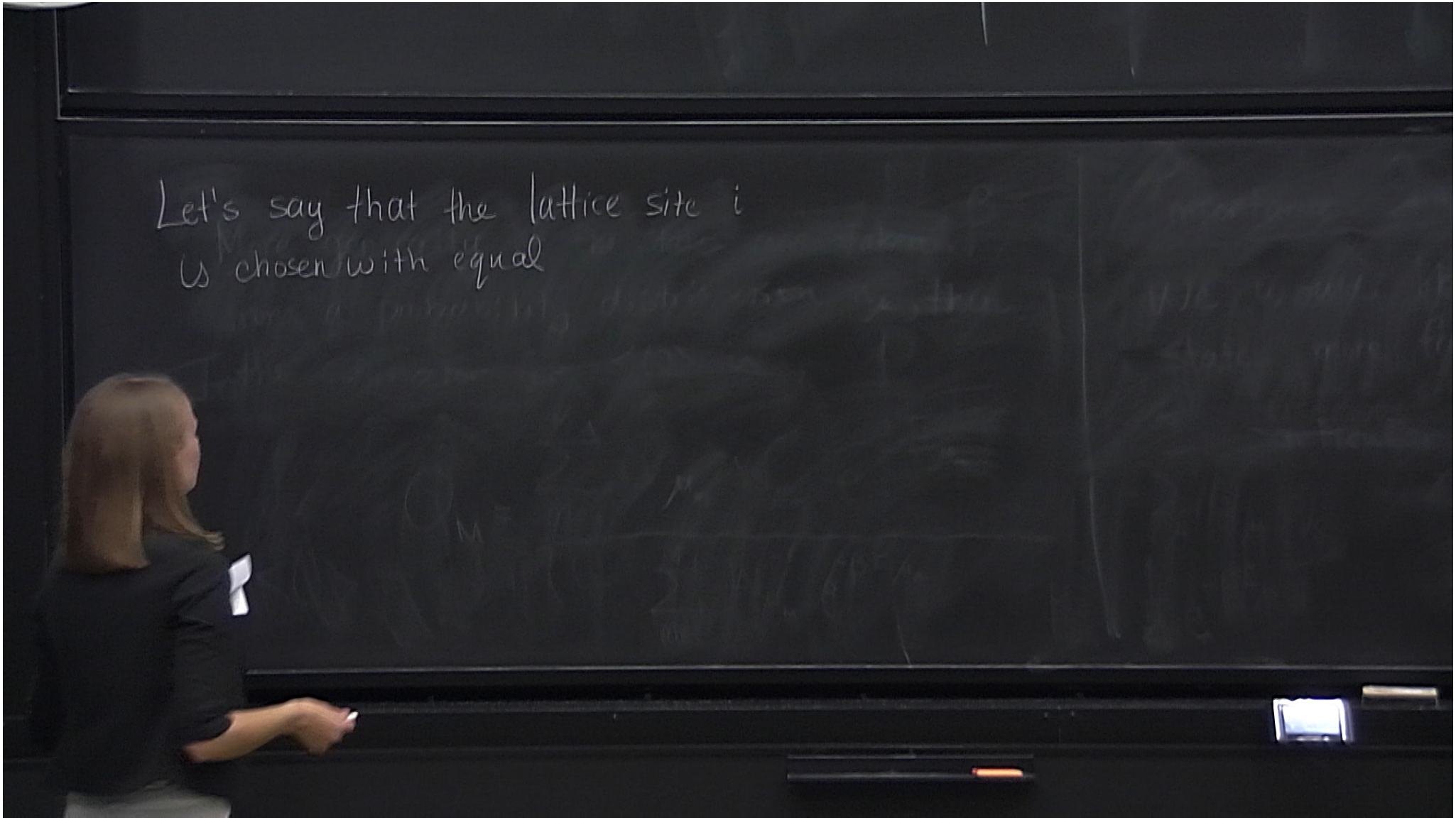


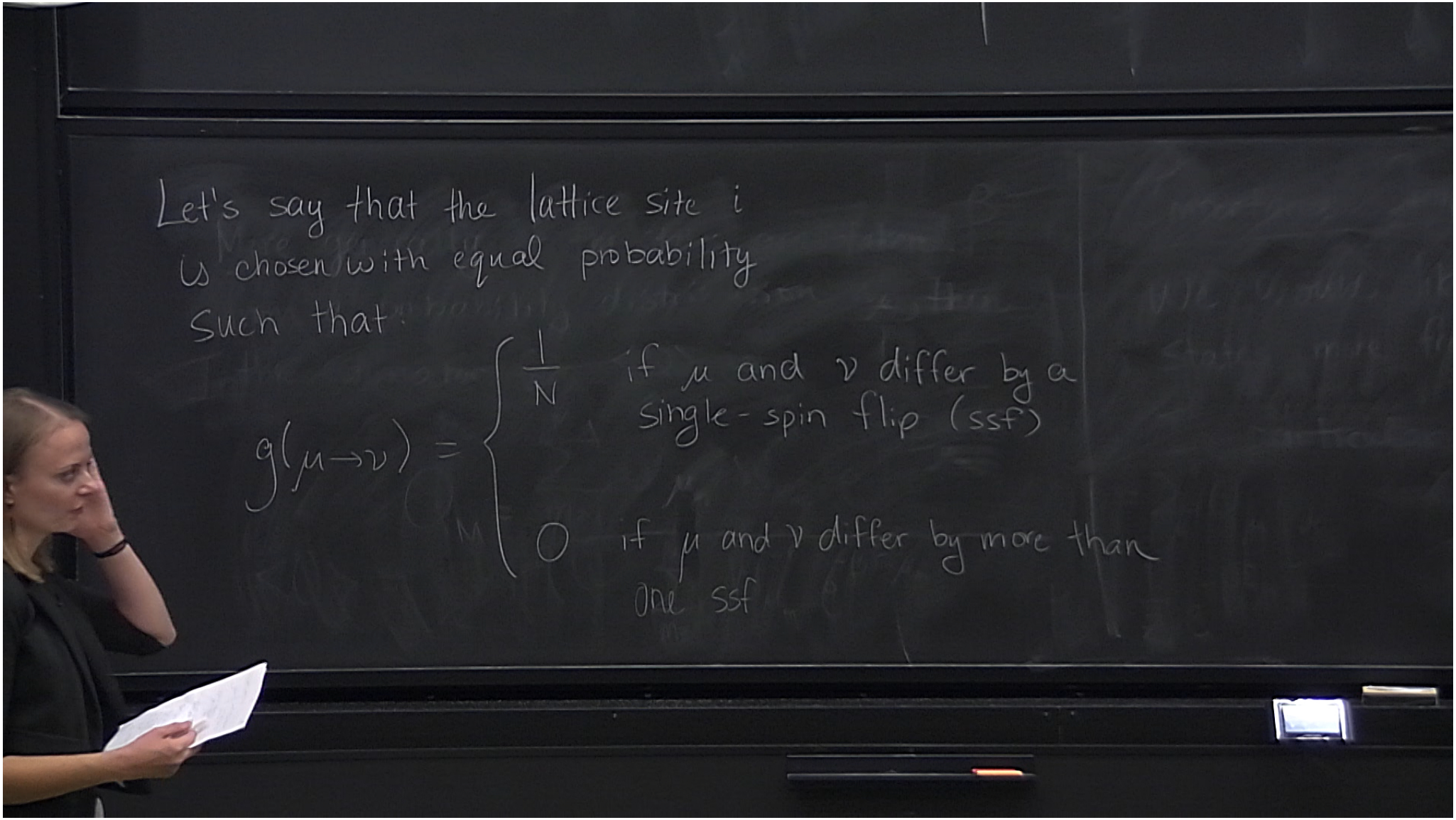
$$Q_M = \frac{1}{M} \sum_{m=1}^{M} Q_{\mu_m}$$

Single-Spin-Flip Algorithms for the Ising Model

Given a state μ_m of N spins at step m of the Markov chain, propose a new state μ_{m+1} that differs from μ_m by a single spin flip at some lattice site i




Let's say that the lattice site i
is chosen with equal


Let's say that the lattice site i is chosen with equal probability such that:
$g(\mu \to \nu) = \begin{cases} \frac{1}{N} & \text{if } \mu \text{ and } \nu \text{ differ by a single-spin flip (ssf)} \\ 0 & \text{if } \mu \text{ and } \nu \text{ differ by more than one ssf} \end{cases}$

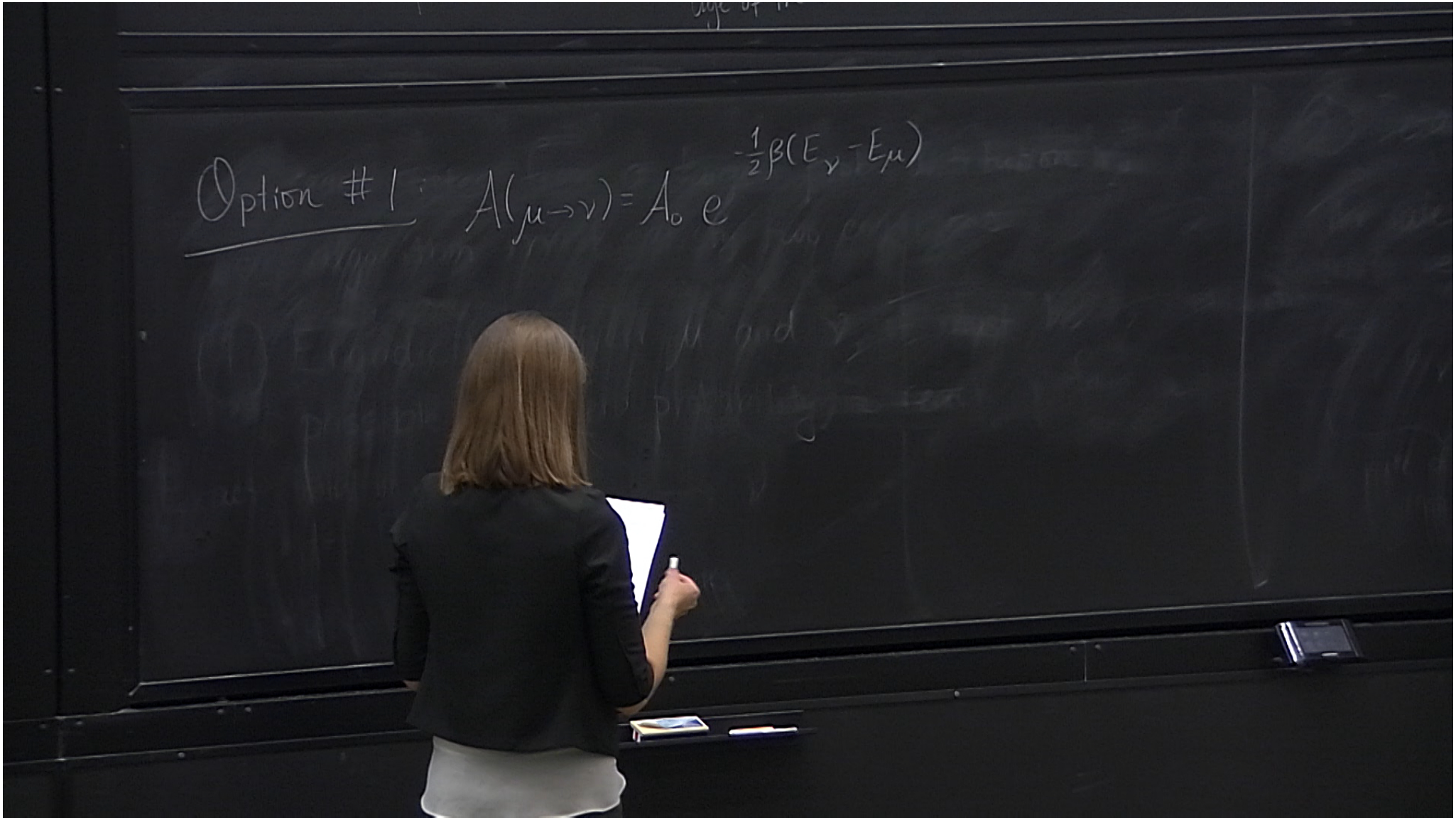
By DB, for such a ssf

$$e^{-\beta(E_\nu - E_\mu)} = \frac{A(\mu \to \nu)\,\frac{1}{N}}{A(\nu \to \mu)\,\frac{1}{N}} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)}$$

By DB, for such a ssf

$$e^{-\beta(E_\nu - E_\mu)} = \frac{A(\mu \to \nu)\frac{1}{N}}{A(\nu \to \mu)\frac{1}{N}} = \frac{A(\mu \to \nu)}{A(\nu \to \mu)}$$

Let's consider two options for choosing the acceptance probabilities


Option #1: $A(\mu \to \nu) = A_0 e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$

Option #1: $A(\mu \to \nu) = A_0 \, e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$

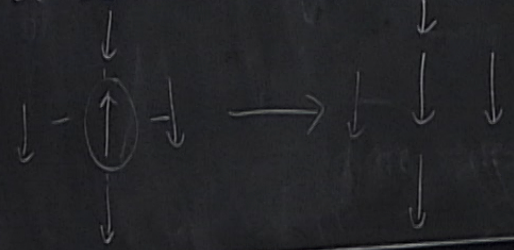
We are free to choose A_0 provided $A(\mu \to \nu) \leq 1$ for any ssf. Since the minimum energy difference

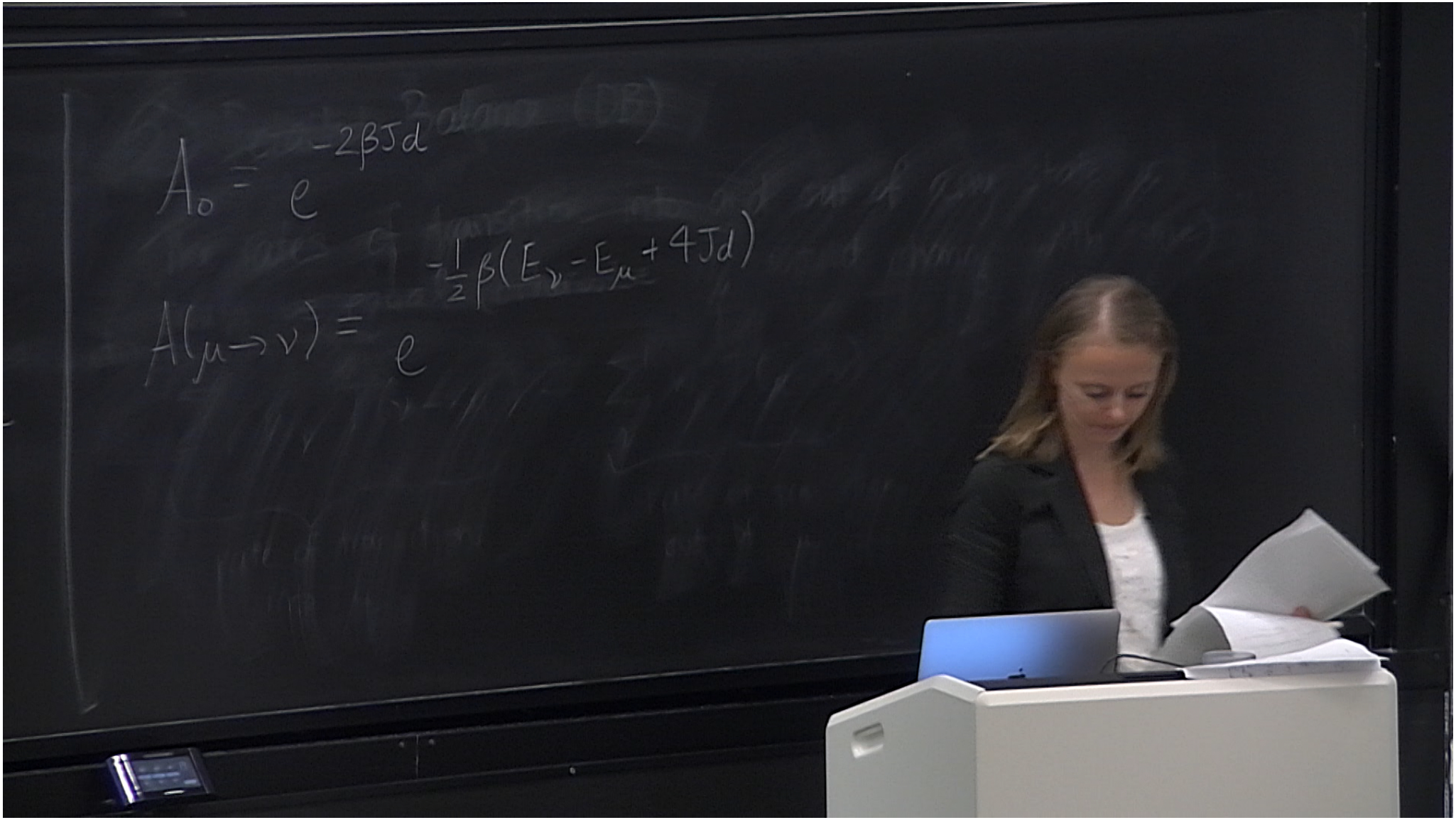
Option #1: $A(\mu \to \nu) = A_0 \, e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$

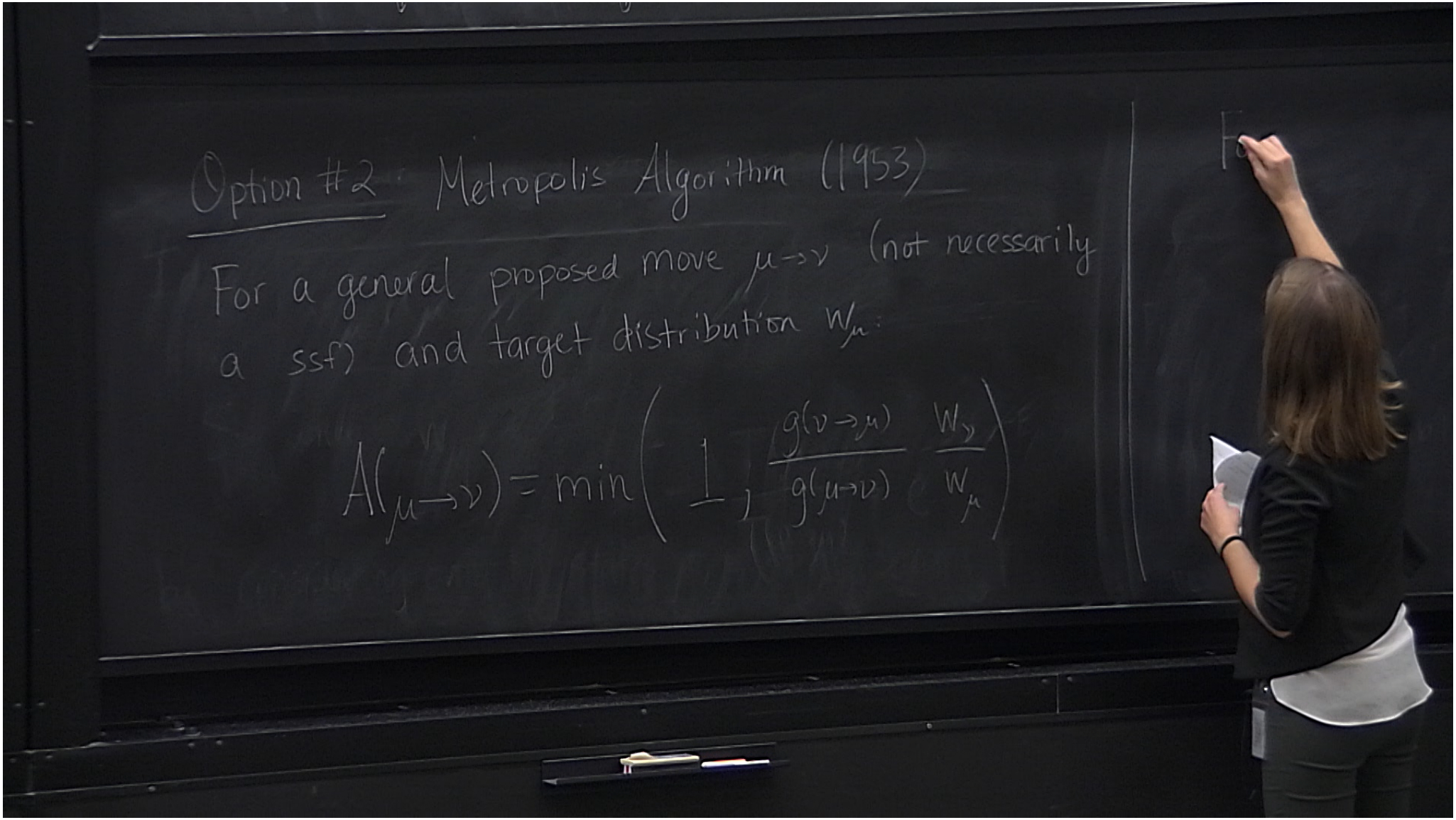
We are free to choose A_0 provided $A(\mu \to \nu) \leq 1$ for any ssf. Since the minimum energy difference for a ssf on a d-dim cubic lattice is

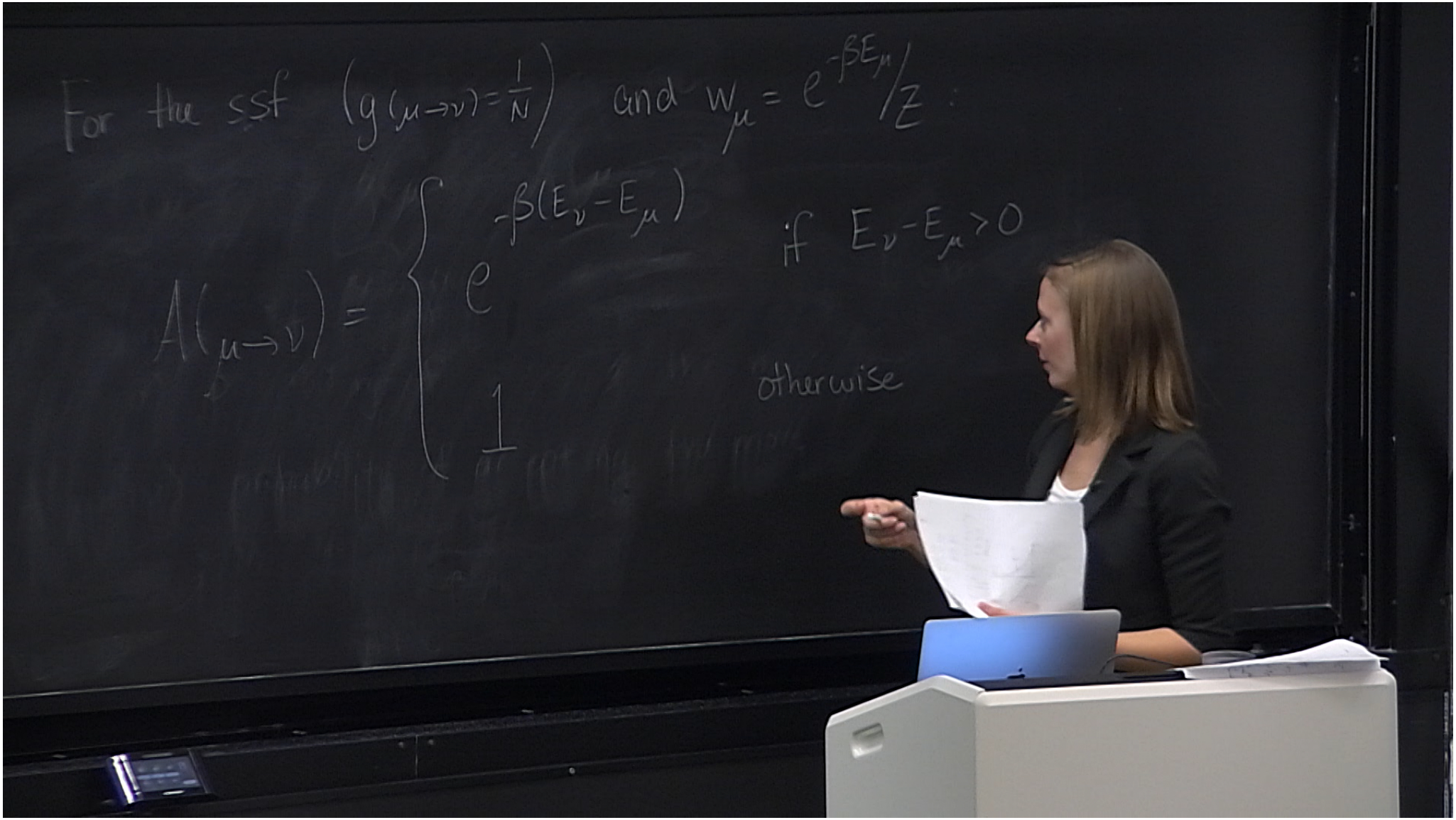
Option #1: $A(\mu \to \nu) = A_0 \, e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}$

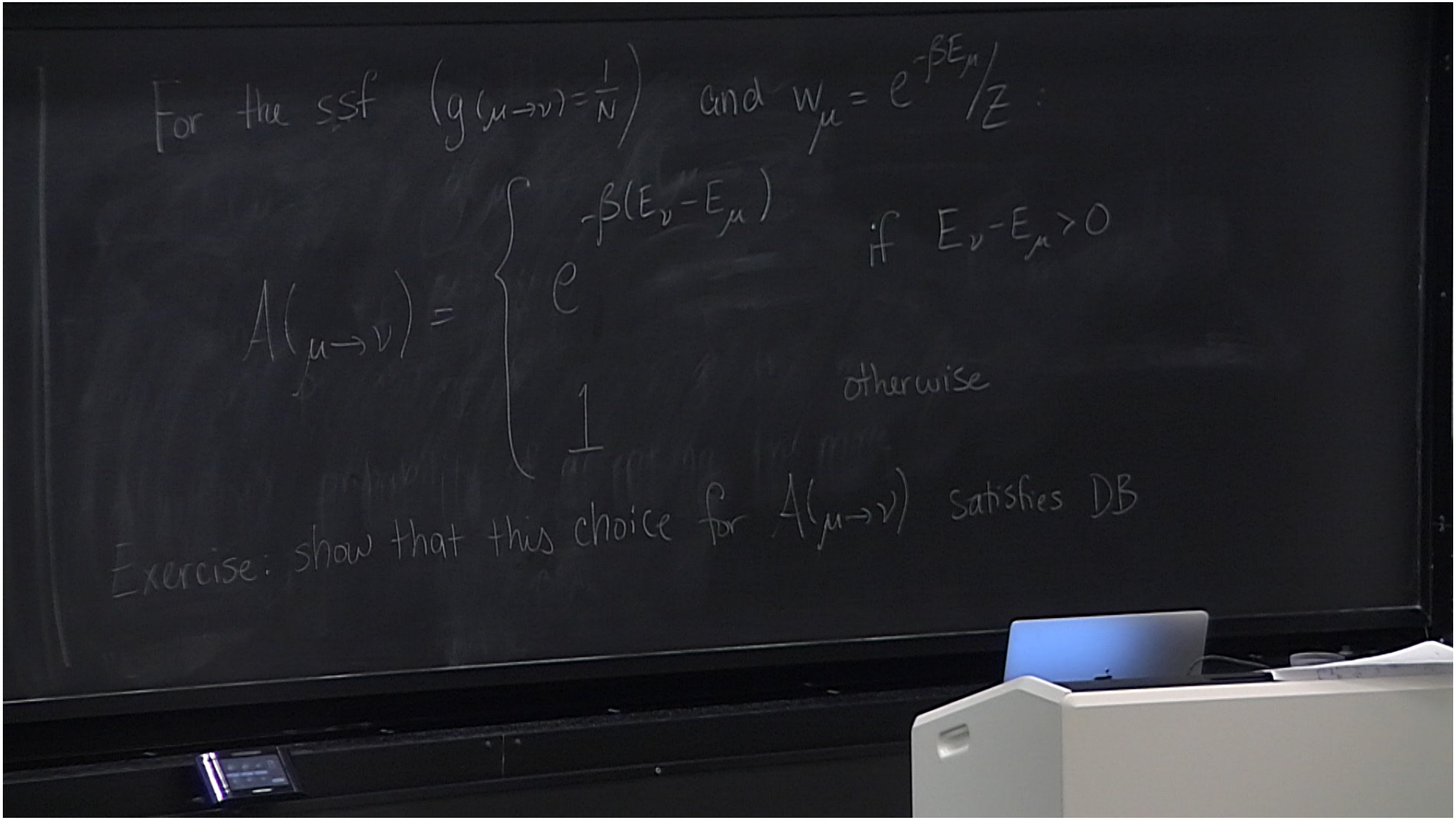
We are free to choose A_0 provided $A(\mu \to \nu) \leq 1$ for any ssf. Since the minimum energy difference for a ssf on a d-dim cubic lattice is $-4Jd$

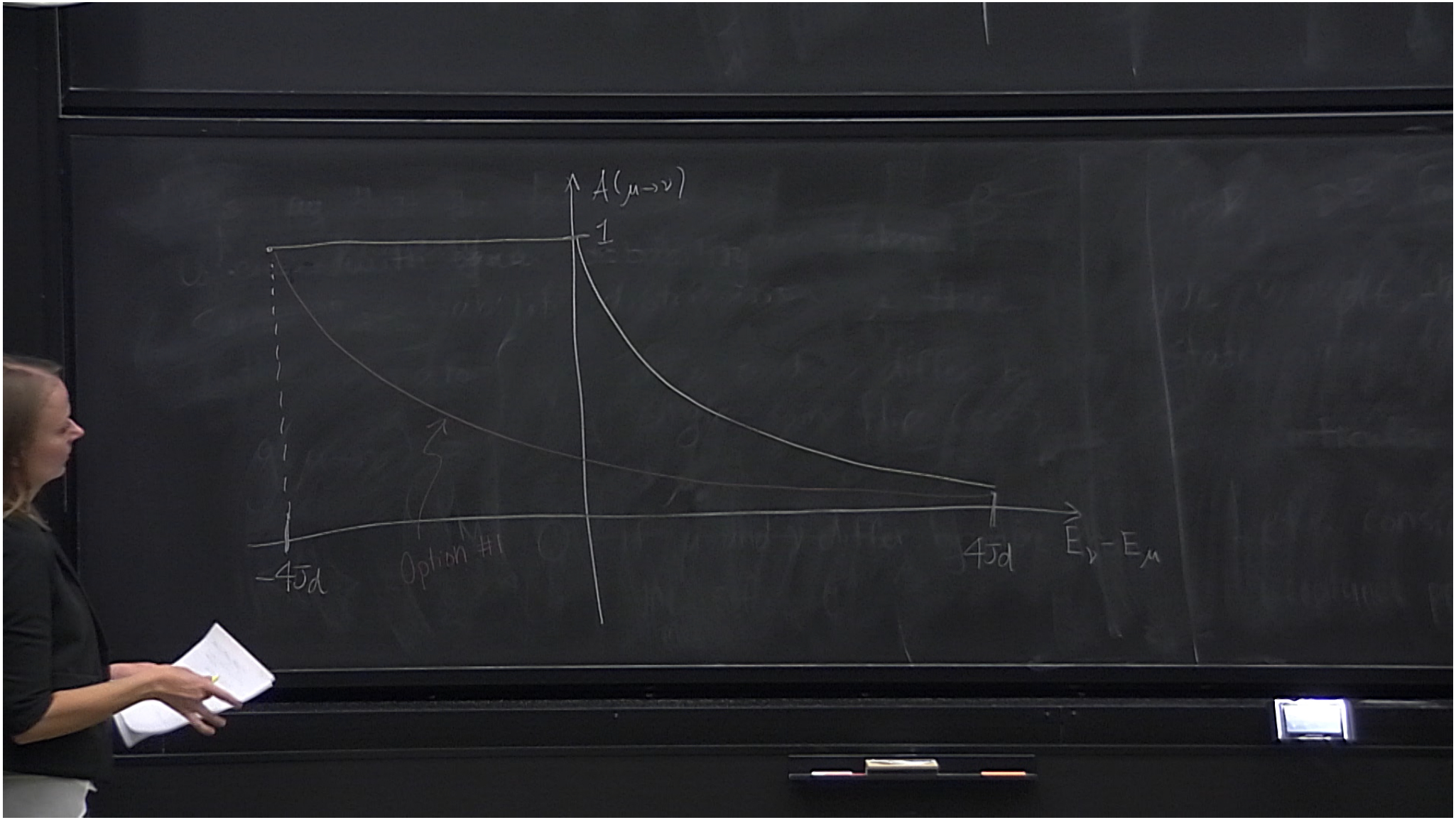


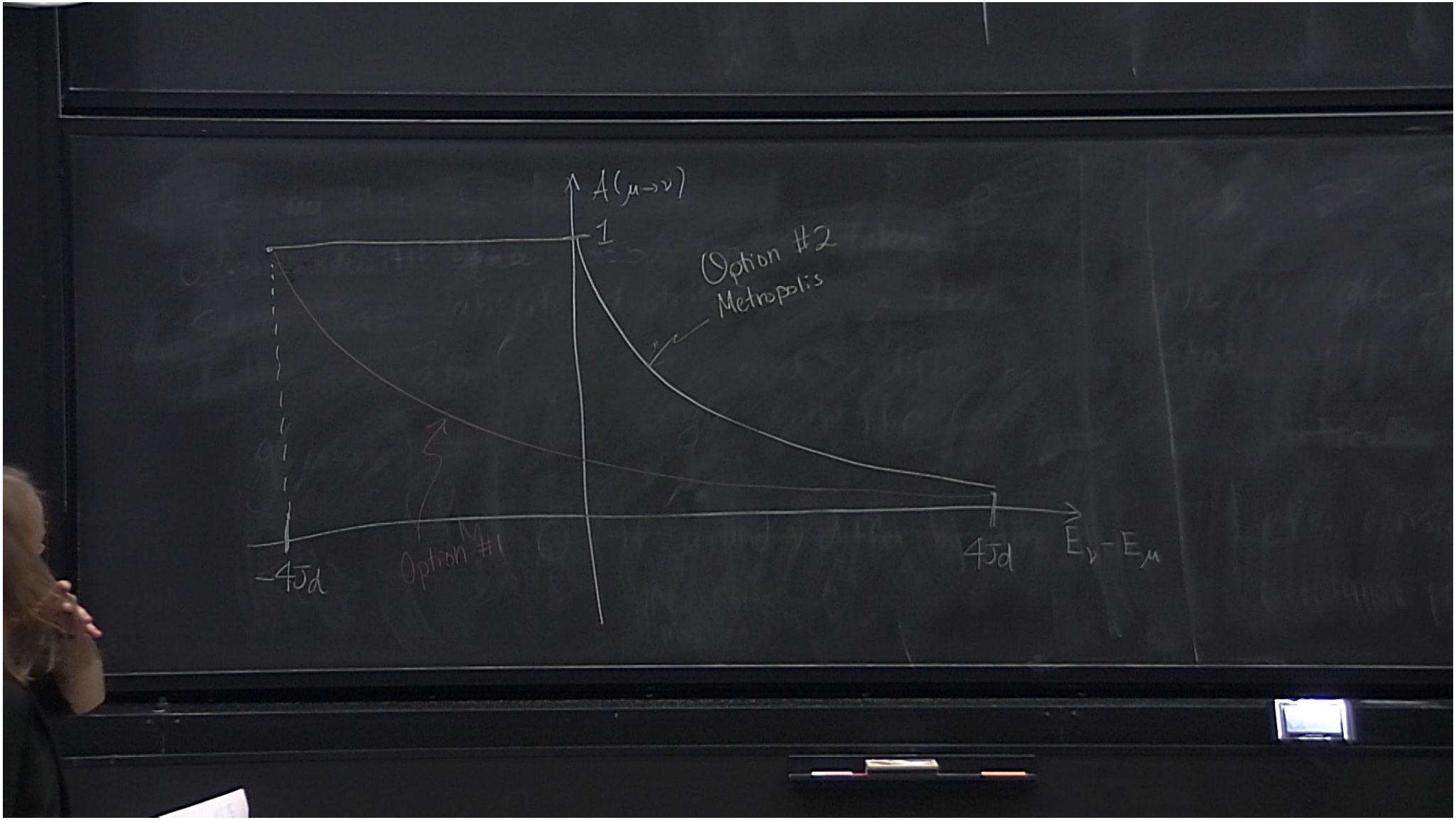

$A_0 = e^{-2\beta J d}$
$A(\mu \to \nu) = e^{-\frac{1}{2}\beta(E_\nu - E_\mu + 4Jd)}$

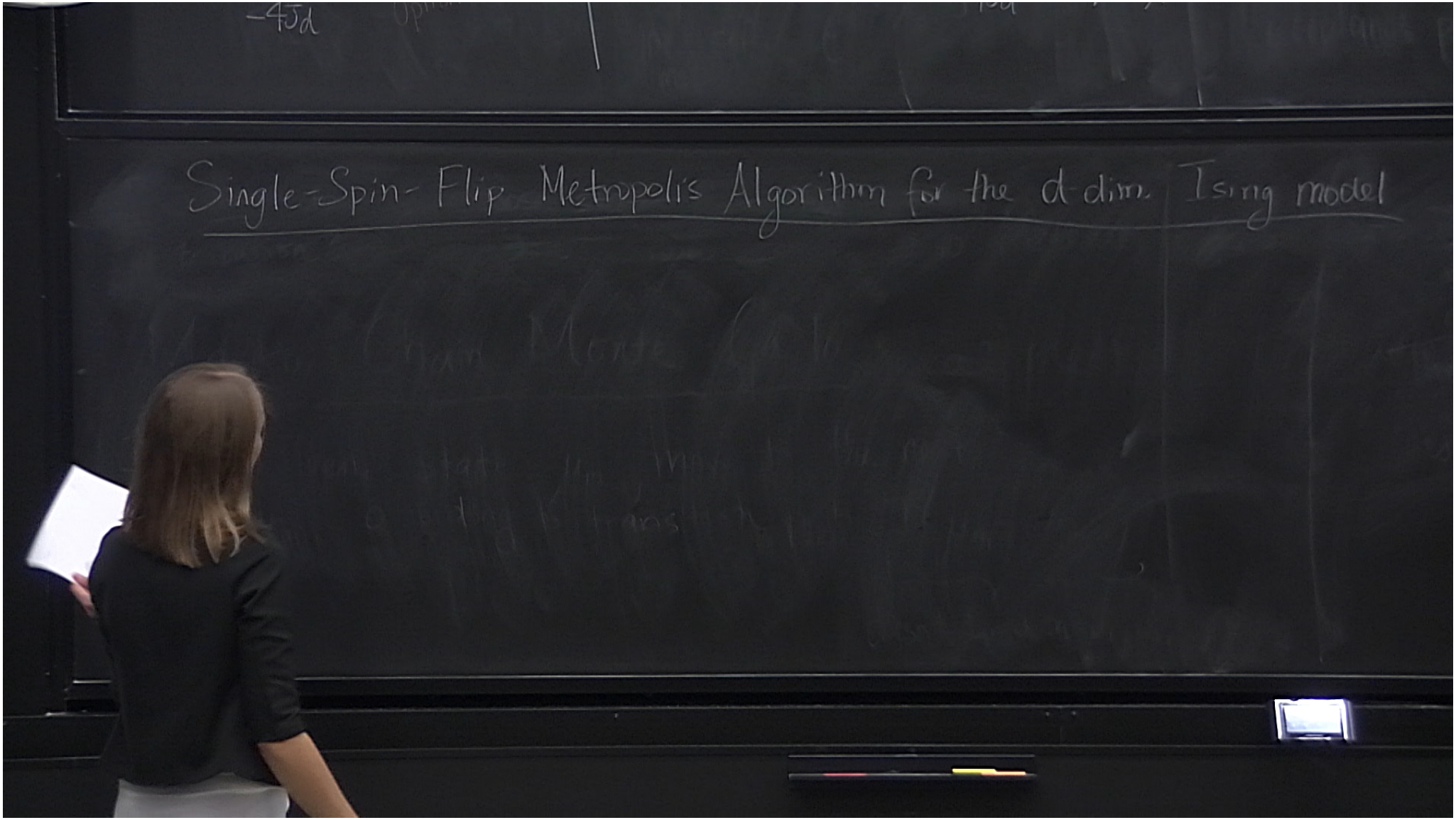

Option #2: Metropolis Algorithm (1953)
For a general proposed move $\mu \to \nu$ (not necessarily a ssf) and target distribution W_μ:
$$A(\mu \to \nu) = \min\left(1, \frac{g(\nu \to \mu)}{g(\mu \to \nu)} \frac{W_\nu}{W_\mu}\right)$$

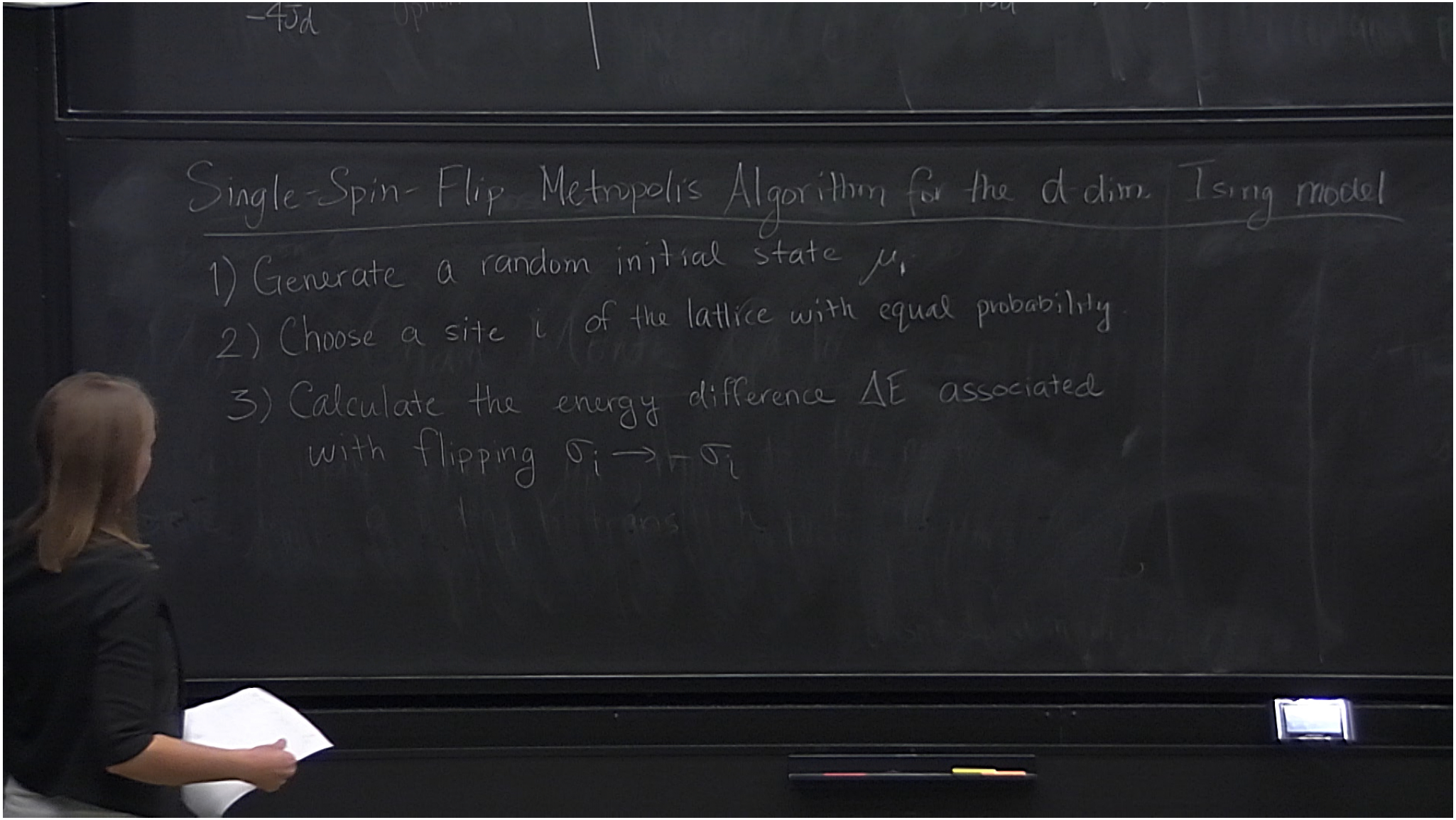

For the ssf $\left(g(\mu \to \nu) = \frac{1}{N}\right)$ and $w_\mu = e^{-\beta E_\mu}/Z$:
$A(\mu \to \nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & \text{if } E_\nu - E_\mu > 0 \\ 1 & \text{otherwise} \end{cases}$

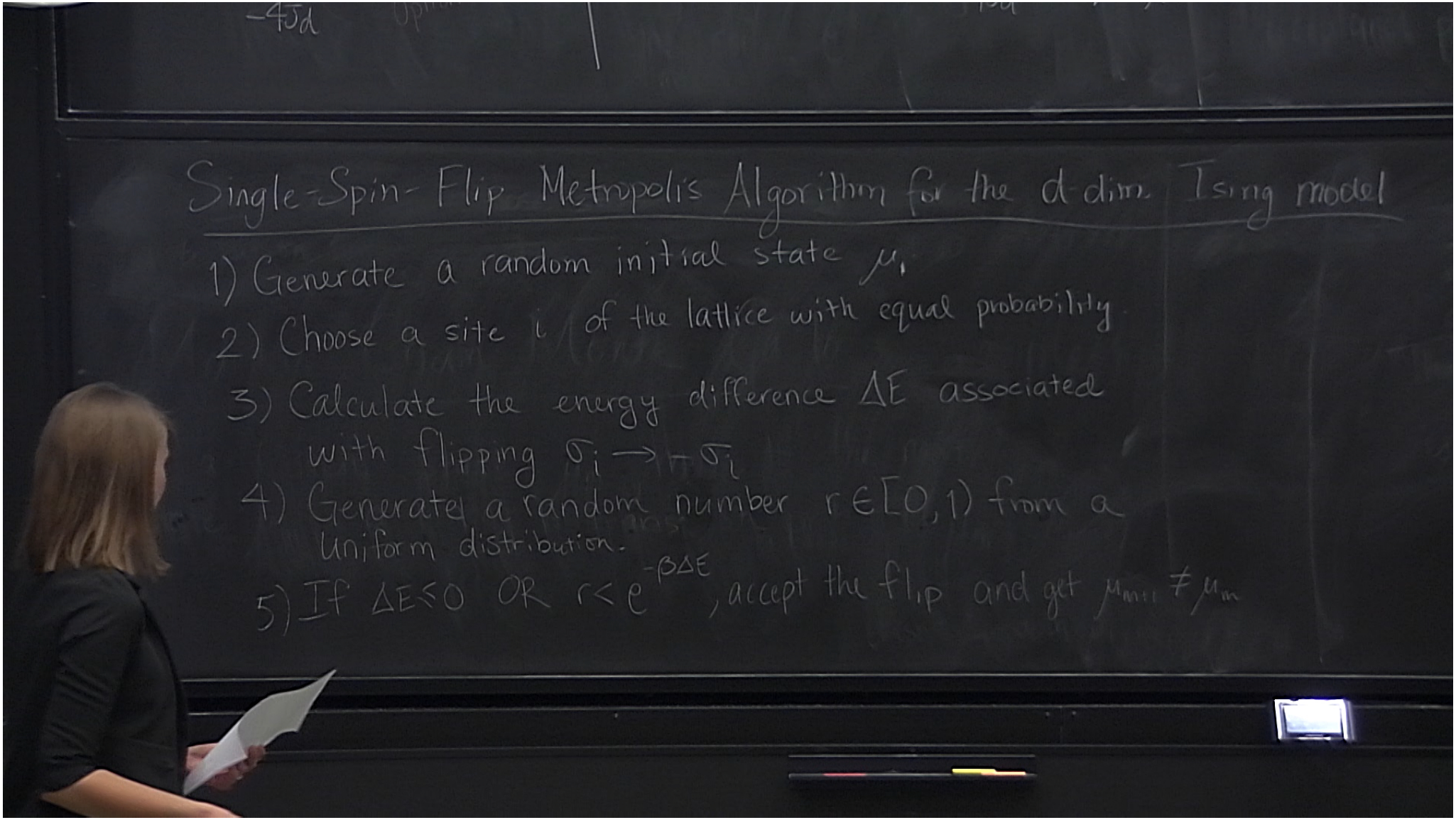

For the sst $(g(\mu\to\nu)=\frac{1}{N})$ and $W_\mu = e^{-\beta E_\mu}/Z$:
$A(\mu\to\nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & \text{if } E_\nu - E_\mu > 0 \\ 1 & \text{otherwise} \end{cases}$
Exercise: show that this choice for $A(\mu\to\nu)$ satisfies DB

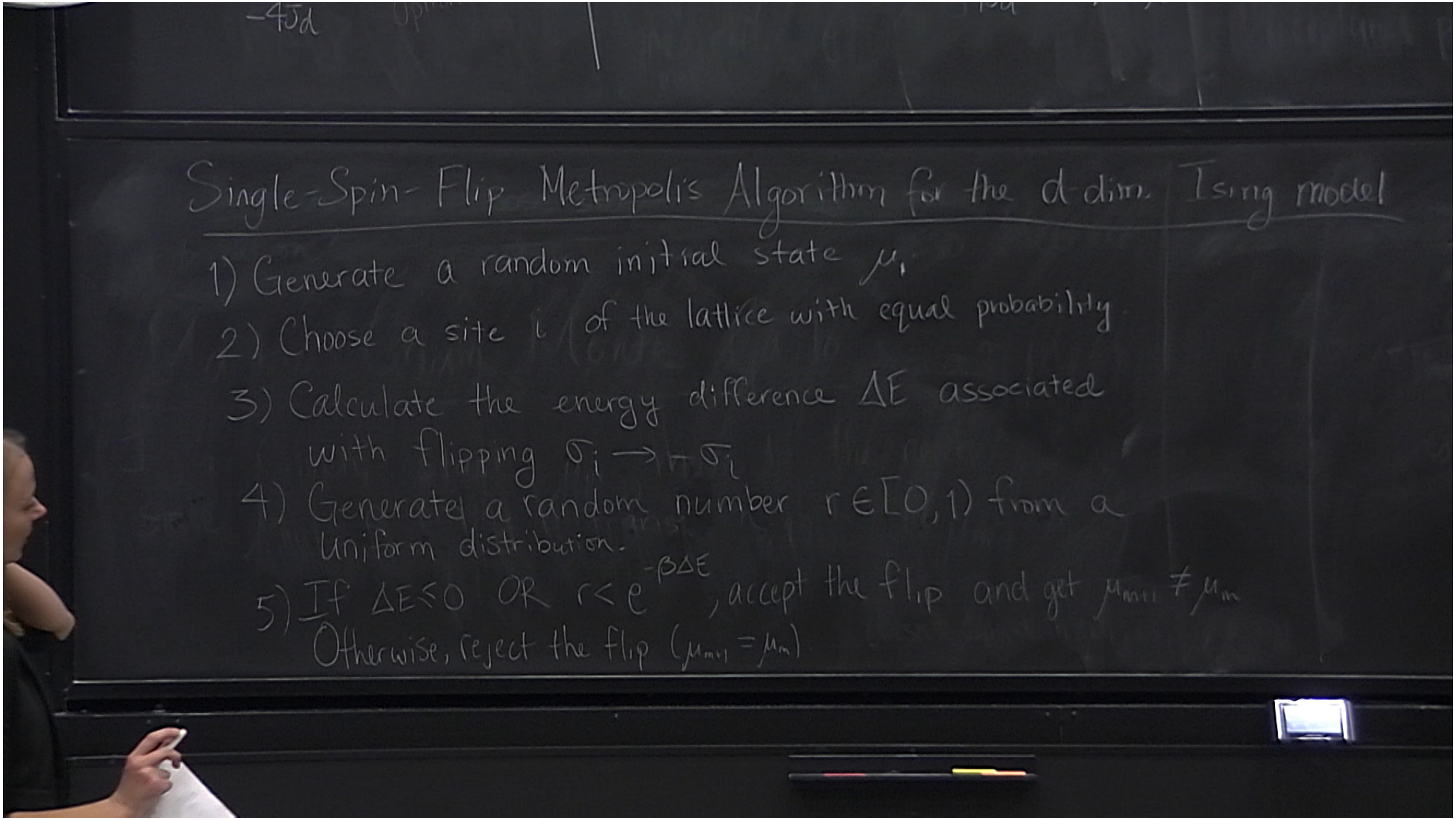

A(μ→ν)
1
-4Δd
Option #1
4Δd
E_ν - E_μ

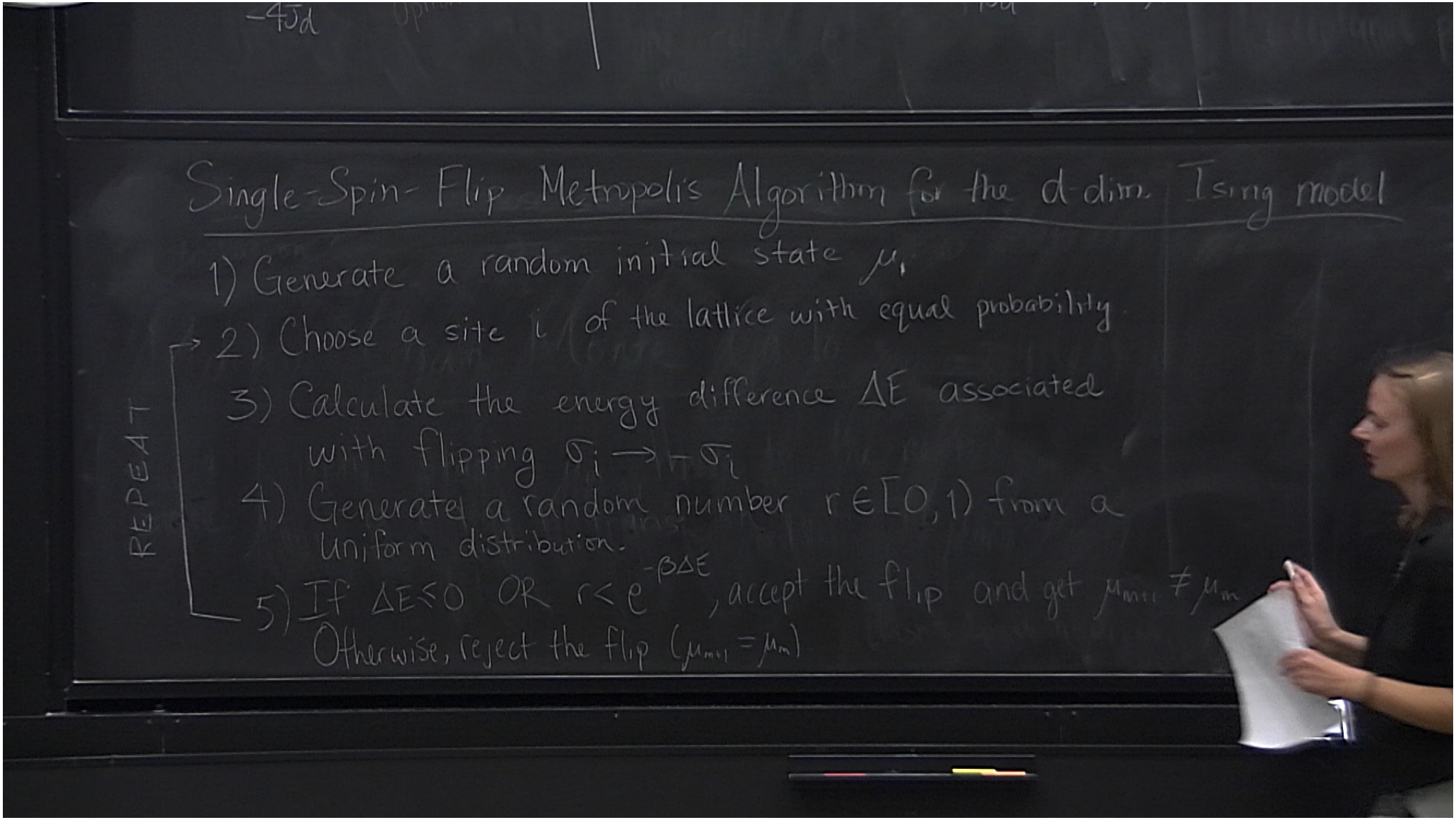

A(μ→ν)
1
Option #2
Metropolis
Option #1
-4Δd
4Δd
E_ν - E_μ


Single-Spin-Flip Metropolis Algorithm for the d-dim. Ising model


Single-Spin-Flip Metropolis Algorithm for the d-dim. Ising model
1) Generate a random initial state μ_1
2) Choose a site i of the lattice with equal probability
3) Calculate the energy difference ΔE associated with flipping $\sigma_i \to -\sigma_i$


Single-Spin-Flip Metropolis Algorithm for the d-dim. Ising model
1) Generate a random initial state μ_1
2) Choose a site i of the lattice with equal probability
3) Calculate the energy difference ΔE associated with flipping $\sigma_i \rightarrow -\sigma_i$
4) Generate a random number $r \in [0,1)$ from a uniform distribution.
5) If $\Delta E \leq 0$ OR $r < e^{-\beta \Delta E}$, accept the flip and get $\mu_{m+1} \neq \mu_m$


Single-Spin-Flip Metropolis Algorithm for the d-dim. Ising model
1) Generate a random initial state μ_1
2) Choose a site i of the lattice with equal probability
3) Calculate the energy difference ΔE associated with flipping $\sigma_i \to -\sigma_i$
4) Generate a random number $r \in [0,1)$ from a uniform distribution.
5) If $\Delta E \leq 0$ OR $r < e^{-\beta \Delta E}$, accept the flip and get $\mu_{m+1} \neq \mu_m$
Otherwise, reject the flip ($\mu_{m+1} = \mu_m$)


Single-Spin-Flip Metropolis Algorithm for the d-dim. Ising model
1) Generate a random initial state μ_1
2) Choose a site i of the lattice with equal probability
3) Calculate the energy difference ΔE associated with flipping $\sigma_i \to -\sigma_i$
4) Generate a random number $r \in [0,1)$ from a uniform distribution.
5) If $\Delta E \leq 0$ OR $r < e^{-\beta \Delta E}$, accept the flip and get $\mu_{m+1} \neq \mu_m$
Otherwise, reject the flip ($\mu_{m+1} = \mu_m$)
REPEAT