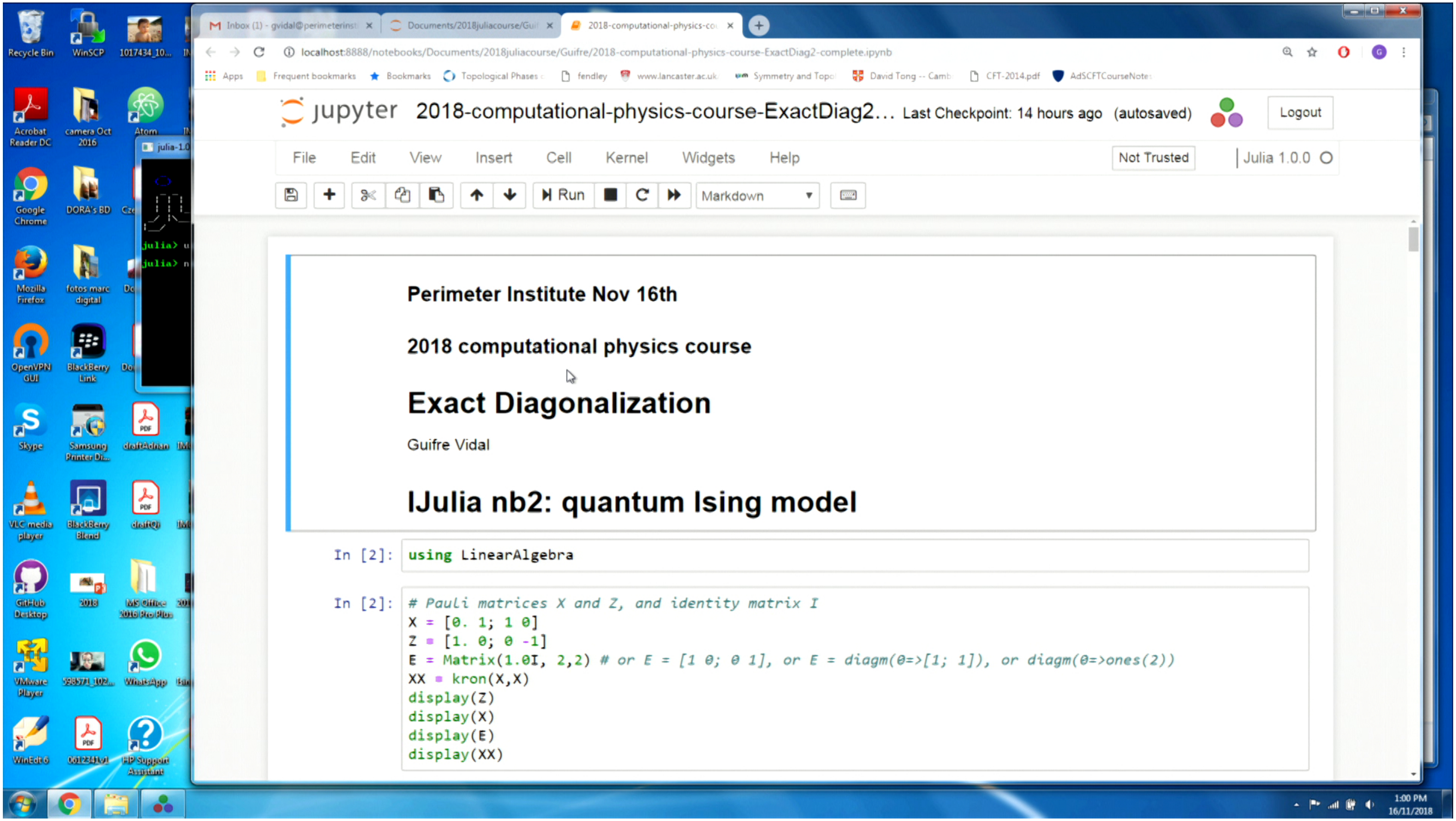
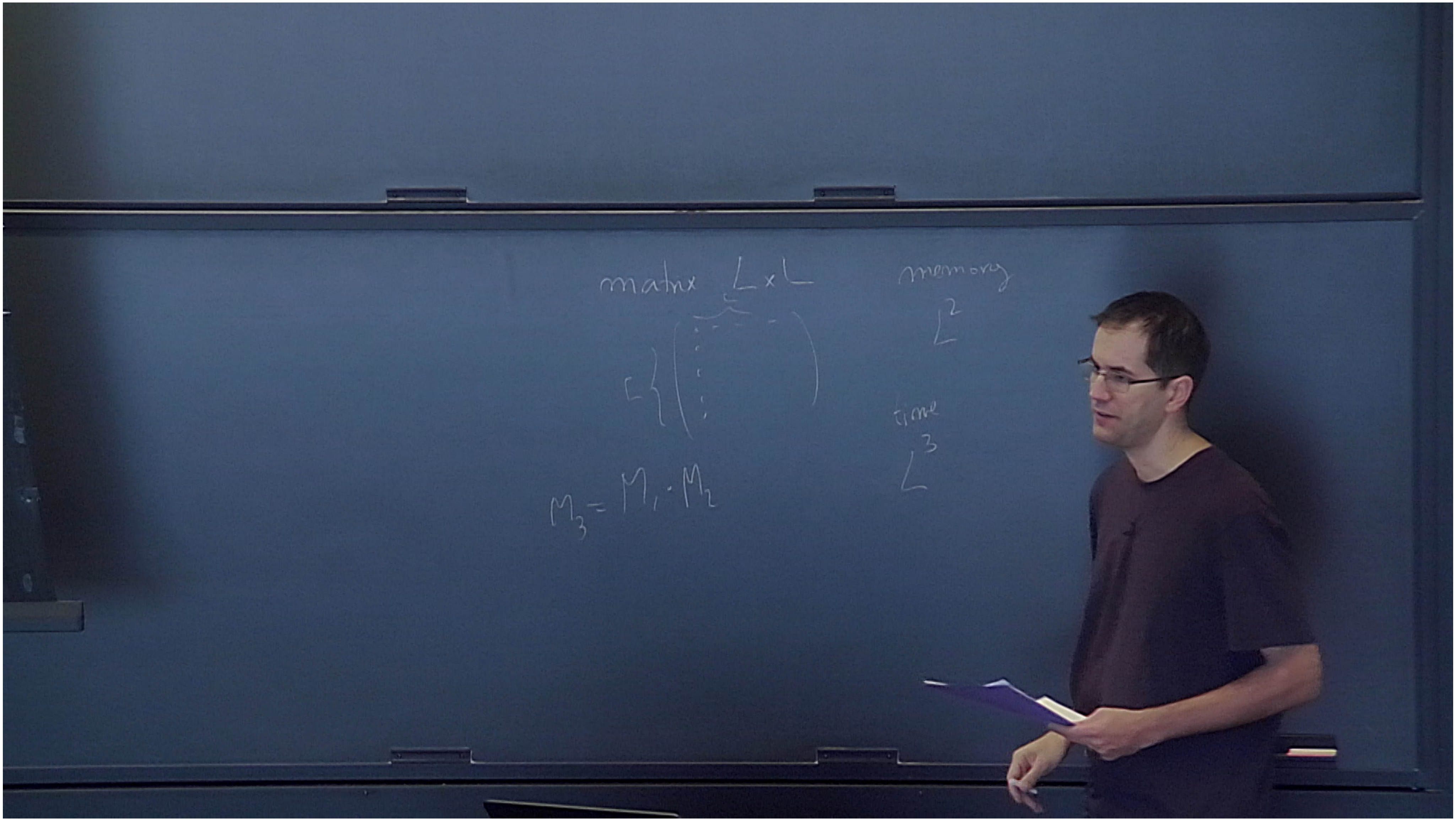
Title: Computational Physics - Lecture 20

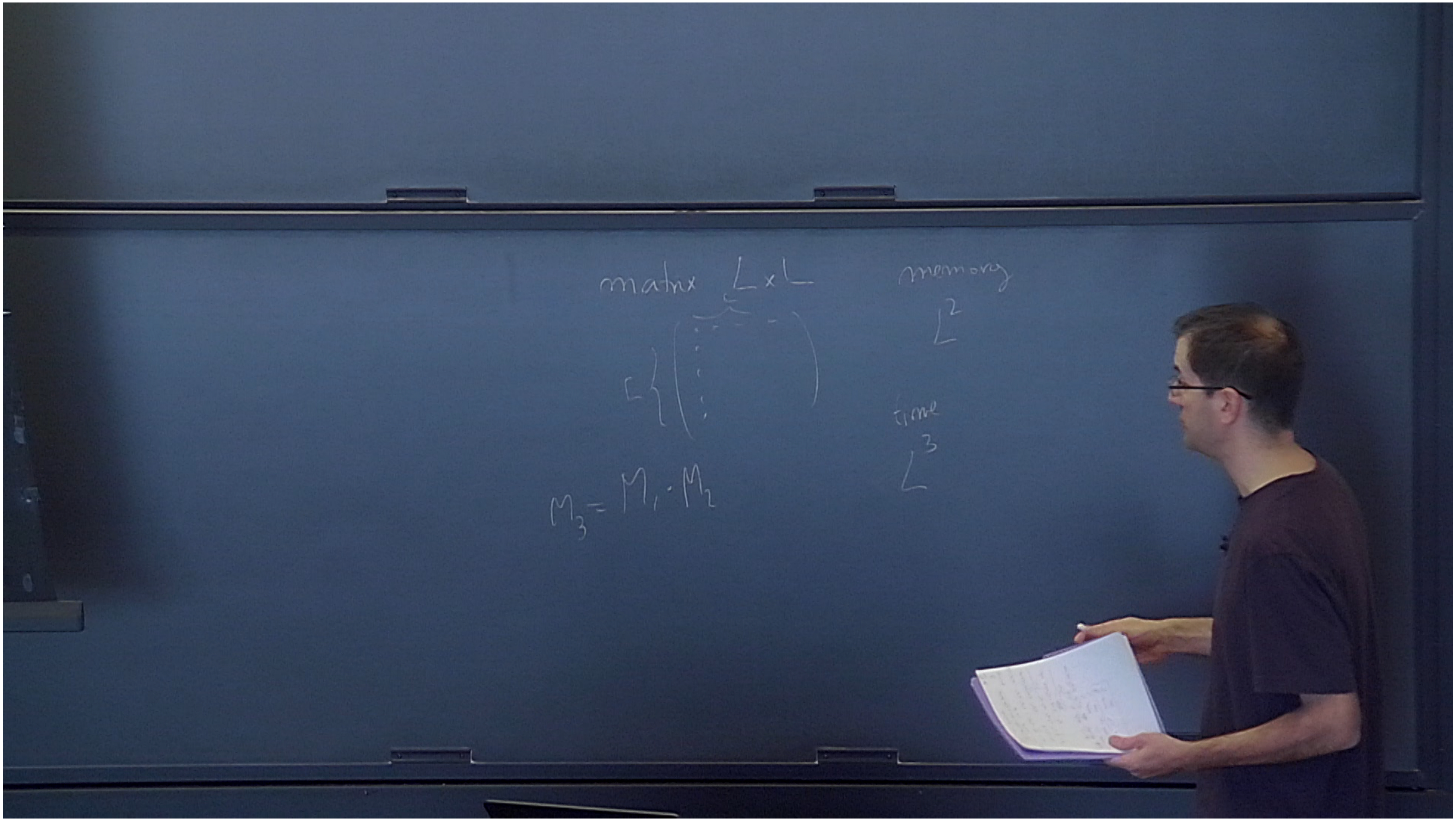
Date: Nov 16, 2018 01:00 PM

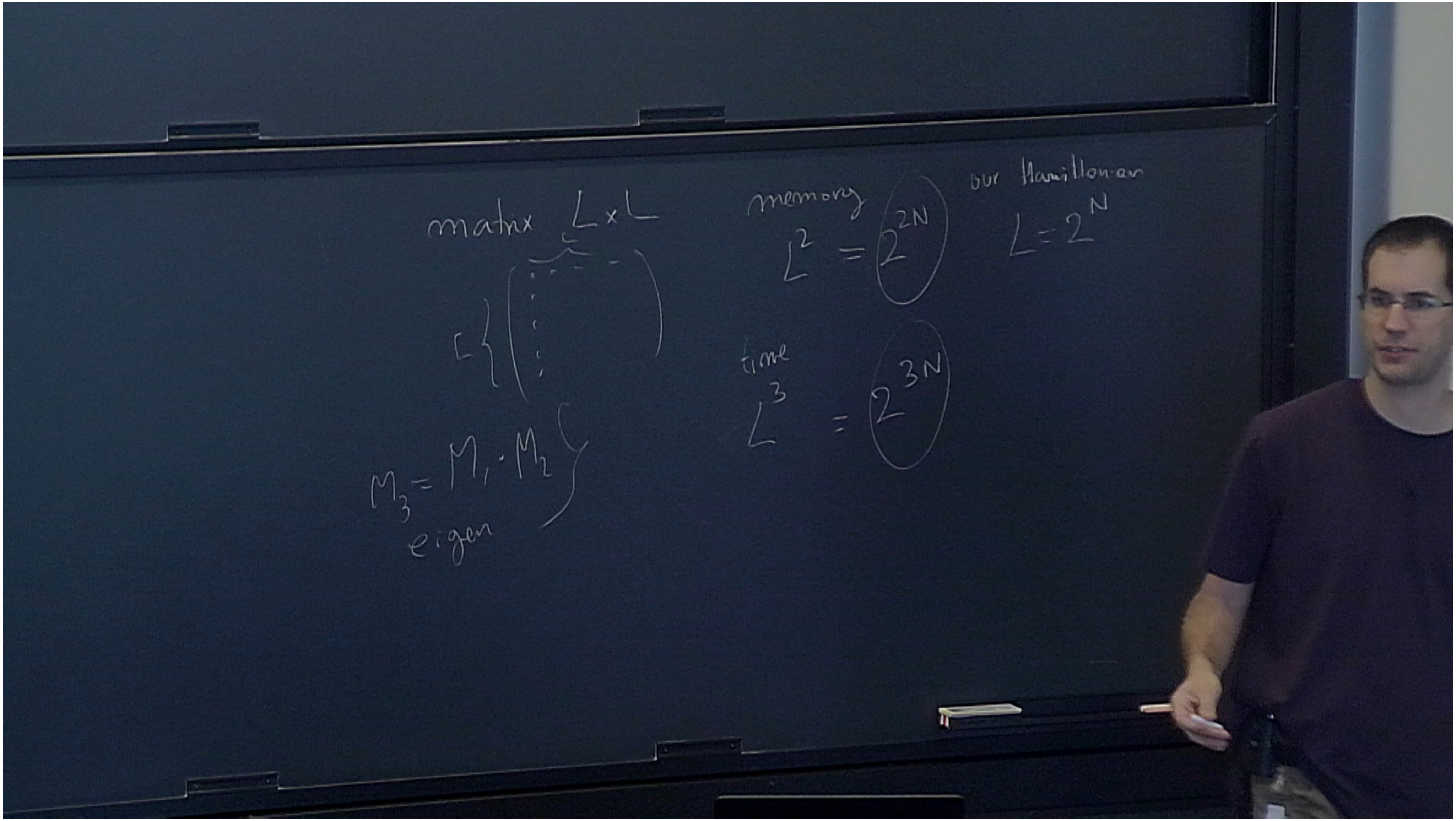
URL: http://pirsa.org/18110046

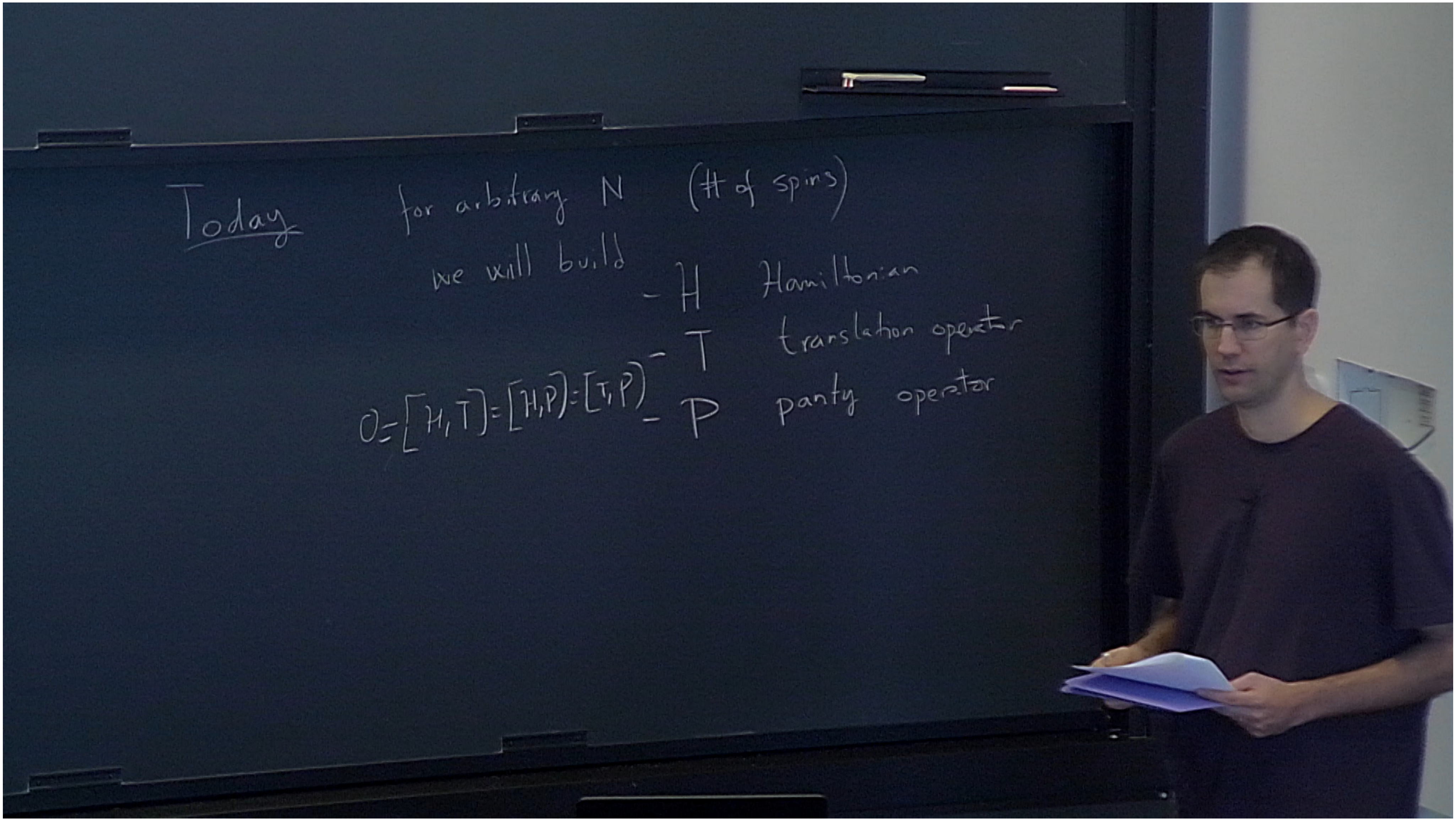
Abstract:


jupyter 2018-computational-physics-course-ExactDiag2... Last Checkpoint: 14 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Julia 1.0.0
Run
Markdown
Perimeter Institute Nov 16th
2018 computational physics course
Exact Diagonalization
Guifre Vidal
IJulia nb2: quantum Ising model
In [2]: using LinearAlgebra
In [2]: # Pauli matrices X and Z, and identity matrix I
X = [0. 1; 1 0]
Z = [1. 0; 0 -1]
E = Matrix(1.0I, 2,2) # or E = [1 0; 0 1], or E = diagm(0=>[1; 1]), or diagm(0=>ones(2))
XX = kron(X,X)
display(Z)
display(X)
display(E)
display(XX)


matrix L×L
memory
L²
time
L³
M₃ = M₁ · M₂


matrix L×L
memory
L^2
time
L^3
M_3 = M_1 · M_2


matrix L×L
L
memory
L² = 2^2N
our Hamiltonian
L = 2^N
time
L³ = 2^3N
M3 = M1 · M2
eigen


Today
for arbitrary N (# of spins)
we will build
- H Hamiltonian
- T translation operator
- P parity operator
0 = [H, T] = [H, P] = [T, P]

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```julia
display(E)
display(XX)
println(XX)
```

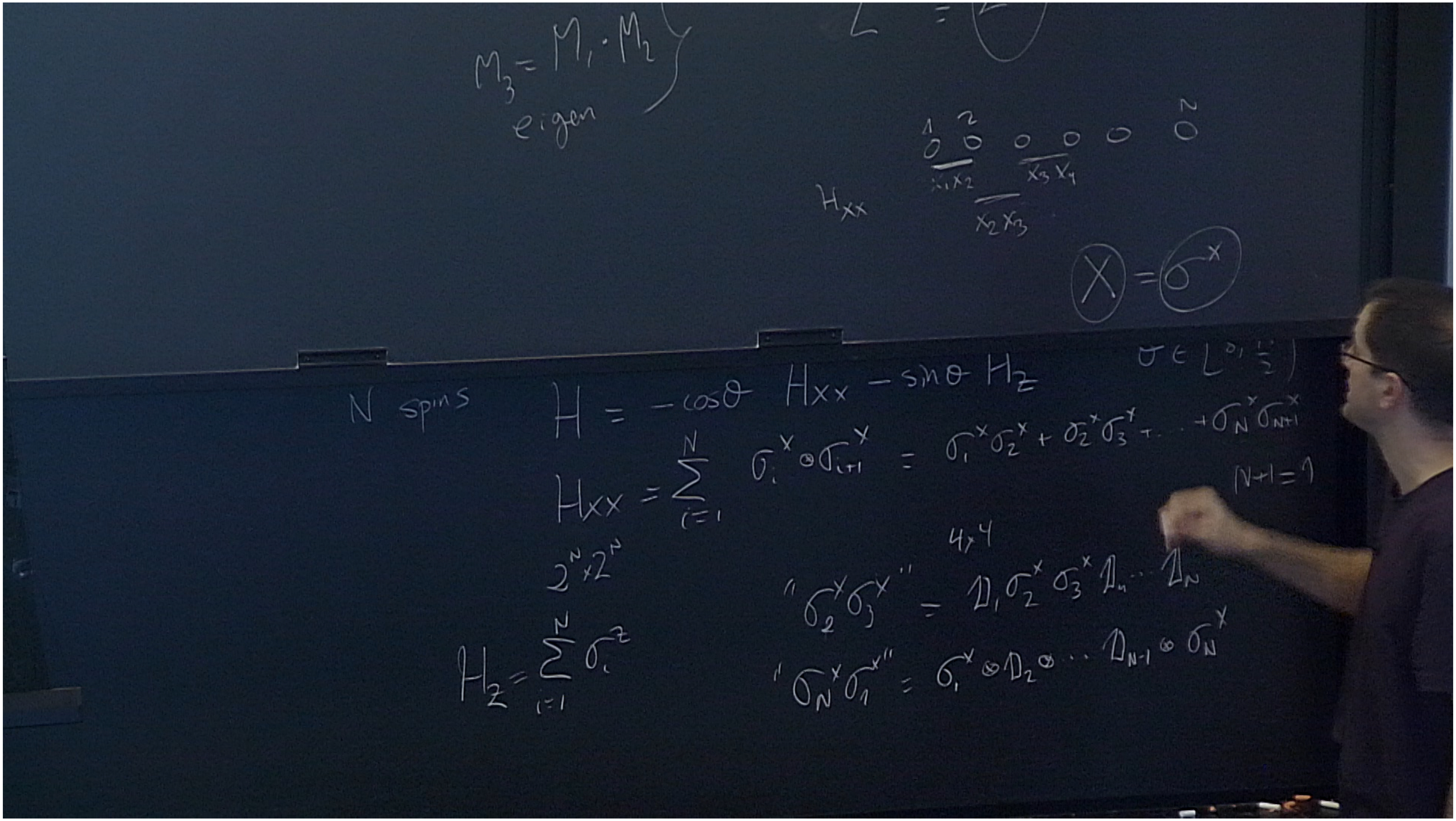
```
2×2 Array{Float64,2}:
 1.0   0.0
 0.0  -1.0

2×2 Array{Float64,2}:
 0.0  1.0
 1.0  0.0

2×2 Array{Float64,2}:
 1.0  0.0
 0.0  1.0

4×4 Array{Float64,2}:
 0.0  0.0  0.0  1.0
 0.0  0.0  1.0  0.0
 0.0  1.0  0.0  0.0
 1.0  0.0  0.0  0.0
```

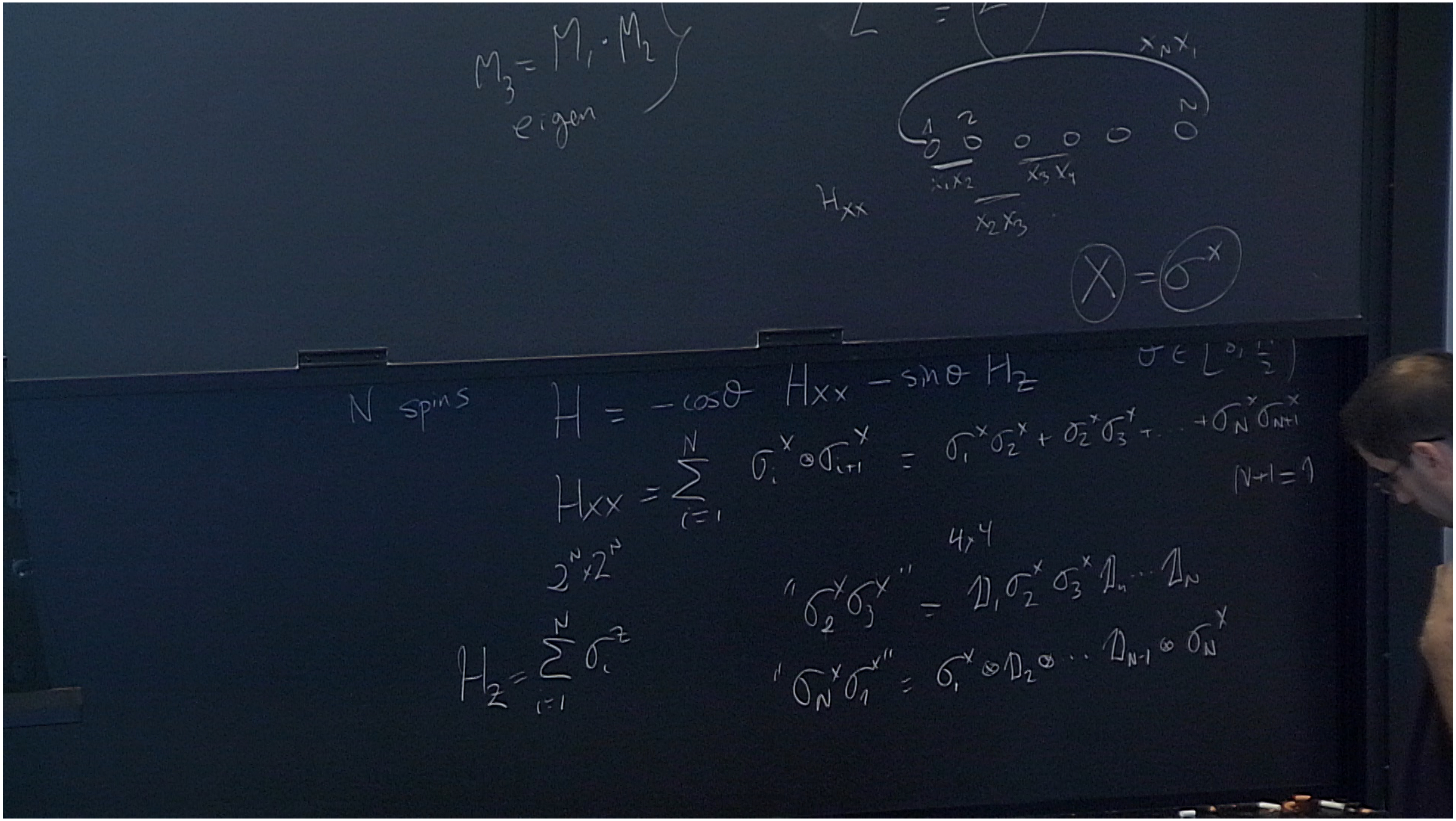
jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Hamiltonian for N spins (use $N \leq 10$)

In [8]:
```
# Ising Hamiltonian for N spins  with exact diagonalization
N = 3
theta = pi/4 # magnetic field
HXX = XX
HZ = kron(Z,E) + kron(E,Z)
for n = 3:N
    HXX = kron(HXX,E)+kron(diagm(0=>ones(2^(n-2))), XX)
    HZ =  kron(HZ,E) + kron(diagm(0=>ones(2^(n-1))),Z)
end
HXX = HXX + kron(X,kron(diagm(0=>ones(2^(N-2))),X))
H = -cos(theta)*HXX - sin(theta)*HZ

display(H)
```

```
8×8 Array{Float64,2}:
 -2.12132   -0.0        -0.0        …  -0.707107  -0.707107  -0.0
 -0.0       -0.707107   -0.707107      -0.0       -0.0       -0.707107
```


M3 = M1 · M2
eigen
Hxx
x1 x2
x3 x4
x2 x3
X = σx
N spins
H = −cosθ Hxx − sinθ Hz
Hxx = Σ σix ⊗ σi+1x = σ1x σ2x + σ2x σ3x + ... + σNx σN+1x
N+1 = 1
2N × 2N
4×4
"σ2x σ3x" = 1 1 σ2x σ3x 1 4 ... 1 N
"σNx σ1x" = σ1x ⊗ 1 2 ⊗ ... 1 N-1 ⊗ σNx
Hz = Σ σiz


$M_3 = M_1 \cdot M_2$
eigen
$x_N x_1$
1 2 N
$x_1 x_2$
$x_3 x_4$
$x_2 x_3$
H_{xx}
$X = \sigma^x$
$\sigma \in [0, \frac{1}{2})$
N spins
$H = -\cos\theta \, H_{xx} - \sin\theta \, H_Z$
$H_{xx} = \sum_{i=1}^{N} \sigma_i^x \otimes \sigma_{i+1}^x = \sigma_1^x \sigma_2^x + \sigma_2^x \sigma_3^x + \dots + \sigma_N^x \sigma_{N+1}^x$
$N+1 = 1$
$2^N \times 2^N$
4×4
"$\sigma_2^x \sigma_3^x$" $= \mathbb{1}_1 \sigma_2^x \sigma_3^x \mathbb{1}_4 \cdots \mathbb{1}_N$
"$\sigma_N^x \sigma_1^x$" $= \sigma_1^x \otimes \mathbb{1}_2 \otimes \cdots \mathbb{1}_{N-1} \otimes \sigma_N^x$
$H_Z = \sum_{i=1}^{N} \sigma_i^z$

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

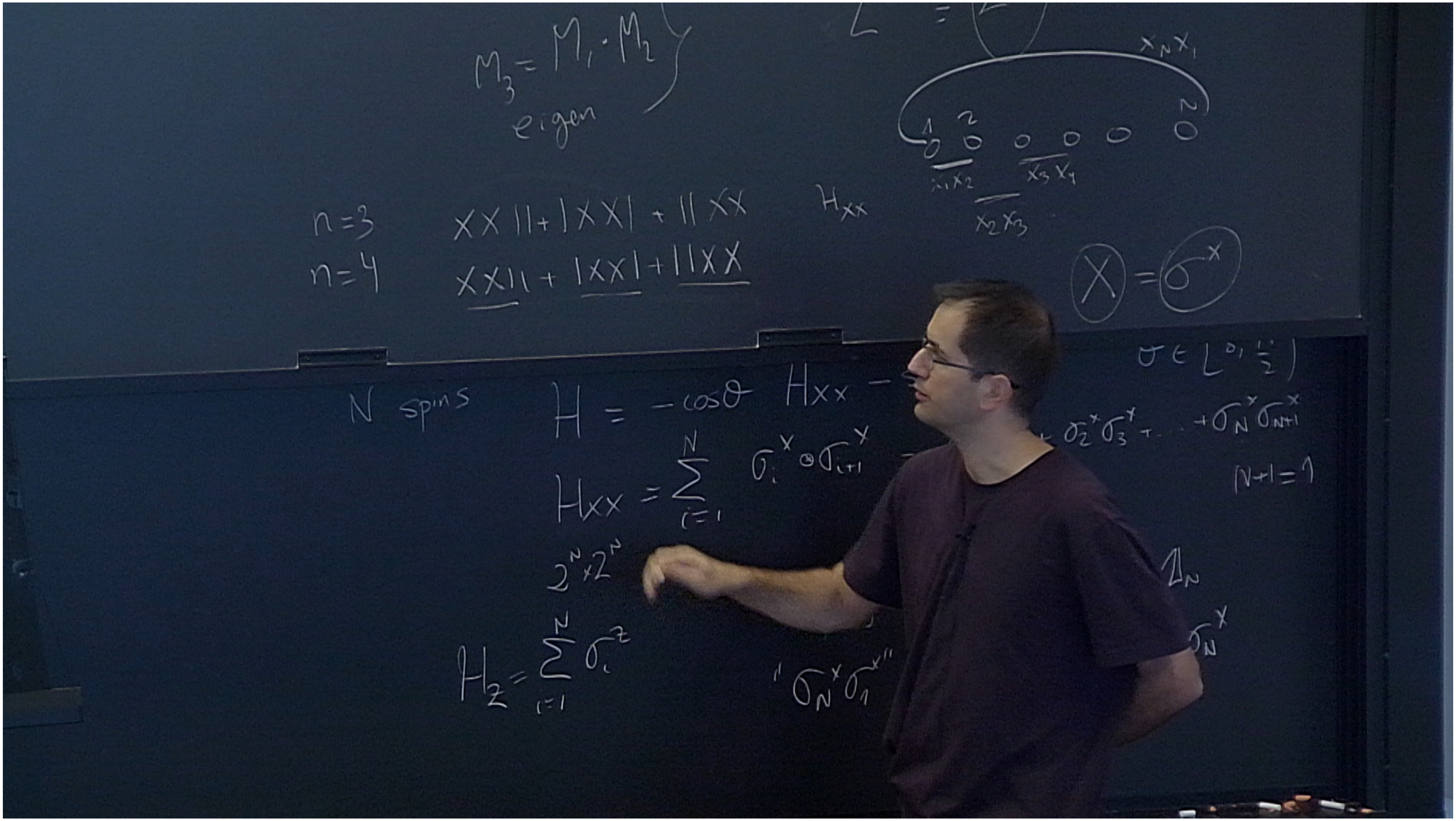
File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Hamiltonian for N spins (use $N \leq 10$)

In [8]:

```
# Ising Hamiltonian for N spins  with exact diagonalization
N = 3
theta = pi/4 # magnetic field
HXX = XX
HZ = kron(Z,E) + kron(E,Z)
for n = 3:N
    HXX = kron(HXX,E)+kron(diagm(0=>ones(2^(n-2))), XX)
    HZ =  kron(HZ,E) + kron(diagm(0=>ones(2^(n-1))),Z)
end
HXX = HXX + kron(X,kron(diagm(0=>ones(2^(N-2))),X))
H = -cos(theta)*HXX - sin(theta)*HZ

display(H)
```

```
8×8 Array{Float64,2}:
 -2.12132   -0.0        -0.0       …  -0.707107  -0.707107  -0.0
 -0.0       -0.707107  -0.707107     -0.0       -0.0       -0.707107
```


M3 = M1 · M2
eigen
n=3 XX11 + 1XX1 + 11XX
n=4 XX11 + 1XX1 + 11XX
Hxx
X = σx
N spins
H = −cosθ Hxx −
Hxx = Σ σix ⊗ σi+1x
2N×2N
Hz = Σ σiz
"σNx σ1x"
N+1 = 1

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Hamiltonian for N spins (use $N \leq 10$)

In [8]:
```julia
# Ising Hamiltonian for N spins  with exact diagonalization
N = 3
theta = pi/4 # magnetic field
HXX = XX
HZ = kron(Z,E) + kron(E,Z)
for n = 3:N
    HXX = kron(HXX,E)+kron(diagm(0=>ones(2^(n-2))), XX)
    HZ =  kron(HZ,E) + kron(diagm(0=>ones(2^(n-1))),Z)
end
HXX = HXX + kron(X,kron(diagm(0=>ones(2^(N-2))),X))
H = -cos(theta)*HXX - sin(theta)*HZ

display(H)
```

```
8×8 Array{Float64,2}:
 -2.12132   -0.0        -0.0        …  -0.707107  -0.707107  -0.0
 -0.0       -0.707107   -0.707107      -0.0       -0.0       -0.707107
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

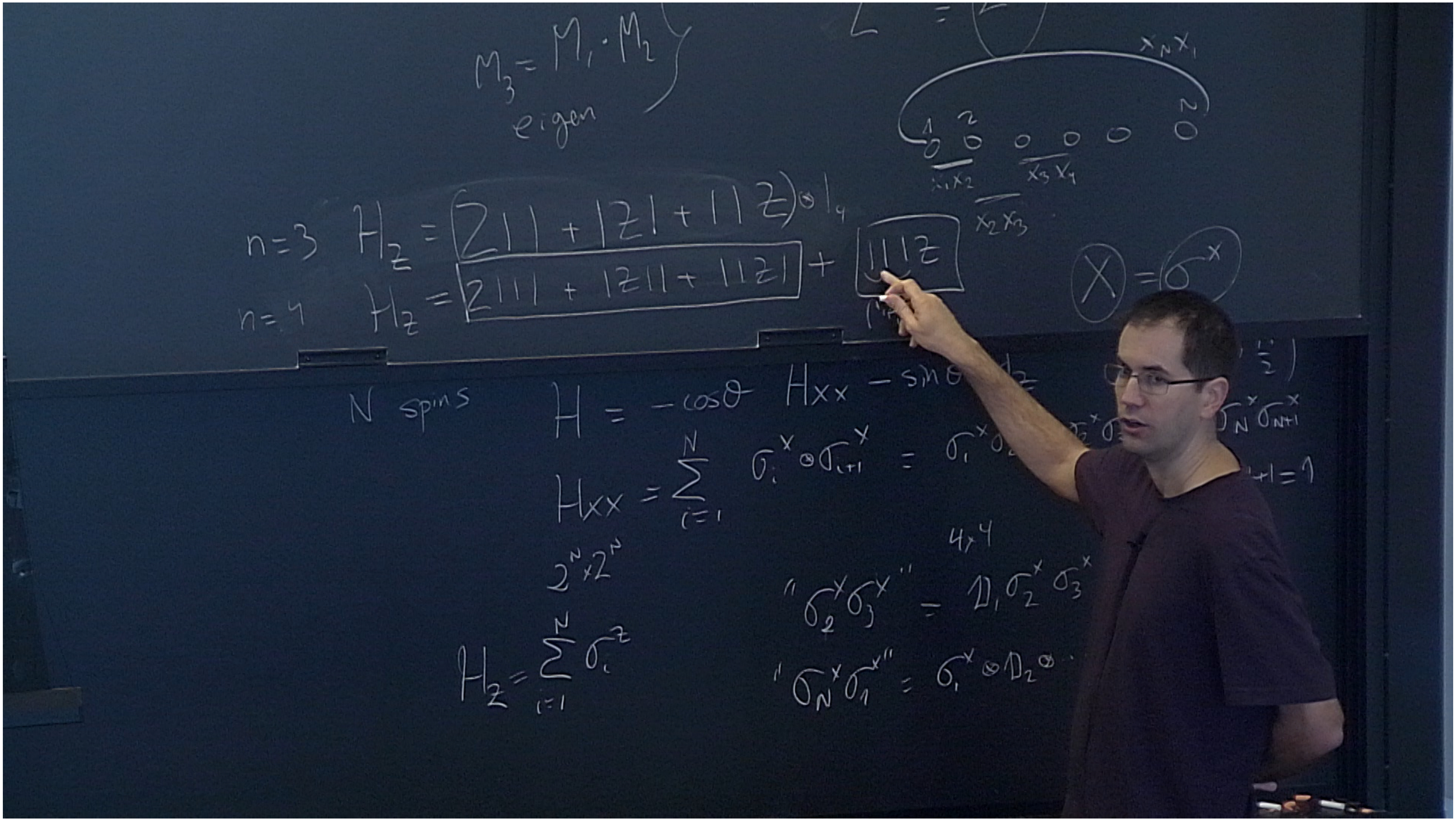
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

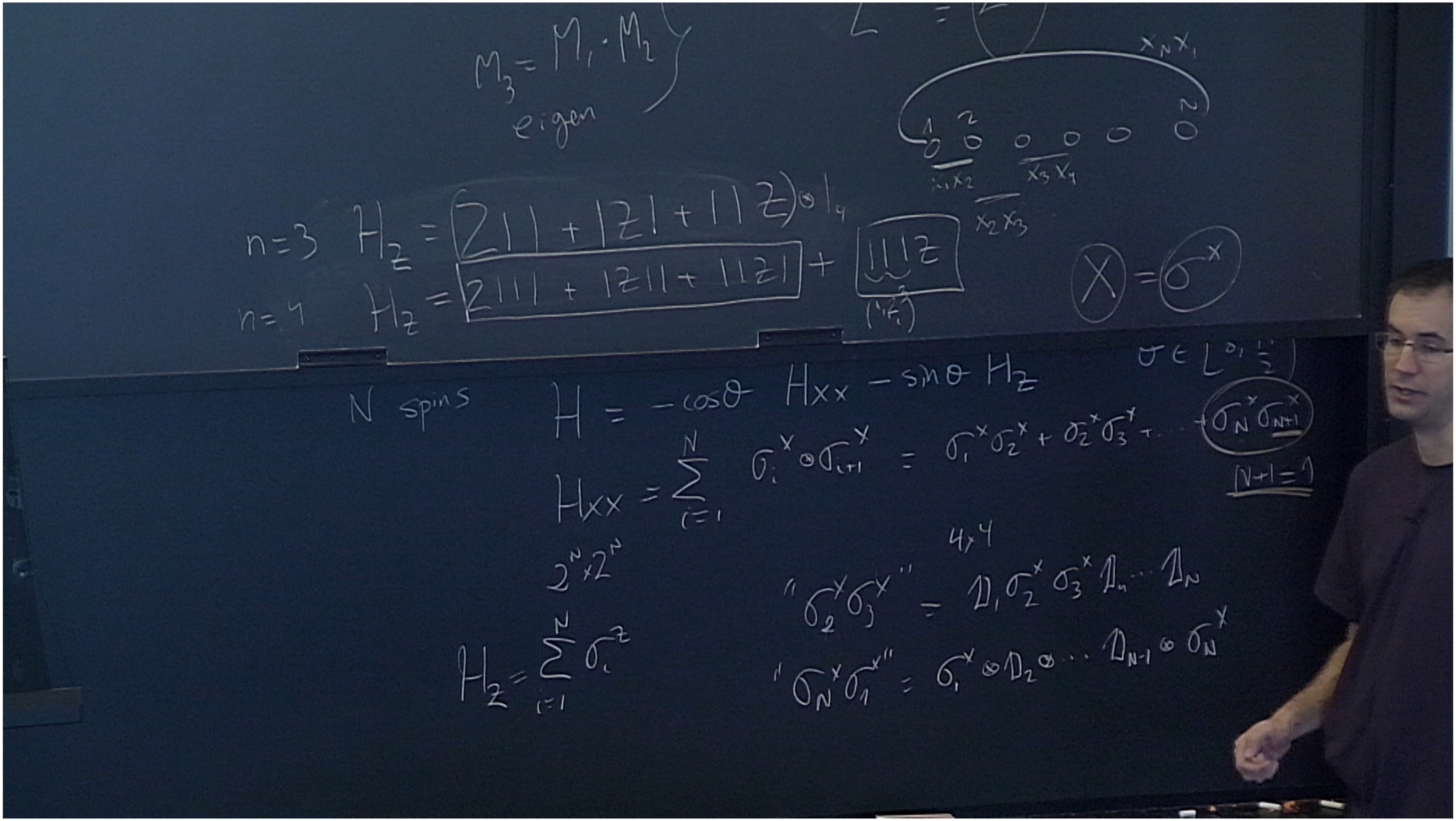
Hamiltonian for N spins (use $N \leq 10$)

In [8]:

```julia
# Ising Hamiltonian for N spins  with exact diagonalization
N = 3
theta = pi/4 # magnetic field
HXX = XX
HZ = kron(Z,E) + kron(E,Z)
for n = 3:N
    HXX = kron(HXX,E)+kron(diagm(0=>ones(2^(n-2))), XX)
    HZ =  kron(HZ,E) + kron(diagm(0=>ones(2^(n-1))),Z)
end
HXX = HXX + kron(X,kron(diagm(0=>ones(2^(N-2))),X))
H = -cos(theta)*HXX - sin(theta)*HZ

display(H)
```

```
8×8 Array{Float64,2}:
 -2.12132   -0.0        -0.0       …  -0.707107  -0.707107  -0.0
 -0.0       -0.707107   -0.707107     -0.0       -0.0       -0.707107
```


$M_3 = M_1 \cdot M_2$
eigen
$n=3 \quad H_Z = (Z11 + 1Z1 + 11Z) \otimes 1_4$
$n=4 \quad H_Z = Z111 + 1Z11 + 11Z1 + 111Z$
$X = \sigma^x$
N spins
$H = -\cos\theta \, H_{xx} - \sin\theta \, H_Z$
$H_{xx} = \sum_{i=1}^{N} \sigma_i^x \otimes \sigma_{i+1}^x$
$2^N \times 2^N$
$H_Z = \sum_{i=1}^{N} \sigma_i^z$



M3 = M1 · M2
eigen
x1x2
x3x4
x2x3
xNx1
n=3 HZ = (Z11 + 1Z1 + 11Z) ⊗ 14
n=4 HZ = Z111 + 1Z11 + 11Z1 + 111Z
X = σx
N spins
H = −cosθ Hxx − sinθ HZ
Hxx = Σ σix ⊗ σi+1x = σ1xσ2x + σ2xσ3x + ... + σNxσN+1x
N+1 = 1
2N x 2N
4x4
"σ2xσ3x" = 11 σ2x σ3x 14 ... 1N
"σNxσ1x" = σ1x ⊗ 12 ⊗ ... 1N-1 ⊗ σNx
HZ = Σ σiz

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Code

```
for n = 3:N
    HXX = kron(HXX,E)+kron(diagm(0=>ones(2^(n-2))), XX)
    HZ =  kron(HZ,E) + kron(diagm(0=>ones(2^(n-1))),Z)
end
HXX = HXX + kron(X,kron(diagm(0=>ones(2^(N-2))),X))
H = -cos(theta)*HXX - sin(theta)*HZ

display(H)
```

```
8×8 Array{Float64,2}:
 -2.12132    -0.0        -0.0       …  -0.707107  -0.707107  -0.0
 -0.0        -0.707107   -0.707107     -0.0       -0.0       -0.707107
 -0.0        -0.707107   -0.707107     -0.0       -0.0       -0.707107
 -0.707107   -0.0        -0.0          -0.707107  -0.707107  -0.0
 -0.0        -0.707107   -0.707107     -0.0       -0.0       -0.707107
 -0.707107   -0.0        -0.0       …   0.707107  -0.707107  -0.0
 -0.707107   -0.0        -0.0          -0.707107   0.707107  -0.0
 -0.0        -0.707107   -0.707107     -0.0       -0.0        2.12132
```

magnetic field angle $\theta \in [0, \pi/2]$

jupyter 2018-computational-physics-course-ExactDiag… (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```
 -0.707107  -0.0       -0.0          -0.707107  -0.707107  -0.0
 -0.0       -0.707107  -0.707107     -0.0       -0.0       -0.707107
 -0.707107  -0.0       -0.0      …    0.707107  -0.707107  -0.0
 -0.707107  -0.0       -0.0          -0.707107   0.707107  -0.0
 -0.0       -0.707107  -0.707107     -0.0       -0.0        2.12132
```

magnetic field angle $\theta \in [0, \pi/2]$

In [4]:

```julia
N_theta=21  # number of values of magnetic field
d_theta = pi/(2*(N_theta-1)) # increments of theta
theta_val = collect(0:d_theta:pi/2) # list of magnetic field angle theta [0, pi/2
Energies = zeros(2^3,N_theta)
for i=1:N_theta
    H = -cos(theta_val[i])*HXX - sin(theta_val[i])*HZ # build the Hamiltonian
    D,U = eigen(H)          # diagonalize the Hamiltonian
    Energies[:,i] = D
end
[theta_val Energies[1,:]]
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

```
    H = -cos(theta_val[i])*HXX - sin(theta_val[i])*HZ # build the Hamiltonian
    D,U = eigen(H)          # diagonalize the Hamiltonian
    Energies[:,i] = D
end
[theta_val Energies[1,:]]
```

Out[5]:

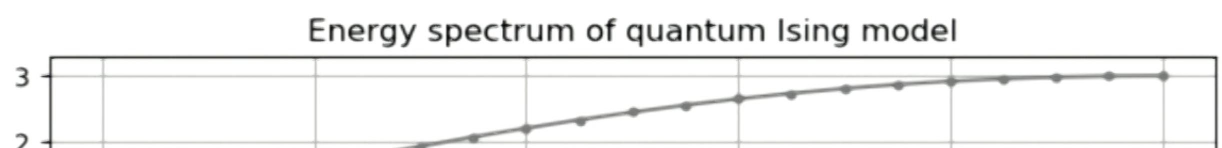
```
21×2 Array{Float64,2}:
 0.0        -3.0
 0.0785398  -2.99557
 0.15708    -2.98314
 0.235619   -2.96423
 0.314159   -2.94068
 0.392699   -2.9146
 0.471239   -2.88836
 0.549779   -2.86443
 0.628319   -2.84521
 0.706858   -2.83275
 0.785398   -2.82843
 0.863938   -2.83275
 0.942478   -2.84521
```

```
1.5708      -3.0
```

Let us plot this!

In [6]:
```
using PyPlot
```

In [7]:
```
figure("My_figure",figsize=(8,3))
grid("on")
for i in 1:2^N
    plot(2*theta_val/pi,vec(Energies[i,:]), marker=".")
end
title("Energy spectrum of quantum Ising model")
xlabel("magnetic field theta")
ylabel("Energy levels");
```



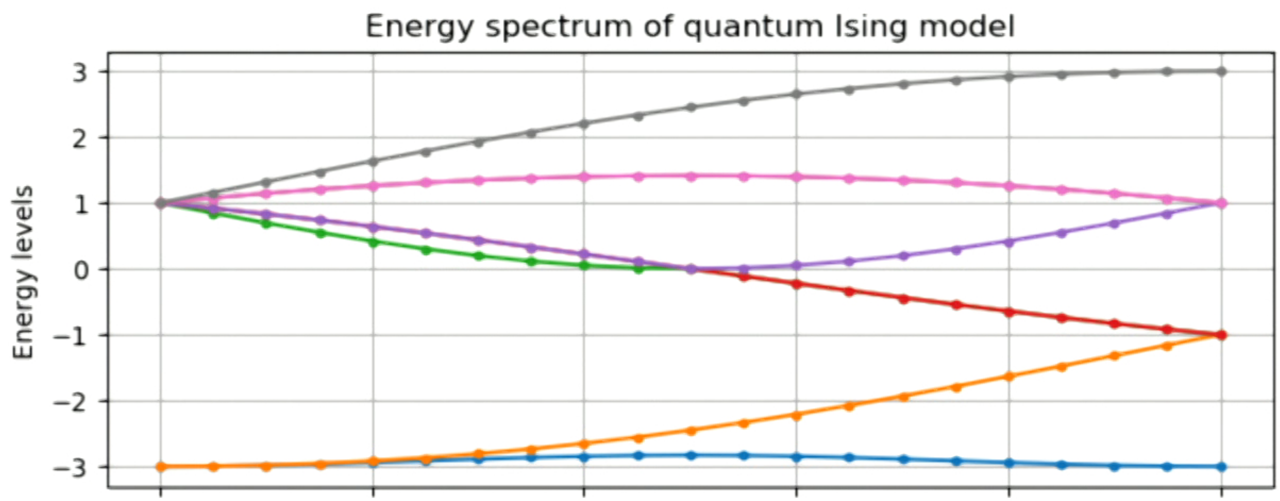
jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

Logout

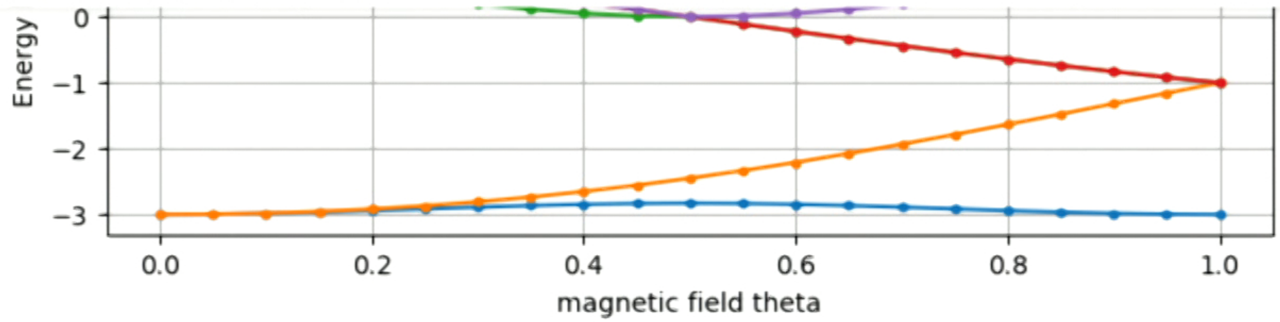
File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Markdown

```
for i in 1:2^N
    plot(2*theta_val/pi,vec(Energies[i,:]), marker=".")
end
title("Energy spectrum of quantum Ising model")
xlabel("magnetic field theta")
ylabel("Energy levels");
```



jupyter 2018-computational-physics-course-ExactDiag2-co… (autosaved)



Now it's your turn! (1/3)

1) write a function buildH that builds the Hamiltonian for N spins (use only for $N \leq 10$)

Solution

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Out[3]:

```
8×8 Array{Float64,2}:
 -2.12132    -0.0        -0.0       …  -0.707107  -0.707107  -0.0
 -0.0        -0.707107  -0.707107     -0.0       -0.0       -0.707107
 -0.0        -0.707107  -0.707107     -0.0       -0.0       -0.707107
 -0.707107  -0.0        -0.0          -0.707107  -0.707107  -0.0
 -0.0        -0.707107  -0.707107     -0.0       -0.0       -0.707107
 -0.707107  -0.0        -0.0       …   0.707107  -0.707107  -0.0
 -0.707107  -0.0        -0.0          -0.707107   0.707107  -0.0
 -0.0        -0.707107  -0.707107     -0.0       -0.0        2.12132
```

2) plot the energy spectrum as a function of angle theta, $\theta \in [0, \pi/2]$, for $N = 8$ spins ¶

Solution

In [5]:

```
N=8  # spin chain size N=10 and Nh = 21 takes approx 10 seconds
N_theta=21  # number of values of magnetic field
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

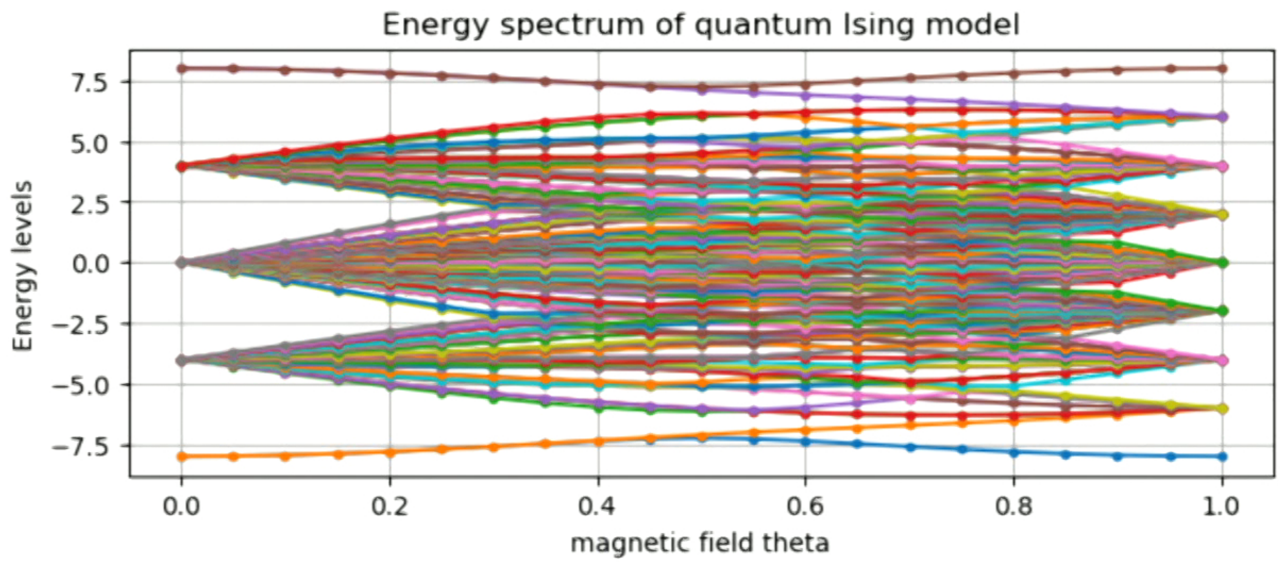
File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Translation operator T, with $[H, T] = 0$ (translation invariance)

In [43]:

```
# Let us quickly rebuild the Hamiltonian for N=3 spins and magnetic field angle t
EE = kron(E,E)
HZ = kron(Z,EE) + kron(E,kron(Z,E)) + kron(EE,Z)
HXX = kron(XX,E)+kron(E,XX)+kron(X,kron(E,X))
H = -(HXX+HZ)/sqrt(2)

# The translation operator T is
SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
N = 3
T = copy(SWAP)
for n=3:N
    SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
    T =  SWAPn*kron(T,E)
end
#display(SWAP)
#display(SWAP^2)
display(T)
#display(T^3)
```

```
title("Energy spectrum of quantum Ising model")
xlabel("magnetic field theta")
ylabel("Energy levels");
```

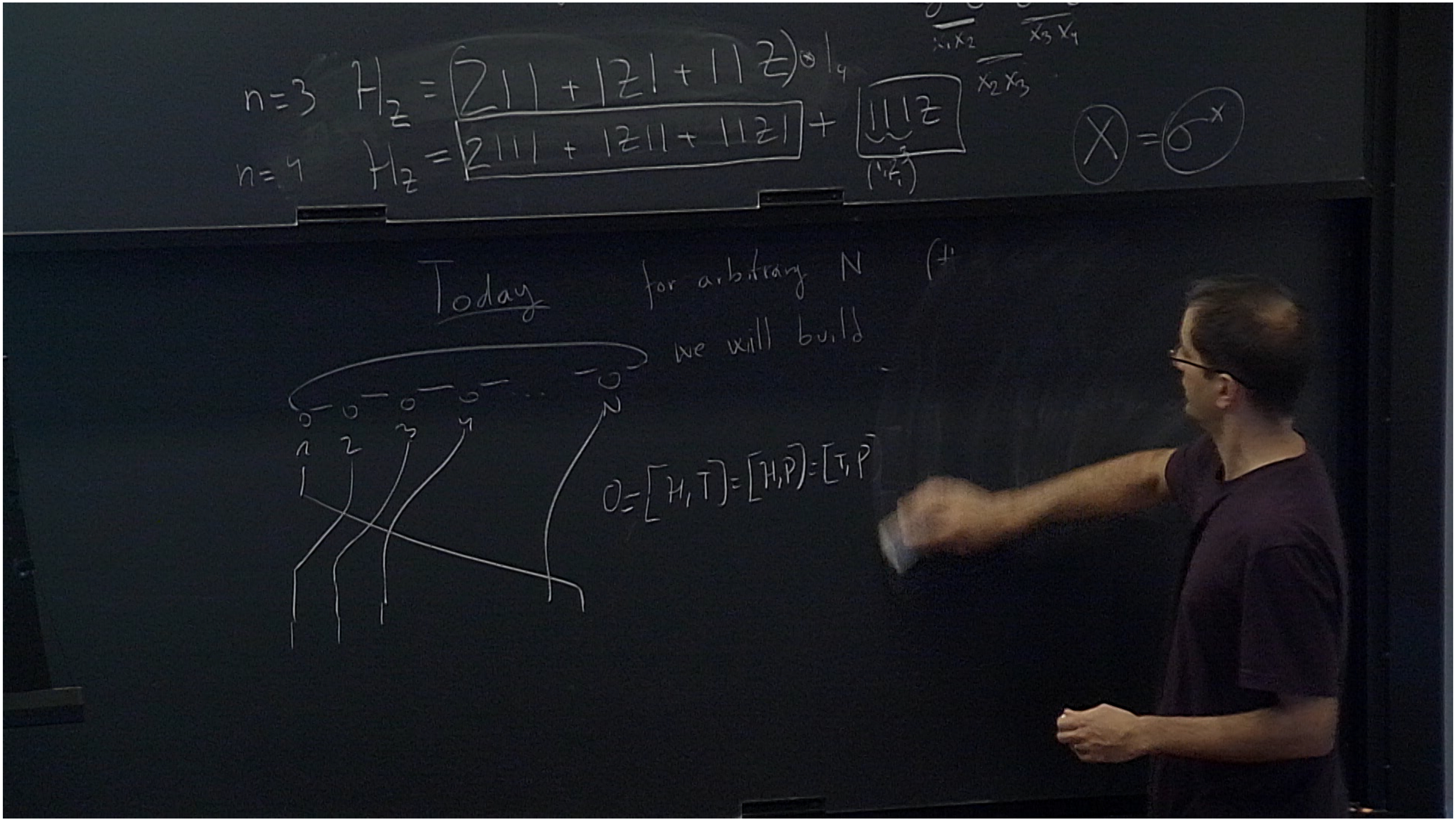


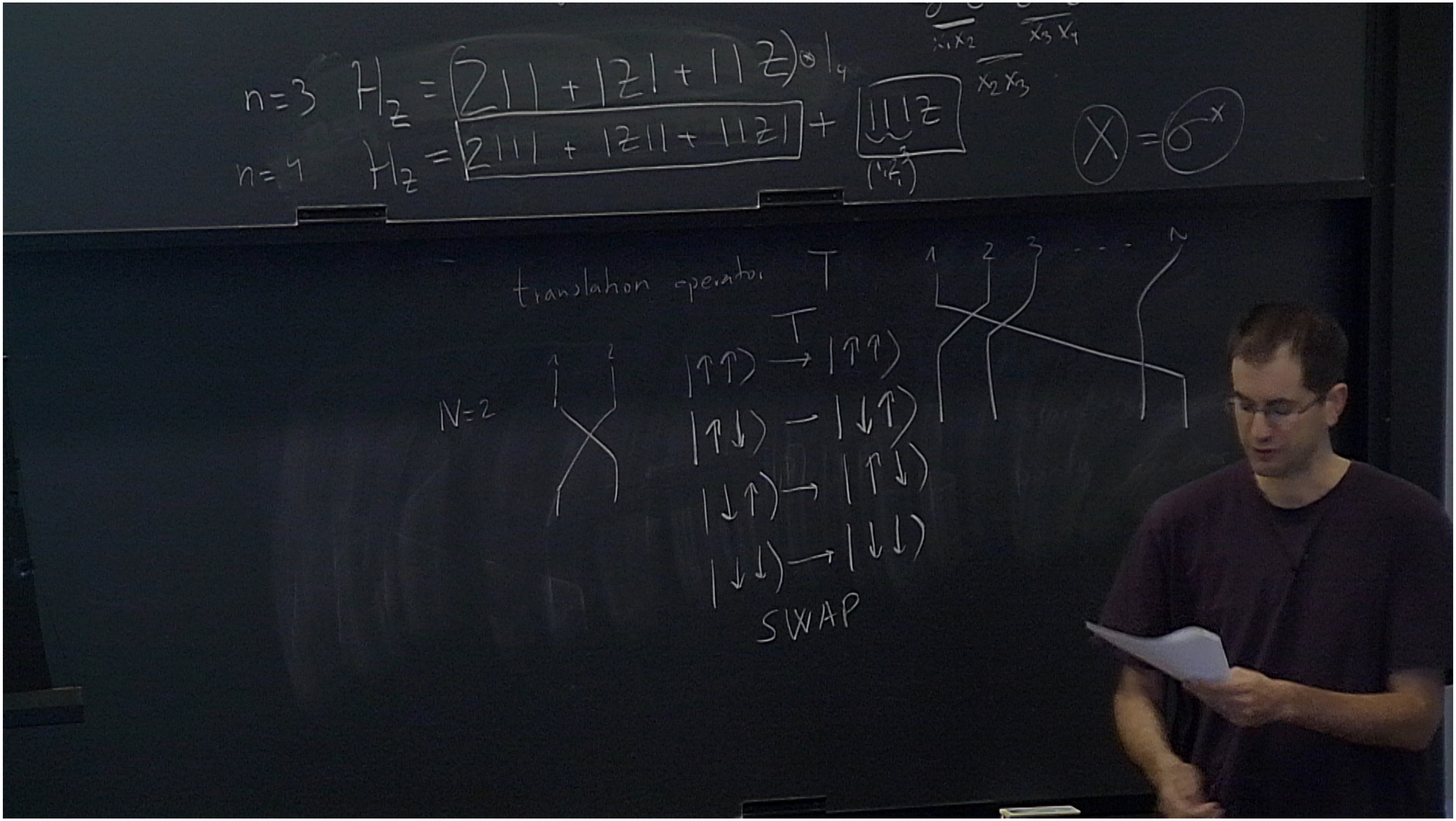
jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

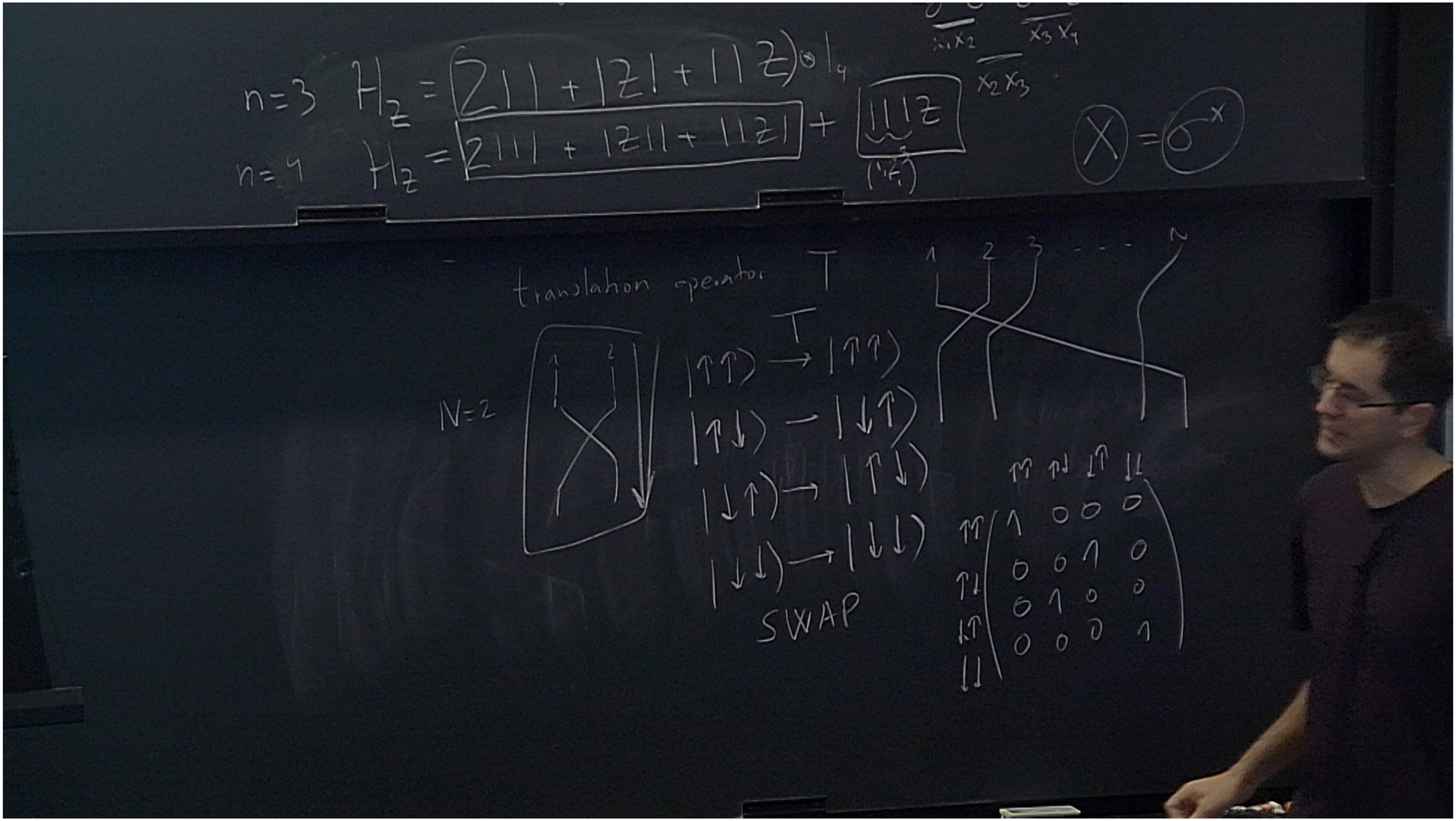
Let us continue:

Translation operator T, with $[H,T] = 0$ (translation invariance)

In [43]:
```julia
# Let us quickly rebuild the Hamiltonian for N=3 spins and magnetic field angle t
EE = kron(E,E)
HZ = kron(Z,EE) + kron(E,kron(Z,E)) + kron(EE,Z)
HXX = kron(XX,E)+kron(E,XX)+kron(X,kron(E,X))
H = -(HXX+HZ)/sqrt(2)

# The translation operator T is
SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
N = 3
T = copy(SWAP)
for n=3:N
    SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
    T =  SWAPn*kron(T,E)
end
#display(SWAP)
```


n=3 H_Z = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 H_Z = Z111 + 1Z11 + 11Z1 + 111Z
Today
for arbitrary N
we will build
0 = [H, T] = [H, P] = [T, P]


n=3
n=4
translation operator T
N=2
SWAP


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1₄
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator T
N=2
|↑↑) → |↑↑)
|↑↓) → |↓↑)
|↓↑) → |↑↓)
|↓↓) → |↓↓)
SWAP
1 2 3 . . . N

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Markdown

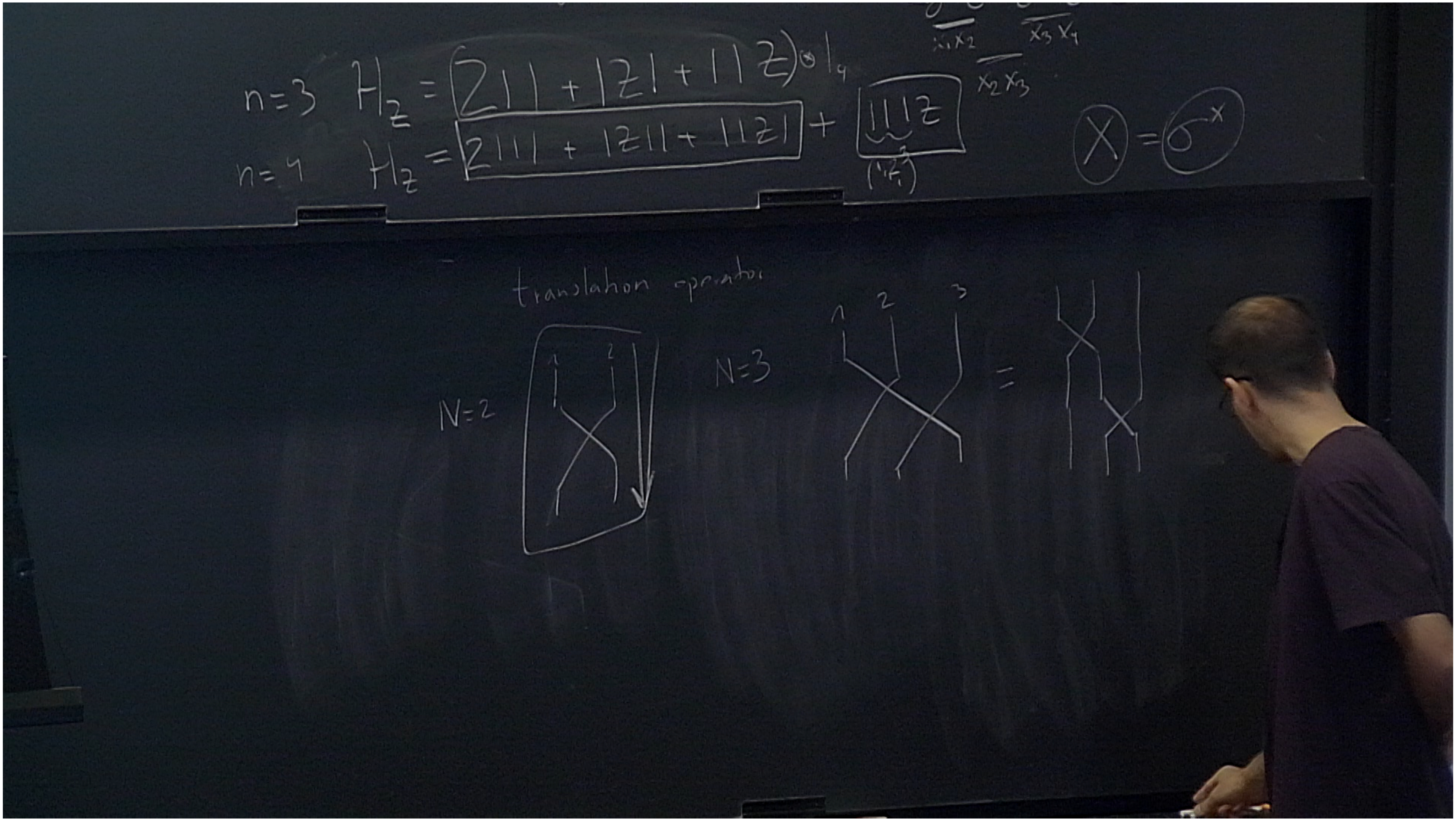
```
end
display(SWAP)
#display(SWAP^2)
#display(T)
#display(T^3)
#display(H3*T-T*H3)
```

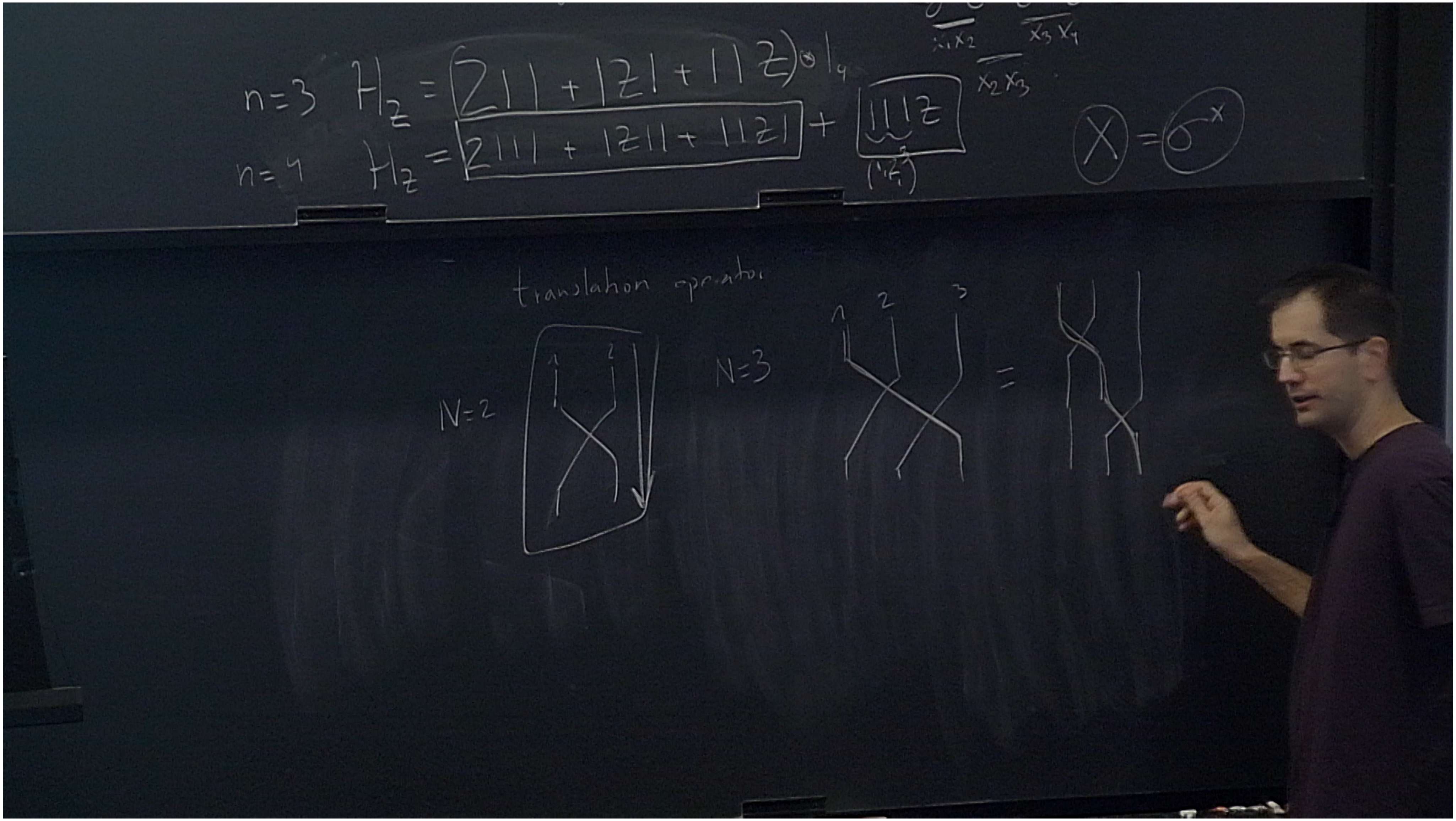
```
4×4 Array{Int64,2}:
 1  0  0  0
 0  0  1  0
 0  1  0  0
 0  0  0  1
```

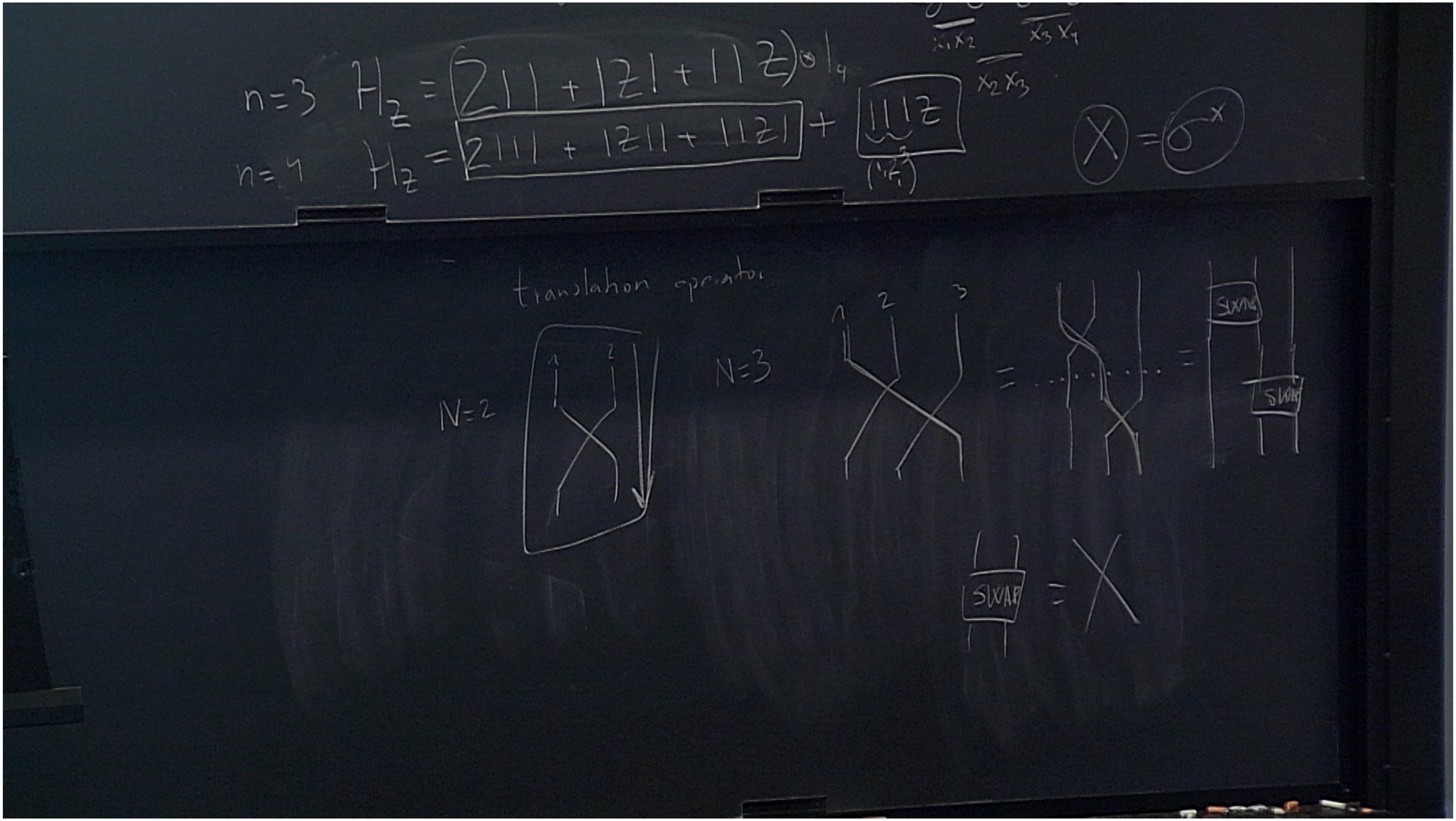
Let us diagonalize H and T simultaneously,

$H|\Psi_i\rangle = e_i|\Psi_i\rangle$ **and** $T|\Psi_i\rangle = \exp(ik_i)|\Psi_i\rangle$

```
In [44]: HT = H + 0.0001*T
```


n=3 $H_Z = (Z11 + 1Z1 + 11Z) \otimes 1_4$
n=4 $H_Z = Z111 + 1Z11 + 11Z1 + 111Z$
$x_1 x_2$
$x_3 x_4$
$x_2 x_3$
X = σ^x
translation operator
N=2
N=3
1 2 3


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1₄
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
translation operator
N=2
N=3
1 2 3

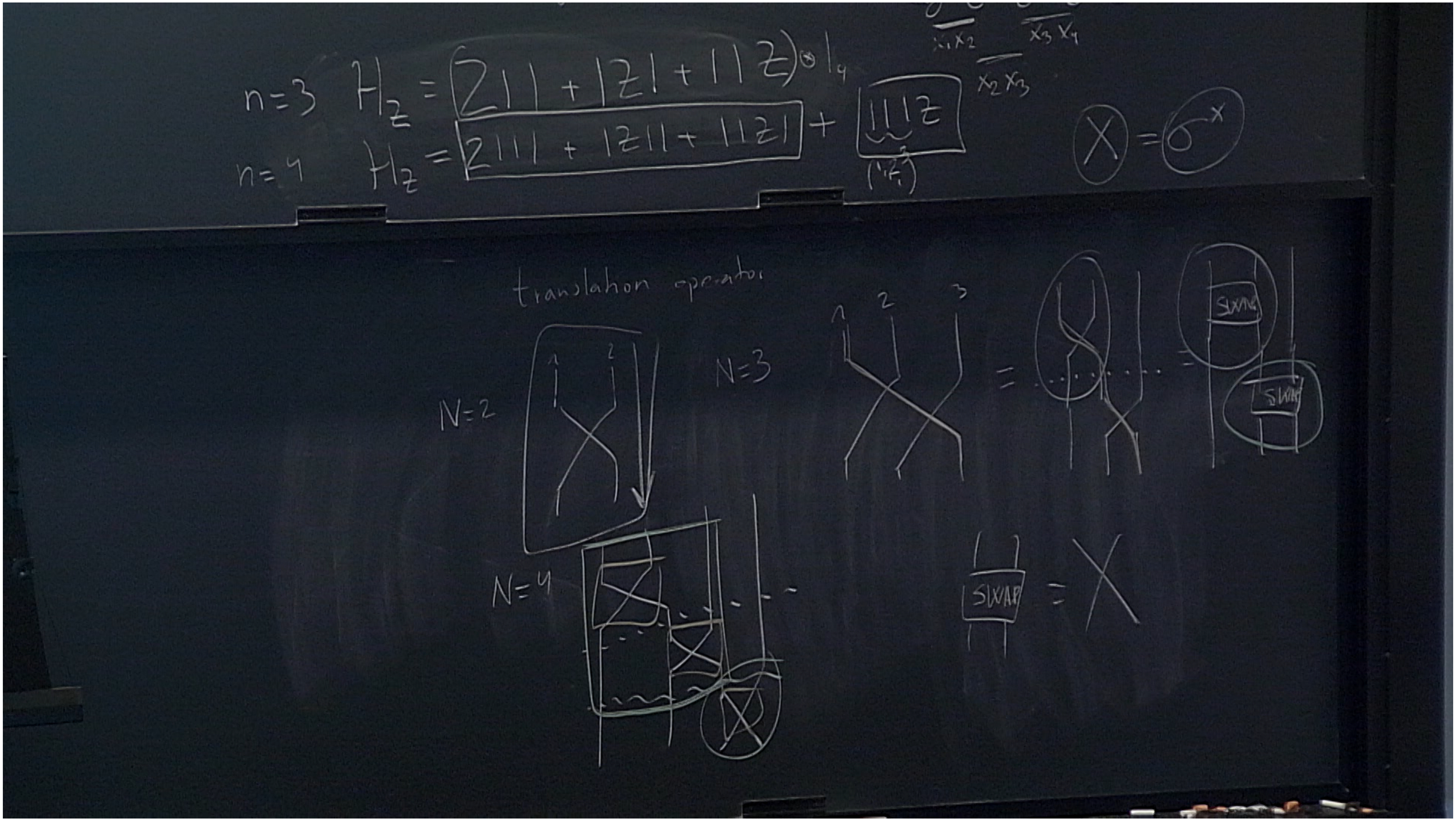

n=3
n=4
translation operator
N=2
N=3
SWAP
SWAP
SWAP

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

```julia
HZ = kron(Z,EE) + kron(E,kron(Z,E)) + kron(EE,Z)
HXX = kron(XX,E)+kron(E,XX)+kron(X,kron(E,X))
H = -(HXX+HZ)/sqrt(2)

# The translation operator T is
SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
N = 3
T = copy(SWAP)
for n=3:N
    SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
    T =  SWAPn*kron(T,E)
end
display(SWAP)
#display(SWAP^2)
#display(T)
#display(T^3)
#display(H3*T-T*H3)
```

```
4×4 Array{Int64,2}:
 1  0  0  0
 0  0  1  0
```



n=3
n=4
translation operator
N=2
N=3
N=4
SWAP

Inbox - gvidal@perimeterinstitut × | Documents/2018juliacourse/Gui × | 2018-computational-physics-co ×

localhost:8888/notebooks/Documents/2018juliacourse/Guifre/2018-computational-physics-course-ExactDiag2-complete.ipynb

Apps | Frequent bookmarks | Bookmarks | Topological Phases | fendley | www.lancaster.ac.uk/ | Symmetry and Topo | David Tong -- Camb | CFT-2014.pdf | AdSCFTCourseNote

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved) Logout

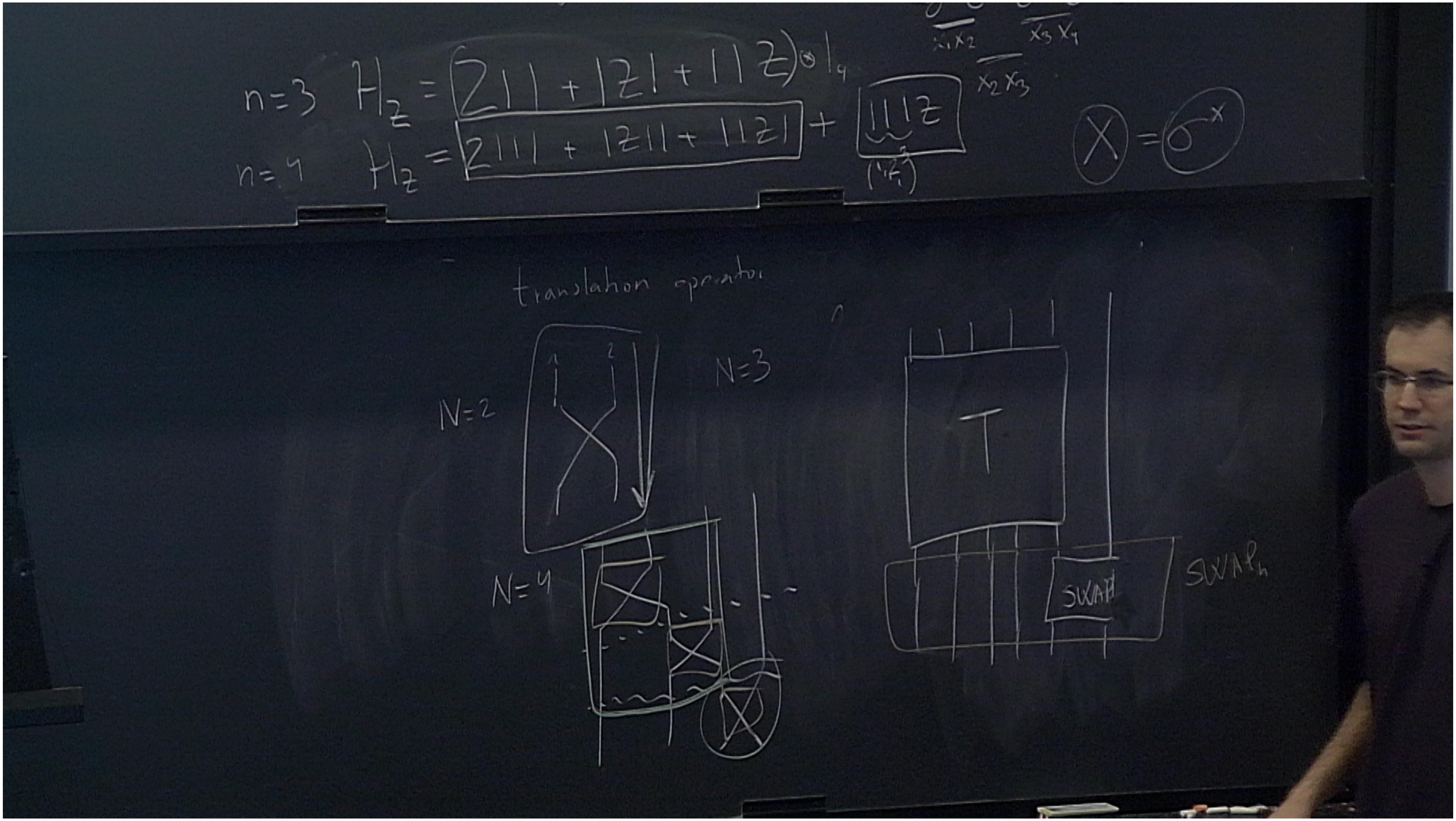
File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

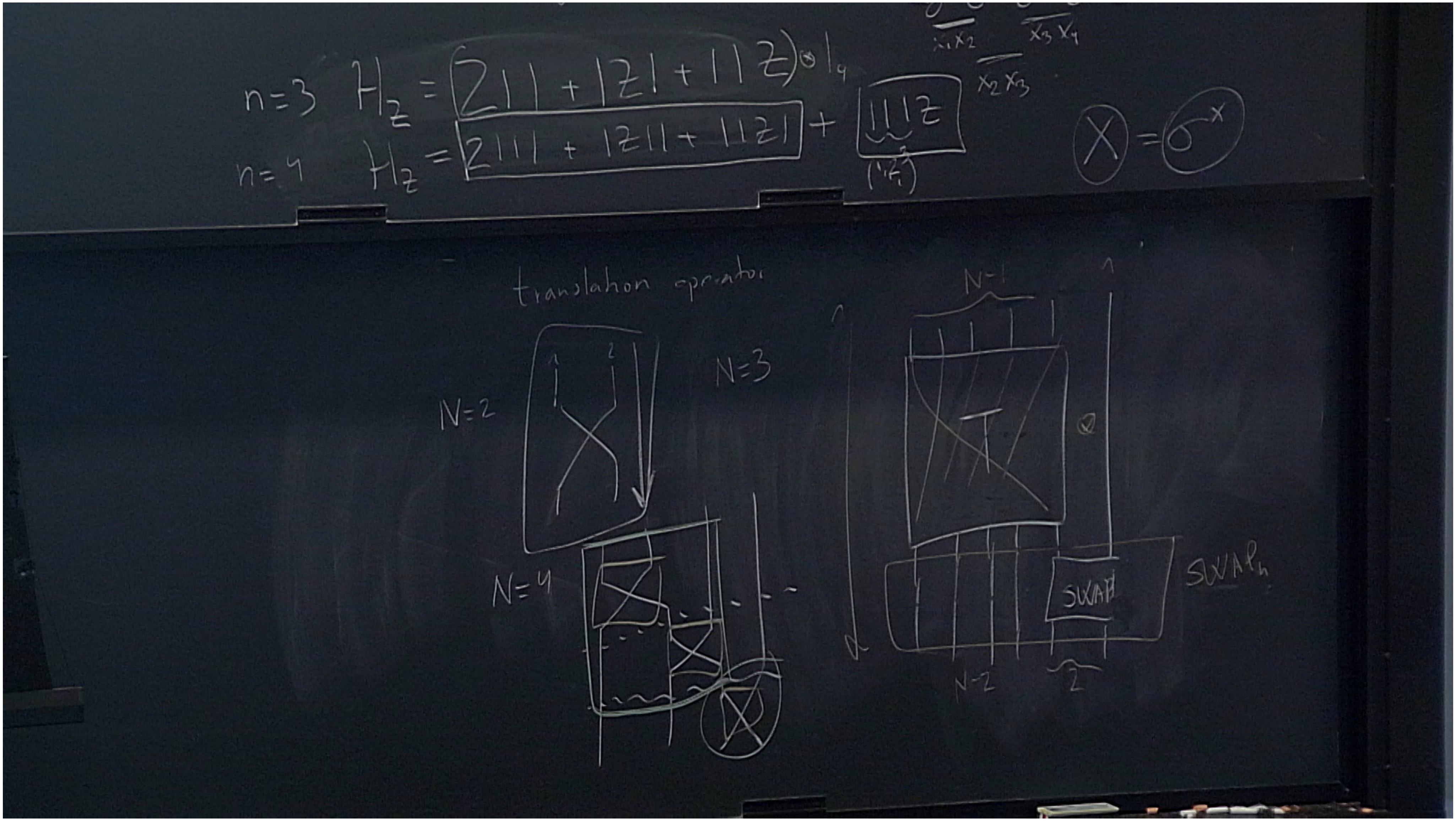
Run Markdown

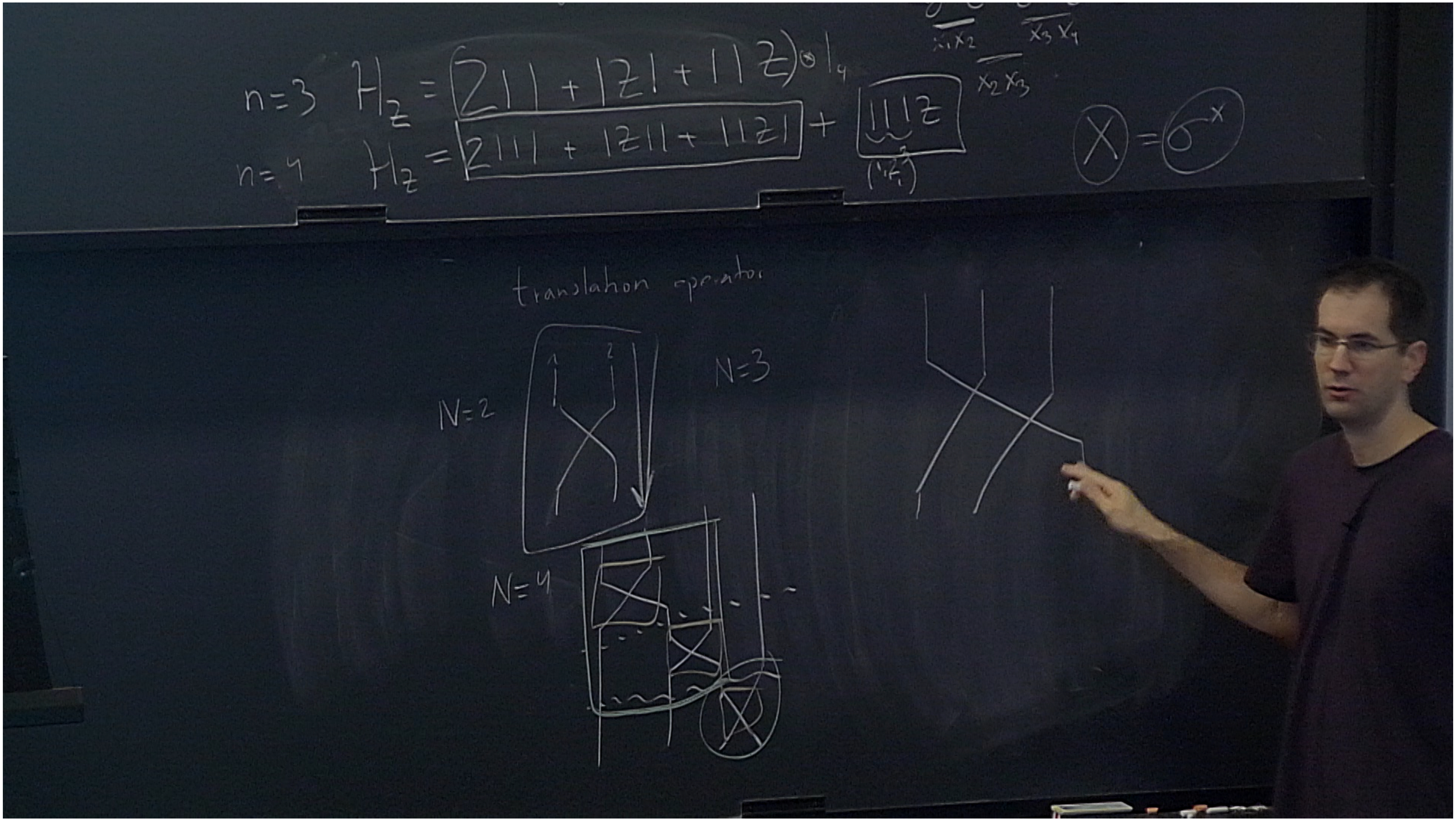
```
HZ = kron(Z,EE) + kron(E,kron(Z,E)) + kron(EE,Z)
HXX = kron(XX,E)+kron(E,XX)+kron(X,kron(E,X))
H = -(HXX+HZ)/sqrt(2)

# The translation operator T is
SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
N = 3
T = copy(SWAP)
for n=3:N
    SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
    T =  SWAPn*kron(T,E)
end
display(SWAP)
#display(SWAP^2)
#display(T)
#display(T^3)
#display(H3*T-T*H3)
```

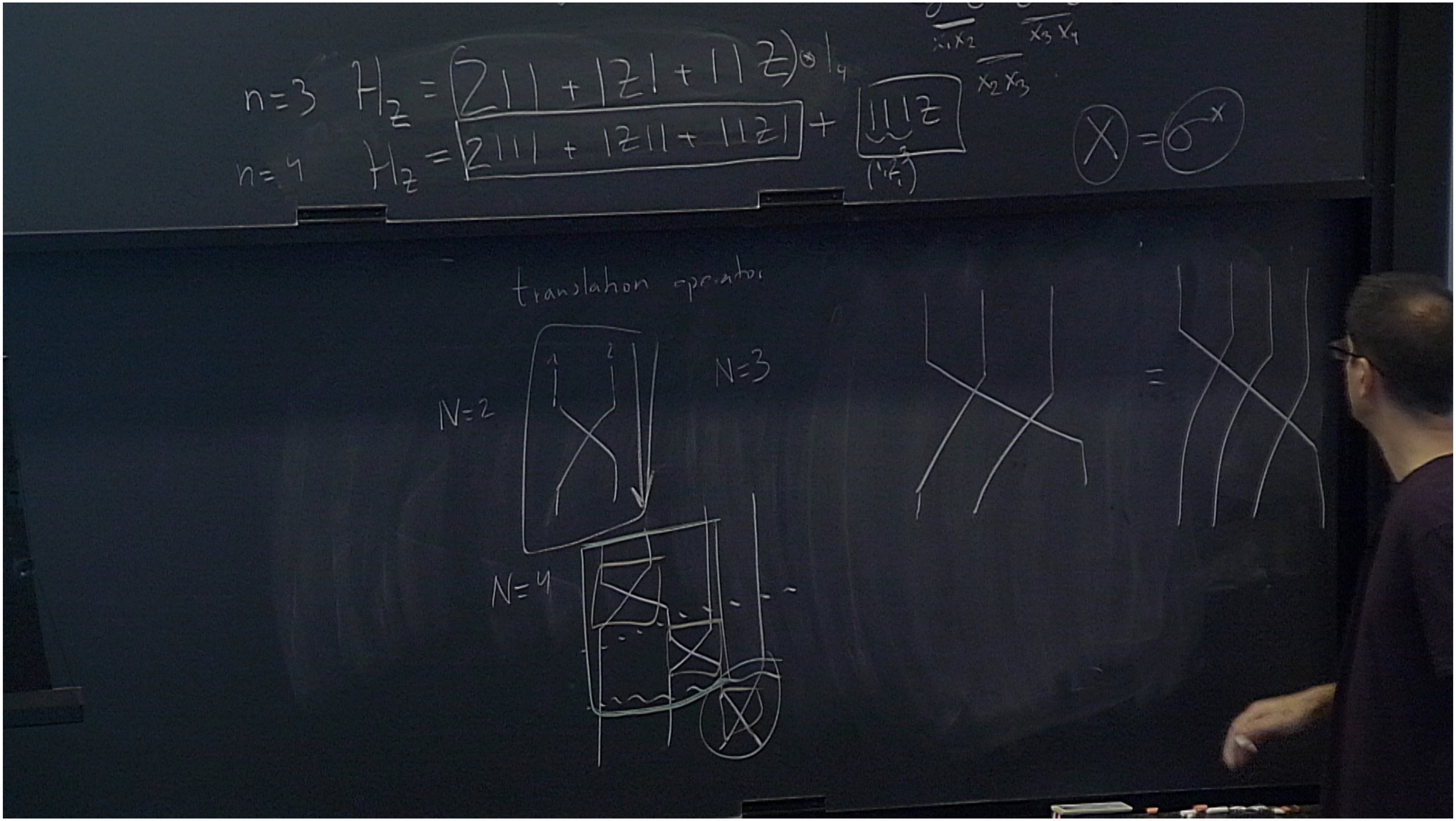
```
4×4 Array{Int64,2}:
 1  0  0  0
 0  0  1  0
```

2:03 PM 16/11/2018

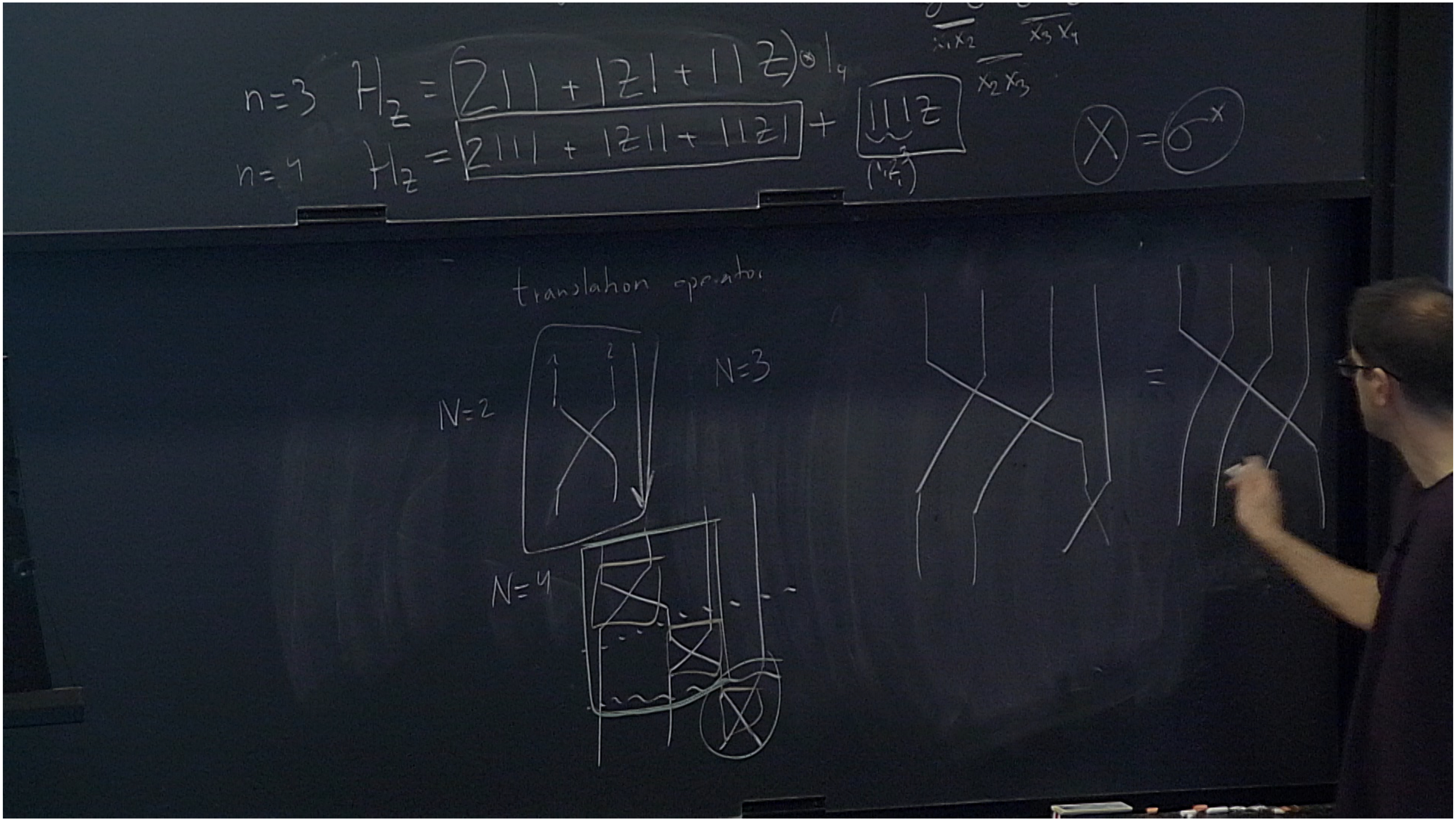

n=3
n=4
translation operator
N=2
N=3
N=4
T
SWAP


n=3 H_Z = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 H_Z = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N=2
N=3
N=4
N-1
N-2
SWAP
SWAP_n


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1₄
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N=2
N=3
N=4



n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
translation operator
N=2
N=3
N=4

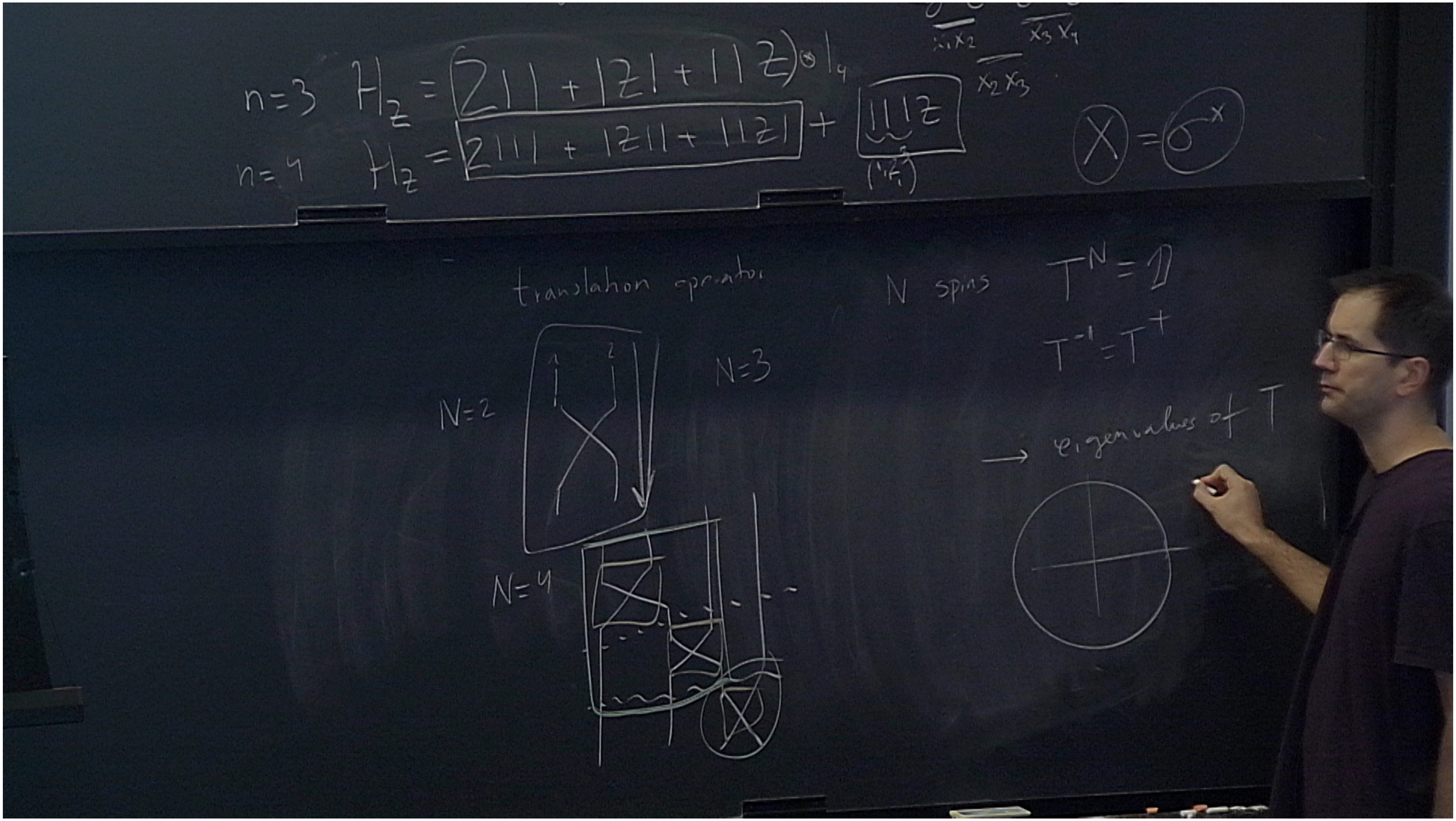

n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σˣ
translation operator
N=2
N=3
N=4

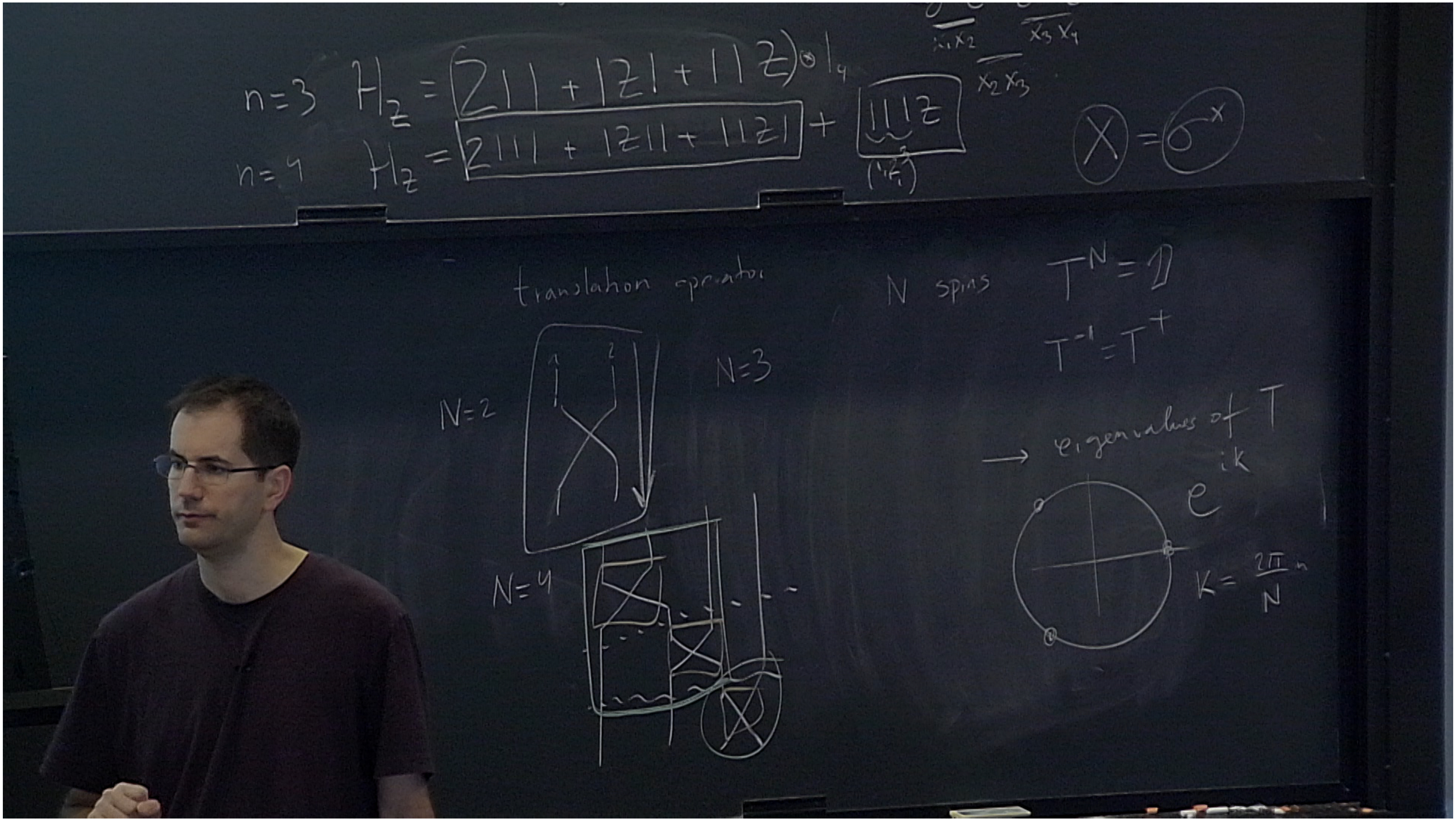
jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

```
display(T^3)
#display(H3*T-T*H3)
```

```
8×8 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
 0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0

8×8 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
```


n=3 H_Z = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 H_Z = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N spins
T^N = 1
T^-1 = T^+
→ eigenvalues of T
N=2
N=3
N=4


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N spins
T^N = 1
T^-1 = T^+
N=2
N=3
N=4
→ eigenvalues of T
e^ik
K = 2π/N n

Inbox (1) - gvidal@perimeterinst × | Documents/2018juliacourse/Gui × | 2018-computational-physics-co ×

localhost:8888/notebooks/Documents/2018juliacourse/Guifre/2018-computational-physics-course-ExactDiag2-complete.ipynb

Apps | Frequent bookmarks | Bookmarks | Topological Phases | fendley | www.lancaster.ac.uk | Symmetry and Topo | David Tong -- Camb | CFT-2014.pdf | AdSCFTCourseNotes

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

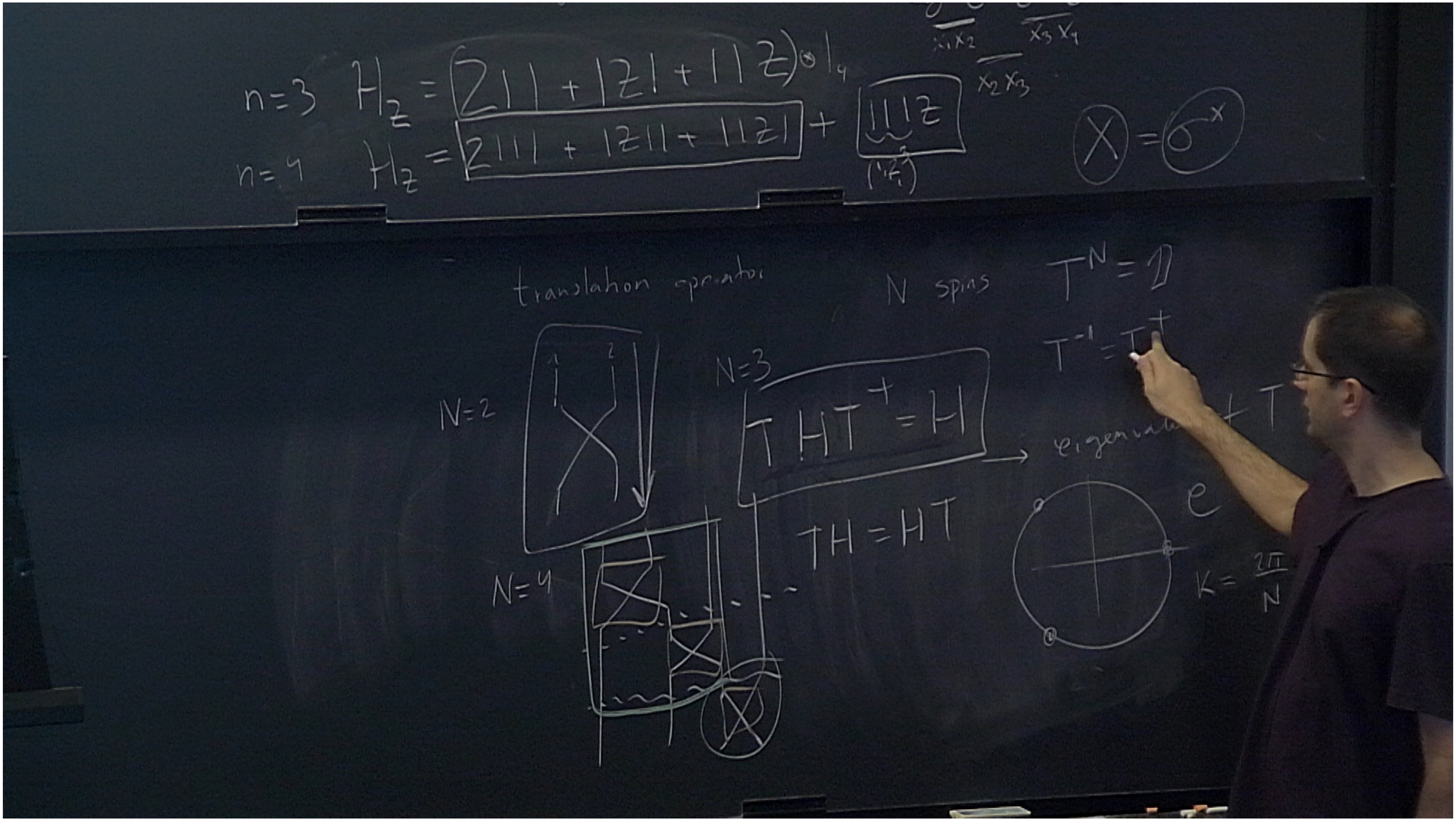
Run | Markdown

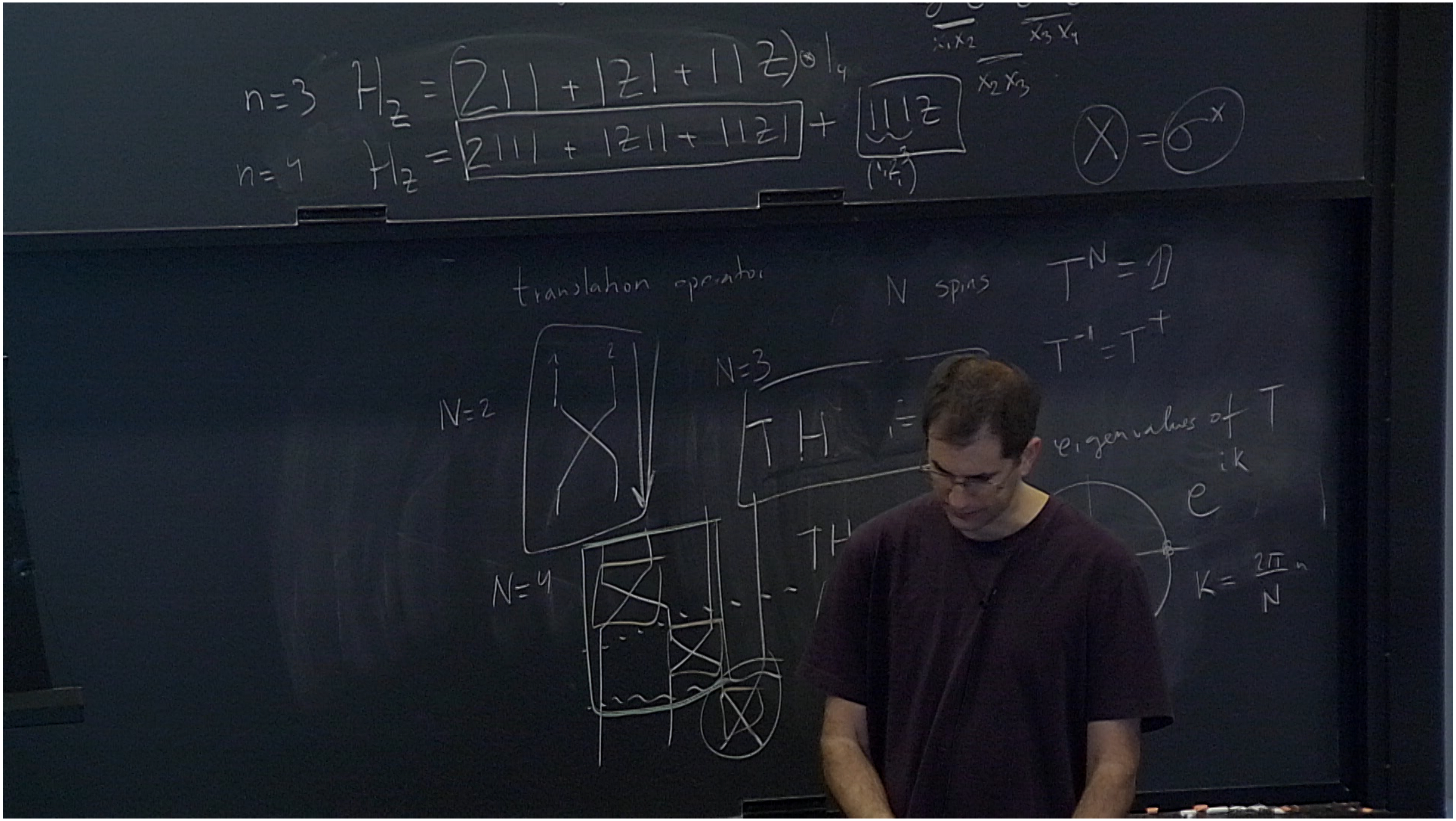
```
    SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
    T =  SWAPn*kron(T,E)
end
#display(SWAP)
#display(SWAP^2)
#display(T)
#display(T^3)
display(H*T-T*H)
```

```
8×8 Array{Float64,2}:
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

Let us diagonalize H and T simultaneously

2:14 PM 16/11/2018


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N spins
T^N = 1
T^-1 = T^†
N=2
N=3
THT^† = H
TH = HT
eigenval
N=4
K = 2π/N


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
translation operator
N spins T^N = 1
T^-1 = T^†
N=2
N=3
N=4
TH
eigenvalues of T
e^ik
K = 2π/N n

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Let us diagonalize H and T simultaneously,

$$H|\Psi_i\rangle = e_i|\Psi_i\rangle \text{ and } T|\Psi_i\rangle = \exp(ik_i)|\Psi_i\rangle$$

In [44]:
```
HT = H + 0.0001*T
D,U = eigen(HT);
```

Let us compute and plot energies e_i as a function of momenta k_i

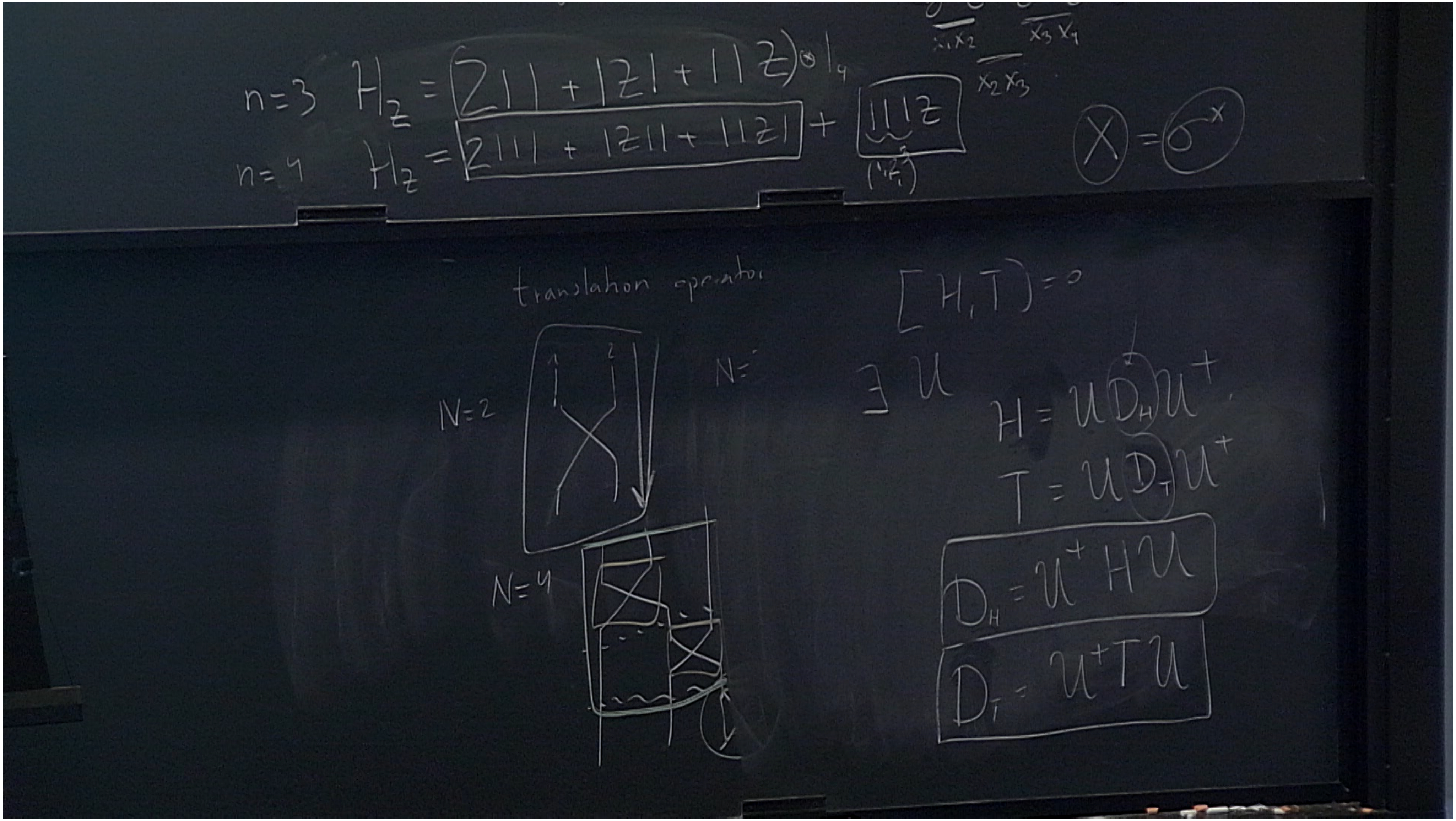
In [45]:
```
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

In [13]:
```julia
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]

[e k]
```

Out[13]:
```
8×2 Array{Float64,2}:
 -2.82843       0.0
 -2.44949       0.0
  0.0           0.0
  2.34557e-31   2.0944
  2.34557e-31  -2.0944
  1.41421       2.0944
  1.41421      -2.0944
  2.44949       0.0
```


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
x1 x2
x3 x4
x2 x3
X = σ^x
translation operator
[H, T] = 0
∃ U
N=2
N=4
H = U D_H U^+
T = U D_T U^+
D_H = U^+ H U
D_T = U^+ T U

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

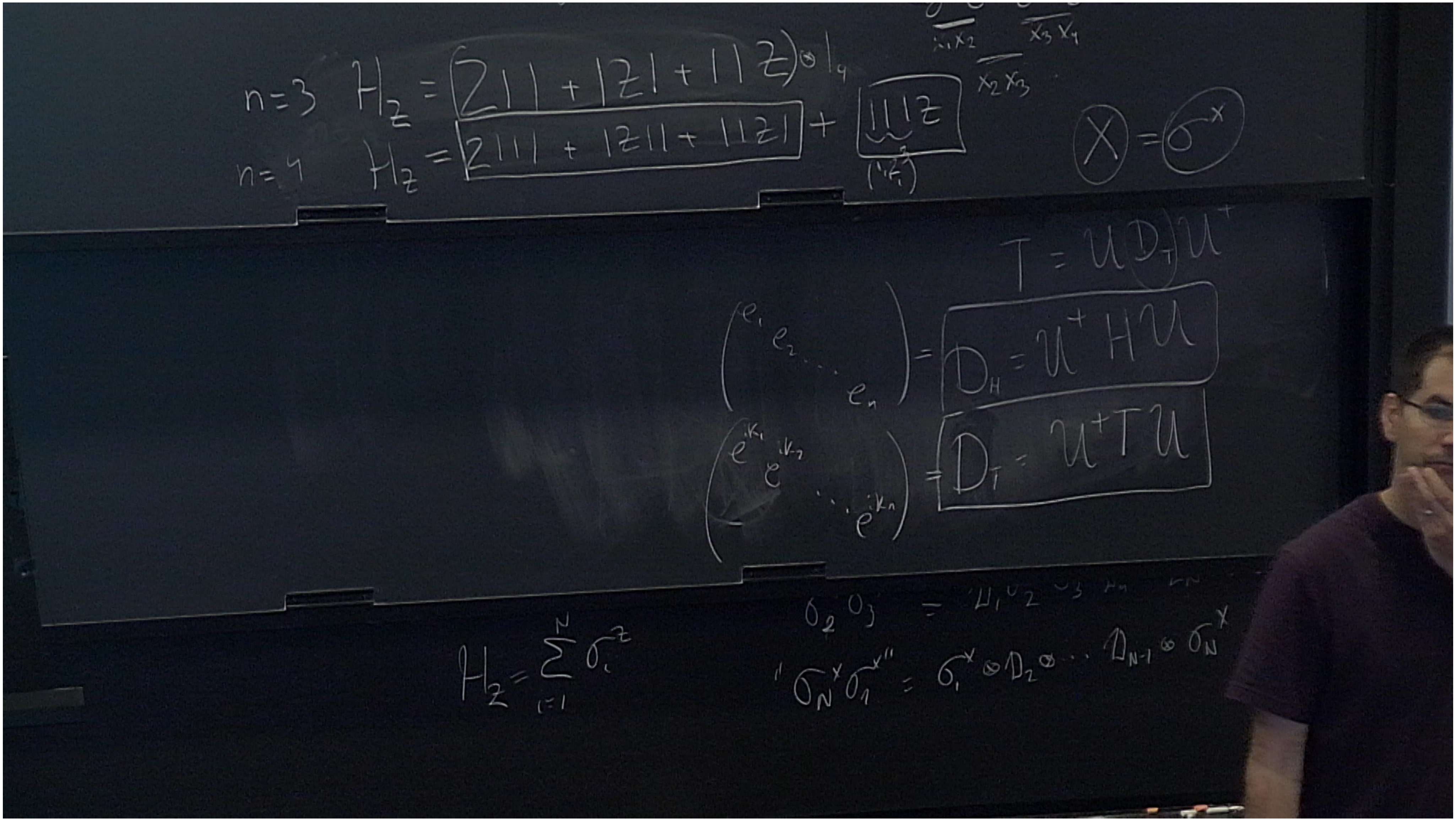
File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Julia 1.0.0

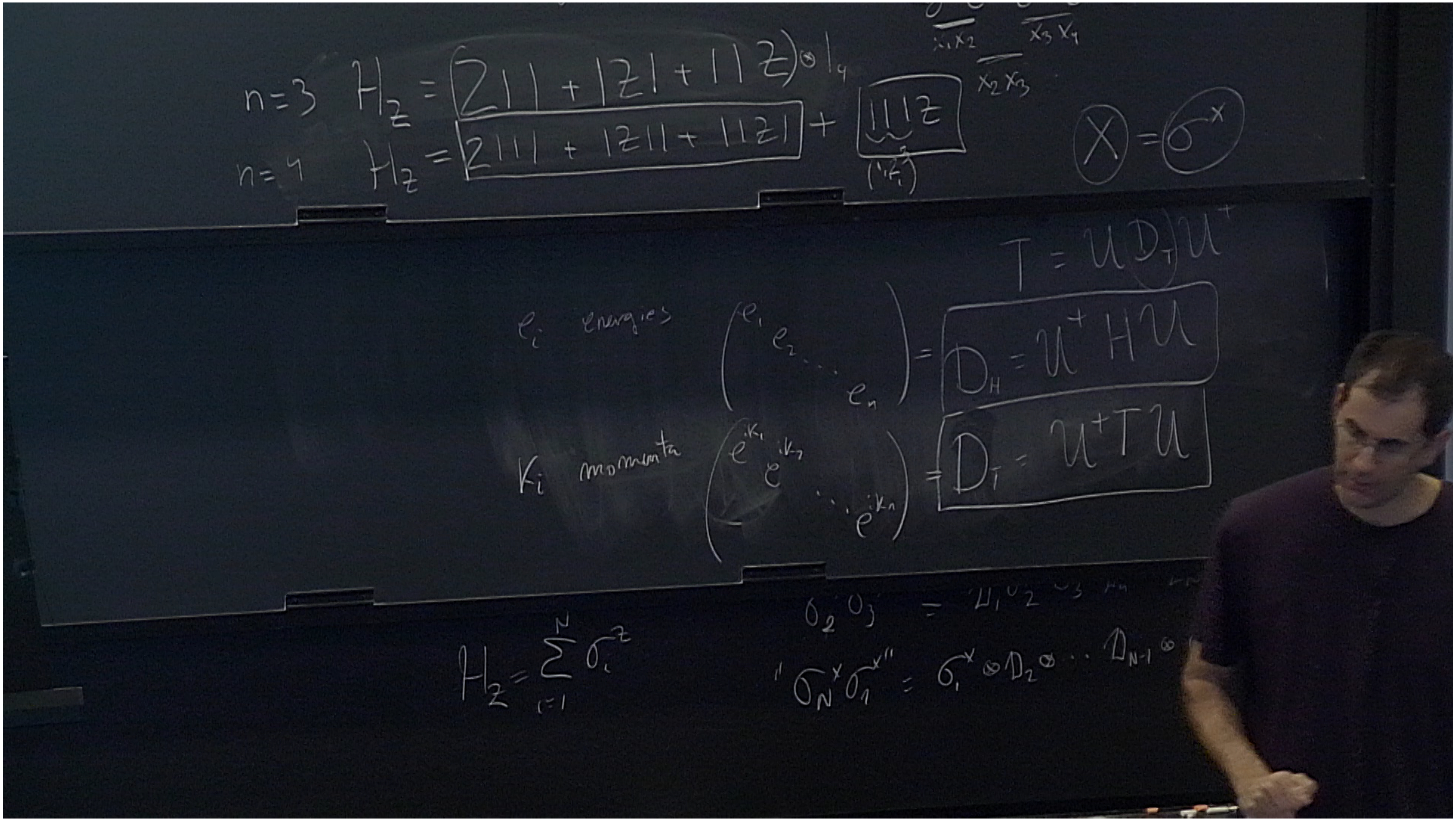
In [13]:

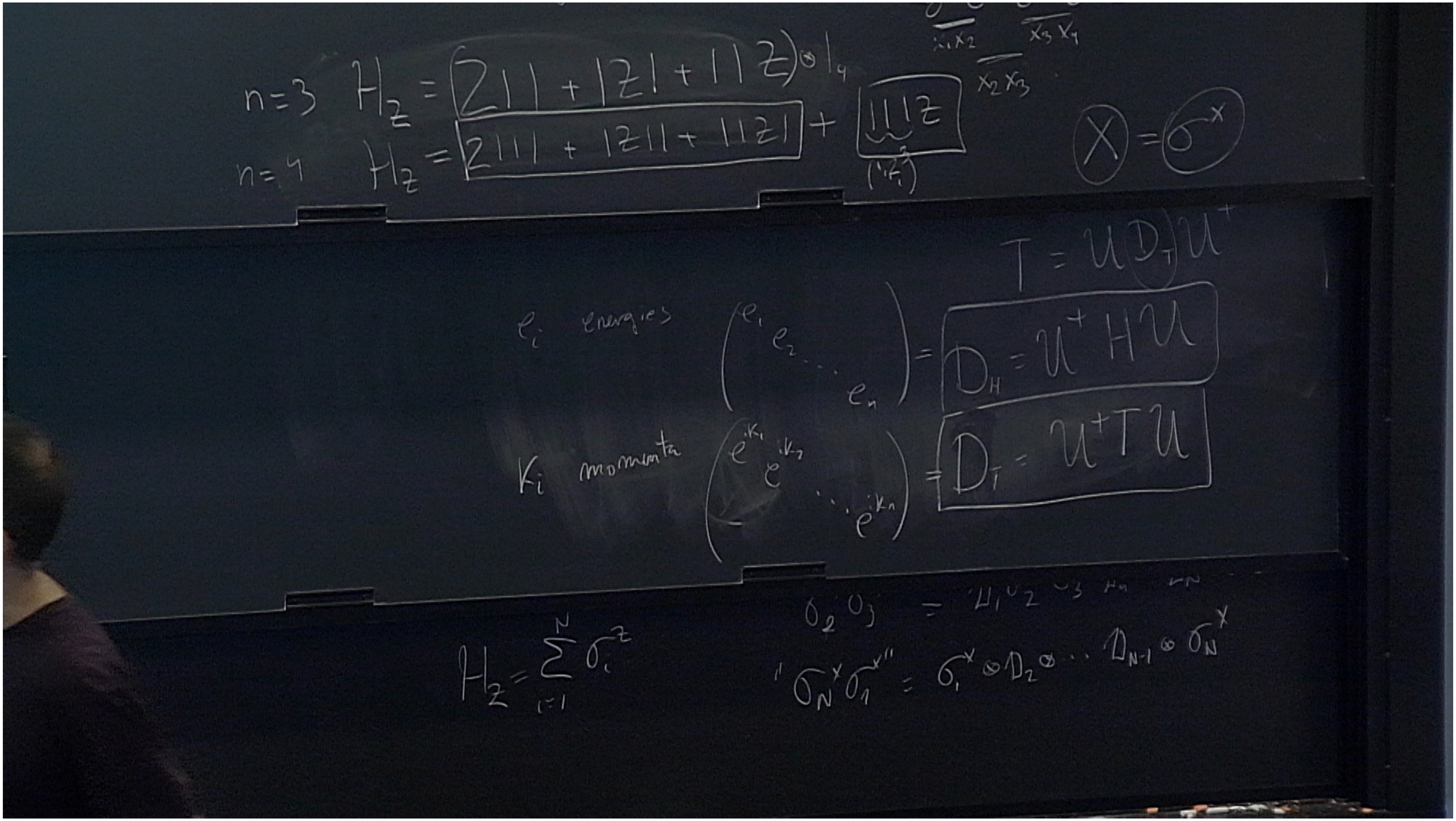
```
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]

[e k]
```

Out[13]:

```
8×2 Array{Float64,2}:
 -2.82843       0.0
 -2.44949       0.0
  0.0           0.0
  2.34557e-31   2.0944
  2.34557e-31  -2.0944
  1.41421       2.0944
  1.41421      -2.0944
  2.44949       0.0
```


n=3 H_Z = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 H_Z = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
T = U D_T U^†
D_H = U^† H U
D_T = U^† T U
H_Z = Σ_{i=1}^N σ_i^z
"σ_N^x σ_1^x" = σ_1^x ⊗ D_2 ⊗ ... D_{N-1} ⊗ σ_N^x


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 14
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
T = U DT U+
ei energies
DH = U+ H U
Ki momenta
DT = U+ T U
Hz = Σ σiz


n=3 Hz = (Z11 + 1Z1 + 11Z) ⊗ 1_4
n=4 Hz = Z111 + 1Z11 + 11Z1 + 111Z
X = σ^x
T = U D_T U^+
e_i energies
D_H = U^+ H U
k_i momenta
D_T = U^+ T U
Hz = Σ_{i=1}^N σ_i^z
"σ_N^x σ_1^x" = σ_1^x ⊗ D_2 ⊗ ... D_{N-1} ⊗ σ_N^x

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Julia 1.0.0

In [13]:

```julia
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]

[e k]
```

Out[13]:

```
8×2 Array{Float64,2}:
 -2.82843       0.0
 -2.44949       0.0
  0.0           0.0
  2.34557e-31   2.0944
  2.34557e-31  -2.0944
  1.41421       2.0944
  1.41421      -2.0944
  2.44949       0.0
```

```
In [13]: e = real(diag(U'*H*U))
         k = angle.(diag(U'*T*U))

         perm = sortperm(e) # re-order eigenvalues according to increasing energies e
         e = e[perm]
         k = k[perm]

         [e k]

Out[13]: 8×2 Array{Float64,2}:
          -2.82843       0.0
          -2.44949       0.0
           0.0           0.0
           2.34557e-31   2.0944
           2.34557e-31  -2.0944
           1.41421       2.0944
           1.41421      -2.0944
           2.44949       0.0
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```
e = e[perm]
k = k[perm]

[e k]
perm
```

Out[15]:

```
8-element Array{Int64,1}:
 1
 3
 2
 5
 6
 7
 8
 4
```

In [46]:

```
figure("low_energies",figsize=(8,3))

grid("on") # Create a grid on the axis
title("low energies spin chain")
```

Inbox (4) - gvidal@perimeterinst | Documents/2018juliacourse/Gui | 2018-computational-physics-co

localhost:8888/notebooks/Documents/2018juliacourse/Guifre/2018-computational-physics-course-ExactDiag2-complete.ipynb

Apps | Frequent bookmarks | Bookmarks | Topological Phases | fendley | www.lancaster.ac.uk/ | Symmetry and Topo | David Tong -- Camb | CFT-2014.pdf | AdSCFTCourseNotes

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Code

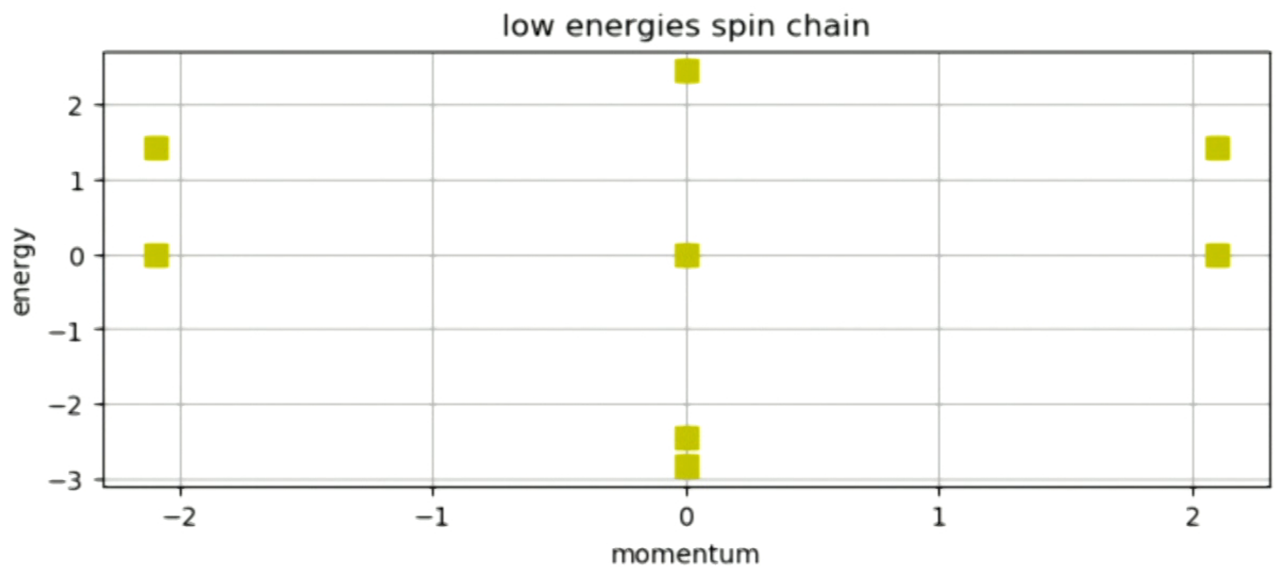
```
[e k]
```

Out[16]:
```
8×2 Array{Float64,2}:
 -2.82843       0.0
 -2.44949       0.0
  0.0           0.0
  2.34557e-31   2.0944
  2.34557e-31  -2.0944
  1.41421       2.0944
  1.41421      -2.0944
  2.44949       0.0
```

In [46]:
```
figure("low_energies",figsize=(8,3))

grid("on") # Create a grid on the axis
title("low energies spin chain")
#ax = gca()
#ax[:set_xlim]([-3,3])
#ax[:set_ylim]([-0.2,2.8])
```

2:24 PM 16/11/2018

```
    plot(k[i],e[i], marker = "s", markersize = 8, color = "y")
end
```



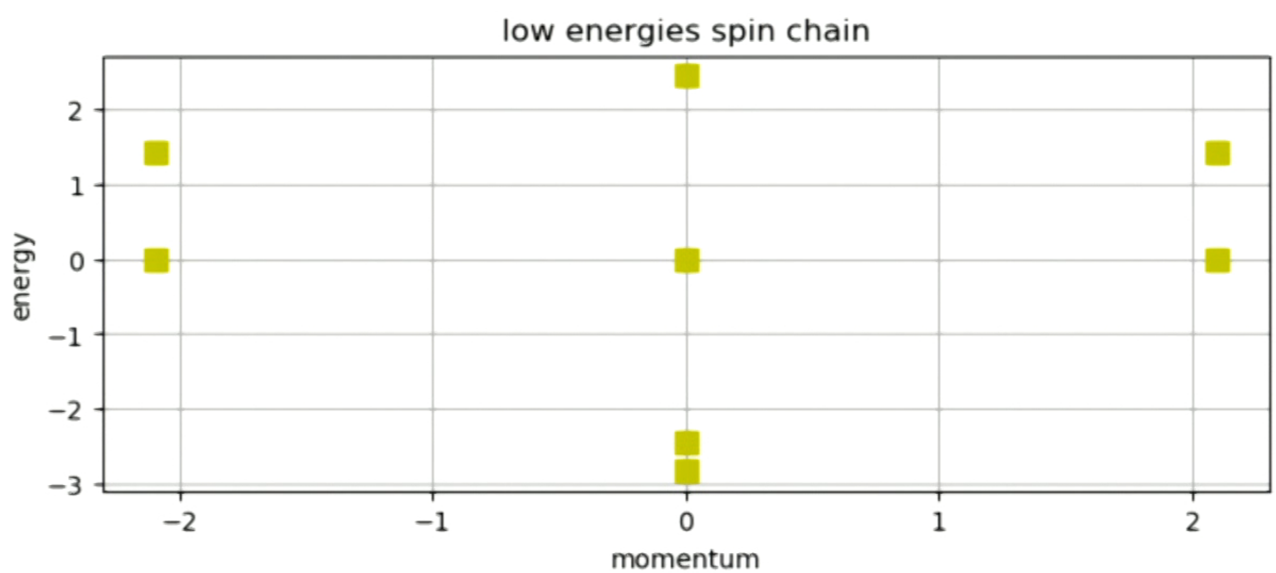
jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Markdown

```
    plot(k[i],e[i], marker = "s", markersize = 8, color = "y")
end
```



jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```
[e k]
```

```
8-element Array{Complex{Float64},1}:
   -2.8284271247461885 + 0.0im
                   0.0 + 0.0im
    -2.449489742783178 + 0.0im
    2.4494897427831788 + 0.0im
  2.345571399448979e-31 + 3.799215150296093 5e-33im
  2.345571399448979e-31 - 3.7992151502960935e-33im
     1.414213562373095 + 8.338914357139747e-18im
     1.414213562373095 - 8.338914357139747e-18im
```

Out[18]:
```
8×2 Array{Float64,2}:
 -2.82843       0.0
 -2.44949       0.0
  0.0           0.0
  2.34557e-31   2.0944
  2.34557e-31  -2.0944
  1.41421       2.0944
  1.41421      -2.0944
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

In [12]:
```julia
HT = H + 0.0001*T
D,U = eigen(HT);
```

Let us compute and plot energies e_i as a function of momenta k_i

In [18]:
```julia
display(diag(U'*H*U))
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]

[e k]
```

```
8-element Array{Complex{Float64},1}:
  2.8284271247461885 + 0.0im
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

```
In [12]: HT = H + 0.0001*T
         D,U = eigen(HT);
```

Let us compute and plot energies e_i as a function of momenta k_i

```
In [18]:
         e = real(diag(U'*H*U))
         k = angle.(diag(U'*T*U))

         perm = sortperm(e) # re-order eigenvalues according to increasing energies e
         e = e[perm]
         k = k[perm]

         [e k]
```

```
8-element Array{Complex{Float64},1}:
 -2.8284271247461885 + 0.0im
```

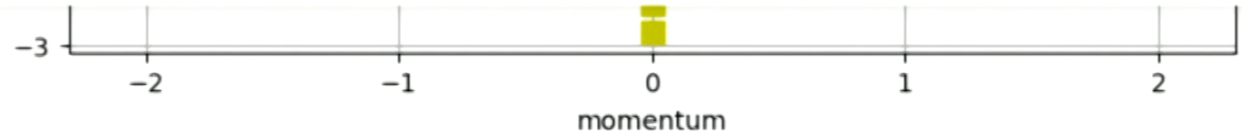
jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

```
In [19]: HT = H + 0.0001*T
         D,U = eigen(HT);
         D
```

```
Out[19]: 8-element Array{Complex{Float64},1}:
             -2.8283271247461896 + 0.0im
            9.999999999987796e-5 + 0.0im
             -2.4493897427831772 + 0.0im
              2.4495897427831794 + 0.0im
          -4.9999999999968974e-5 + 8.66025403783867e-5im
          -4.9999999999968974e-5 - 8.66025403783867e-5im
              1.4141635623730955 + 8.660254037823334e-5im
              1.4141635623730955 - 8.660254037823334e-5im
```

Let us compute and plot energies e_i as a function of momenta k_i

```
In [18]: e = real(diag(U'*H*U))
         k = angle.(diag(U'*T*U))
```



It's your turn again! (2/3)

3) write a function buildT that builds the translation operator T for N spins (use only for $N \leq 10$)

In [9]:
```
# function buildT takes as input N (number of spins) and outputs T (translatio op
function buildT(N::Int64)::Array{Float64,2}
    SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
    E = diagm(0=>ones(2))
    T = copy(SWAP)
    for n=3:N
        SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
        T = SWAPn*kron(T,E)
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

```
0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

4) compute and plot energies e_i as a function of momenta k_i for $N = 8$ spins and $\theta = \pi/4$

In [10]:
```
N=8
H = buildH(N,pi/4)
T = buildT(N)
HT = H + 0.0001*T
D,U = eigen(HT);
```

In [13]:
```
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))

perm = sortperm(e) # re-order eigenvalues according to increasing energies e
e = e[perm]
k = k[perm]
[e k];
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

```
end

buildT(3) # test for N=3 and critical angle pi/4
```

Out[9]:

```
8×8 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
 0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

4) compute and plot energies e_i as a function of momenta k_i for $N = 8$ spins and $\theta = \pi/4$

In [10]:

```
N=8
H = buildH(N,pi/4)
```

jupyter 2018-computational-physics-course-ExactDiag… (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Run Markdown

spins (use only for $N \le 10$)

In [20]:

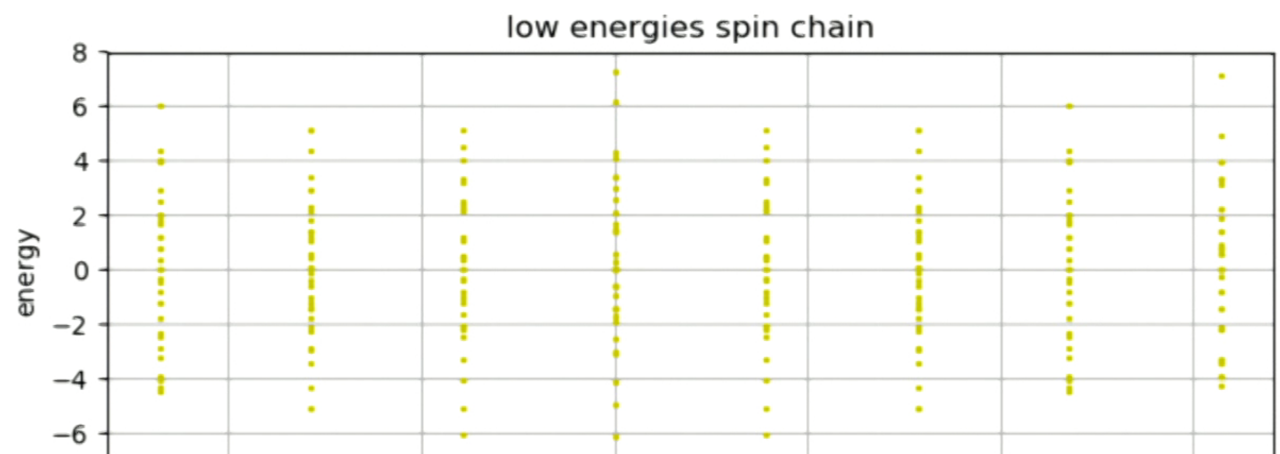
```julia
# function buildT takes as input N (number of spins) and outputs T (translatio op
function buildT(N::Int64)::Array{Float64,2}
    SWAP = [1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]
    E = diagm(0=>ones(2))
    T = copy(SWAP)
    for n=3:N
        SWAPn = kron(diagm(0=>ones(2^(n-2))), SWAP)
        T =  SWAPn*kron(T,E)
    end
    return T
end

buildT(3) # test for N=3 and critical angle pi/4
```

Out[20]:

```
8×8 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

```
#ax = gca()
#ax[:set_xlim]([-3,3])
#ax[:set_ylim]([-0.2,2.8])
xlabel("momentum")
ylabel("energy")
for i in 1:size(e,1)
    plot(k[i],e[i], marker = "s", markersize = 2, color = "y")
end
```



Let us continue:

We add the parity operator P, with $[H, P] = [T, P] = 0$ (parity conservation)

In [23]:
```
# Let us quickly rebuild the Hamiltonian for N=3 spins and magnetic field angle t
EE = kron(E,E)
HZ = kron(Z,EE) + kron(E,kron(Z,E)) + kron(EE,Z)
HXX = kron(XX,E)+kron(E,XX)+kron(X,kron(E,X))
H = -(HXX+HZ)/sqrt(2)
# ...and the translation operator for N=3
T = kron(E,SWAP)*kron(SWAP,E)

# The parity operator P is
Z = [1 0; 0 -1]
N=3
P = copy(Z)
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Julia 1.0.0

```
display(P)
#display(P^2)
#display(norm(H3*P-P*H3))
#display(norm(T*P-P*T))
```

```
8×8 Array{Int64,2}:
 1   0   0  0   0  0  0   0
 0  -1   0  0   0  0  0   0
 0   0  -1  0   0  0  0   0
 0   0   0  1   0  0  0   0
 0   0   0  0  -1  0  0   0
 0   0   0  0   0  1  0   0
 0   0   0  0   0  0  1   0
 0   0   0  0   0  0  0  -1
```

Let us diagonalize H, T, and P simultaneously,

$H|\Psi_i\rangle = e_i|\Psi_i\rangle$, $T|\Psi_i\rangle = \exp(ik_i)|\Psi_i\rangle$, **and** $Z|\Psi_i\rangle = (-1)^p|\Psi_i\rangle$

```
0   0   0  0   0  0  0  -1
```

Let us diagonalize H, T, and P simultaneously,

$$H|\Psi_i\rangle = e_i|\Psi_i\rangle, \quad T|\Psi_i\rangle = \exp(ik_i)|\Psi_i\rangle, \text{ and } \quad Z|\Psi_i\rangle = (-1)^p|\Psi_i\rangle$$

In [48]:
```
HTP = H + 0.0001*T + im*0.01*P
D,U = eigen(HTP)
```

Out[48]:
```
Eigen{Complex{Float64},Complex{Float64},Array{Complex{Float64},2},Array{Complex
{Float64},1}}
eigenvalues:
8-element Array{Complex{Float64},1}:
      -2.82832712474619 + 0.01im
   9.999999999990363e-5 + 0.010000000000000002im
     -2.4493897427831812 - 0.009999999999999998im
      2.4495897427831794 - 0.01im
  -4.999999999980888e-5 - 0.010086602540378445im
```

localhost:8888/notebooks/Documents/2018juliacourse/Guifre/2018-computational-physics-course-ExactDiag2-complete.ipynb

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

Let us compute and plot energies e_i, momenta k_i, and parity p_i

In [50]:

```
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))
p = real(diag(U'*P*U))
p = round.((1 .- p)/2)
perm = sortperm(e) # re-order eigenvalues according to E
e = e[perm]
k = k[perm]
p = p[perm]
[e k p]
```

Out[50]:

```
8×3 Array{Float64,2}:
 -2.82843        0.0          0.0
 -2.44949        0.0          1.0
 -1.13883e-31    2.0944       1.0
 -5.54732e-32    0.0          0.0
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

Let us compute and plot energies e_i, momenta k_i, and parity p_i

In [50]:

```
e = real(diag(U'*H*U))
k = angle.(diag(U'*T*U))
p = real(diag(U'*P*U))
p = round.((1 .- p)/2)
perm = sortperm(e) # re-order eigenvalues according to E
e = e[perm]
k = k[perm]
p = p[perm]
[e k p]
```

Out[50]:

```
8×3 Array{Float64,2}:
 -2.82843      0.0       0.0
 -2.44949      0.0       1.0
 -1.13883e-31  2.0944    1.0
 -5.54732e-32  0.0       0.0
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

```
DimensionMismatch("matrix A has dimensions (256,256), matrix B has dimensions (8,8)")

Stacktrace:
 [1] _generic_matmatmul!(::Array{Complex{Float64},2}, ::Char, ::Char, ::Array{Complex{Float64},2}, ::Array{Float64,2}) at C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.0\LinearAlgebra\src\matmul.jl:588
 [2] generic_matmatmul!(::Array{Complex{Float64},2}, ::Char, ::Char, ::Array{Complex{Float64},2}, ::Array{Float64,2}) at C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.0\LinearAlgebra\src\matmul.jl:578
 [3] mul! at C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.0\LinearAlgebra\src\matmul.jl:290 [inlined]
 [4] * at C:\cygwin\home\Administrator\buildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.0\LinearAlgebra\src\matmul.jl:141 [inlined]
 [5] *(::Adjoint{Complex{Float64},Array{Complex{Float64},2}}, ::Array{Float64,2}, ::Array{Complex{Float64},2}) at .\operators.jl:502
 [6] top-level scope at In[28]:1
```

jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```
Out[31]: 8×3 Array{Float64,2}:
 -2.82843       0.0           0.0
 -2.44949       0.0           1.0
 -1.13883e-31   2.0944        1.0
 -5.54732e-32   0.0           0.0
 -4.32111e-32  -2.0944        1.0
  1.41421       2.0944        0.0
  1.41421      -2.0944       -0.0
  2.44949      -3.85186e-32   1.0
```

```
In [56]: figure("low_energies",figsize=(8,3))

subplot(121) # Create the 1st axis of a 2x2 arrax of axes
grid("on") # Create a grid on the axis
title("Even Parity, p=0")
xlabel("momentum")
ylabel("energy")

subplot(122) # Create the 1st axis of a 2x2 arrax of axes
grid("on") # Create a grid on the axis
title("Even Parity, p=1")
```

jupyter 2018-computational-physics-course-ExactDiag2-co… (autosaved)

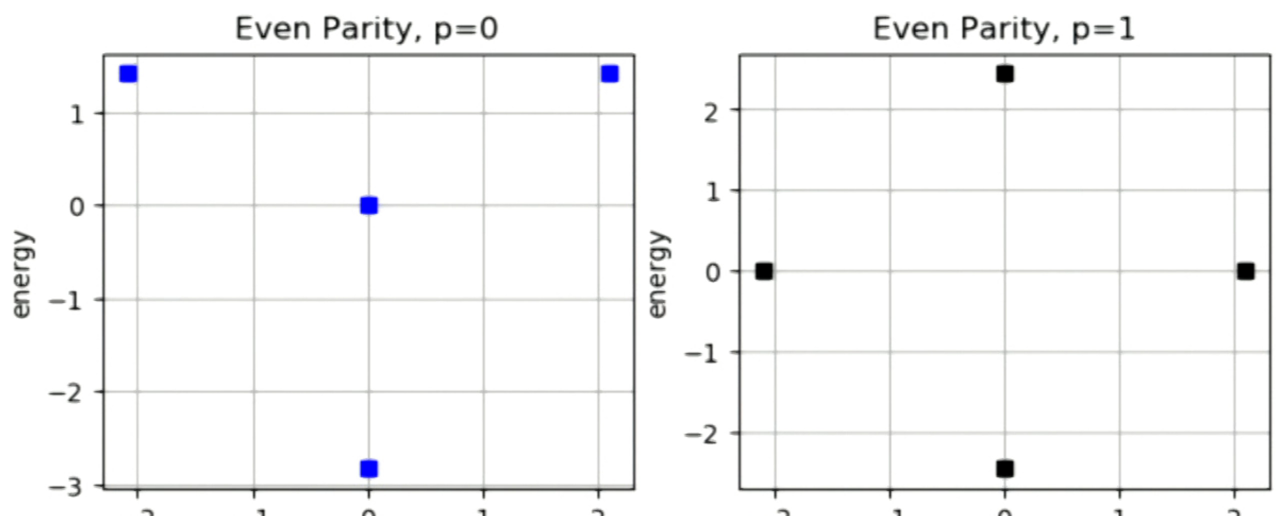
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

```julia
subplot(121) # Create the 1st axis of a 2x2 arrax of axes
grid("on") # Create a grid on the axis
title("Even Parity, p=0")
xlabel("momentum")
ylabel("energy")

subplot(122) # Create the 1st axis of a 2x2 arrax of axes
grid("on") # Create a grid on the axis
title("Even Parity, p=1")
xlabel("momentum")
ylabel("energy")

for i in 1:size(e,1)
    if p[i] == 0
        subplot(121)
        plot(k[i],e[i], marker = "s", color = "b")
    elseif p[i] == 1
        subplot(122)
        plot(k[i],e[i], marker = "s", color = "k")
    end
end
```

jupyter 2018-computational-physics-course-ExactDiag... (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Julia 1.0.0

```
        plot(k[i],e[i], marker = "s", color = "b")
    elseif p[i] == 1
        subplot(122)
        plot(k[i],e[i], marker = "s", color = "k")
    end
end
```



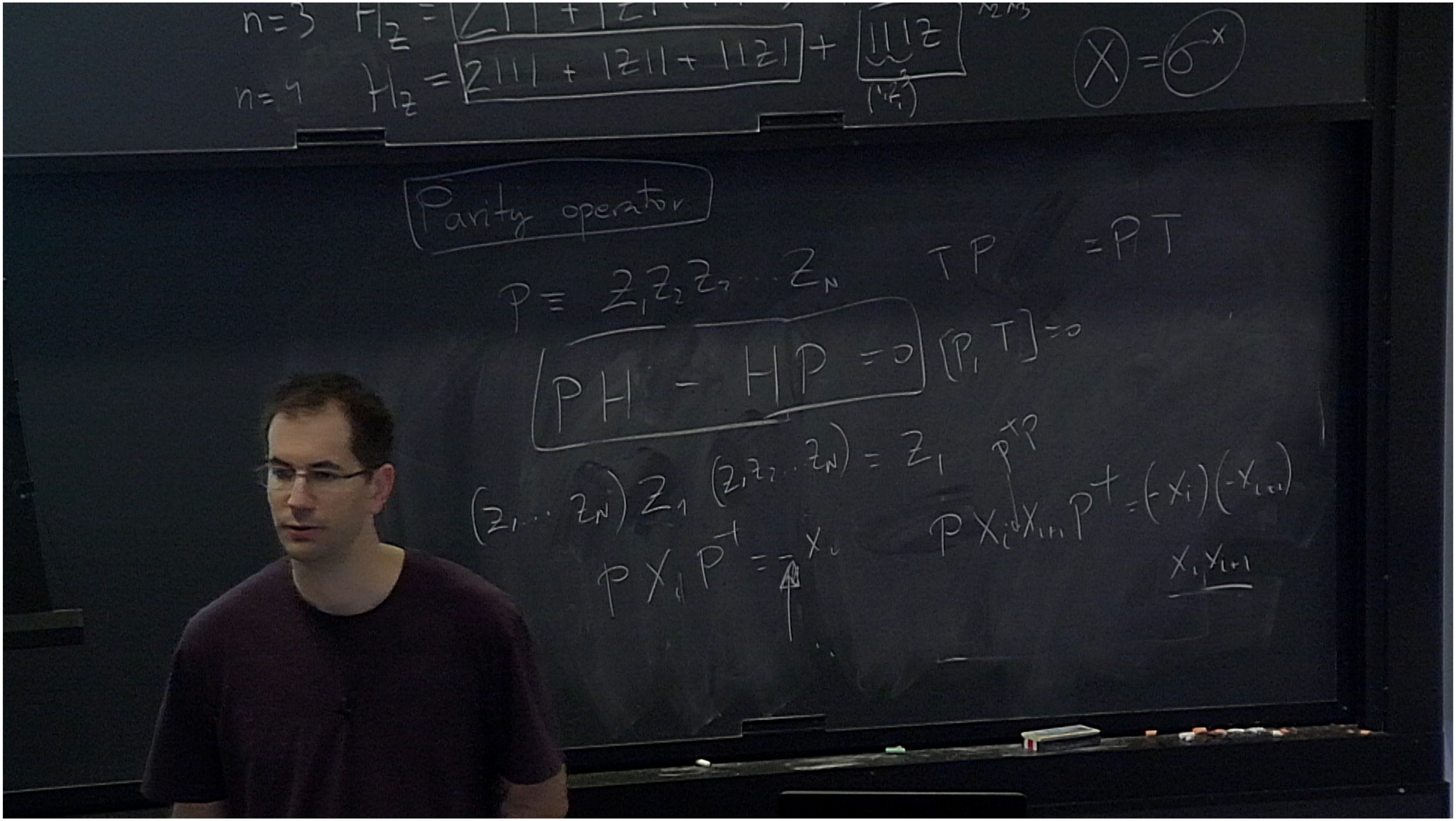
jupyter 2018-computational-physics-course-ExactDiag2-co... (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Julia 1.0.0

It's your turn for the last time! (3/3)

5) write a function buildP that builds the parity operator P for N spins (use only for $N \leq 10$)

In [20]:
```
# function buildP takes as input N (number of spins) and outputs P (parity operat
function buildP(N::Int64)::Array{Float64,2}
    Z = [1 0; 0 -1]
    P = copy(Z)
    for i=2:N
        P = kron(P,Z)
    end
    return P
```


Parity operator
P ≡ Z1 Z2 Z3 ... ZN
PH - HP = 0
[P, T] = 0
TP = PT
X = σ^x