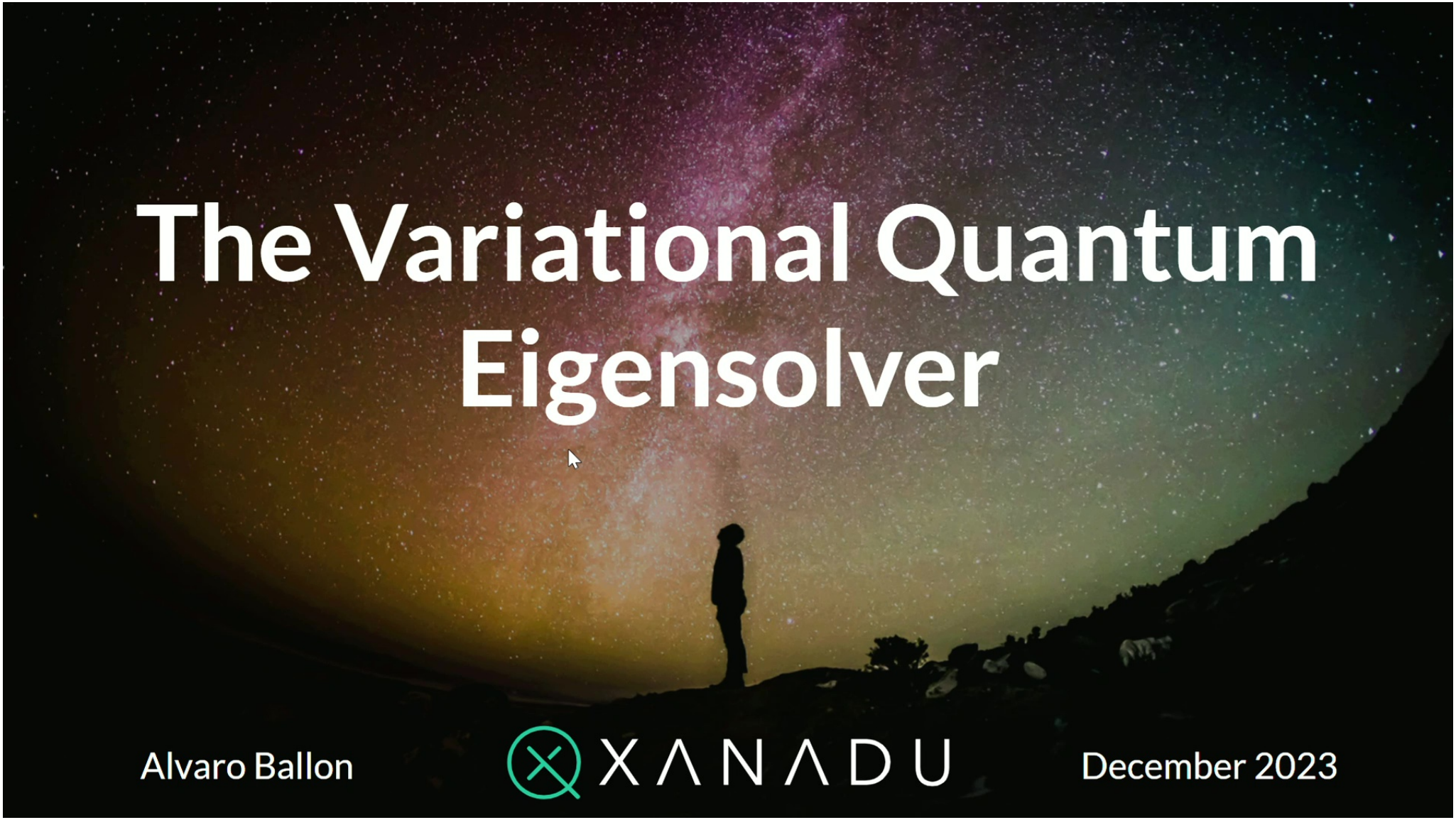
Title: Quantum Machine Learning

Speakers: Alvaro Ballon Bordo

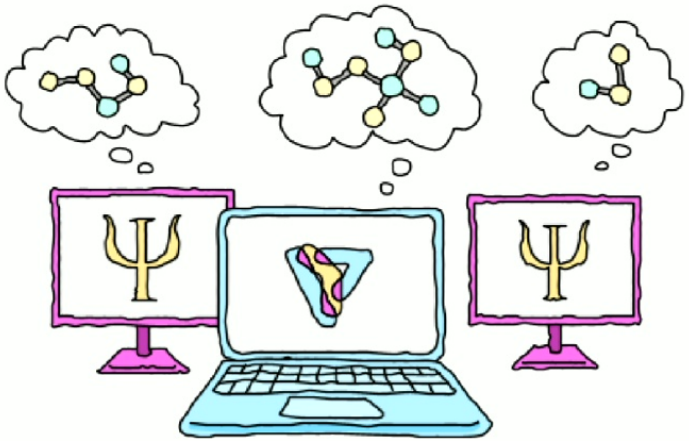
Collection: Navigating Quantum and AI Career Trajectories: A Beginnerâ€™s Mini-Course on Computational Methods and their Applications

Date: May 22, 2024 - 9:30 AM

URL: https://pirsa.org/24050084


The Variational Quantum Eigensolver
Alvaro Ballon
XANADU
December 2023

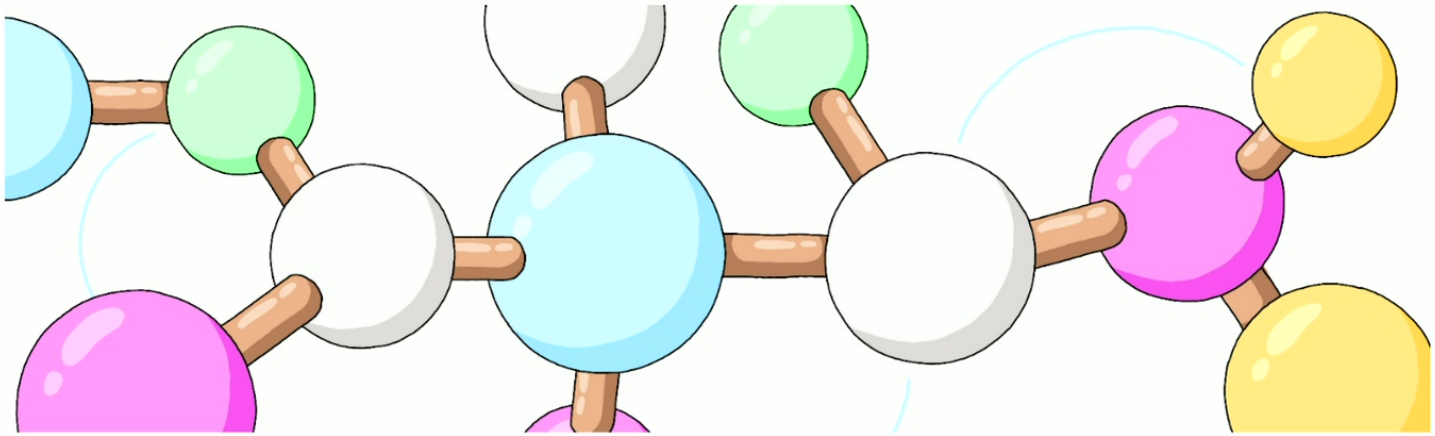
The ultimate goal of Quantum Chemistry



The dream: predict large-scale properties of matter by simulating its small scale physics.

Why? Too many applications to count. Simulating chemical reactions without going to the lab has applications in the energy, pharmaceutical, and manufacturing industries.

The challenge



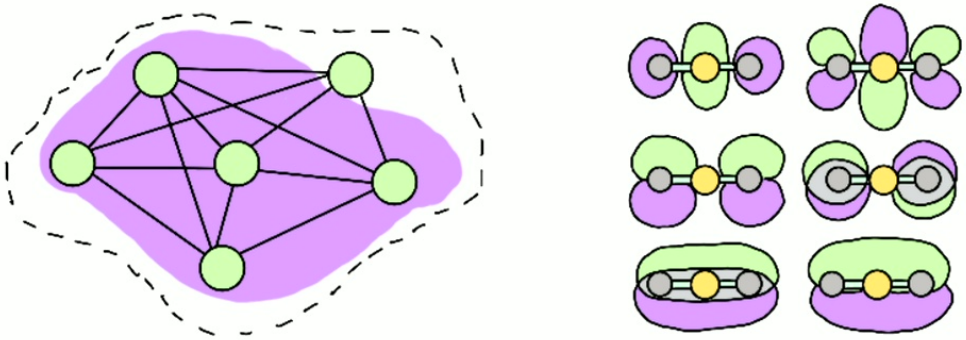
Matter is a large collection of interacting atoms, which also have nuclear and electronic substructure. Described by a Hamiltonian $\hat{H}$ which is usually large.

We need to solve Schrodinger's equation for this Hamiltonian, which is too hard!

$$\hat{H}\,\Psi(\mathbf{R},\mathbf{r}) = E\,\Psi(\mathbf{R},\mathbf{r})$$

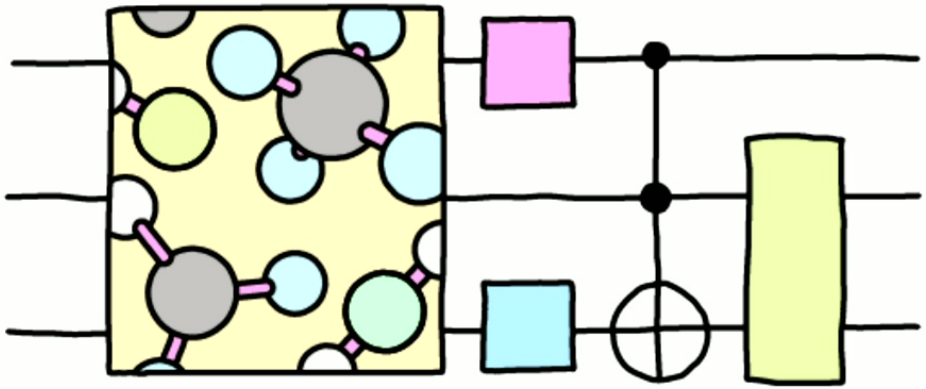
Can we try anyway?

Idea: Use an approximation, called the Hartree-Fock approximation.



Treat electrons as independent and average out effects of electrons on other electrons. Works out nicely! But for our dream applications, **we need more.**

Enter quantum computers

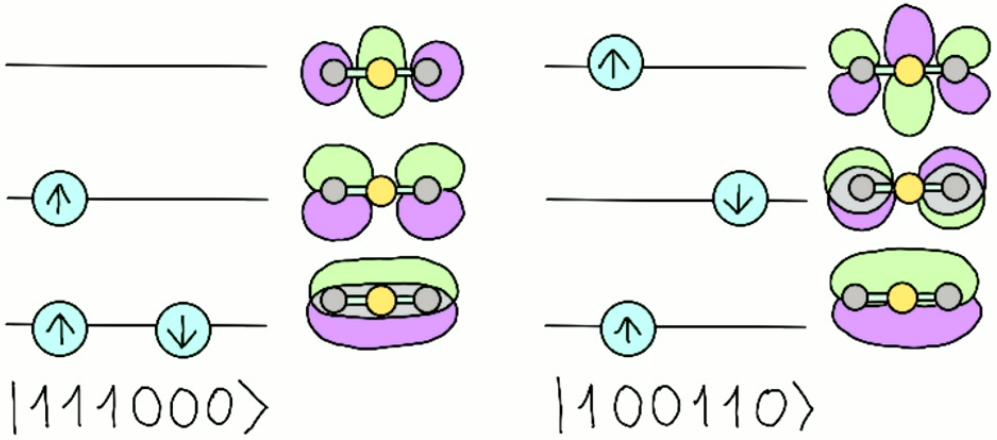


Use quantum computers to **improve on** the Hartree-Fock approximation.

These are quantum **post-Hartree Fock methods**. We will assume we know the Hartree-Fock approximated solution.

We need a way to represent molecules and Hamiltonians using qubits!

Molecule wave function as qubits



Special interest: Ground state. In the Hartree-Fock approximation, it is the Hartree-Fock state $|1,1,\ldots,1,0,0,\ldots,0\rangle$.

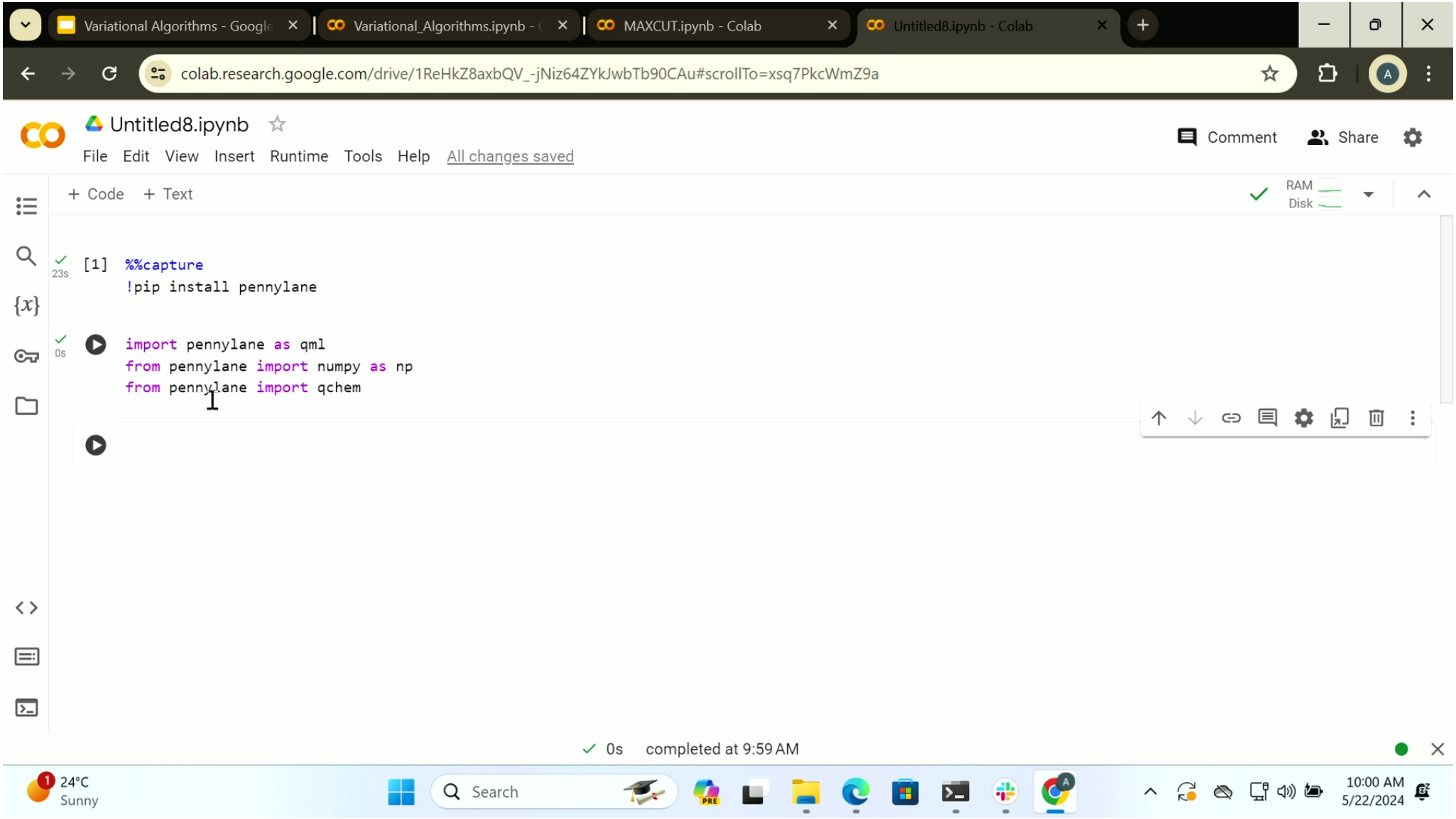
Hamiltonian as a qubit operator

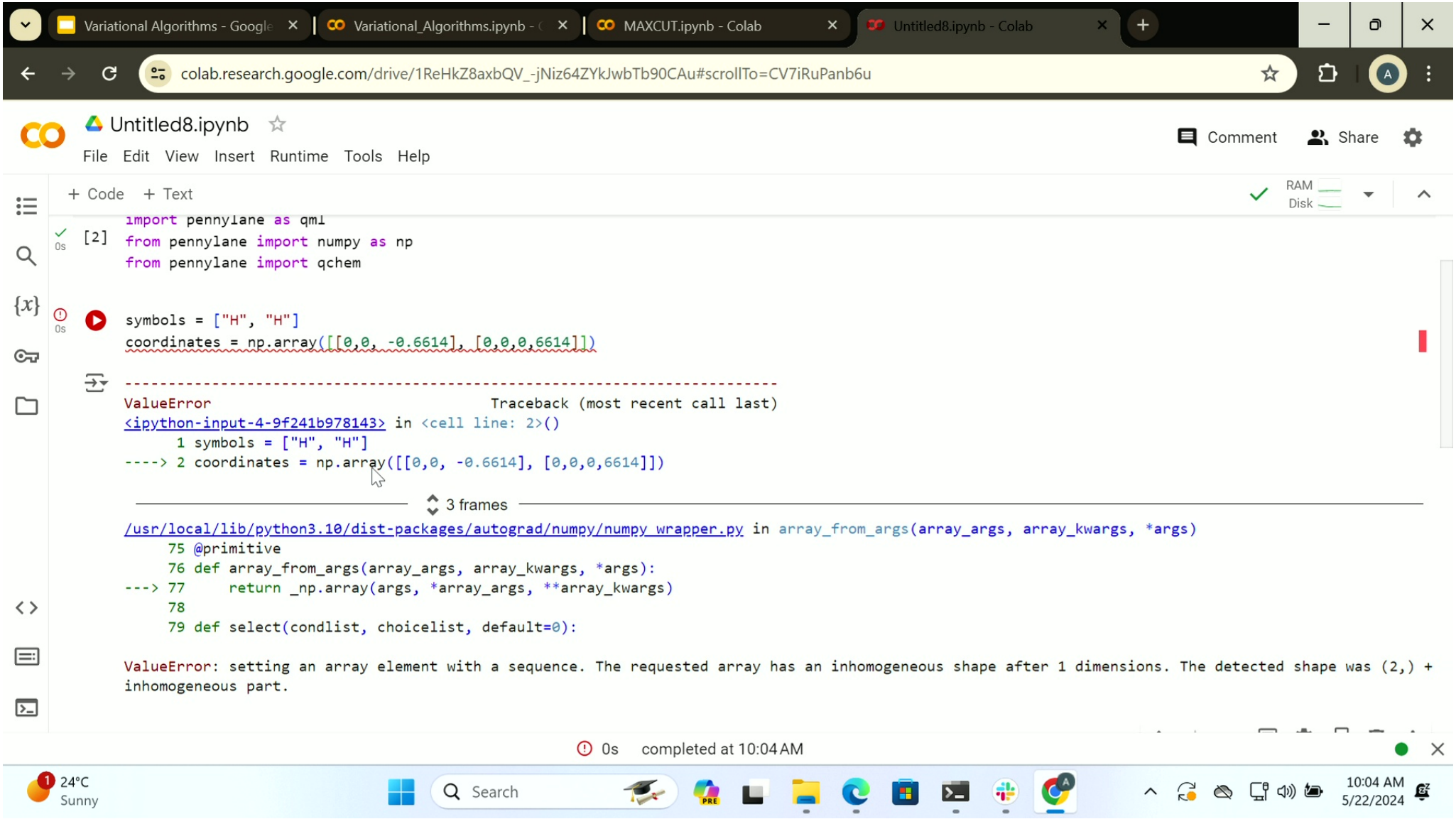
$$\hat{H}\,\Psi(\mathbf{R},\mathbf{r}) = E\,\Psi(\mathbf{R},\mathbf{r})$$

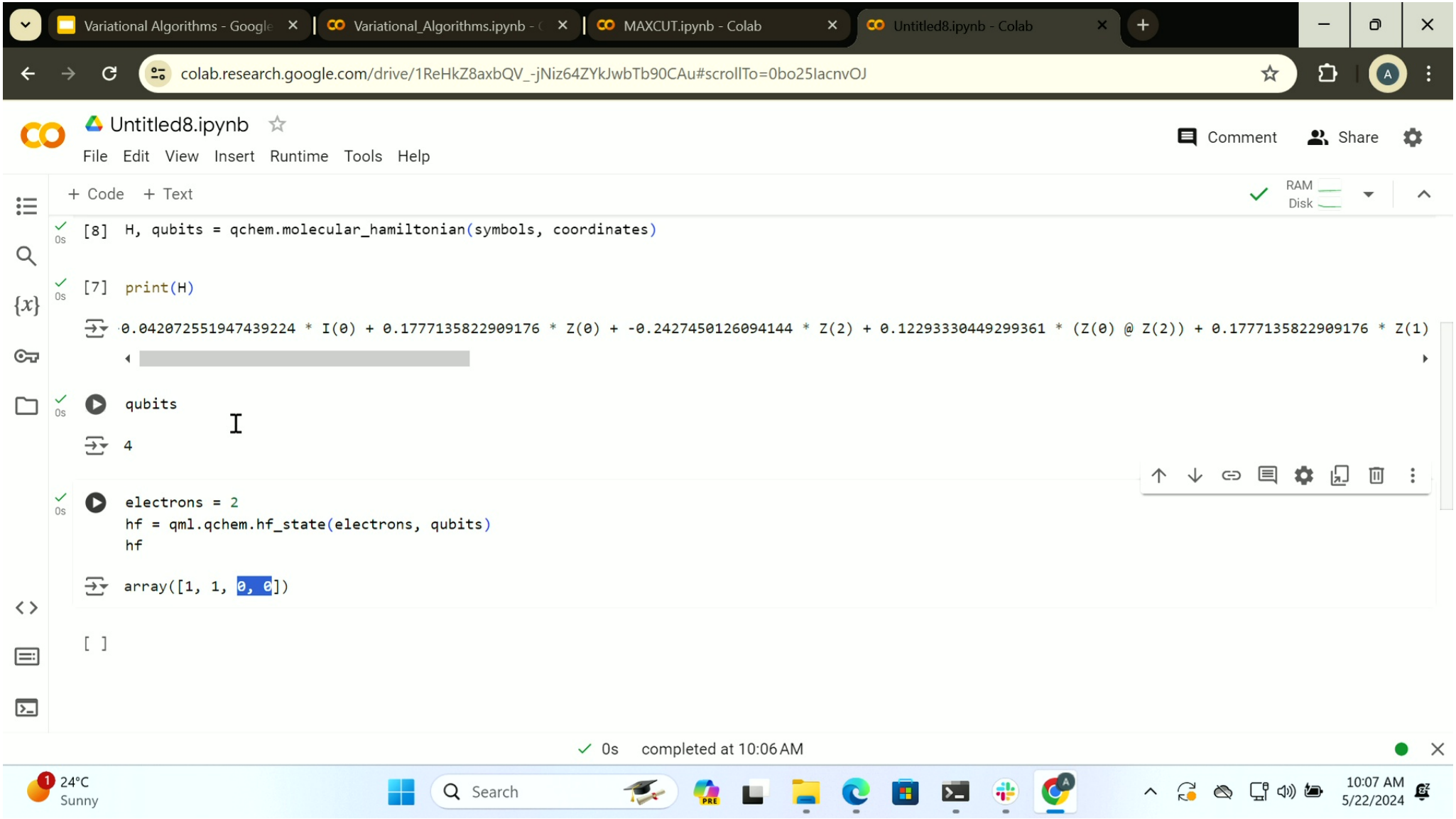
$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

$$\begin{aligned} \hat{a}_p &= \tfrac{1}{2}(X_p + iY_p) \otimes Z_{p-1} \cdots \otimes Z_0 \\ \hat{a}_p^\dagger &= \tfrac{1}{2}(X_p - iY_p) \otimes Z_{p-1} \cdots \otimes Z_0 \end{aligned}$$

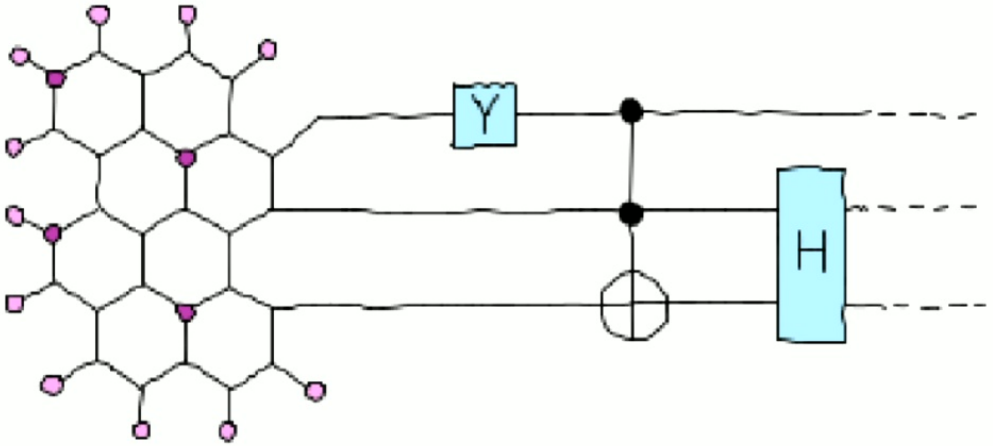
Overwhelmed much? PennyLane can save you all this work!


Variational Algorithms - Google
Variational_Algorithms.ipynb -
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=xsq7PkcWmZ9a
Untitled8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
23s
[1] %%capture
!pip install pennylane
0s
import pennylane as qml
from pennylane import numpy as np
from pennylane import qchem
0s completed at 9:59 AM
24°C
Sunny
Search
10:00 AM
5/22/2024


Variational Algorithms - Google
Variational_Algorithms.ipynb -
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=CV7iRuPanb6u
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
[2]
0s
import pennylane as qml
from pennylane import numpy as np
from pennylane import qchem
symbols = ["H", "H"]
coordinates = np.array([[0,0, -0.6614], [0,0,0,6614]])
ValueError Traceback (most recent call last)
<ipython-input-4-9f241b978143> in <cell line: 2>()
1 symbols = ["H", "H"]
----> 2 coordinates = np.array([[0,0, -0.6614], [0,0,0,6614]])
3 frames
/usr/local/lib/python3.10/dist-packages/autograd/numpy/numpy_wrapper.py in array_from_args(array_args, array_kwargs, *args)
75 @primitive
76 def array_from_args(array_args, array_kwargs, *args):
---> 77 return _np.array(args, *array_args, **array_kwargs)
78
79 def select(condlist, choicelist, default=0):
ValueError: setting an array element with a sequence. The requested array has an inhomogeneous shape after 1 dimensions. The detected shape was (2,) + inhomogeneous part.
0s completed at 10:04 AM
24°C
Sunny
Search
10:04 AM
5/22/2024


Variational Algorithms - Google
Variational_Algorithms.ipynb -
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=0bo25IacnvOJ
Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
[8] H, qubits = qchem.molecular_hamiltonian(symbols, coordinates)
0s
[7] print(H)
0s
0.042072551947439224 * I(0) + 0.1777135822909176 * Z(0) + -0.2427450126094144 * Z(2) + 0.12293330449299361 * (Z(0) @ Z(2)) + 0.1777135822909176 * Z(1)
qubits
0s
4
electrons = 2
hf = qml.qchem.hf_state(electrons, qubits)
hf
0s
array([1, 1, 0, 0])
[ ]
0s completed at 10:06 AM
24°C
Sunny
Search
10:07 AM
5/22/2024

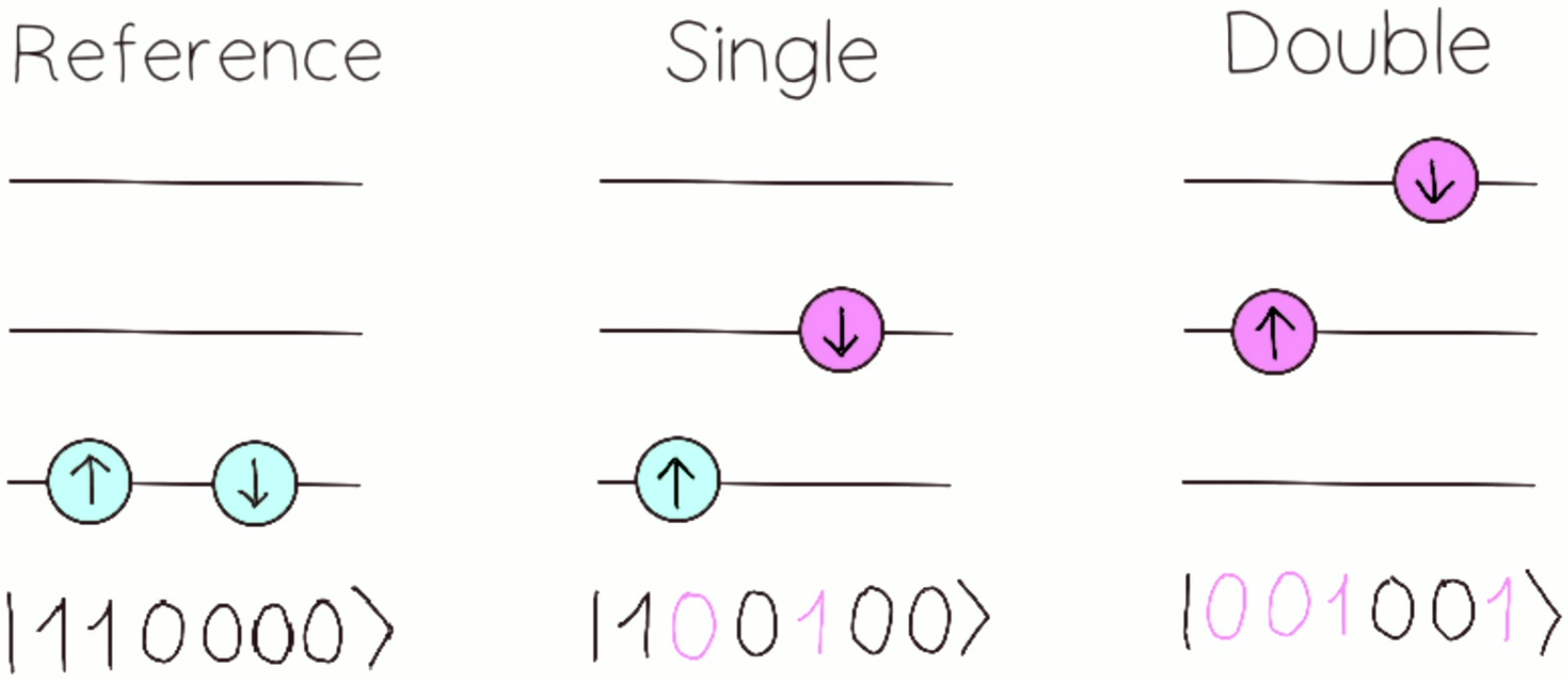
Variational Quantum Eigensolver (VQE)



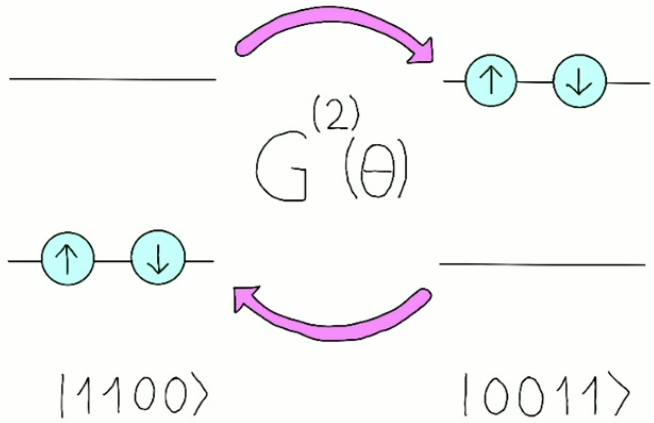
Post-Hartree Fock algorithm to find **ground state** of system described by $\hat{H}$.

1. Prepare Hartree-Fock state.
2. Create an **ansatz** which:
   a. Prepares a family of candidate ground states
   b. Measure the expectation value of the Hamiltonian
3. Optimize circuit parameters to minimize the expectation value.

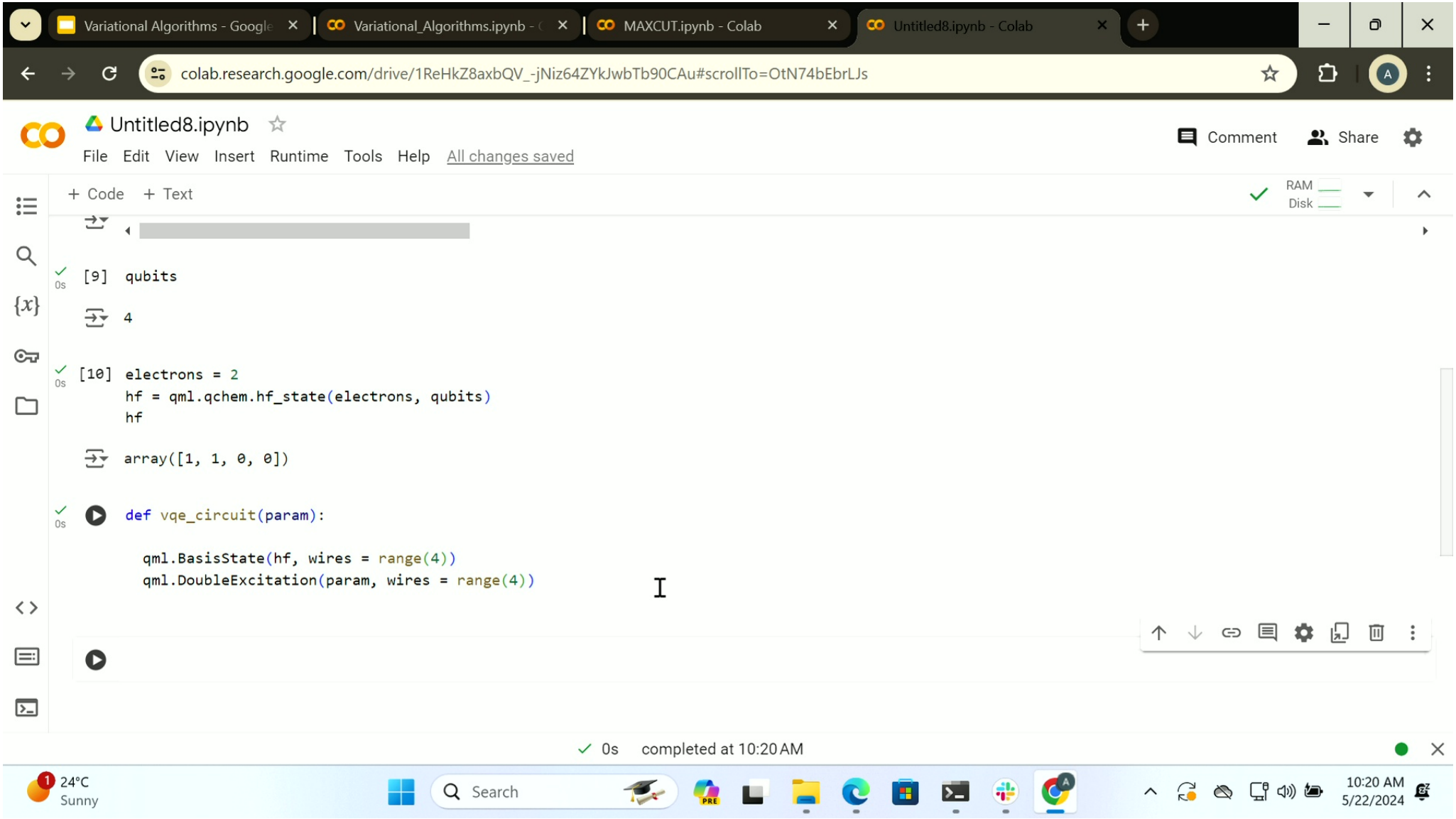
Excitation gates

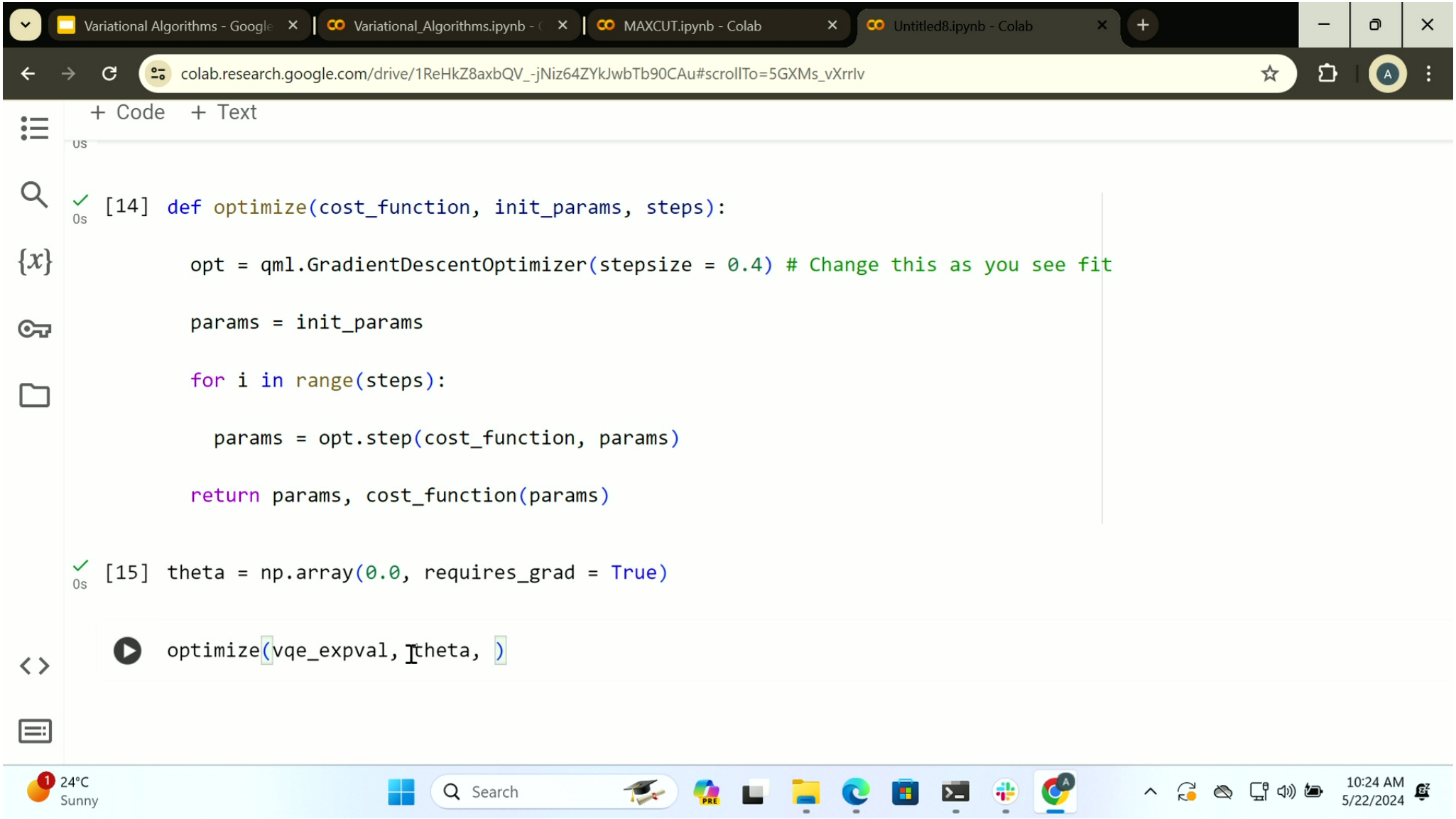


Double excitation gate



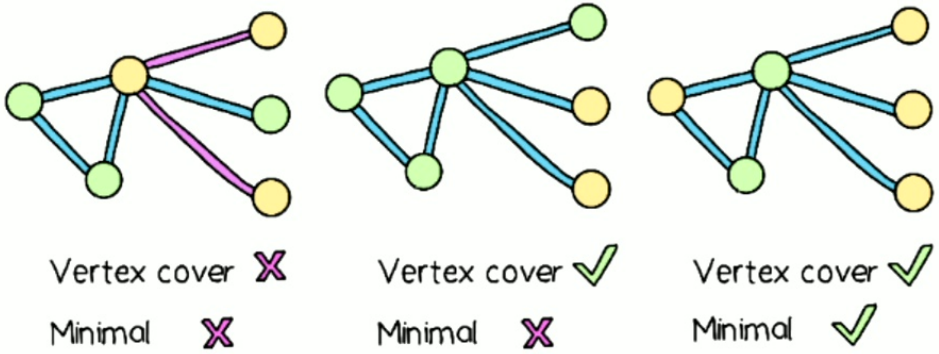
$$G^{(2)}|0011\rangle = \cos(\theta/2)|0011\rangle + \sin(\theta/2)|1100\rangle$$
$$G^{(2)}|1100\rangle = \cos(\theta/2)|1100\rangle - \sin(\theta/2)|0011\rangle$$


Variational Algorithms - Google
Variational_Algorithms.ipynb -
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=OtN74bEbrLJs
Untitled8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[9] qubits
4
[10] electrons = 2
hf = qml.qchem.hf_state(electrons, qubits)
hf
array([1, 1, 0, 0])
def vqe_circuit(param):
qml.BasisState(hf, wires = range(4))
qml.DoubleExcitation(param, wires = range(4))
0s completed at 10:20 AM
24°C
Sunny
Search
10:20 AM
5/22/2024


Variational Algorithms - Google
Variational_Algorithms.ipynb -
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=5GXMs_vXrrIv
+ Code + Text
[14] def optimize(cost_function, init_params, steps):
opt = qml.GradientDescentOptimizer(stepsize = 0.4) # Change this as you see fit
params = init_params
for i in range(steps):
params = opt.step(cost_function, params)
return params, cost_function(params)
[15] theta = np.array(0.0, requires_grad = True)
optimize(vqe_expval, theta,)
24°C
Sunny
Search
10:24 AM
5/22/2024

Let’s try this in PennyLane!

What is combinatorial optimization?

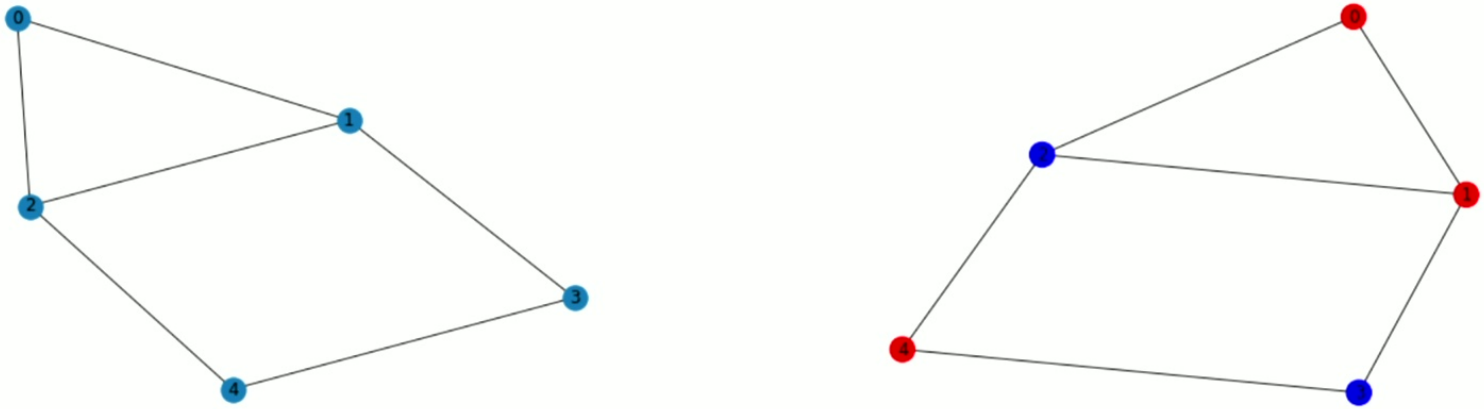


Another Potential Application of Optimization with Quantum Circuits

Finding an optimal combination among the set of all possible combinations. Examples:

Knapsack problem, Minimum vertex cover, MaxCut

The MAXCUT problem



Partition a graph into two sets of vertices

If we maximize the number of edges between elements of different colours (sets of vertices), then we have a MAXCUT of the graph. Finding this partition is a combinatorial optimization problem.

Mathematical formulation of MAXCUT

Maximize the function:

$$C(\vec{s}) = \frac{1}{2} \sum_{(i,j)\in E} (1 - s_i s_j)$$

Where the s take the value 1 if s is in partition 1, and the value -1 if it's in partition 2.

Quantum Formulation

Map cost function into Cost Hamiltonian

$$\hat{C} = -\frac{1}{2} \sum_{(i,j) \in E} (\mathbb{1} - Z_i Z_j)$$

Amounts to finding the ground state of this cost Hamiltonian. Let's implement this Hamiltonian in PennyLane for our MAXCUT graph.

QAOA Theory

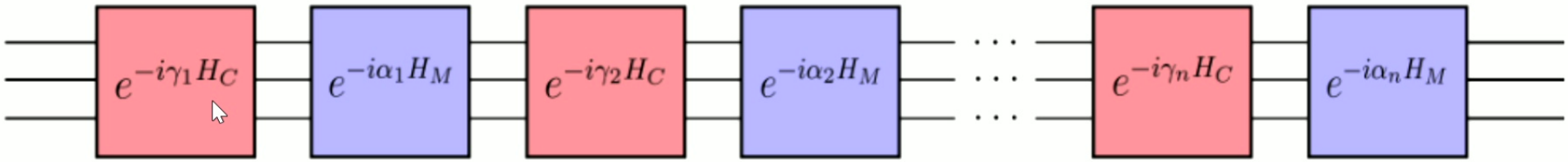
Based on **Adiabatic theorem.** If we evolve the ground state of H_0 under the time dependent Hamiltonian

$$H(t) = (1-t)H_0 + tH_1, \quad t \in [0,1],$$

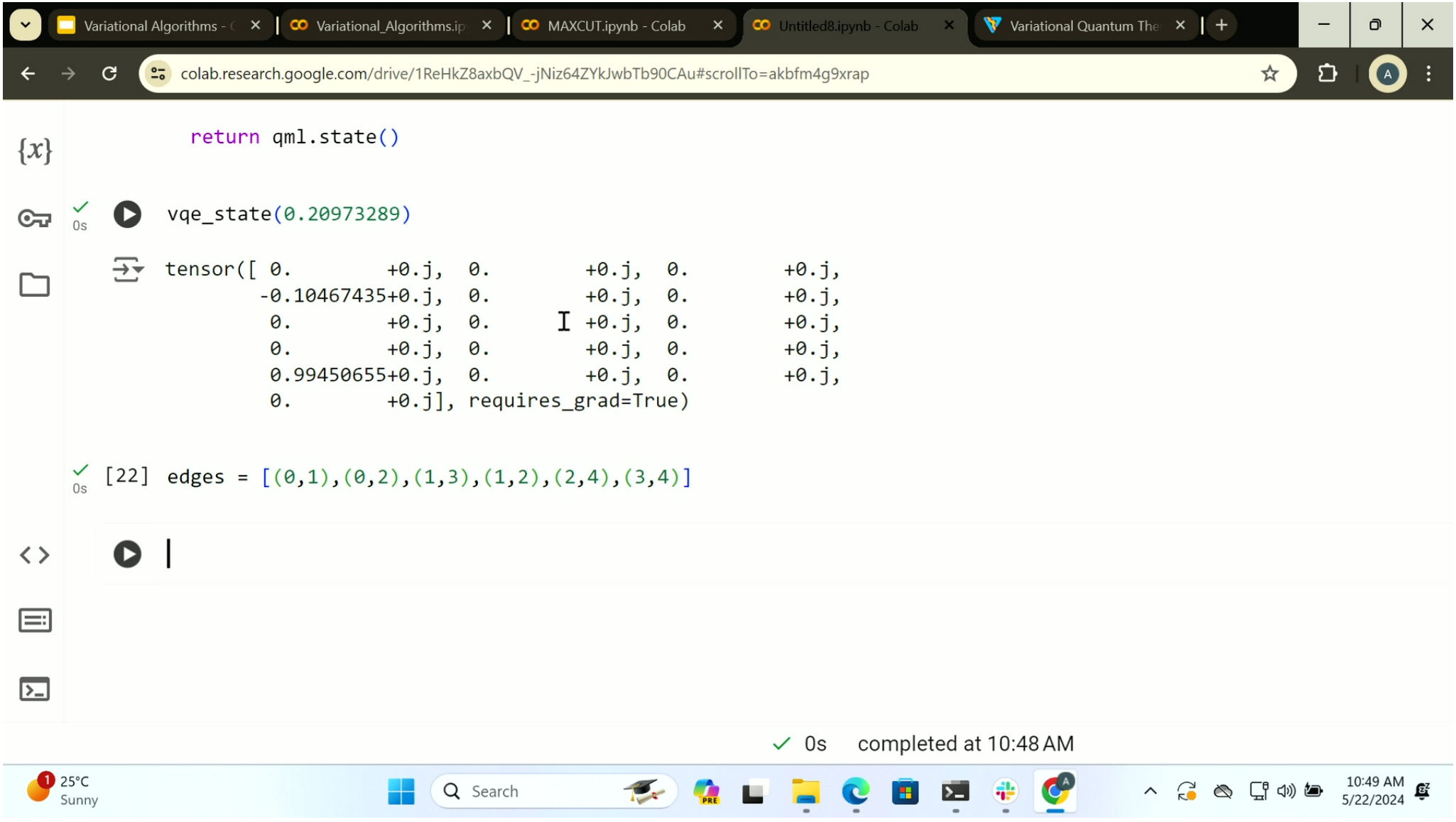
We end up in the ground state of H_1.

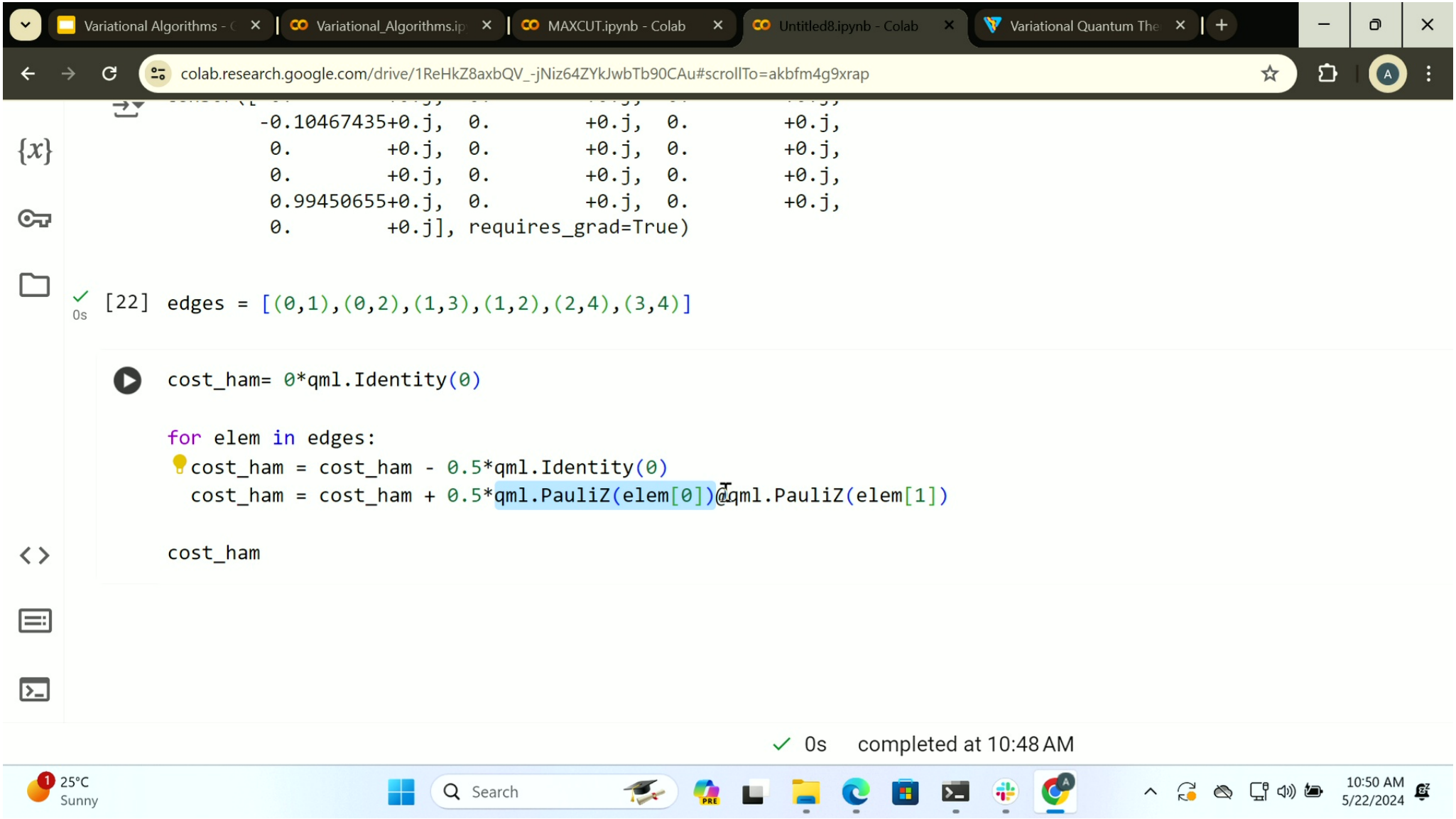
Strategy: Start in the ground state of a Hamiltonian we know (mixer Hamiltonian), and evolve adiabatically to end up in ground state of target Hamiltonian.

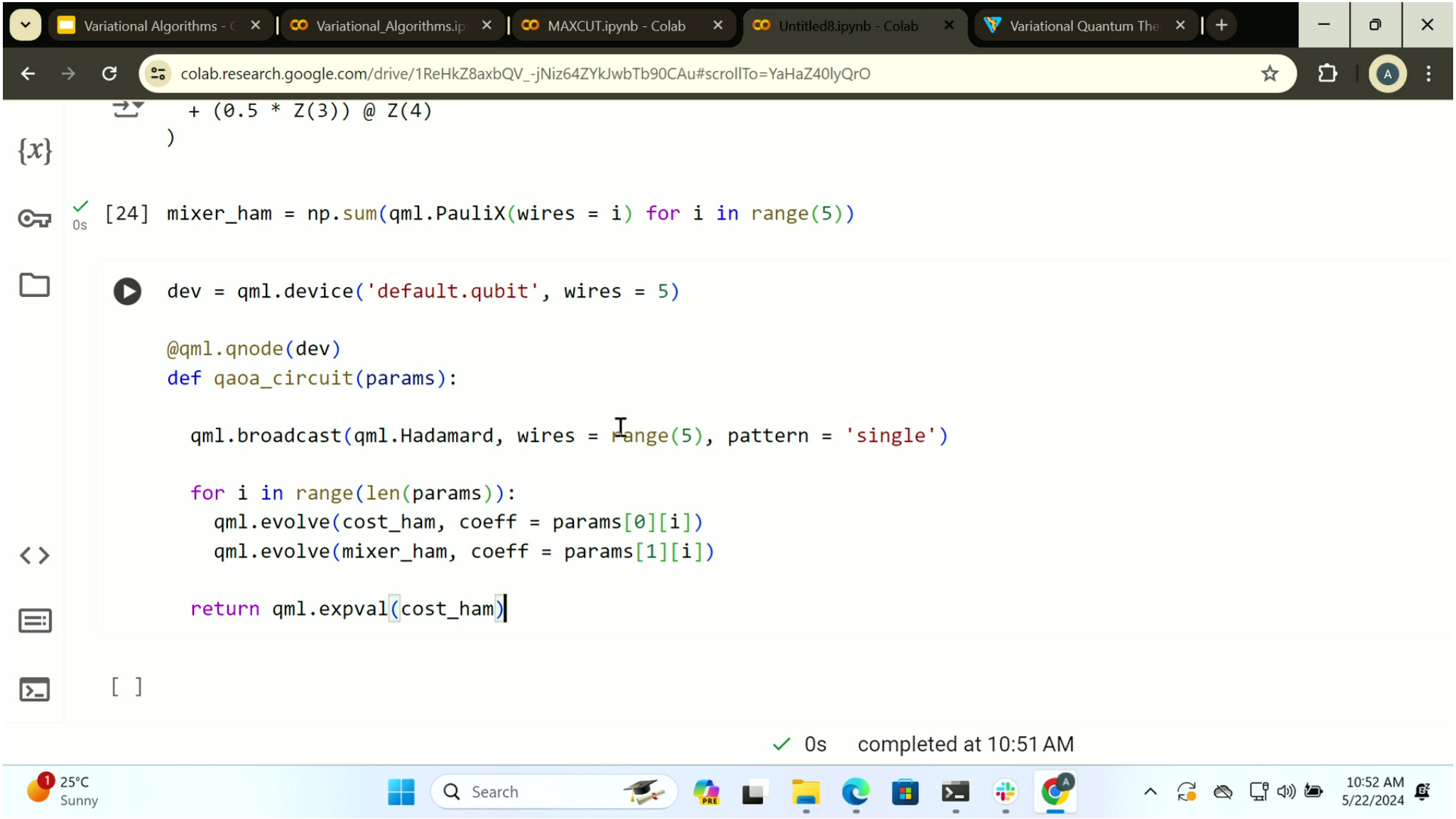
Implementation

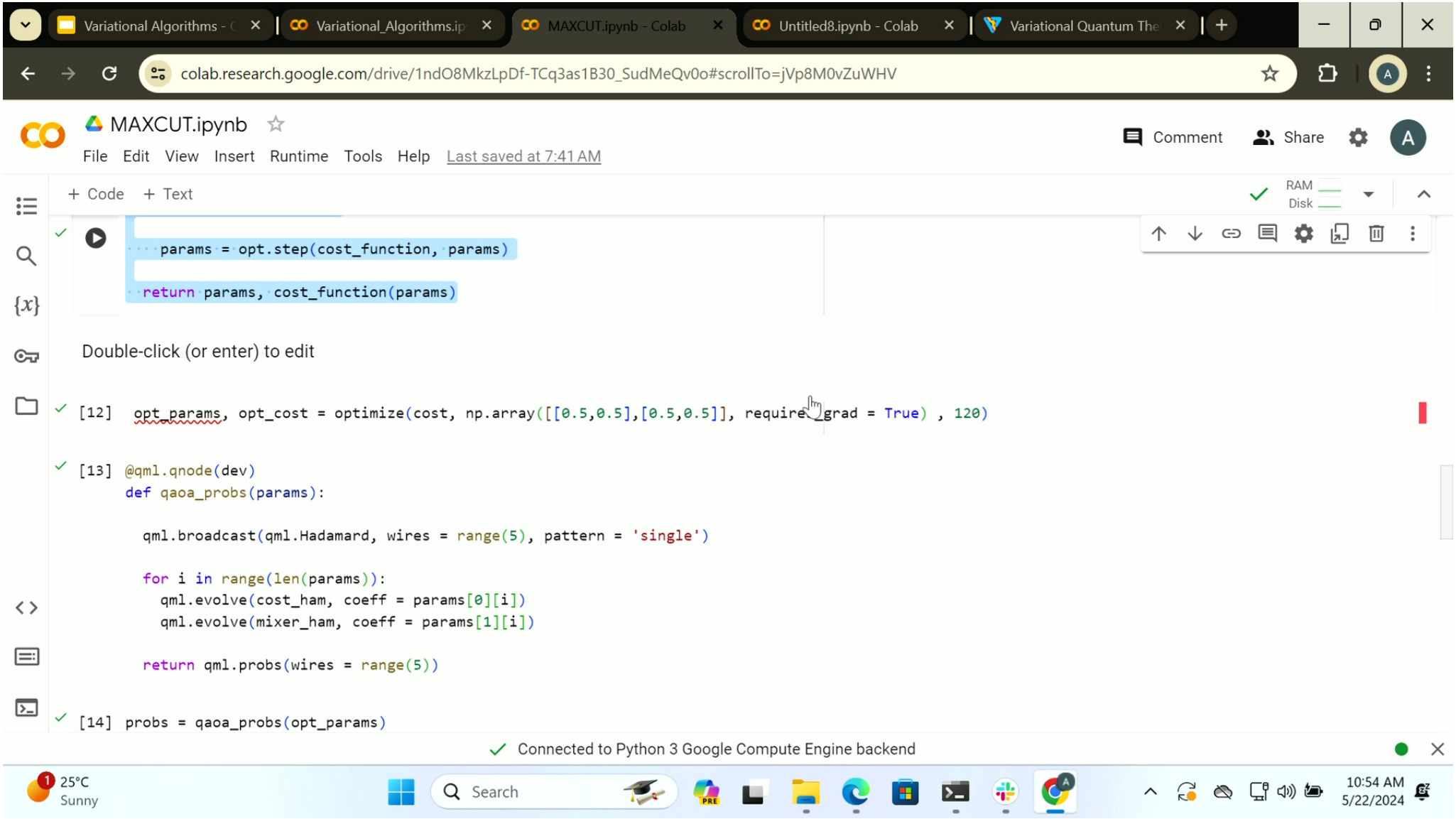


Circuit above implements an **approximate** adiabatic evolution between the mixer and target Hamiltonian, also knowns as **cost Hamiltonian.**


Variational Algorithms -
Variational_Algorithms.ip
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
Variational Quantum The
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=akbfm4g9xrap
return qml.state()
vqe_state(0.20973289)
tensor([0. +0.j, 0. +0.j, 0. +0.j,
-0.10467435+0.j, 0. +0.j, 0. +0.j,
0. +0.j, 0. +0.j, 0. +0.j,
0. +0.j, 0. +0.j, 0. +0.j,
0.99450655+0.j, 0. +0.j, 0. +0.j,
0. +0.j], requires_grad=True)
[22] edges = [(0,1),(0,2),(1,3),(1,2),(2,4),(3,4)]
0s
0s completed at 10:48 AM
25°C
Sunny
Search
10:49 AM
5/22/2024


Variational Algorithms -
Variational_Algorithms.ip
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
Variational Quantum The
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=akbfm4g9xrap
-0.10467435+0.j, 0. +0.j, 0. +0.j,
0. +0.j, 0. +0.j, 0. +0.j,
0. +0.j, 0. +0.j, 0. +0.j,
0.99450655+0.j, 0. +0.j, 0. +0.j,
0. +0.j], requires_grad=True)
[22] edges = [(0,1),(0,2),(1,3),(1,2),(2,4),(3,4)]
0s
cost_ham= 0*qml.Identity(0)
for elem in edges:
cost_ham = cost_ham - 0.5*qml.Identity(0)
cost_ham = cost_ham + 0.5*qml.PauliZ(elem[0])@qml.PauliZ(elem[1])
cost_ham
0s completed at 10:48 AM
25°C
Sunny
Search
10:50 AM
5/22/2024


Variational Algorithms -
Variational_Algorithms.ip
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
Variational Quantum The
colab.research.google.com/drive/1ReHkZ8axbQV_-jNiz64ZYkJwbTb90CAu#scrollTo=YaHaZ40lyQrO
+ (0.5 * Z(3)) @ Z(4)
)
[24] mixer_ham = np.sum(qml.PauliX(wires = i) for i in range(5))
0s
dev = qml.device('default.qubit', wires = 5)
@qml.qnode(dev)
def qaoa_circuit(params):
qml.broadcast(qml.Hadamard, wires = range(5), pattern = 'single')
for i in range(len(params)):
qml.evolve(cost_ham, coeff = params[0][i])
qml.evolve(mixer_ham, coeff = params[1][i])
return qml.expval(cost_ham)
[ ]
0s completed at 10:51 AM
25°C
Sunny
Search
10:52 AM
5/22/2024


Variational Algorithms -
Variational_Algorithms.ip
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
Variational Quantum The
colab.research.google.com/drive/1ndO8MkzLpDf-TCq3as1B30_SudMeQv0o#scrollTo=jVp8M0vZuWHV
MAXCUT.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:41 AM
Comment
Share
+ Code + Text
RAM
Disk
params = opt.step(cost_function, params)
return params, cost_function(params)
Double-click (or enter) to edit
[12] opt_params, opt_cost = optimize(cost, np.array([[0.5,0.5],[0.5,0.5]], requires_grad = True) , 120)
[13] @qml.qnode(dev)
def qaoa_probs(params):
qml.broadcast(qml.Hadamard, wires = range(5), pattern = 'single')
for i in range(len(params)):
qml.evolve(cost_ham, coeff = params[0][i])
qml.evolve(mixer_ham, coeff = params[1][i])
return qml.probs(wires = range(5))
[14] probs = qaoa_probs(opt_params)
Connected to Python 3 Google Compute Engine backend
25°C
Sunny
Search
10:54 AM
5/22/2024

Variational Algorithms - C | Variational_Algorithms.ip | MAXCUT.ipynb - Colab | Untitled8.ipynb - Colab | Variational Quantum The

docs.google.com/presentation/d/1KKt7m9lSGdD56GoouZz-JoFsfuYDNAkb9jcSHh6Q2gM/edit#slide=id.g271fbbab598_0_4

Variational Algorithms

File Edit View Insert Format Slide Arrange Tools Extensions Help

Slideshow | Share

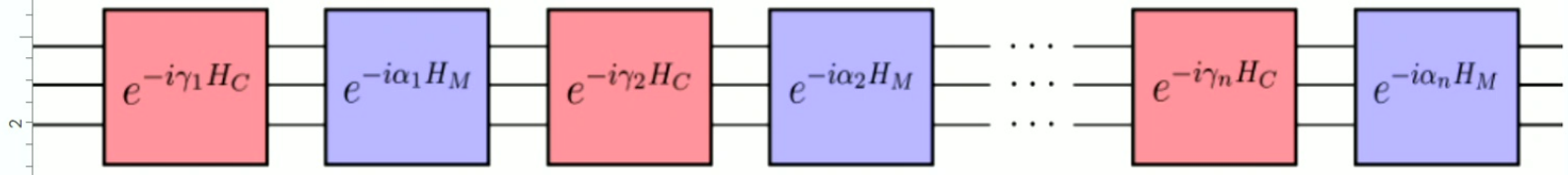
114% | Background | Layout | Theme | Transition | Rec

20 Quantum Formulation

21 QAOA Theory

22 Implementation

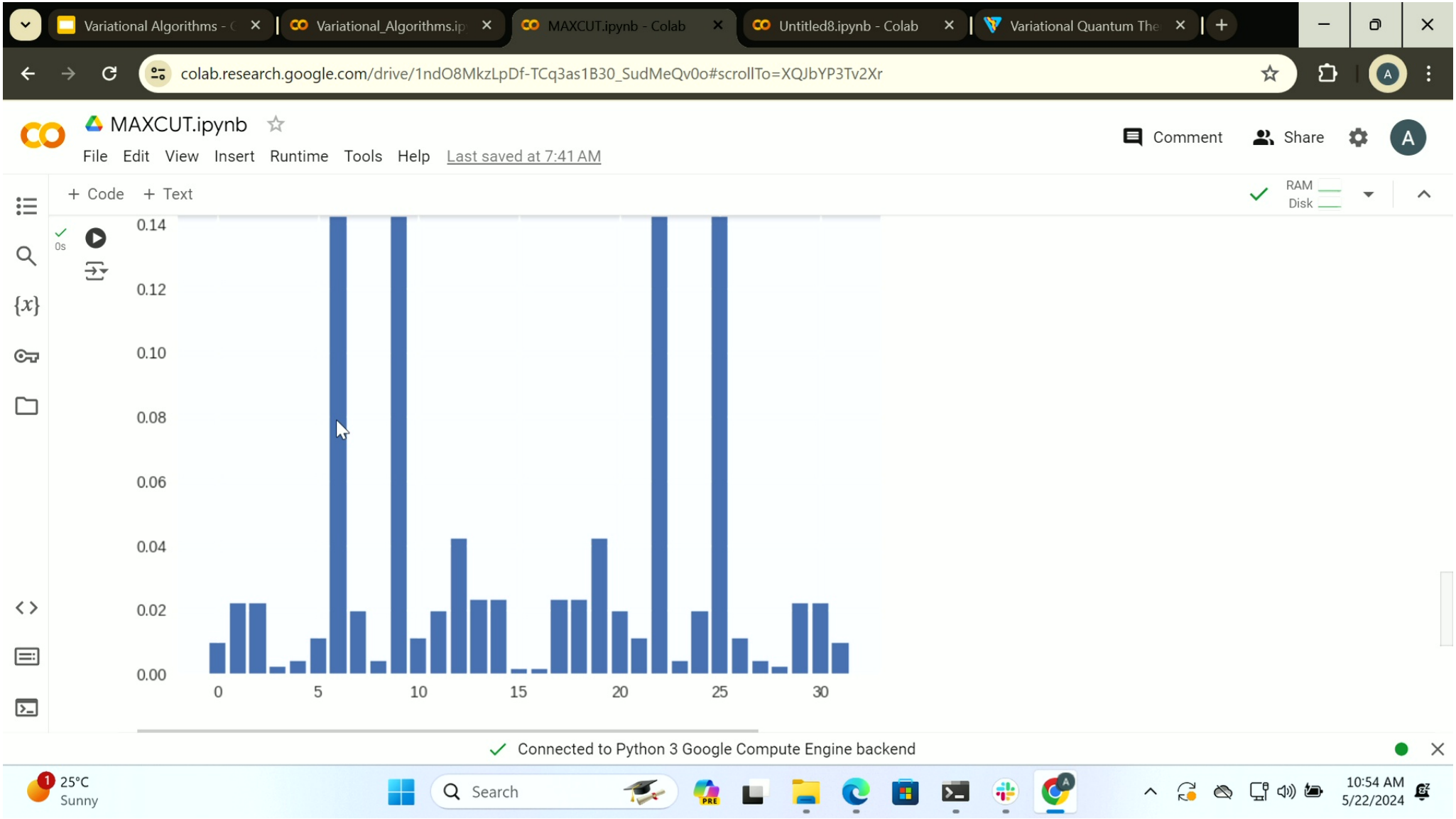
23 Let's try this in PennyLane!



Circuit above implements an **approximate** adiabatic evolution between the mixer and target Hamiltonian, also knowns as **cost Hamiltonian.**

Click to add speaker notes

25°C Sunny | Search | 10:55 AM 5/22/2024


Variational Algorithms -
Variational_Algorithms.ip
MAXCUT.ipynb - Colab
Untitled8.ipynb - Colab
Variational Quantum The
colab.research.google.com/drive/1ndO8MkzLpDf-TCq3as1B30_SudMeQv0o#scrollTo=XQJbYP3Tv2Xr
MAXCUT.ipynb
File Edit View Insert Runtime Tools Help Last saved at 7:41 AM
Comment
Share
+ Code + Text
RAM
Disk
0s
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30
Connected to Python 3 Google Compute Engine backend
25°C
Sunny
Search
10:54 AM
5/22/2024

Variational Algorithms - C | Variational_Algorithms.ip | MAXCUT.ipynb - Colab | Untitled8.ipynb - Colab | Variational Quantum The

colab.research.google.com/drive/1ndO8MkzLpDf-TCq3as1B30_SudMeQv0o#scrollTo=SEIJU7uLwz4p

MAXCUT.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:41 AM

Comment Share A

+ Code + Text

RAM Disk

```
    return qml.probs(wires = range(5))
```

```
[14] probs = qaoa_probs(opt_params)
```

```
from matplotlib import pyplot as plt
plt.style.use("seaborn")
plt.bar(range(2 ** 5), probs)
plt.show()
```

```
<ipython-input-15-6f6f4133b234>:2: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer
  plt.style.use("seaborn")
```

Connected to Python 3 Google Compute Engine backend

26°C Sunny

Search

10:57 AM 5/22/2024