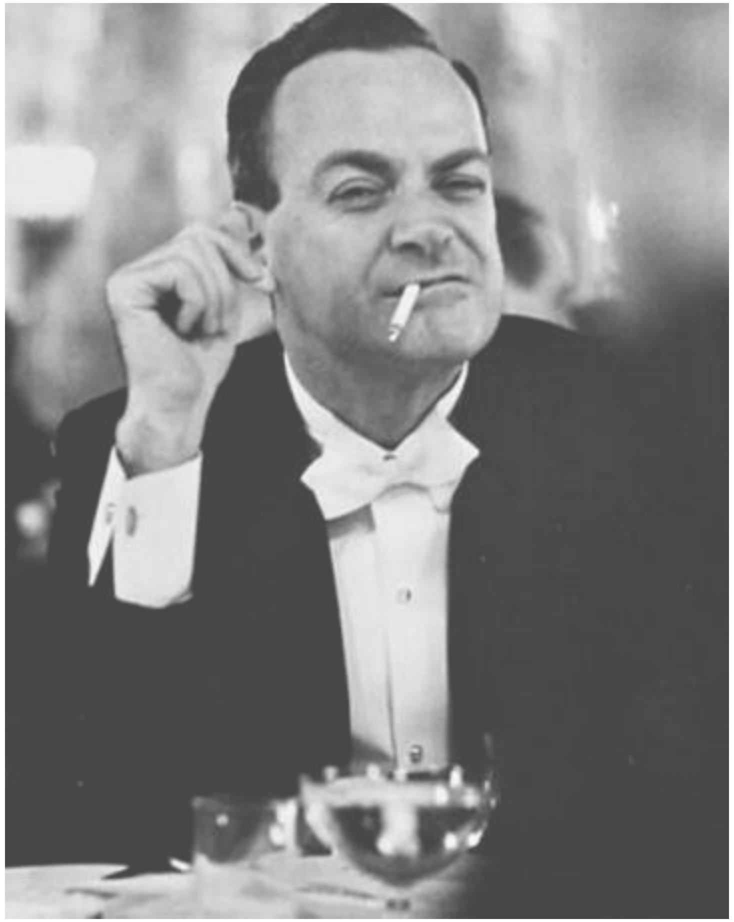
Innovative methods to determine molecular spectra

Politecnico di Torino

What is quantum chemistry?

A branch of physical chemistry focused on the application of *quantum mechanics* to chemical systems to *predict the contribution of electronic systems to physical and chemical properties* of molecules, materials and solutions at the atomic level.

What is quantum chemistry?

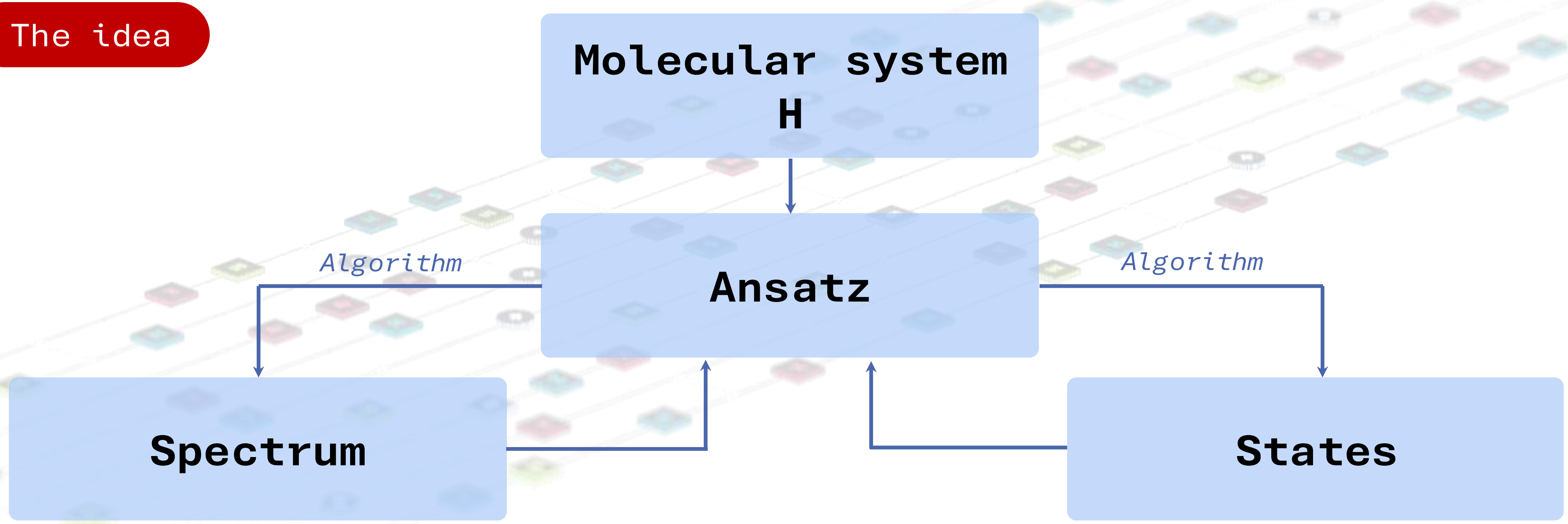


“Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical”

Richard P. Feynman

Solution of a quantum chemistry problem

The idea



Our work

I. Provide an *implementation* of the RODEO algorithm

II. Propose an *alternative algorithm* to retrieve the whole energy spectrum of a molecular system

III. Illustrate the *differences* between the outcomes of the two algorithms test-benched over the same hamiltonians

Our work

Why?

Many algorithms able to determine the *ground state energy* of a quantum system

- Quantum phase estimation
- Quantum annealing
- ...

No *established* algoritmhs to build the *whole* spectrum!

OUTLINE

- The Rodeo Algorithm
- RODEO Implementation & Results
- Variational Quantum Deflation (VQD)
- VQD Implementation & Results
- Comparison
- Conclusions

The RODEO Algorithm

The RODEO Algorithm

Preliminary Example

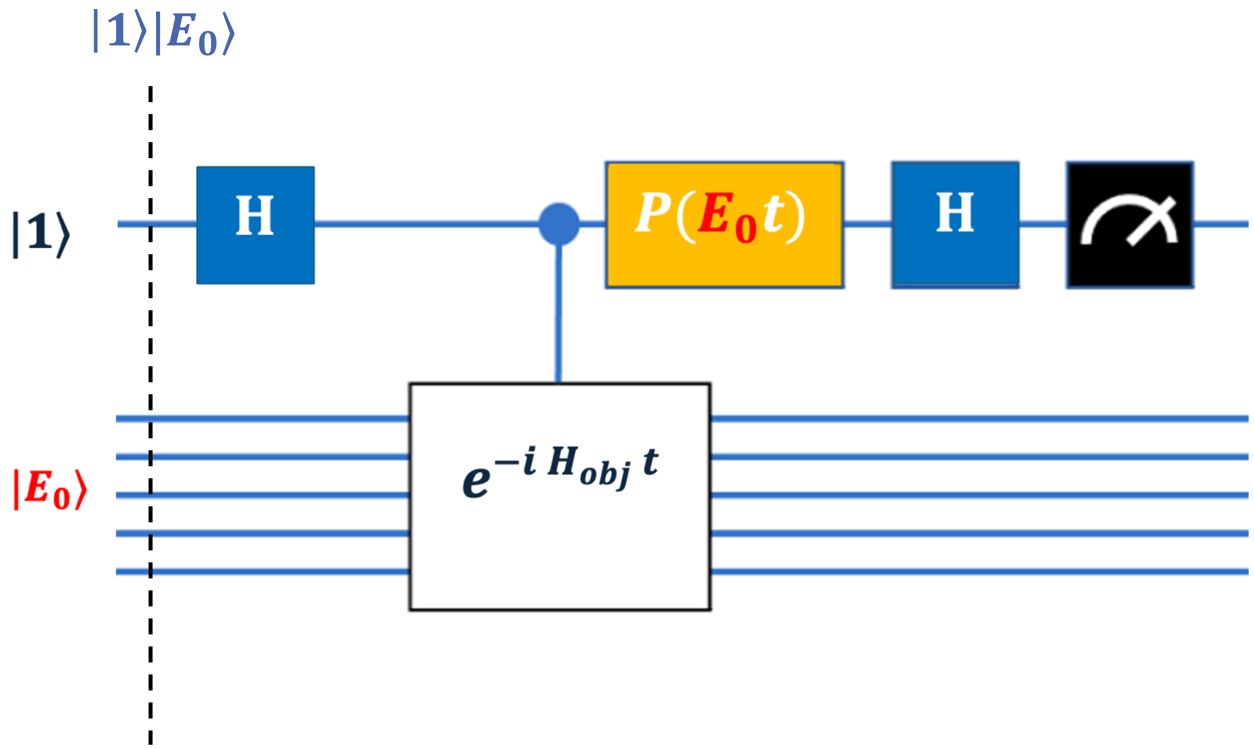
Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$



The RODEO Algorithm

Preliminary Example

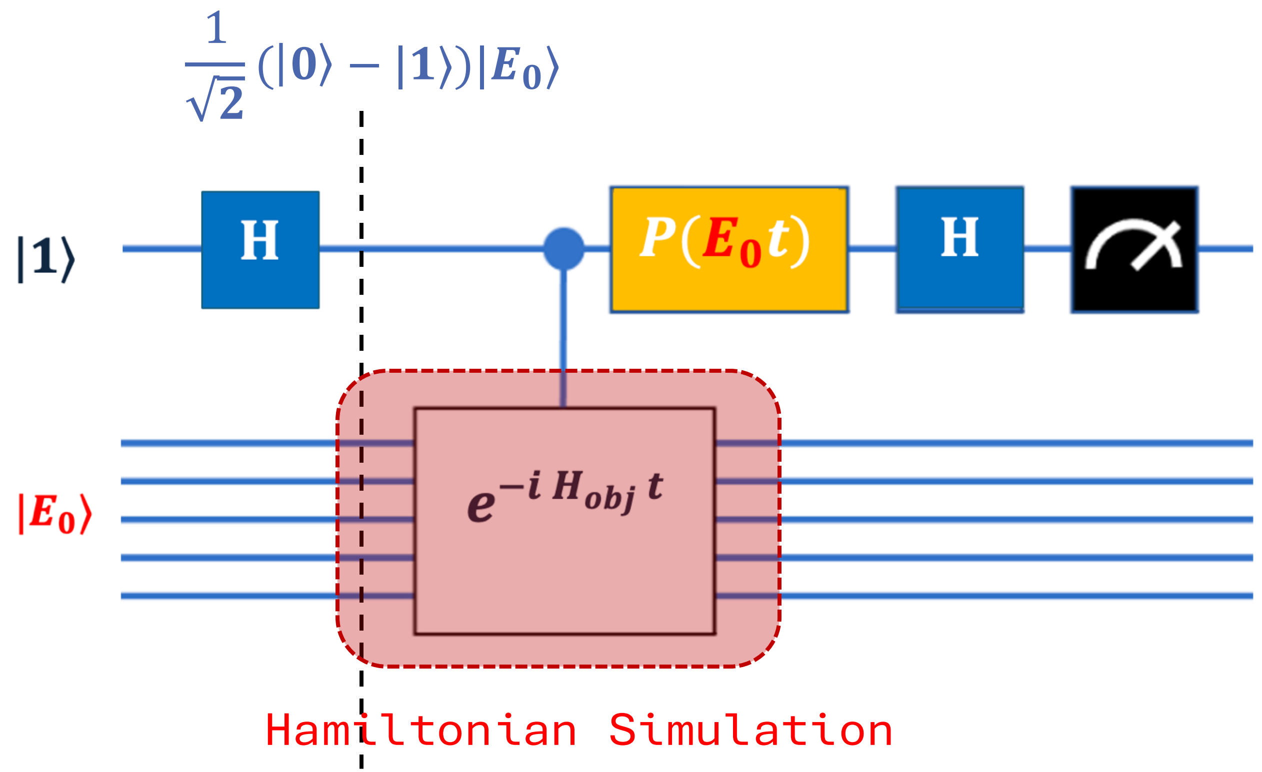
Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$



The RODEO Algorithm

Preliminary Example

Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

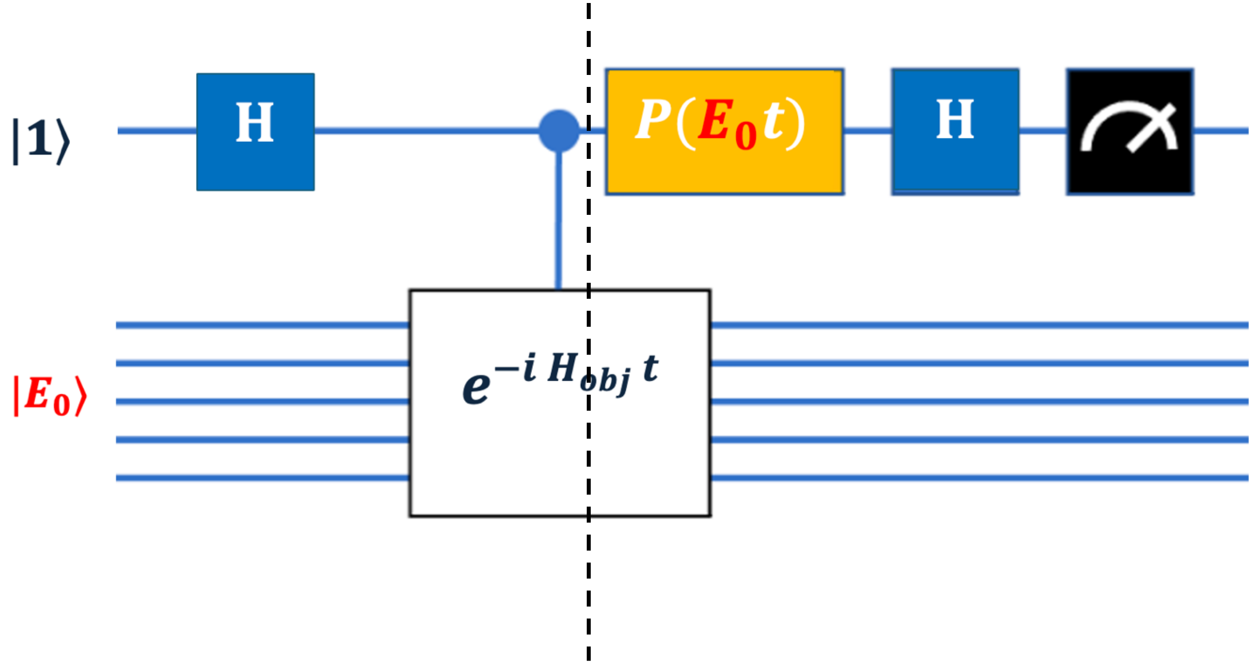
- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle - e^{-i\,E_0 t}|1\rangle\right)|E_0\rangle$$



The RODEO Algorithm

Preliminary Example

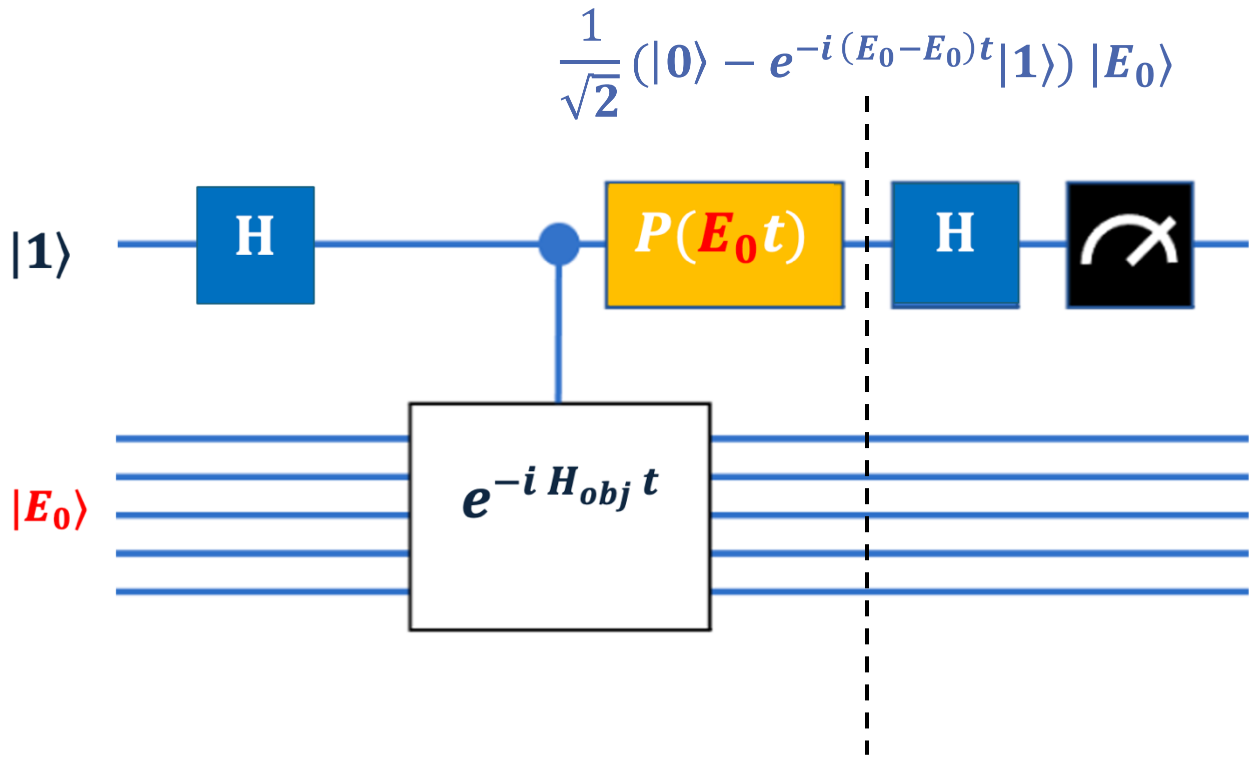
Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i\,|E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E\cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$



The RODEO Algorithm

Preliminary Example

Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

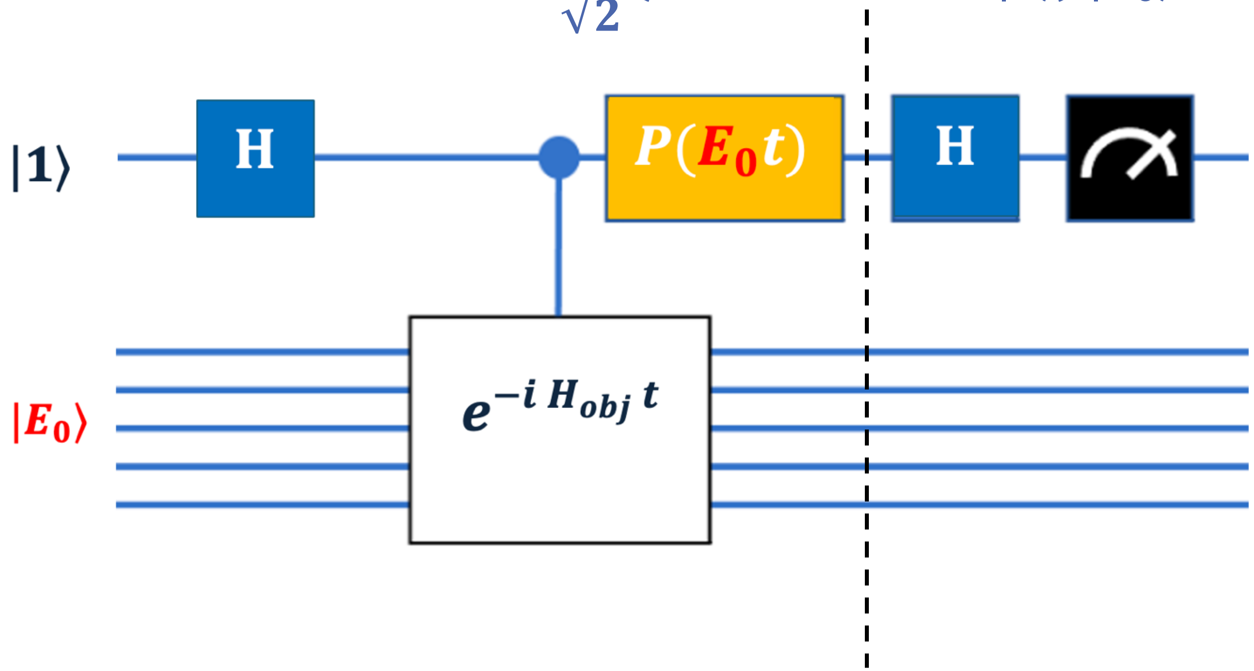
- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\left(|0\rangle - \overbrace{e^{-i(E_0-E_0)t}}^{1}|1\rangle\right)|E_0\rangle$$



The RODEO Algorithm

Preliminary Example

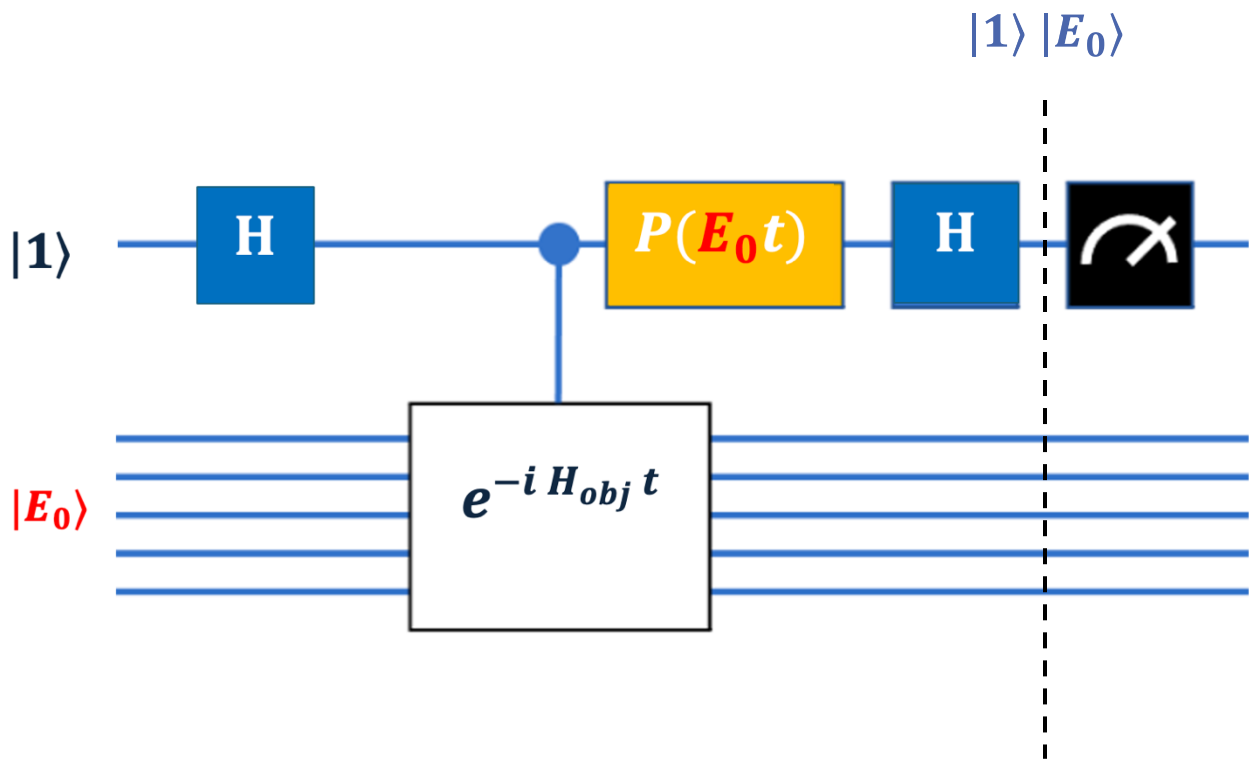
Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$



The RODEO Algorithm

Preliminary Example

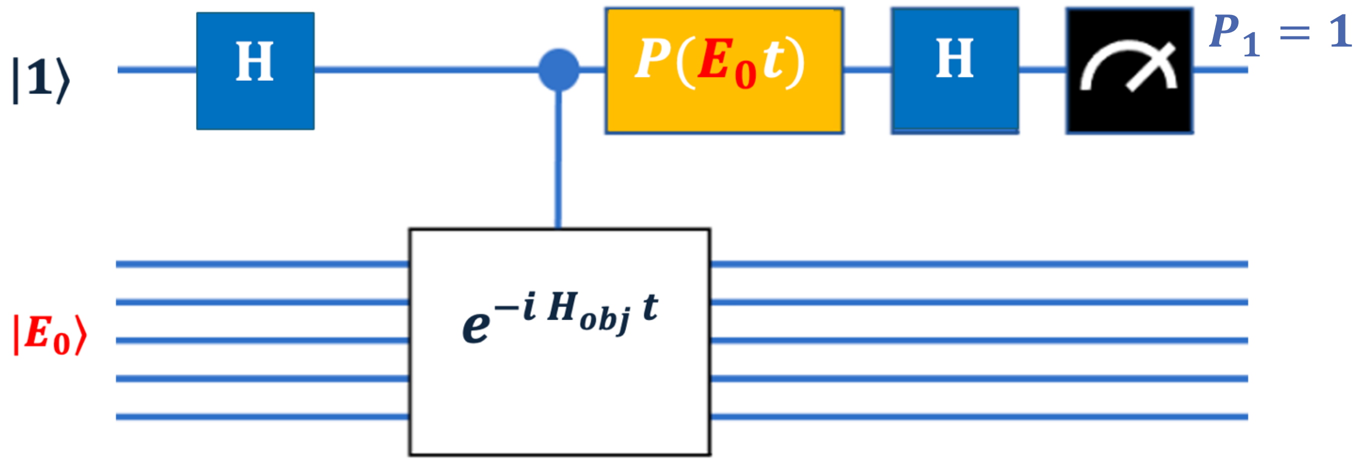
Exploit ***phase kickback*** in controlled-Hamiltonian evolution.

- Objective Hamiltonian:

$$H_{obj} = \sum_i E_i \, |E_i\rangle\langle E_i|$$

- Parametrized Phase Rotation:

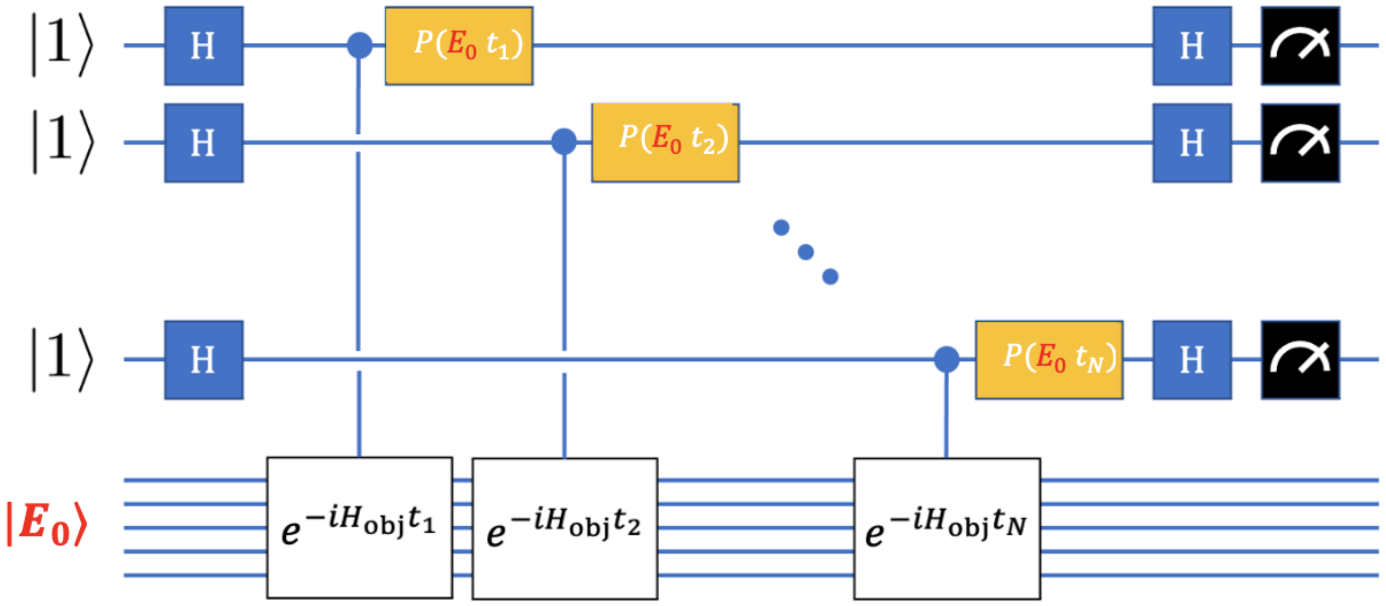
$$P(E \cdot t) = \begin{pmatrix} 1 & 0 \\ 0 & e^{iEt} \end{pmatrix}$$



The RODEO Algorithm

Preliminary Example

- More ***ancillas*** (N) with different ***time parameters*** $\{t_n\}_{n=1}^{N}$

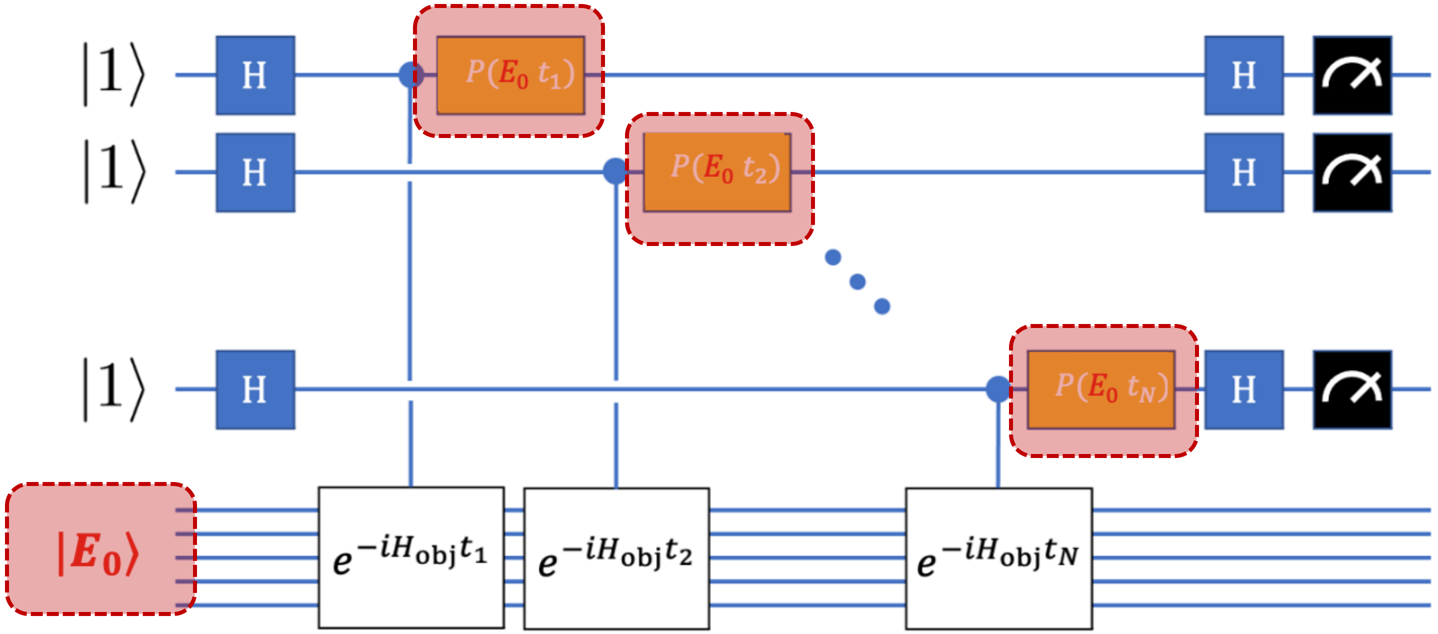


$$P_N = 1$$

The RODEO Algorithm

Preliminary Example

- More ***ancillas*** (N) with different ***time parameters*** $\{t_n\}_{n=1}^N$



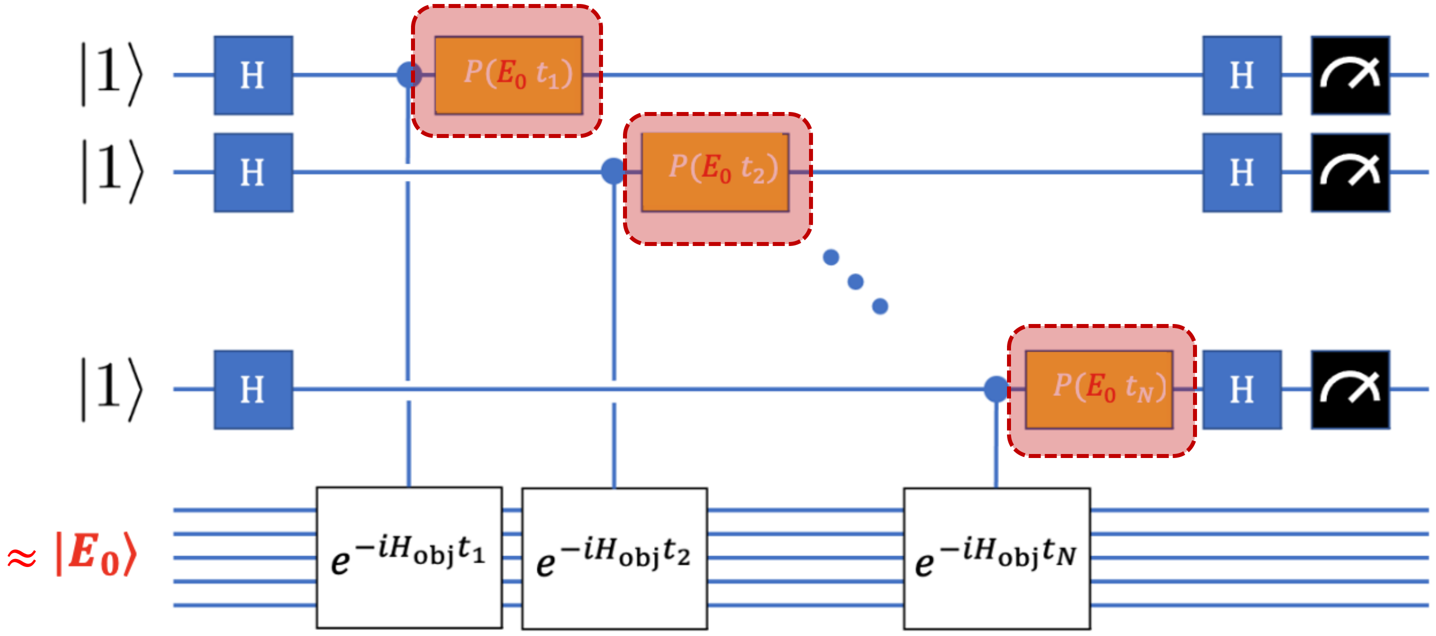
E_0 and $|E_0\rangle$ unknown!

RODEO solves both issues

The RODEO Algorithm

Preliminary Example

- More ***ancillas*** (N) with different ***time parameters*** $\{t_n\}_{n=1}^N$



E_0 **unknown!**

Our Work

RODEO finds E_0

The RODEO Algorithm

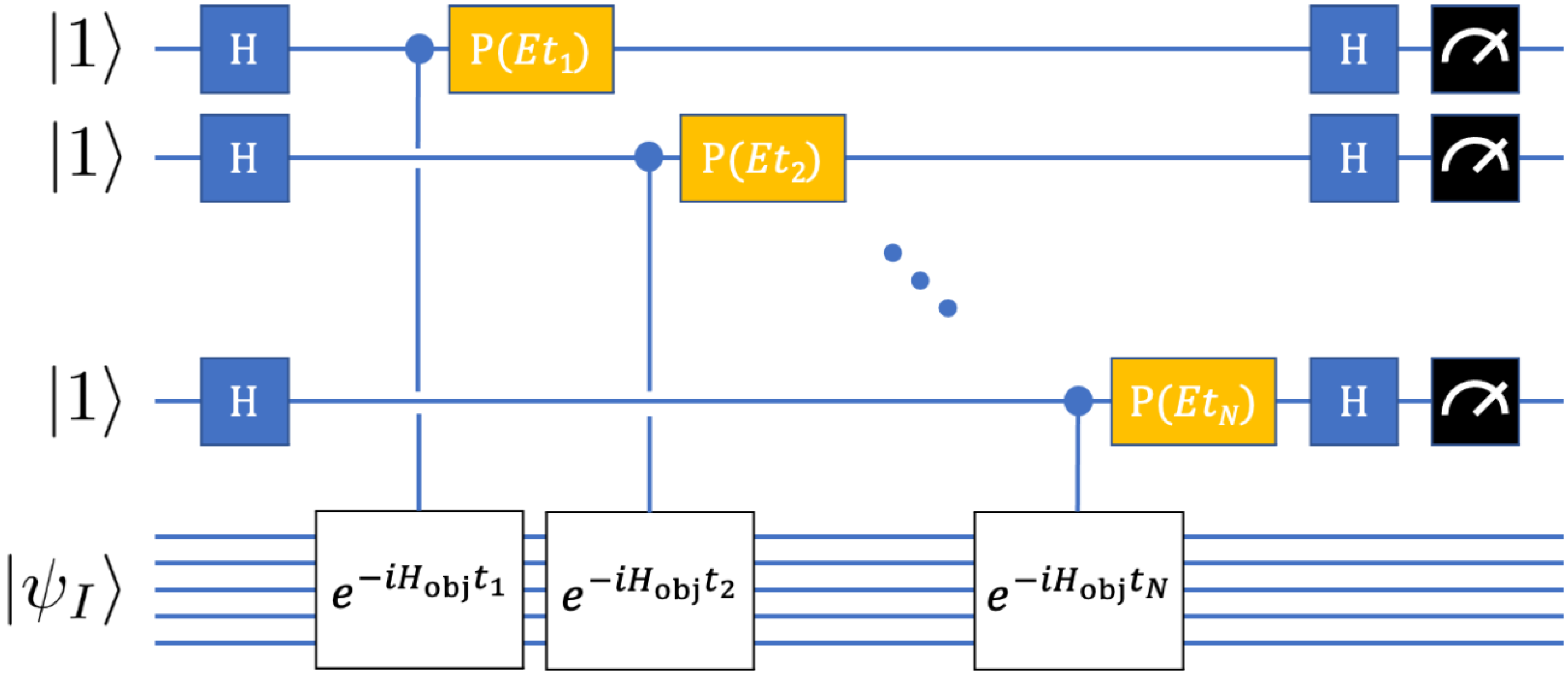
The Circuit

Typical case:

- Known energy ***range*** $E_{obj} \in [E_{min}, E_{max}]$ from $\|H_{obj}\|$
- Initial ***guess*** for the eigenstate $|\psi_I\rangle \approx |E_{obj}\rangle$ (*ansatz*)

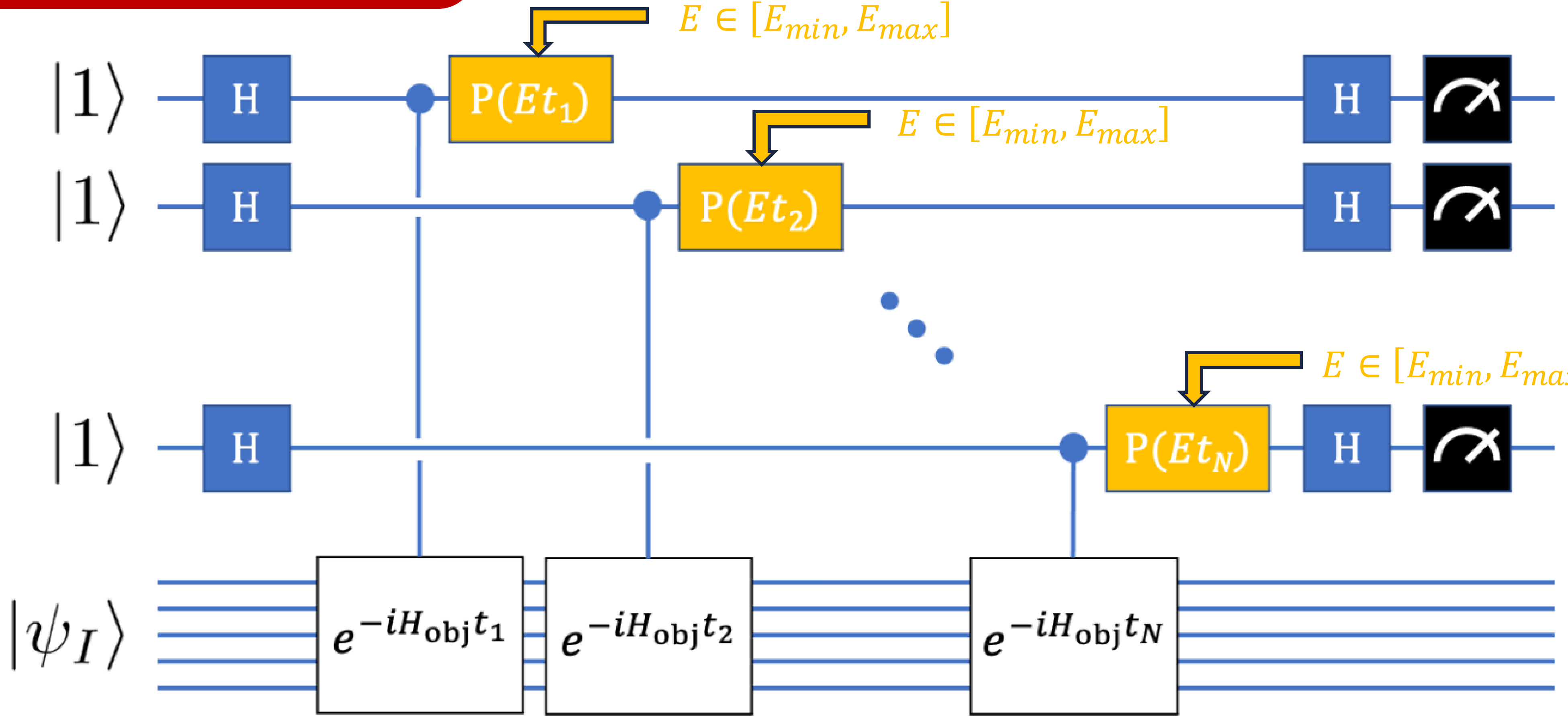
The RODEO Algorithm

The Circuit



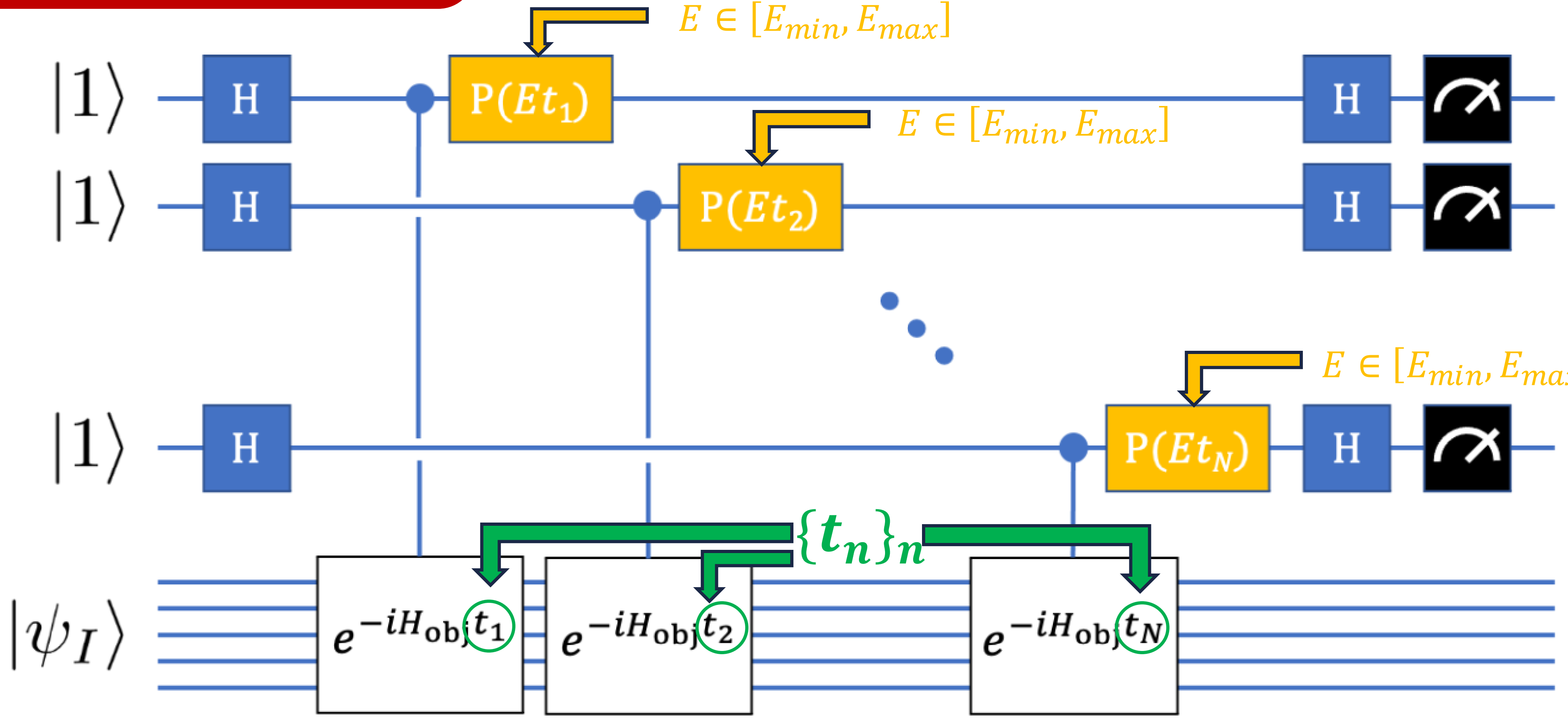
The RODEO Algorithm

The Circuit



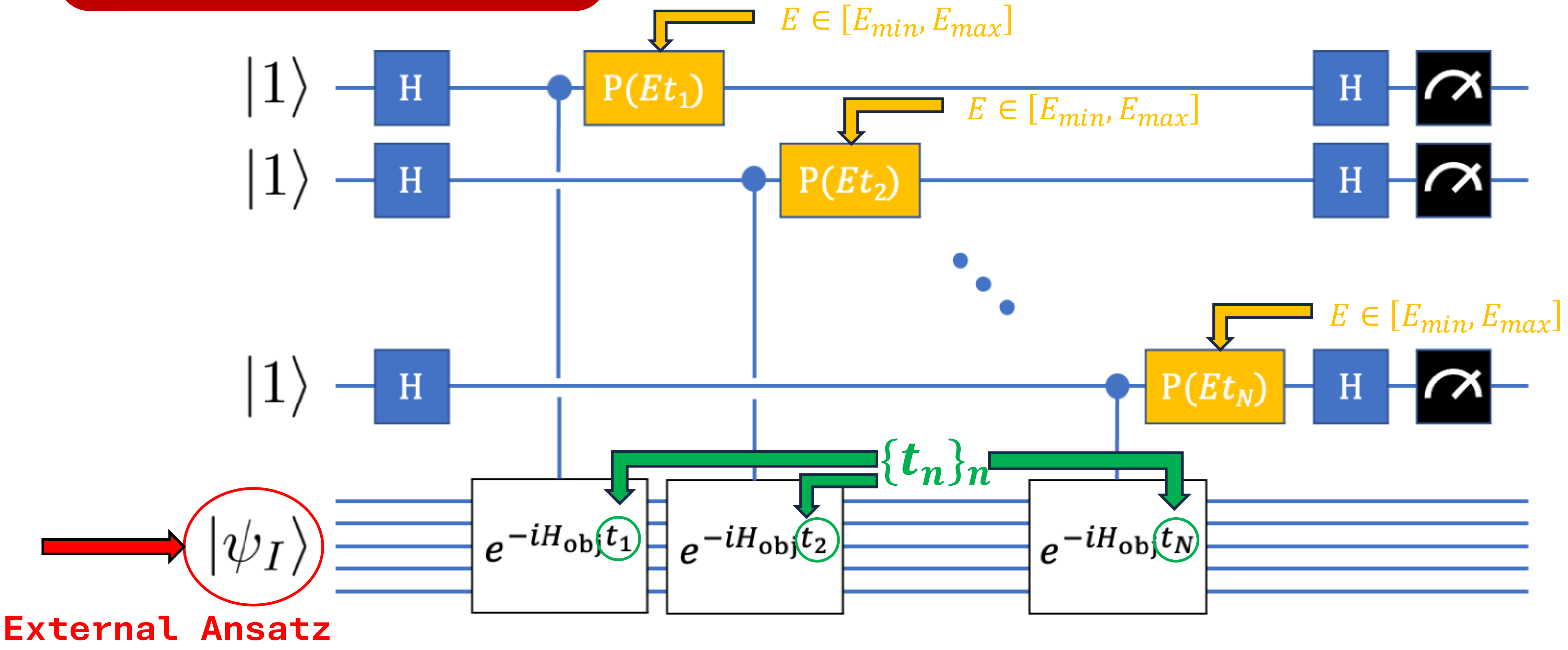
The RODEO Algorithm

The Circuit



The RODEO Algorithm

The Circuit



RODEO: Eigenvalue Determination

Idea

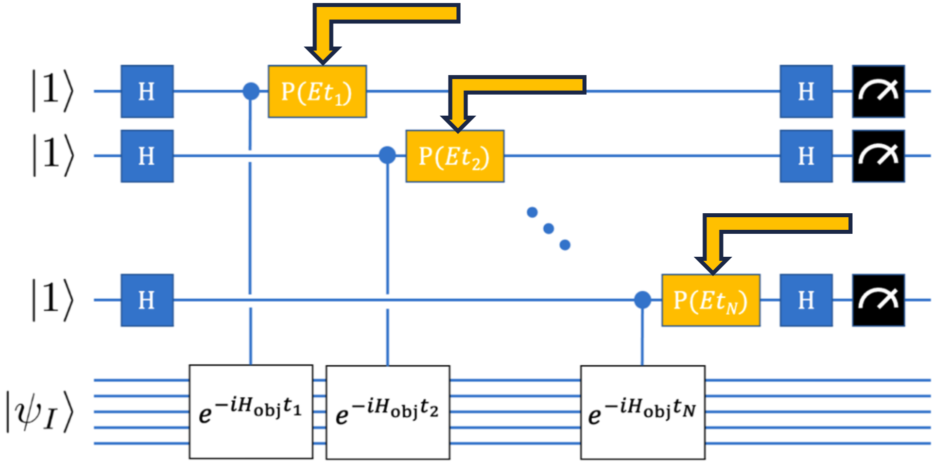
Run the circuit with ***different values of E*** and study measurement outcomes vs E to retrieve the wanted eigenvalue

- Define:

$$P_N = \mathbb{P}(\text{All ancillas in } |1\rangle)$$

- ***Filtering mechanism*** is based on interference:

Peak in P_N $\longleftrightarrow$ $E \approx E_{obj}$



Politecnico di Torino

RODEO: Eigenvalue Determination

Case 1: Input Eigenstate

Simple Case:

$$|\psi_I\rangle = |E_{obj}\rangle, \qquad H_{obj}|E_{obj}\rangle = E_{obj}\,|E_{obj}\rangle$$

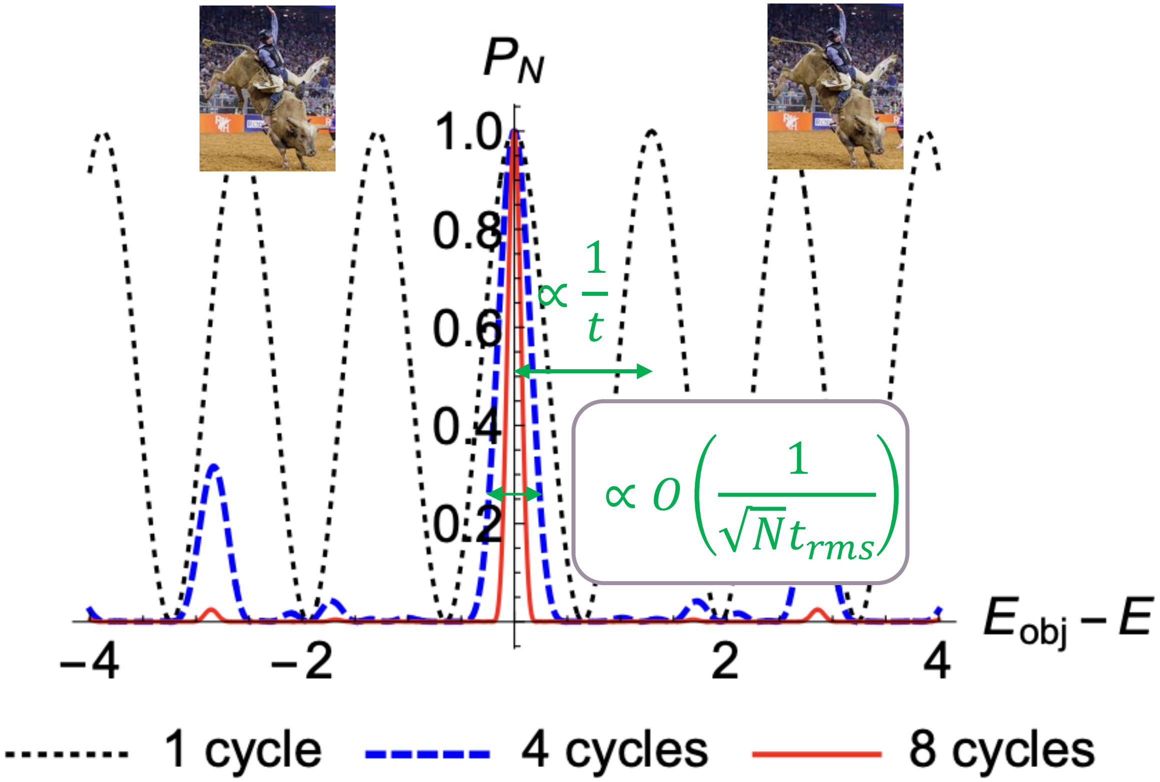
Then:

$$P_N(E) = \prod_{n=1}^{N} \cos^2\left[\frac{(E_{obj}-E)t_n}{2}\right]$$

- ***Energy Filter***: Select t_n according to Gaussian distribution with stdev t_{rms} and increase N (*Rodeo cycle*):

$$P_N(E) = 1 \iff E = E_{obj}$$

- ***Exponential suppression*** of $E \neq E_{obj}$ with N



RODEO: Eigenvalue Determination

Case 2: Generic Input

General case: Initial state(ansatz) $|\psi_I\rangle$:

$$\left|\langle\psi_I|E_{obj}\rangle\right|^2 = p < 1. \quad \mathbf{H}_{obj}|\psi_I\rangle \neq E|\psi_I\rangle$$

Problems:

I . State at target register ***changes*** with Hamiltonian evolution

II. Peak of P_N at $E = E_{obj}$ with ***maximum at p*** instead of 1

Solution:

Run algorithm multiple times with ***increasing*** t_{rms} to improve energy resolution (Heisenberg principle) and isolate peak

Caveat: Strong dependence on initial overlap $p \rightarrow$ ***initial state preparation*** is crucial (see next...)

Politecnico di Torino

RODEO: Eigenvalue Determination

Summary

Parameter tuning is pivotal in the general case:

- Number of Ancillas (N)
- Gaussian Time Distribution (t_{rms})
- Energy range and Spacing (E)
- Input state (ψ_I)

Choice strongly depends on Hamiltonian of interest

- Our case: k −local Hamiltonians

$$\boldsymbol{H} = \sum_{j=1}^{L} \alpha_j H_j$$

Politecnico di Torino

RODEO Implementation

RODEO Implementation

Implementation Intro

PushQuantum – CLASSIQ

Classiq Challange:
Estimating Molecular Spectra Using the Rodeo Algorithm

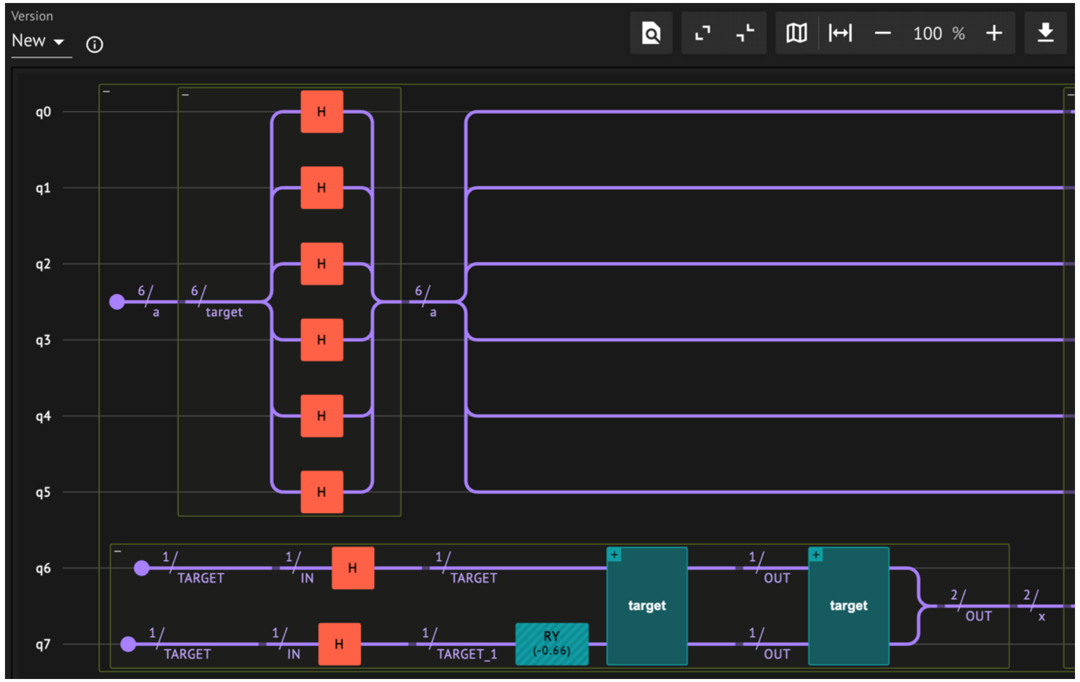
RODEO Implementation

Implementation Intro

CLASSIQ

```python
def main(a: Output[QArray], x: Output[QArray] ... ):
        """
        Quantum function.
        """
        prepare_amplitudes(
          amplitudes=initial_vec,
          bound=0.01,
          out=x
        )

        allocate(N, a)
        hadamard_transform(a)
```

High level code

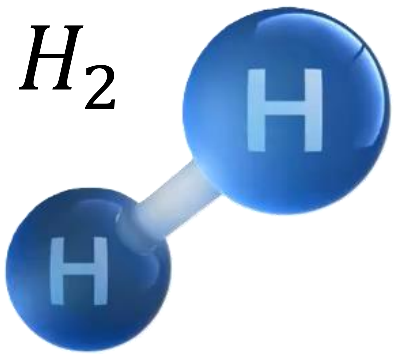


Quantum circuit

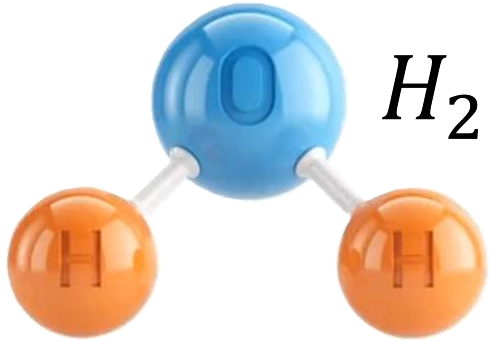
RODEO Implementation

Test-case

H_2



$$H = \sum_{j=1}^{L} \alpha_j H_j$$



H_2O

```
HAMILTONIAN = [
    PauliTerm([Pauli.I, Pauli.I], -1.0523),
    PauliTerm([Pauli.I, Pauli.Z], 0.3979),
    PauliTerm([Pauli.Z, Pauli.I], -0.3979),
    PauliTerm([Pauli.Z, Pauli.Z], -0.0112),
    PauliTerm([Pauli.X, Pauli.X], 0.1809),
]
```

```
HAMILTONIAN = [
    PauliTerm([Pauli.I, Pauli.I, Pauli.I, Pauli.I, Pauli.I, Pauli.I], -12.533),
    PauliTerm([Pauli.Z, Pauli.I, Pauli.I, Pauli.Z, Pauli.I, Pauli.I], -1.276),
    PauliTerm([Pauli.Z, Pauli.Z, Pauli.I, Pauli.I, Pauli.I, Pauli.I], 0.627),
    PauliTerm([Pauli.I, Pauli.Z, Pauli.I, Pauli.I, Pauli.Z, Pauli.I], -0.875),
    PauliTerm([Pauli.I, Pauli.I, Pauli.Z, Pauli.Z, Pauli.I, Pauli.I], 0.452),
    PauliTerm([Pauli.X, Pauli.I, Pauli.X, Pauli.I, Pauli.I, Pauli.I], 0.182),
    PauliTerm([Pauli.I, Pauli.X, Pauli.I, Pauli.X, Pauli.I, Pauli.I], 0.139),
    PauliTerm([Pauli.Y, Pauli.Y, Pauli.I, Pauli.I, Pauli.I, Pauli.I], -0.047),
    PauliTerm([Pauli.Z, Pauli.I, Pauli.Z, Pauli.I, Pauli.Z, Pauli.I], 0.209),
    PauliTerm([Pauli.Z, Pauli.Z, Pauli.Z, Pauli.Z, Pauli.I, Pauli.I], -0.154),
    PauliTerm([Pauli.I, Pauli.Z, Pauli.I, Pauli.Z, Pauli.Z, Pauli.Z], 0.198),
    PauliTerm([Pauli.X, Pauli.I, Pauli.I, Pauli.I, Pauli.X, Pauli.I], 0.061),
    PauliTerm([Pauli.I, Pauli.I, Pauli.Y, Pauli.I, Pauli.Y, Pauli.I], -0.027),
    PauliTerm([Pauli.Z, Pauli.I, Pauli.Z, Pauli.Z, Pauli.I, Pauli.Z], 0.118),
]
```

RODEO Implementation

Setup and Strategy

language

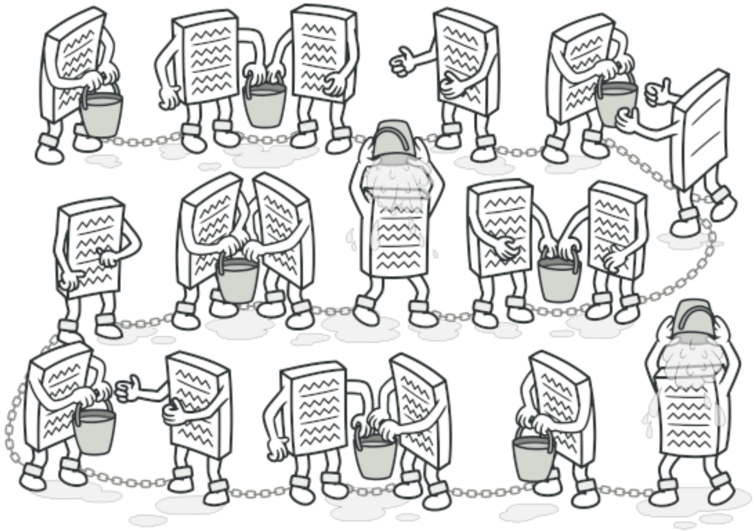
platform

IDE

VCS

Design Pattern

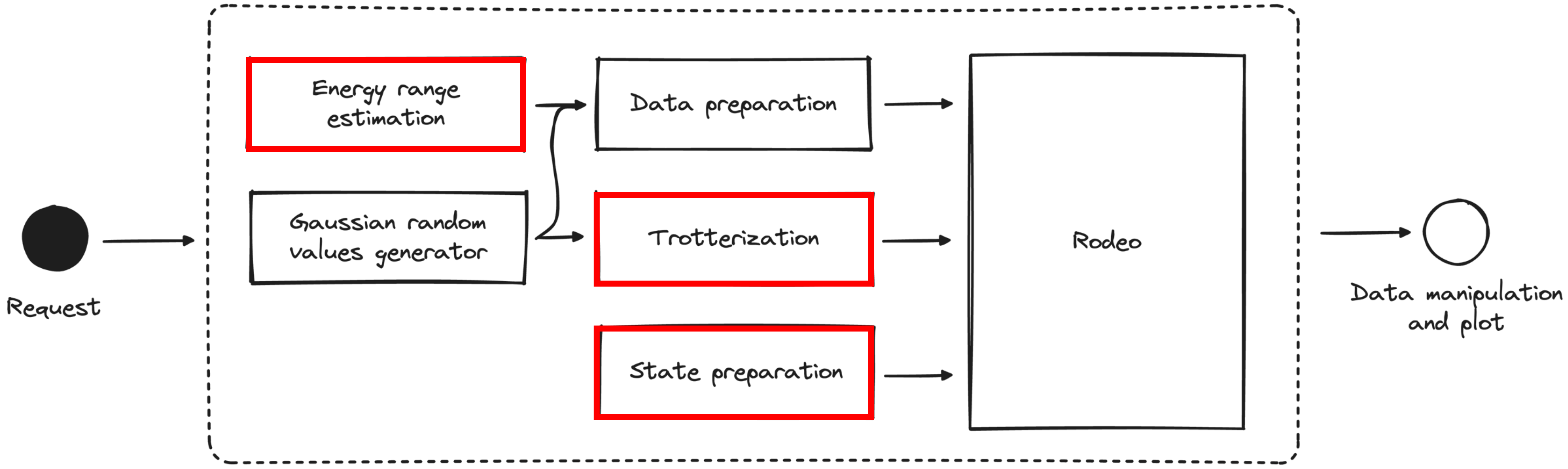
Chain of responsability



https://github.com/Los-Pollos-Quanticos/PushQuantumHackaton2024

RODEO Implementation

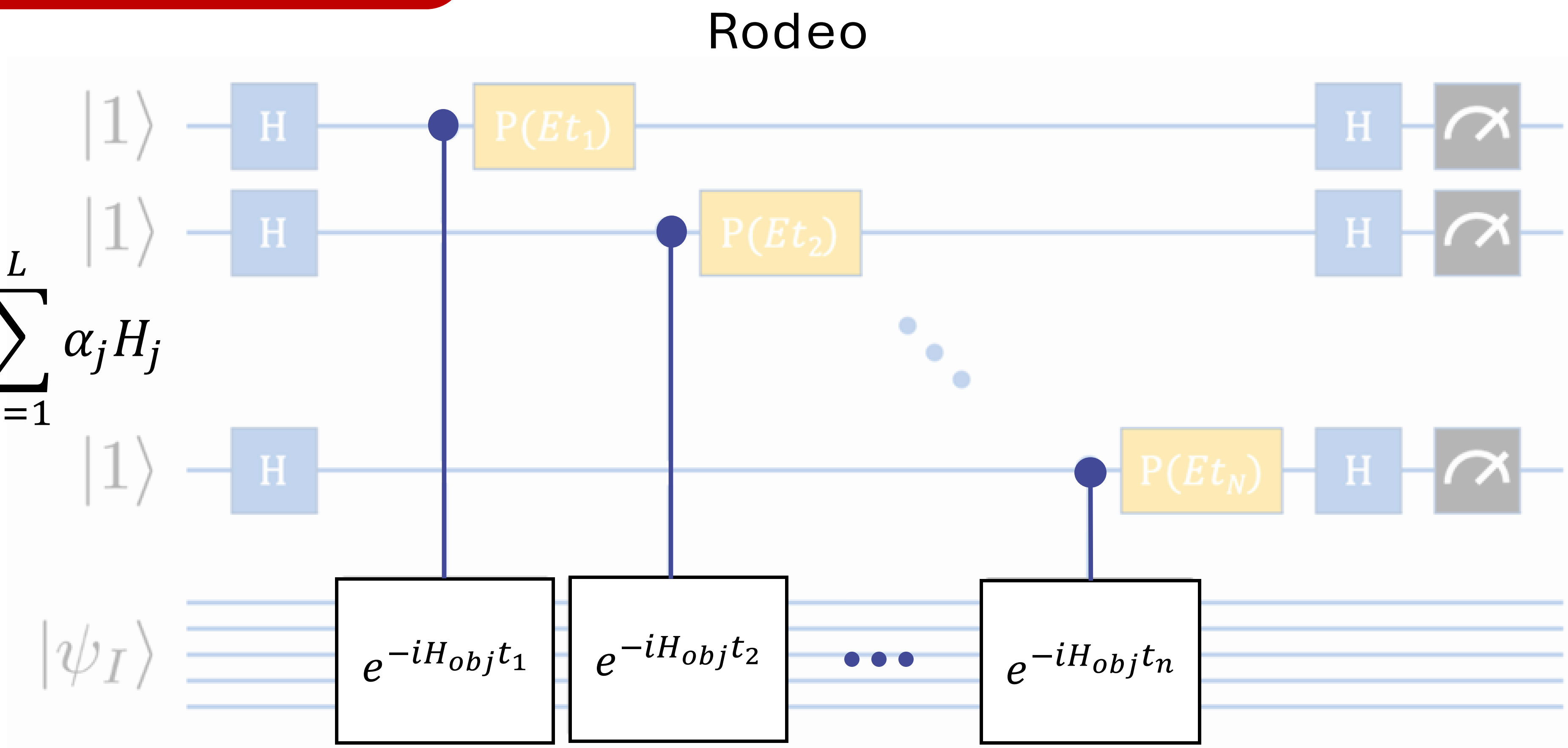
Code structure



RODEO Implementation

Suzuki-Lie Trotter

$$H = \sum_{j=1}^{L} \alpha_j H_j$$



RODEO Implementation

Suzuki-Lie Trotter

product formula

$$H = \sum_{j=1}^{L} \alpha_j H_j \qquad \boxed{e^{-i\sum_{j=1}^{L}\alpha_j H_j t}} \neq \boxed{e^{-i\alpha_1 H_1 t}} \bullet \boxed{e^{-i\alpha_2 H_2 t}} \bullet \cdots \bullet \boxed{e^{-i\alpha_L H_L t}}$$

H_j are non-commuting terms

RODEO Implementation

Suzuki-Lie Trotter

Suzuki-Lie Trotter

$$H = \sum_{j=1}^{L} \alpha_j H_j \qquad \boxed{e^{-i\sum_{j=1}^{L} \alpha_j H_j t}} \approx \underbrace{\boxed{e^{-i\alpha_1 H_1 \frac{t}{r}}} \bullet \boxed{e^{-i\alpha_2 H_2 \frac{t}{r}}} \bullet \cdots \bullet \boxed{e^{-i\alpha_L H_L \frac{t}{r}}}}_{\text{r times}}$$

$$O\left(\frac{(L\Lambda t)^2}{r}\right)$$

$\Lambda \coloneqq \max_j \alpha_j$ $\quad$ L: # of terms

t: evolution time $\quad$ r: # of repetition

RODEO Implementation

Rodeo Test

$$P_A(E) = \prod_{n=1}^{A} \cos^2\left[\frac{(E_i - E)t_n}{2}\right]$$

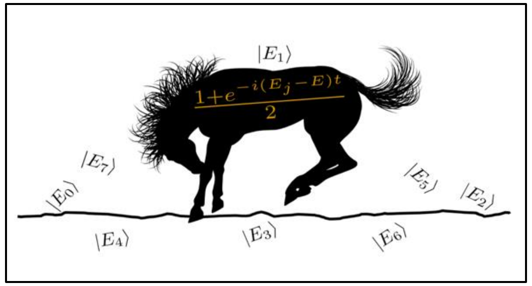
$$|\langle\Psi|\psi_i\rangle|^2 = p = \frac{1}{N}$$

$$H|\psi_i\rangle = E_i|\psi_i\rangle$$

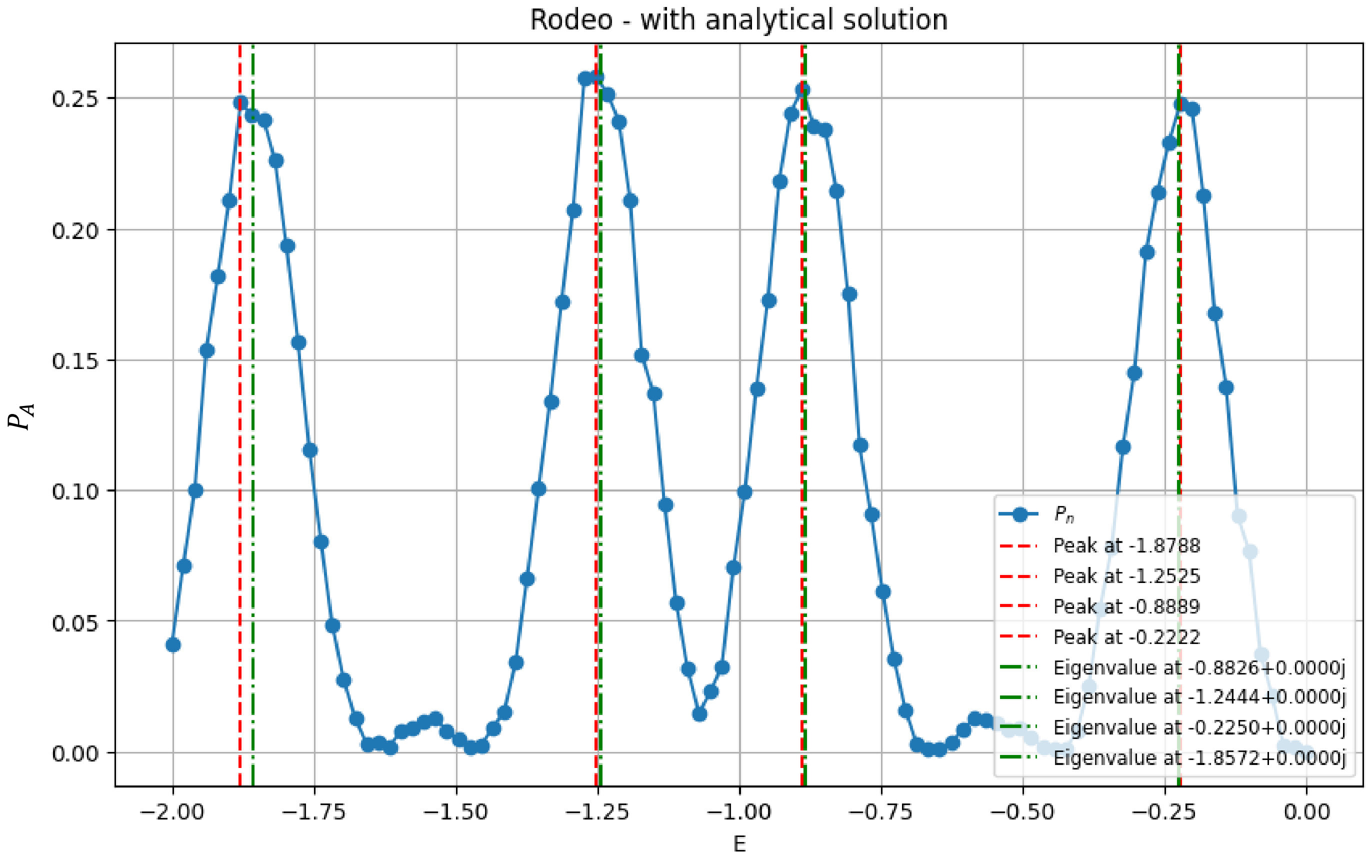
$$\text{for} \quad i = 0, 1, \ldots, N-1.$$

$$[E_0, E_{N-1}]$$

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\psi_i\rangle.$$



A=6, TRMS=7, SAMPLES=100



RODEO Implementation

THIS GUY CAN HELP US!

OK!

HOW DO WE SET THE INITIAL STATE?

HOW DO WE SET THE ENERGY PROPERLY?

works!

RODEO Implementation

Gershgoring Circle Theorem

- **Hypothesis:** let A be a complex n x n matrix with entries a_{ij}. For i $\in \{1 \dots .n\}$ let R_i be the sum of the modulus of the non diagonal entries in the i-th row:

$$R_i = \Sigma_{j \neq i} |a_{ij}|$$

Let $D(a_{ii}, R_i) \subseteq \mathbb{C}$ be a closed disk centered at a_{ii} with radius R_i. Such a disk is called a Gershgorin disk.

- **Statement:** every eigenvalue of A lies within at least one of the Gershgoring disks $D(a_{ii}, R_i)$

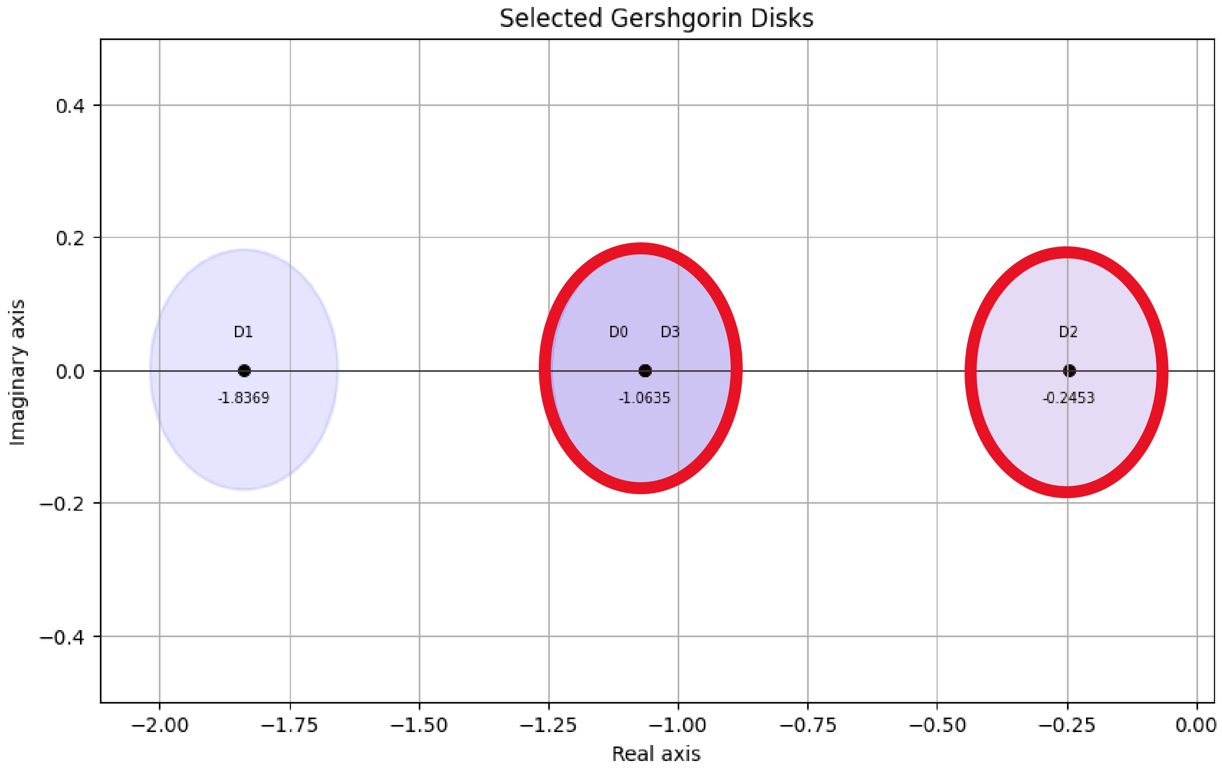
RODEO Implementation

Gershgoring Circle Theorem: H2 Scenario

$$D_0(-1.0635, 0.1809)$$

$$\begin{bmatrix} -1.0635 & 0 & 0 & 0.1809 \\ 0 & -1.8369 & 0.1809 & 0 \\ 0 & 0.1809 & -0.2453 & 0 \\ 0.1809 & 0 & 0 & -1.0635 \end{bmatrix}$$

$$D_2(-0.2453, 0.1809)$$



RODEO Implementation

Gershgoring Circle Theorem: Initial state selection

Key remarks:

- Each Gershgorin disk corresponds to a row of the Hamiltonian.
- The Hamiltonian is diagonally dominant!

$$
\begin{array}{cc}
|00\rangle & \\
|01\rangle & \\
|10\rangle & \\
|11\rangle &
\end{array}
\begin{bmatrix}
\underline{-1.0635} & 0 & 0 & 0.1809 \\
0 & \underline{-1.8369} & 0.1809 & 0 \\
0 & 0.1809 & \underline{-0.2453} & 0 \\
0.1809 & 0 & 0 & \underline{-1.0635}
\end{bmatrix}
$$

Solution:

- The input state for the RODEO will be the computational basis state corresponding to the row index.

Benefits:

- ✓ This ensures a nonzero overlap between input state and associated eigenstate, filtering the desired eigenvalue in the Rodeo algorithm.
- ✓ This approach narrows the energy search range within the bounds of each disk.

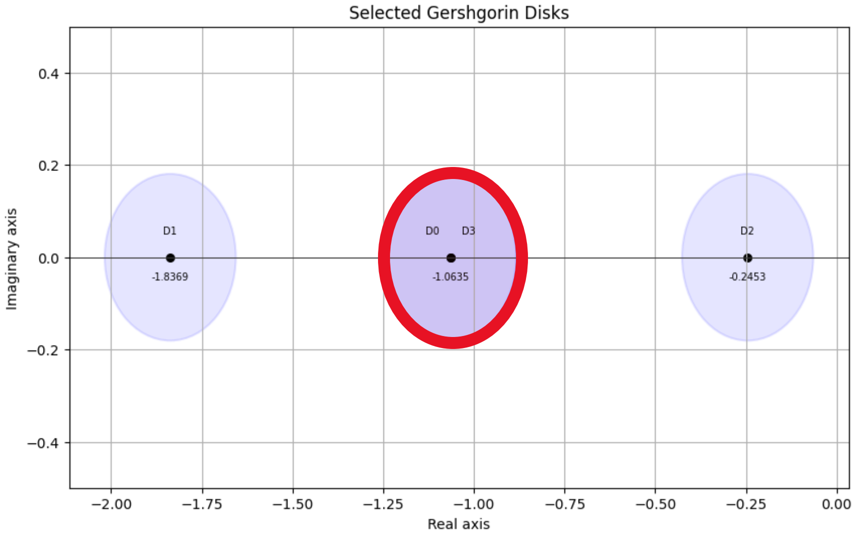
RODEO Implementation

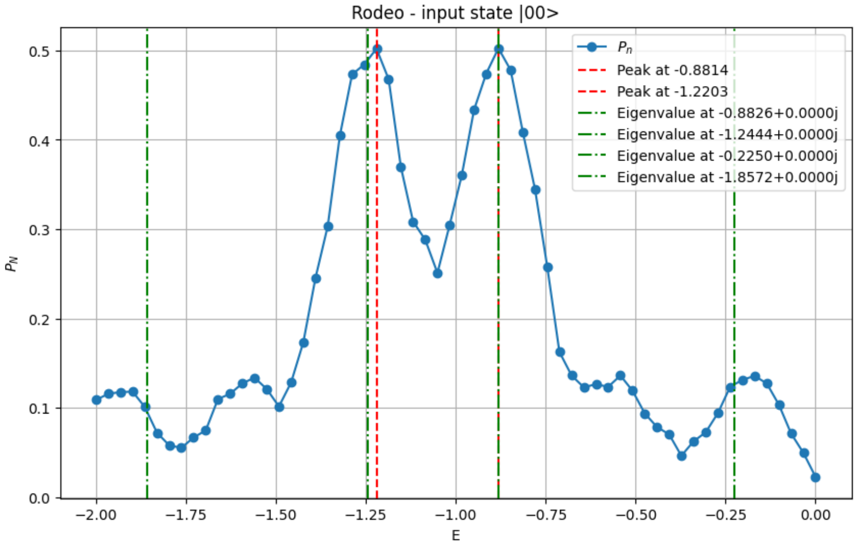
Gershgoring Circle Theorem: H2 results

- **Input state |00>**

$$\begin{bmatrix} -1.0635 & 0 & 0 & 0.1809 \\ 0 & -1.8369 & 0.1809 & 0 \\ 0 & 0.1809 & -0.2453 & 0 \\ 0.1809 & 0 & 0 & -1.0635 \end{bmatrix}$$

→ |00> State





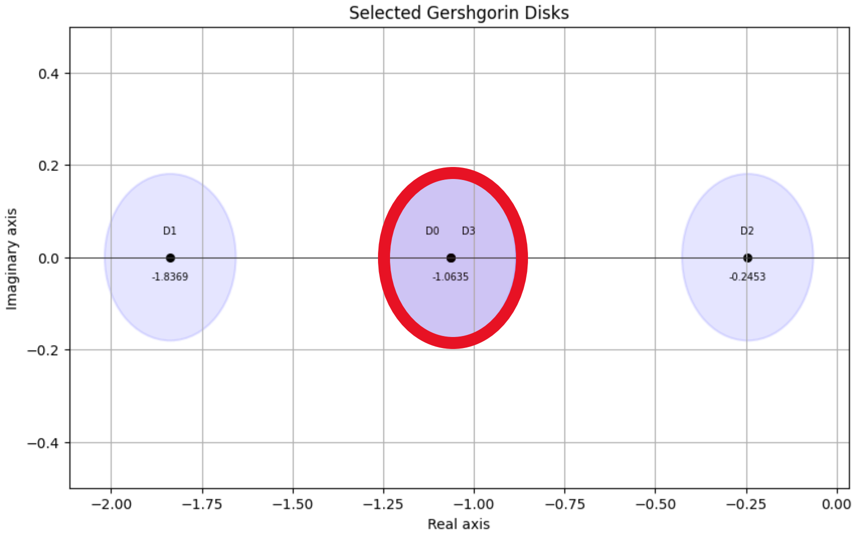
RODEO Implementation

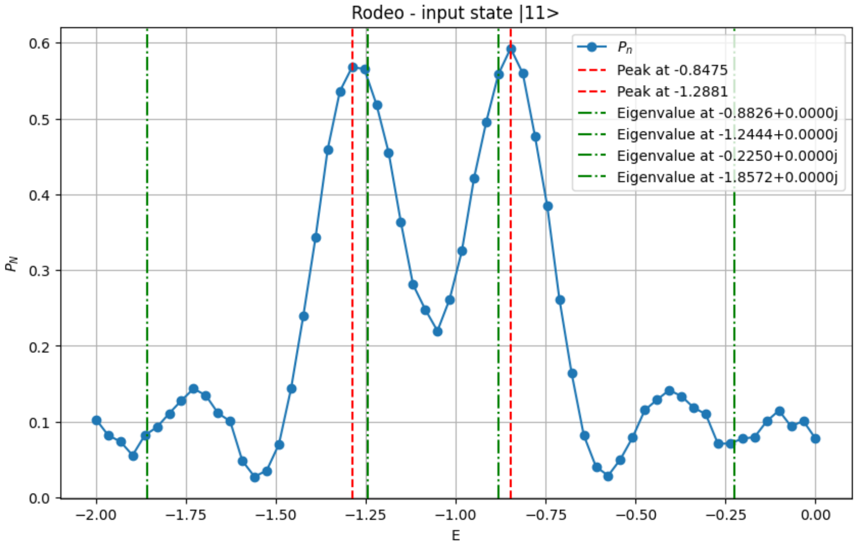
Gershgoring Circle Theorem: H2 results

■ **Input state |11>**

[-1.0635	0	0	0.1809]
[0	-1.8369	0.1809	0]
[0	0.1809	-0.2453	0]
[0.1809	0	0	-1.0635]

→ |11> State



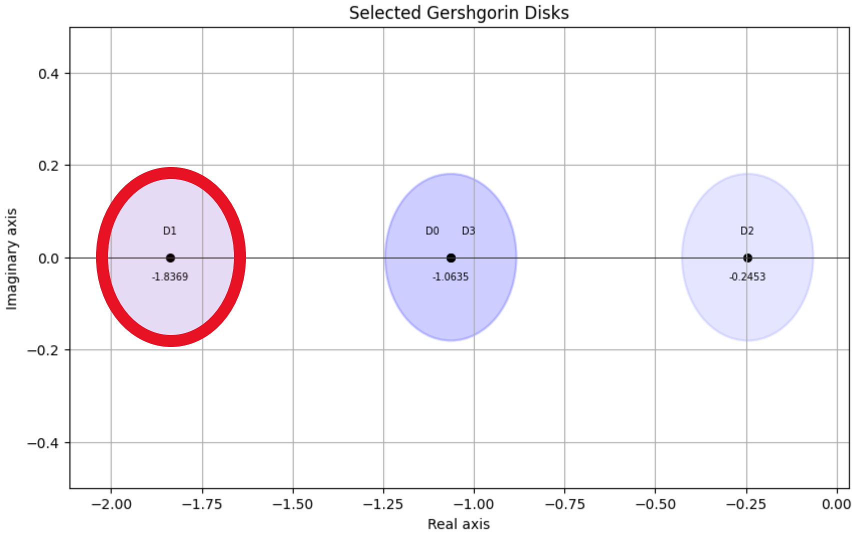


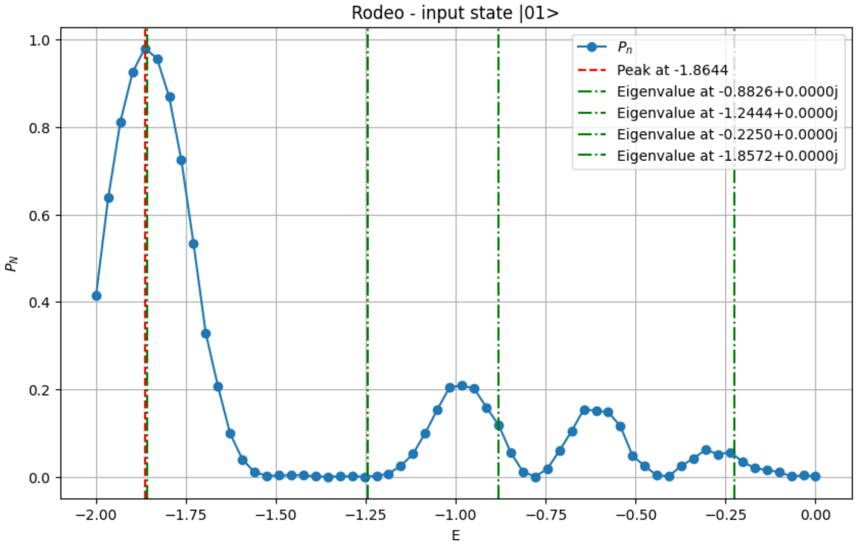
RODEO Implementation

Gershgoring Circle Theorem: H2 results

- **Input state |01>**

$$\begin{bmatrix} -1.0635 & 0 & 0 & 0.1809 \\ 0 & -1.8369 & 0.1809 & 0 \\ 0 & 0.1809 & -0.2453 & 0 \\ 0.1809 & 0 & 0 & -1.0635 \end{bmatrix} \longrightarrow |01\rangle \text{ State}$$





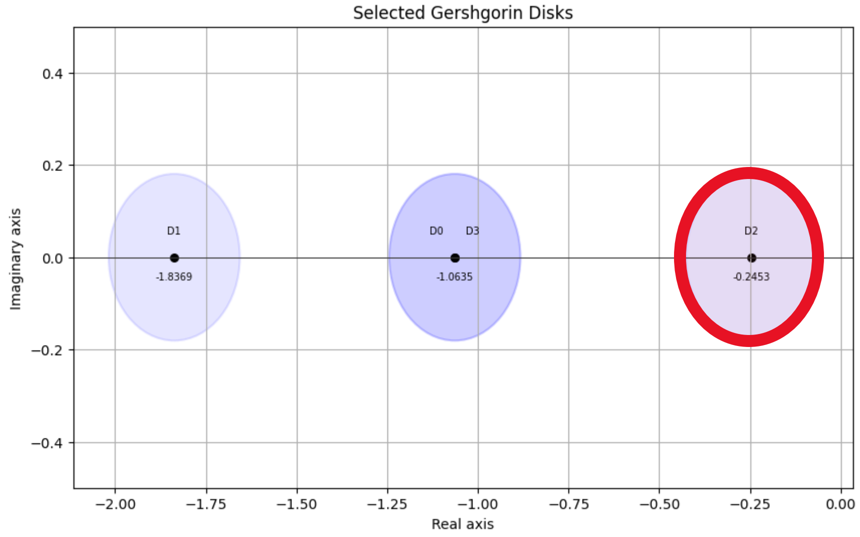
RODEO Implementation

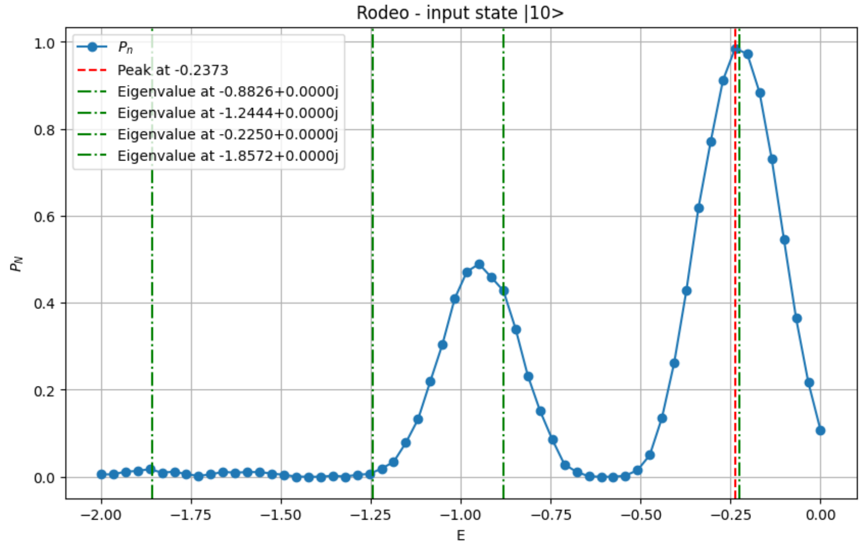
Gershgoring Circle Theorem: H2 results

- **Input state |10>**

$$\begin{bmatrix} -1.0635 & 0 & 0 & 0.1809 \\ 0 & -1.8369 & 0.1809 & 0 \\ 0 & 0.1809 & -0.2453 & 0 \\ 0.1809 & 0 & 0 & -1.0635 \end{bmatrix}$$

→ |10> State





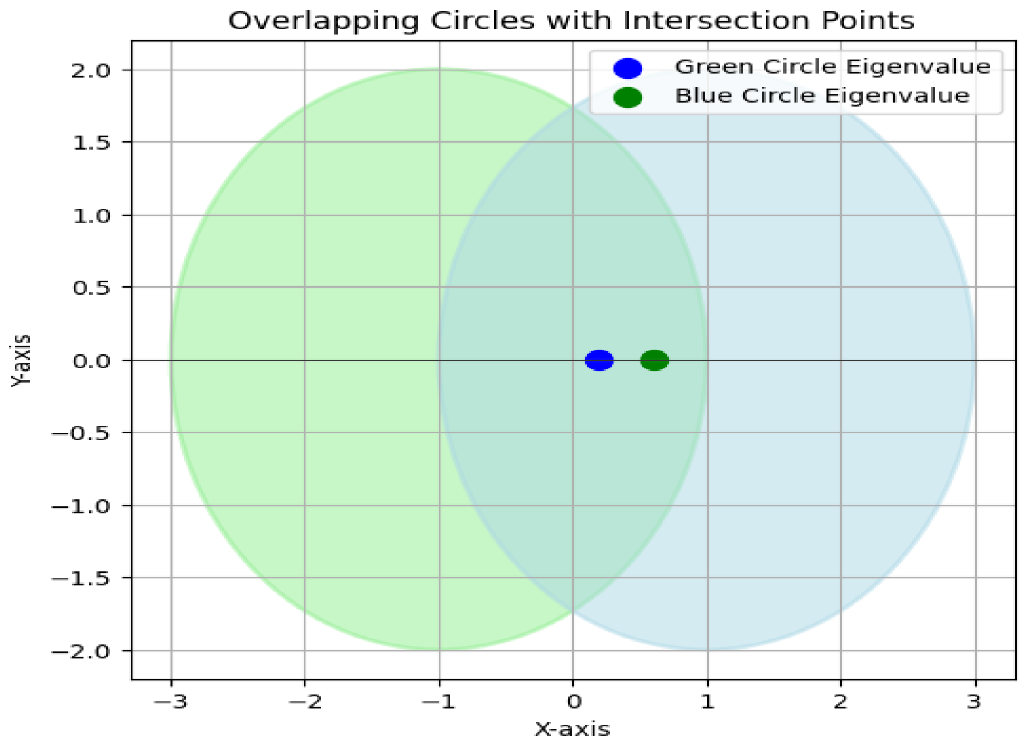
RODEO Implementation

Gershgoring Circle Theorem: H20 Scenario

Task: Find the 5 lowest eigenvalues of the H20 molecule

- Calculation and plot of the Gergshgoring disks.
- Recognition of corresponding computational basis state to give as input for Rodeo Algorithm.

Pathologic situation



RODEO Implementation

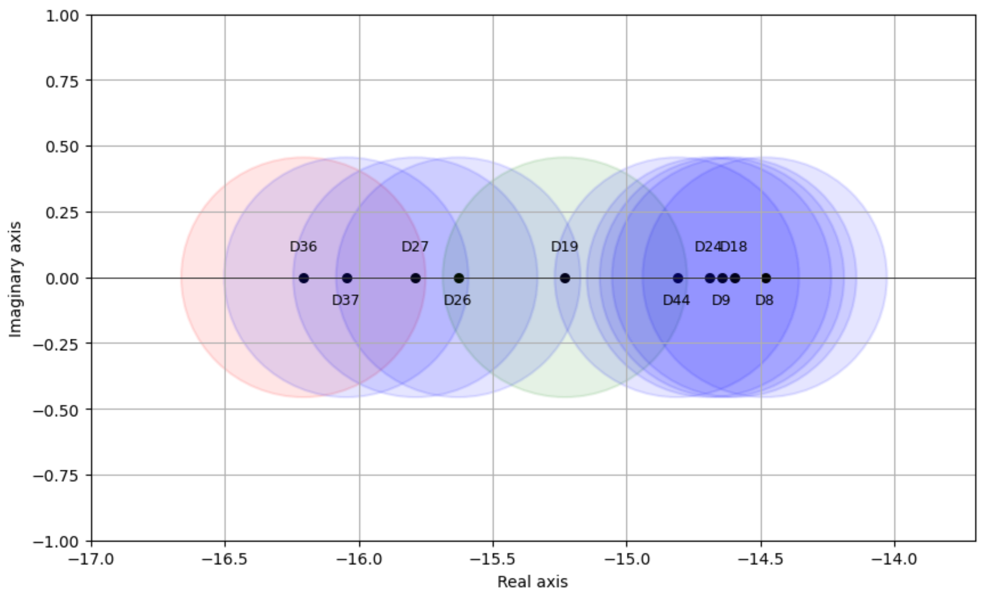
Gershgoring Circle Theorem: H20 Scenario

Main problem:

- Overlapping Gershgorin disks can lead to unknown eigenvalue order.
- The lowest eigenvalues do not necessarily align with the first disks in the E axis.

Strategy

- Sort all Gershgoring disks by their lower bound.
- Start with the first disk “A” and collect all overlapping disk.
- Move to the next non overlapping disk “B” and repeat the collection process.

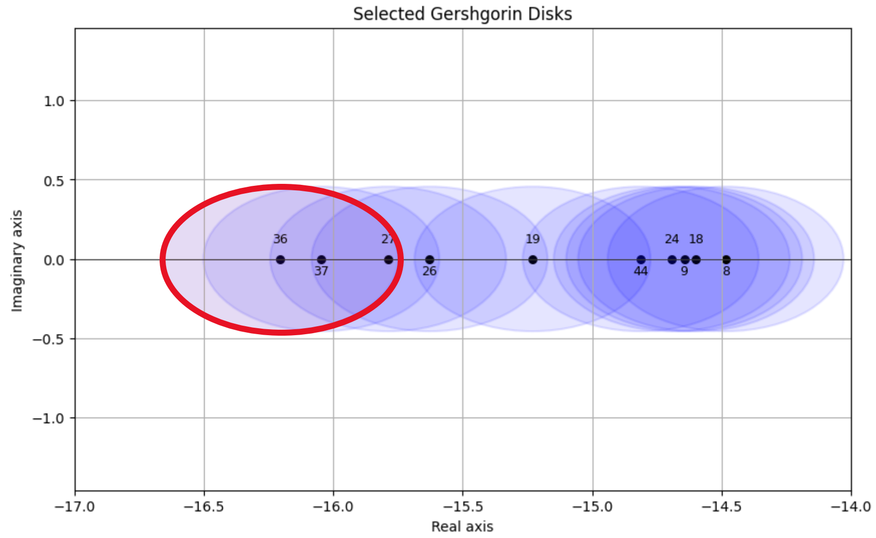


- Stop when five disks are collected.

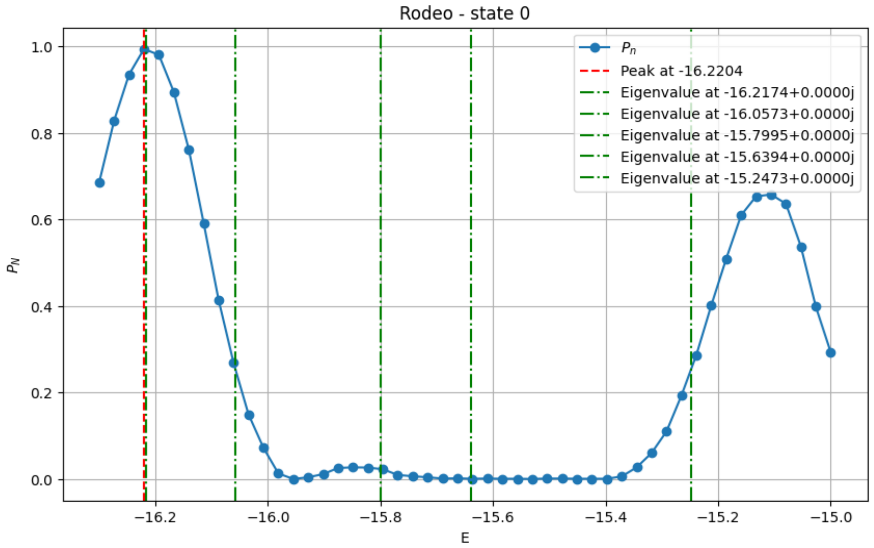
RODEO Implementation

Gershgoring Circle Theorem: H20 Results

- N = 4, RMS = 7: the eigenvalue peak is distinguishable from minor ones.
- Other small peaks appear just due to the periodicity of $P_N(E)$.



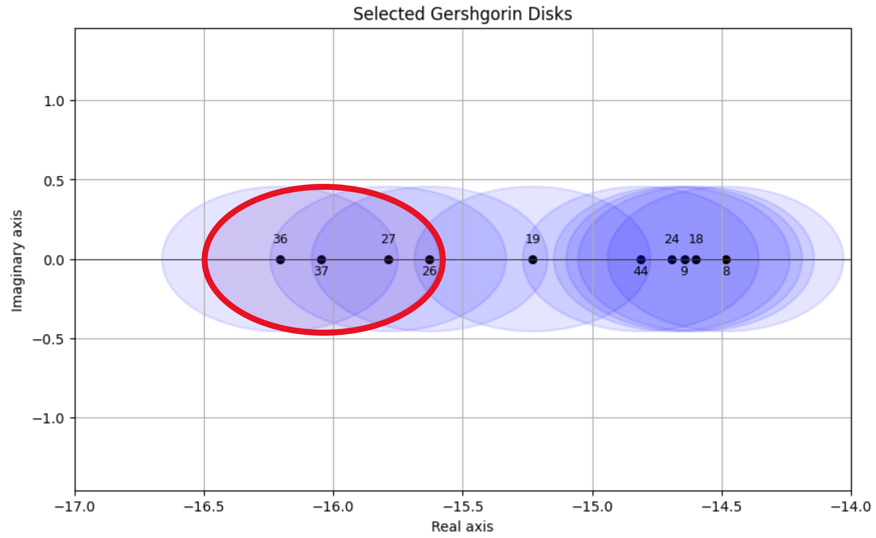
State corresponding to row 36:
|100100>



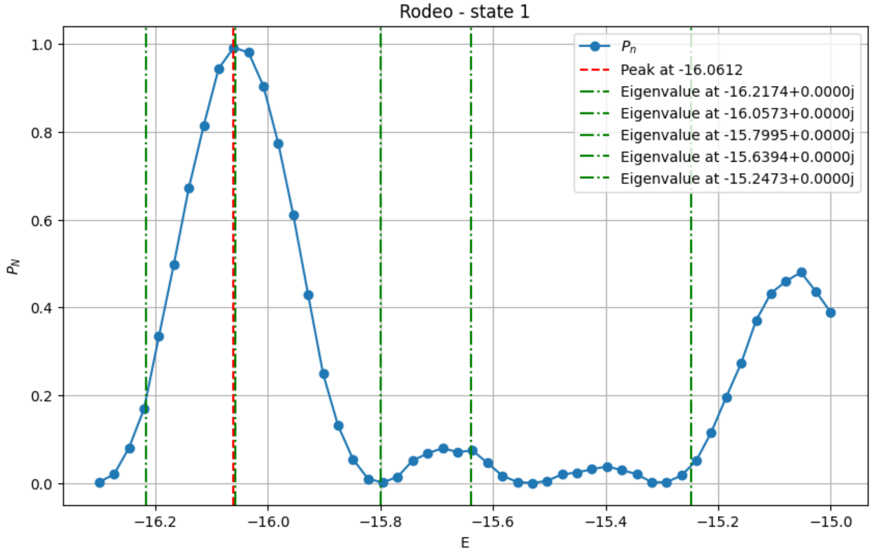
RODEO Implementation

Gershgoring Circle Theorem: H20 Results

- N = 4, RMS = 7: the eigenvalue peak is distinguishable from minor ones.
- Other small peaks appear just due to the periodicity of $P_N(E)$.



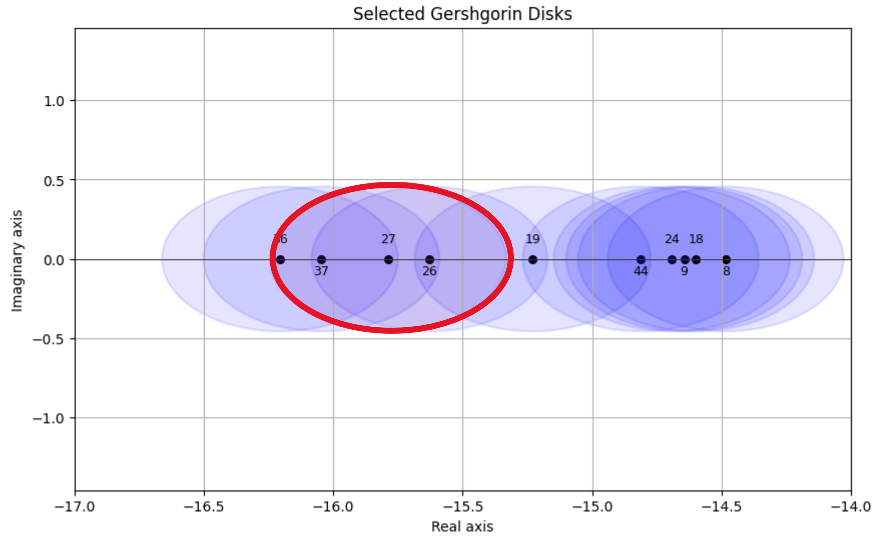
State corresponding to row 37: |100101>



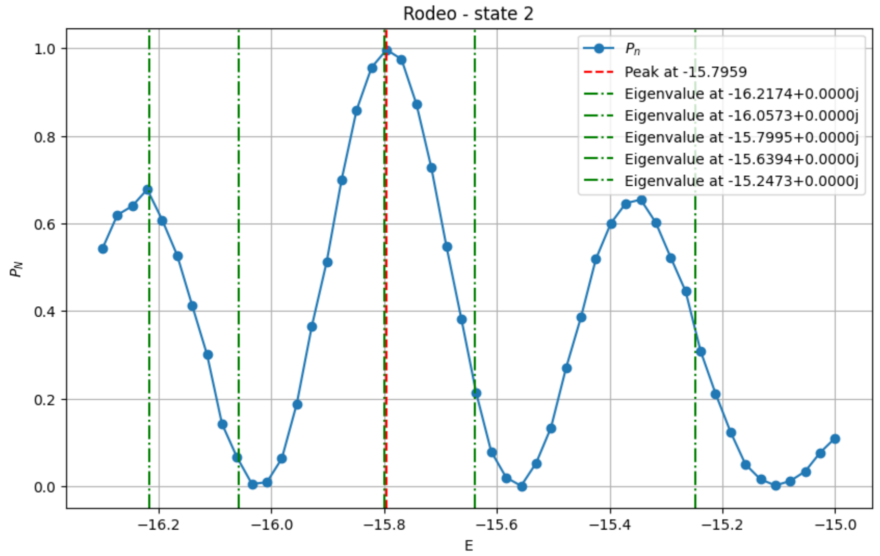
RODEO Implementation

Gershgoring Circle Theorem: H20 Results

- N = 4, RMS = 7: the eigenvalue peak is distinguishable from minor ones.
- Other small peaks appear just due to the periodicity of $P_N(E)$.



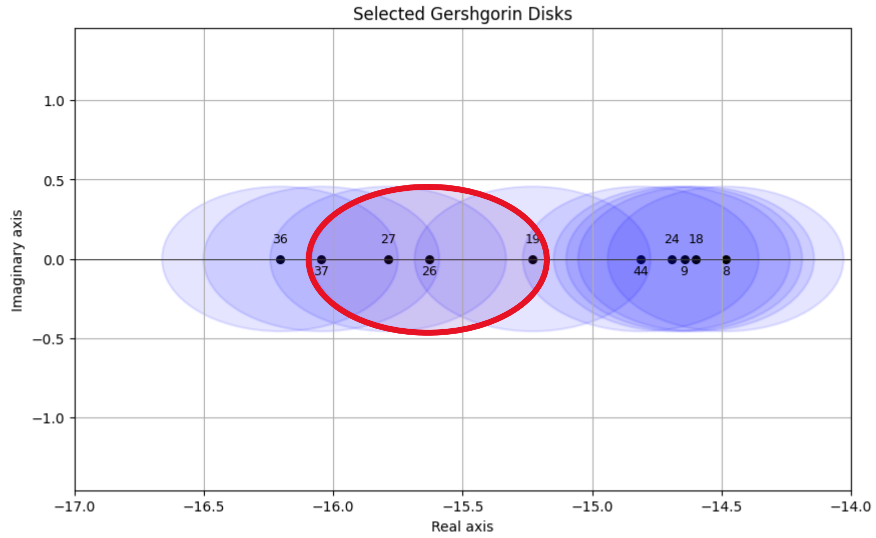
State corresponding to row 27:
|011011>



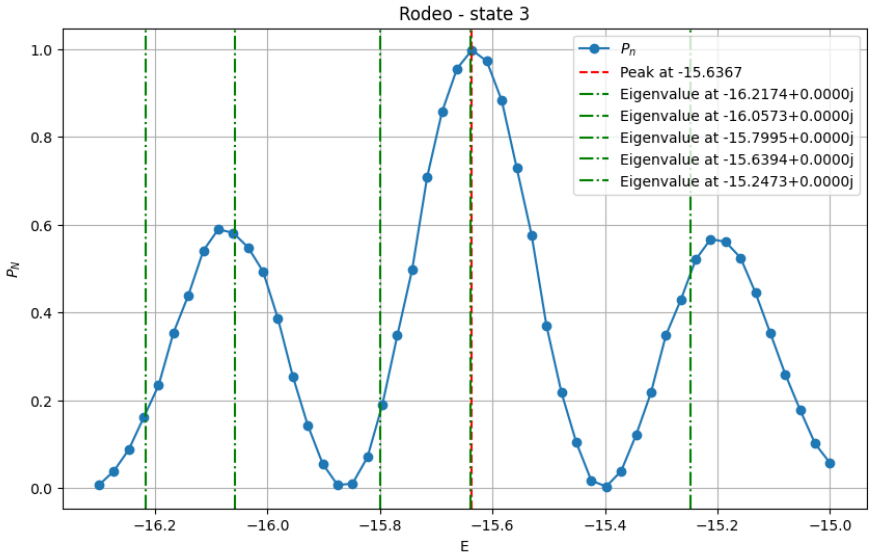
RODEO Implementation

Gershgoring Circle Theorem: H20 Results

- N = 4, RMS = 7: the eigenvalue peak is distinguishable from minor ones.
- Other small peaks appear just due to the periodicity of $P_N(E)$.



State corresponding to row 26: |011010>

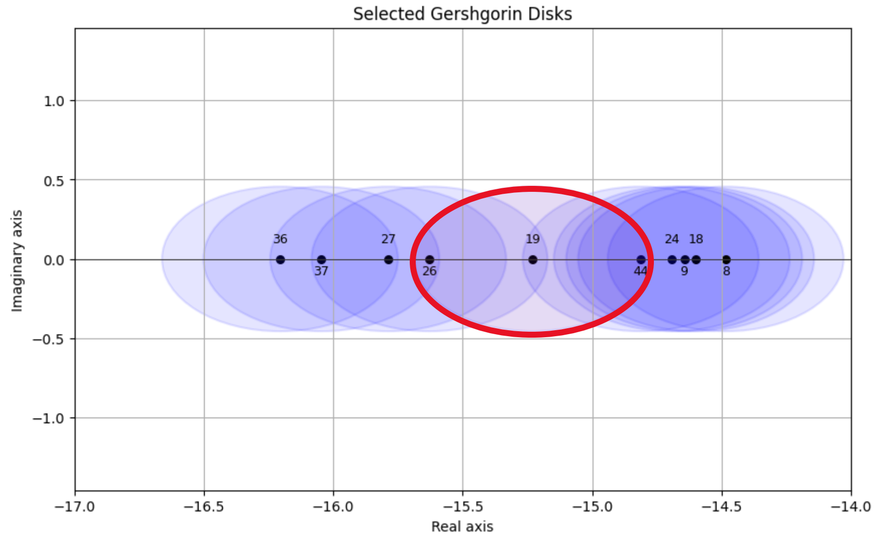


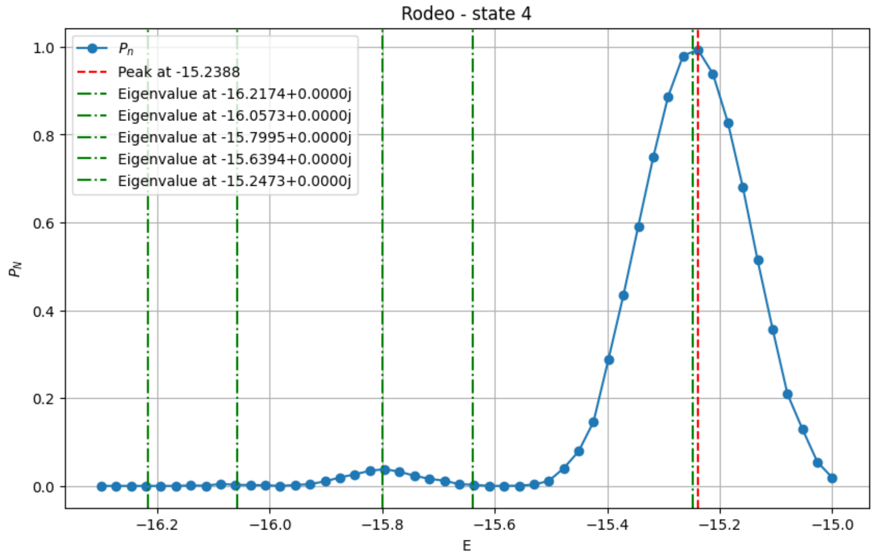
RODEO Implementation

Gershgoring Circle Theorem: H20 Results

- N = 4, RMS = 7: the eigenvalue peak is distinguishable from minor ones.
- Other small peaks appear just due to the periodicity of $P_N(E)$.
- Results show that 5 lowest eigenvalues did correspond to the first five Gershgoring disks.

State corresponding to row 19:
|010011>



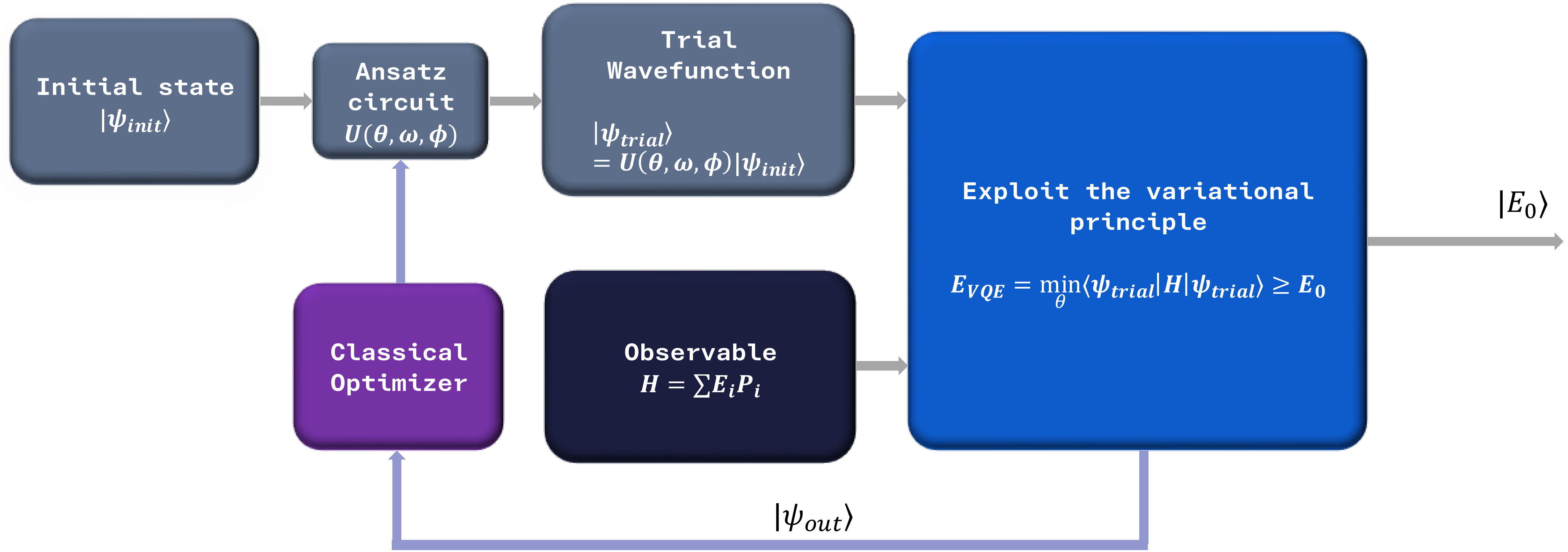


Variational Quantum Deflation (VQD)

Politecnico di Torino

VQD

Brief recall of VQE

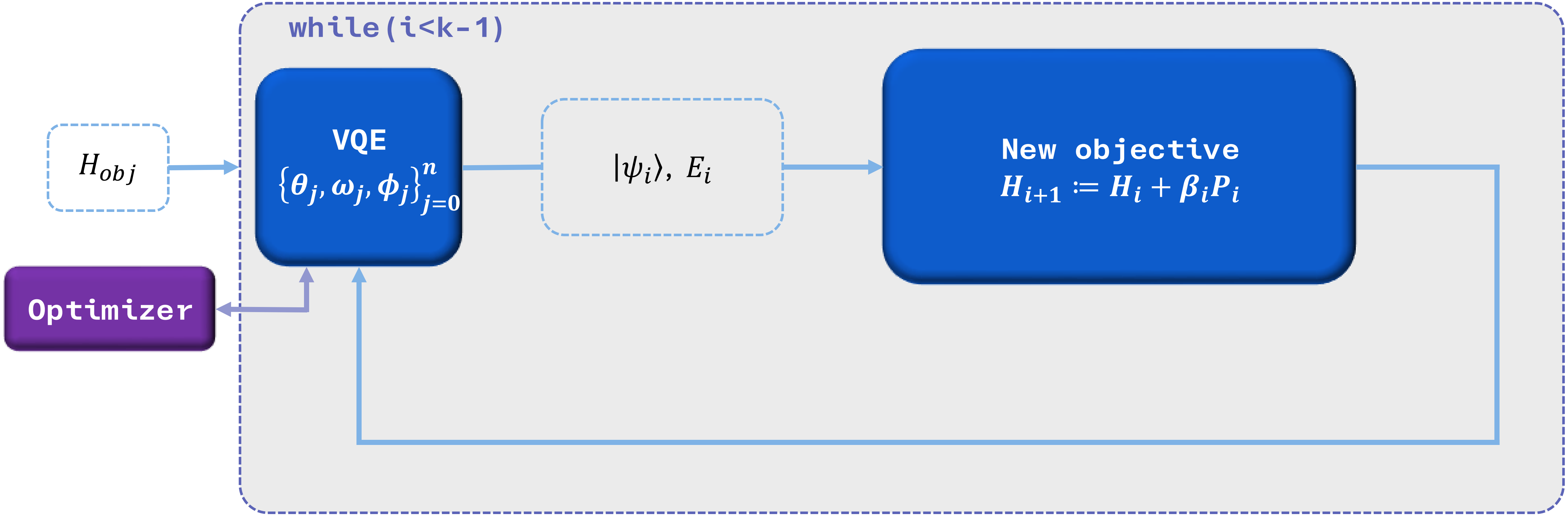


VQD

Building the spectra: the idea [HO19]

Legenda:

- $\{\theta_j, \omega_j, \phi_j\}$ VQE parameters
- $P_i = |\psi_i\rangle\langle\psi_i|$ Projector
- β_i Penalty
- k total number of states



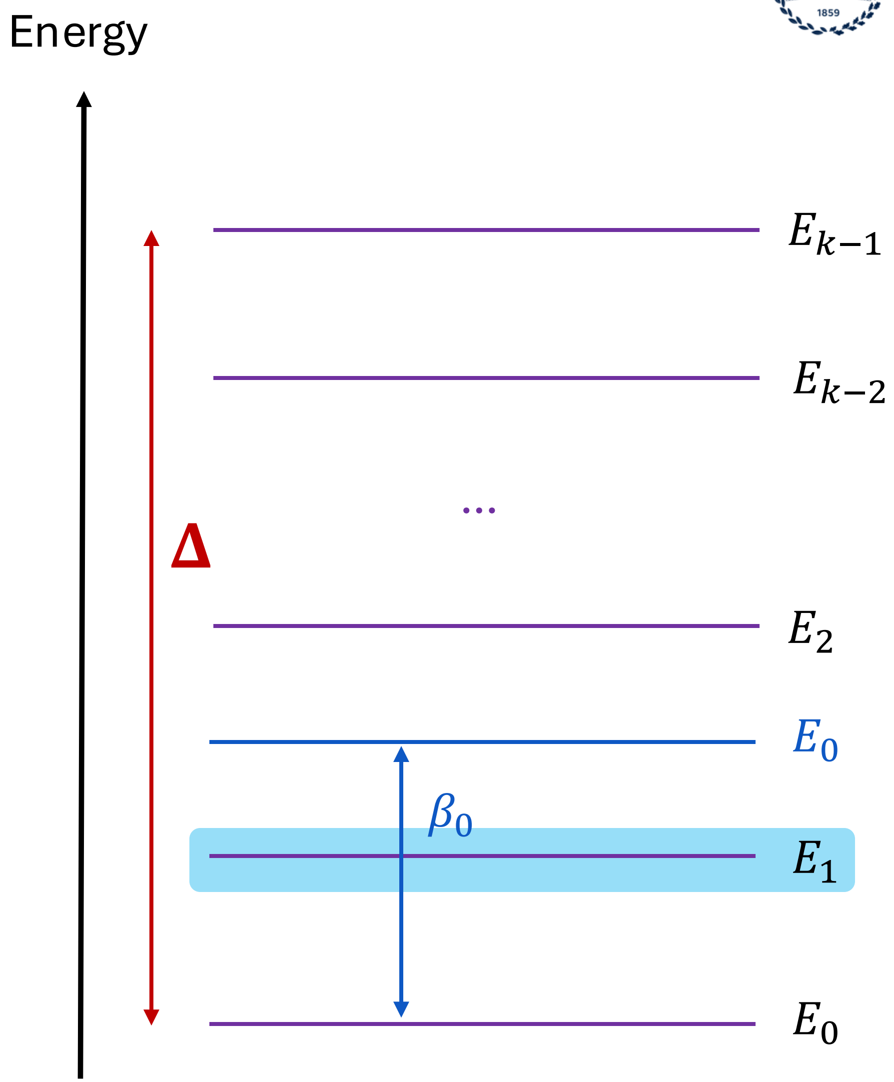
VQD

How to choose the penalties

The penalty must be chosen such that, if we are at stage k, then

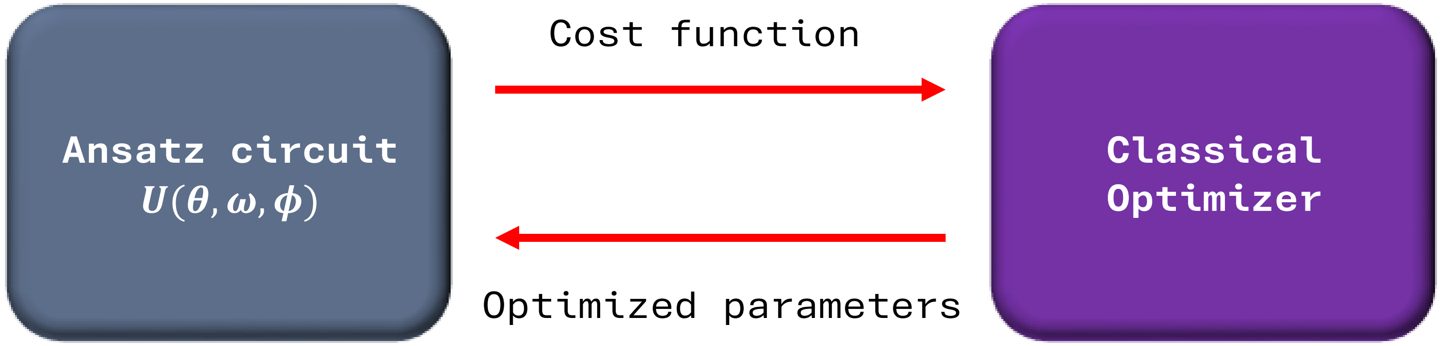
$$\beta_{k-1} + E_{k-1} > E_k$$

So that $|\psi_k\rangle$ is the new ground state.



VQD

implementation intro



VQD

Ansatz – encoding circuit

System with N features

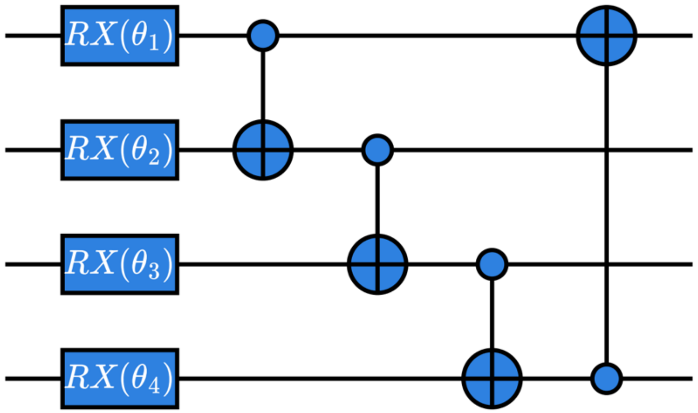
Amplitude Encoding	Angle Encoding	Basis Encoding
$\log_2 N$ *qubits*	N *qubits*	$N \times M$ *qubits* M is the number of bits in the binary string
exponential number of transformations	linear number of transformations	direct encoding into the computational basis

VQD

Ansatz – entangling layer

Basic Entangler Layer

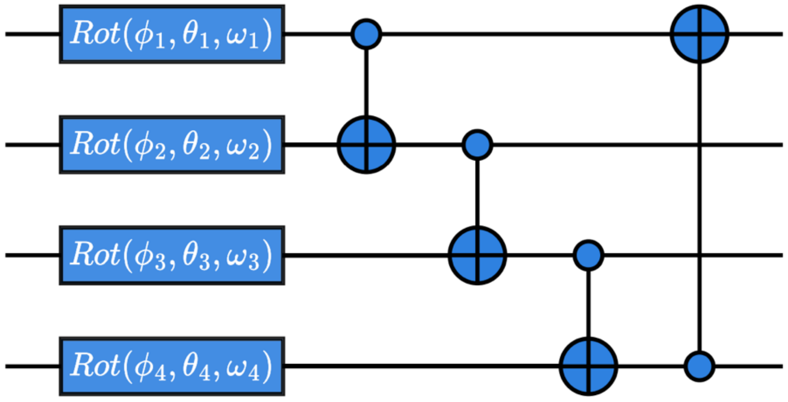
one-parameter θ
single-qubit rotations



Strongly Entangling Layer

ROT gate

$$R(\phi, \theta, \omega) = RZ(\omega)RY(\theta)RZ(\phi)$$



Images taken from IEEE 10.1109/ACCESS.2024.3432312

VQD

The ansatz

Parametric number of layers

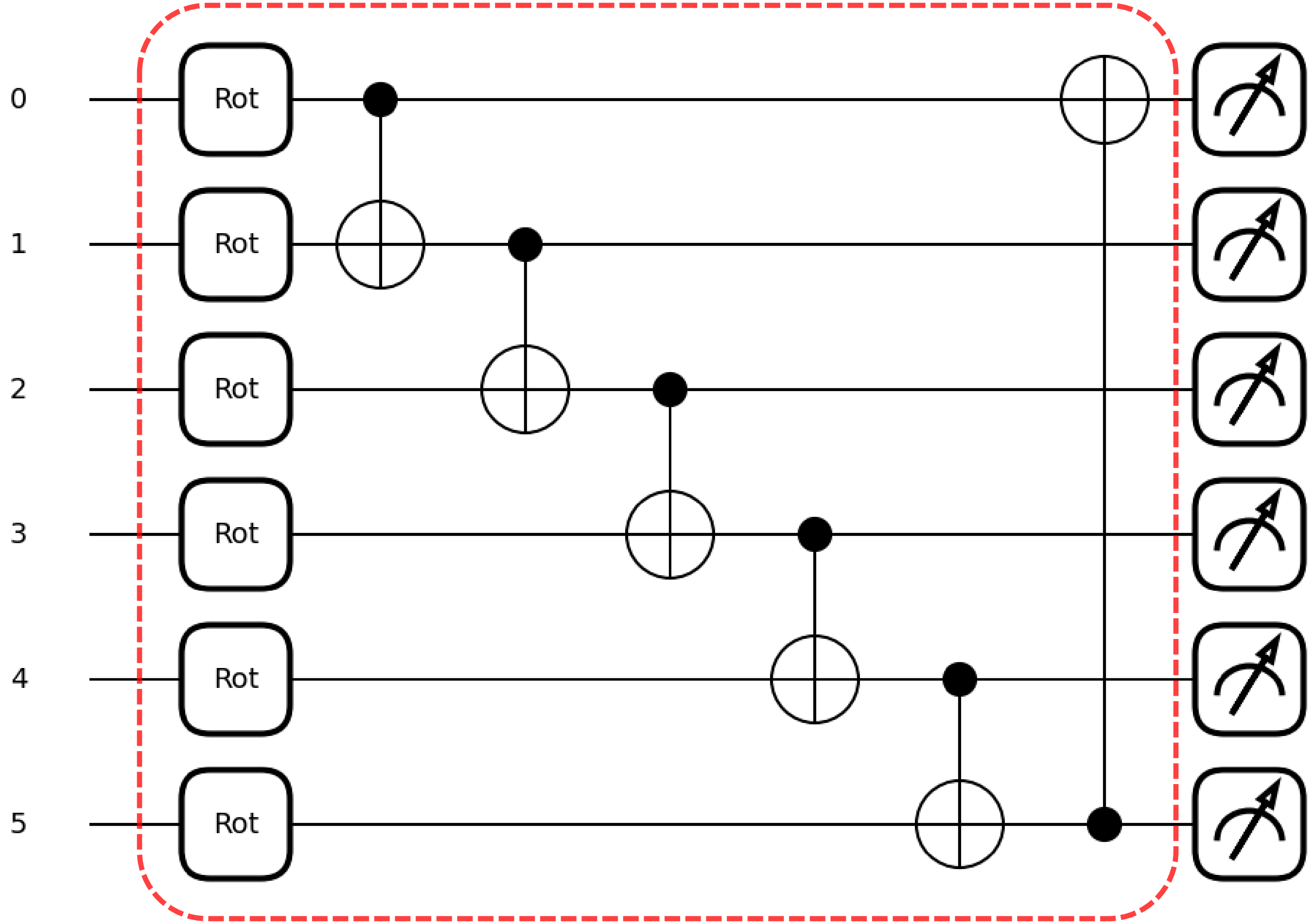
```python
def ansatz(params, num_layers, N):
    for i in range(num_layers):
        for qubit in range(N):
            qml.Rot(params[i, qubit, 0], params[i,
qubit, 1], params[i, qubit, 2], wires=qubit)

        for qubit in range(N - 1):
            qml.CNOT(wires=[qubit, qubit + 1])

        qml.CNOT(wires=[N - 1, 0])
```

Parameters to be optimized: $N \times L \times 3$

N: *# of qubits to represent the state*

L: *# of layer of the circuit*

Single layer



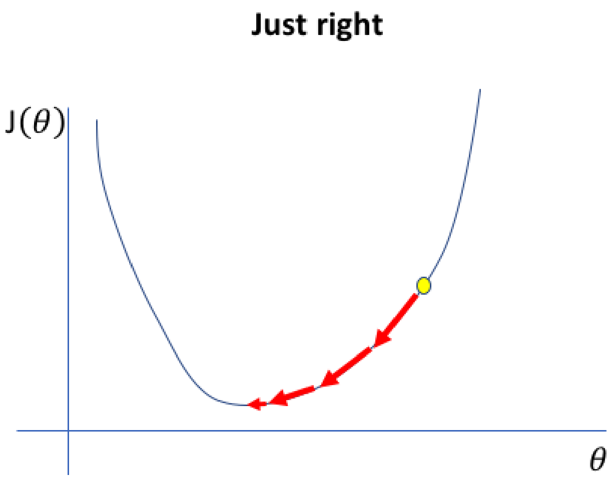
VQD

Optimizer

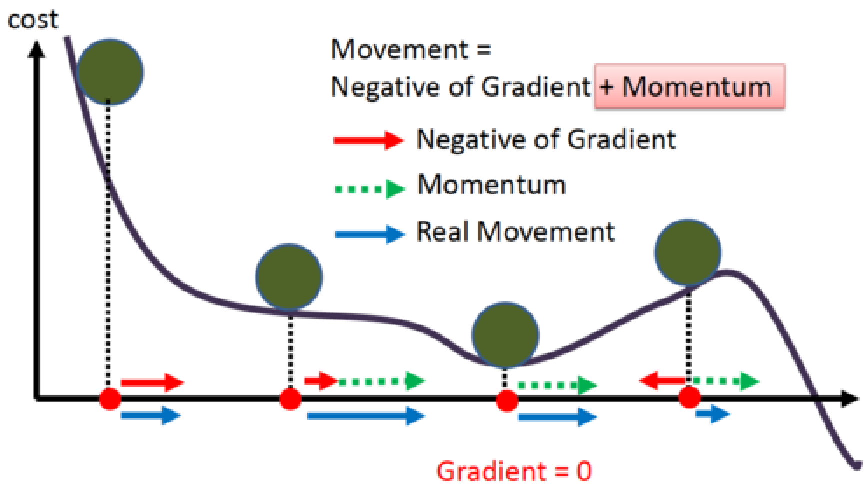
An **optimizer** is an algorithm used to find the parameters that minimize (or maximize) a given function, called a **cost function**

Adam (Adaptive Moment)

Adaptive learning



Momentum

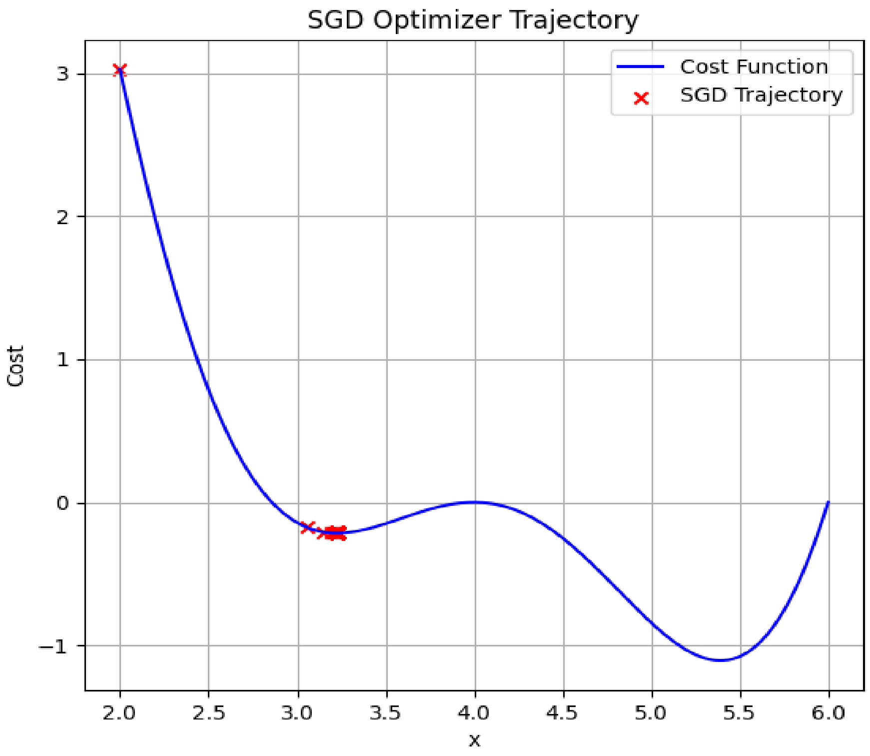


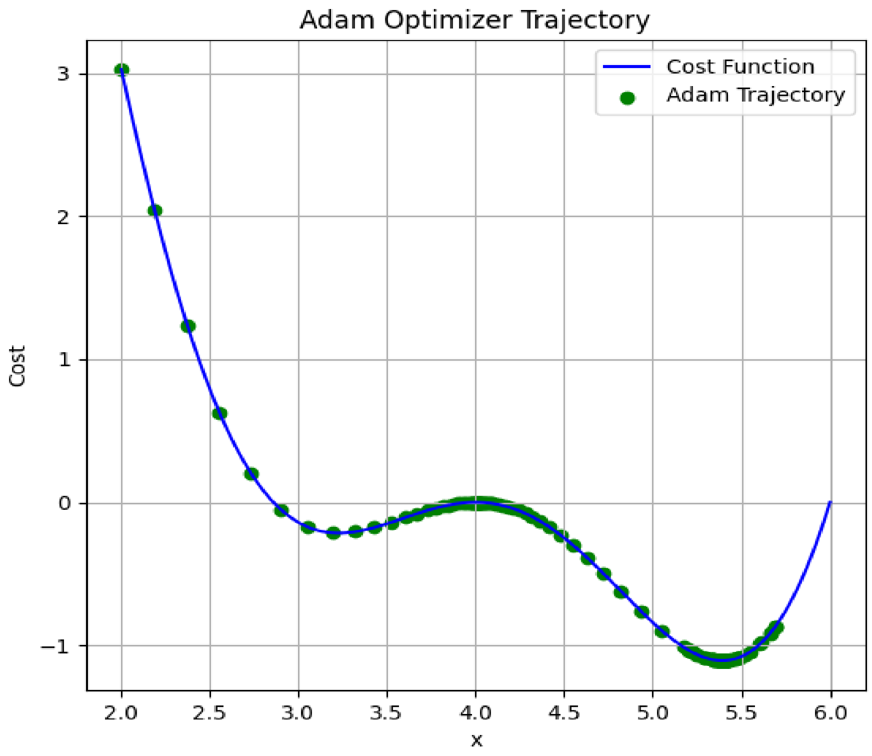
VQD

Optimizer – Toy model

Adam (Adaptive Moment Estimation)

Parameters:
- Starting point
- Cost function
- Tolerance
- Step size

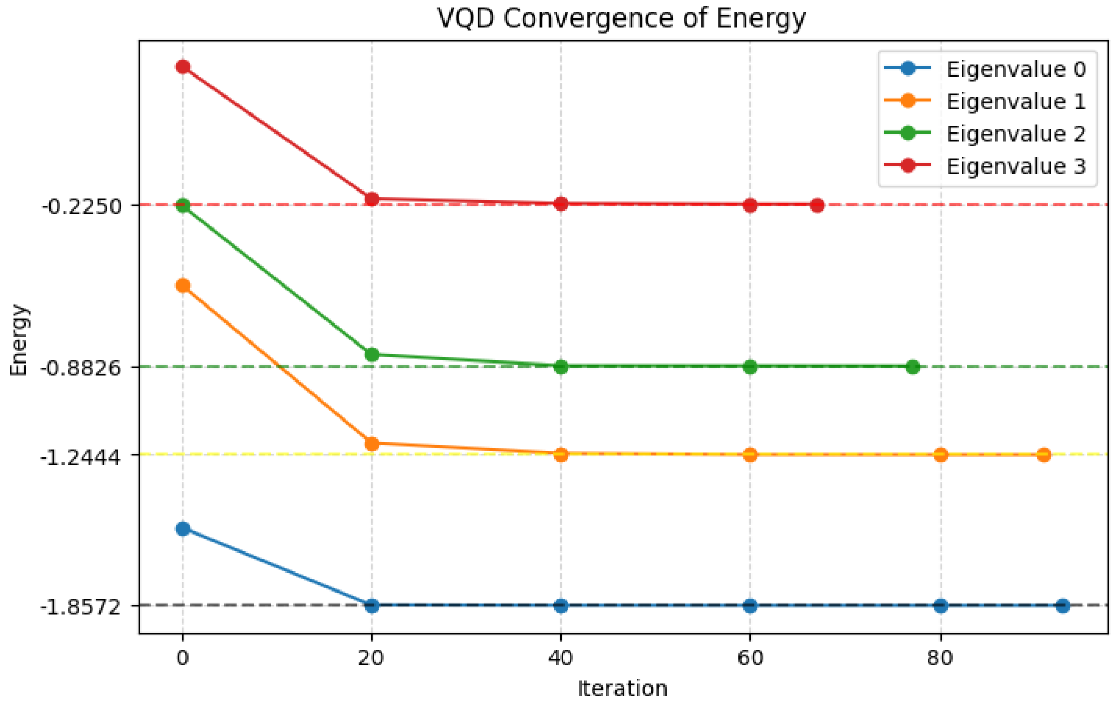


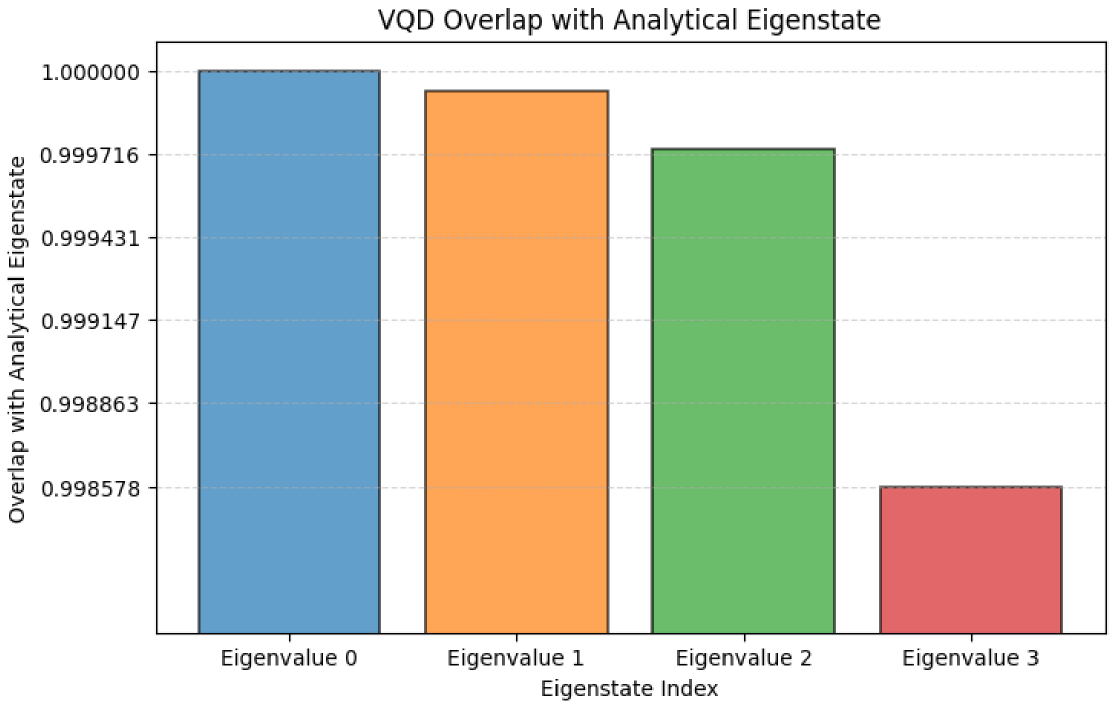


VQD

Results: VQD for H2

- 6 ansatz layer
- Tolerance = 10^{-6}
- Penalty = 2 eV
- Max iterations = 200

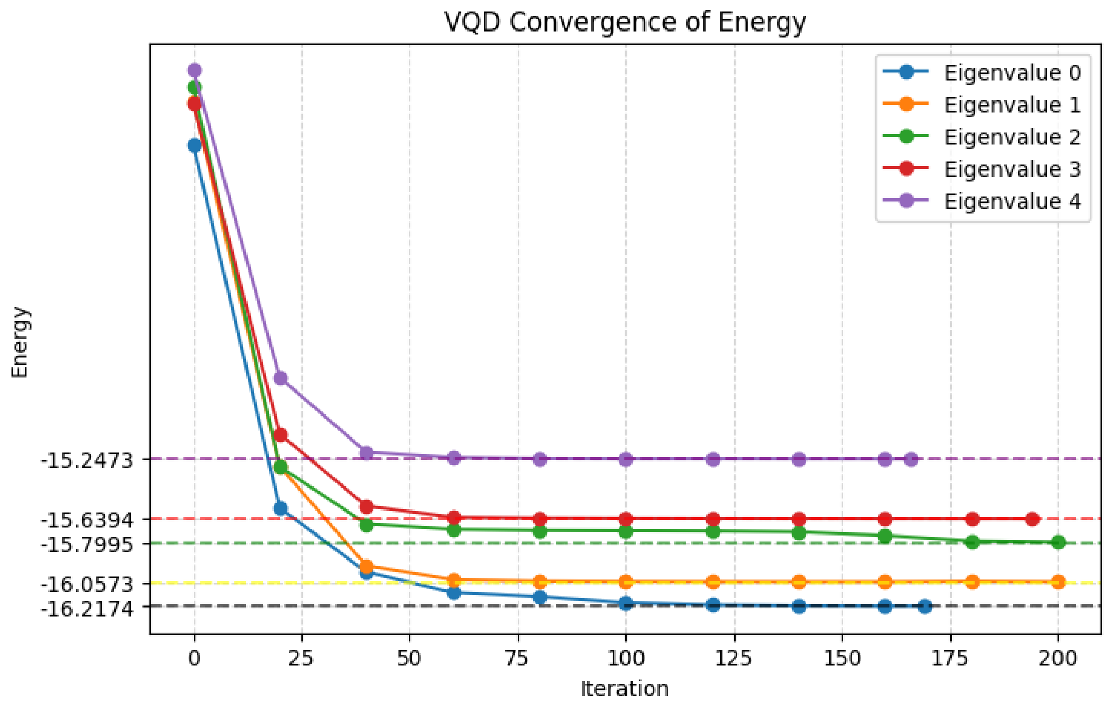


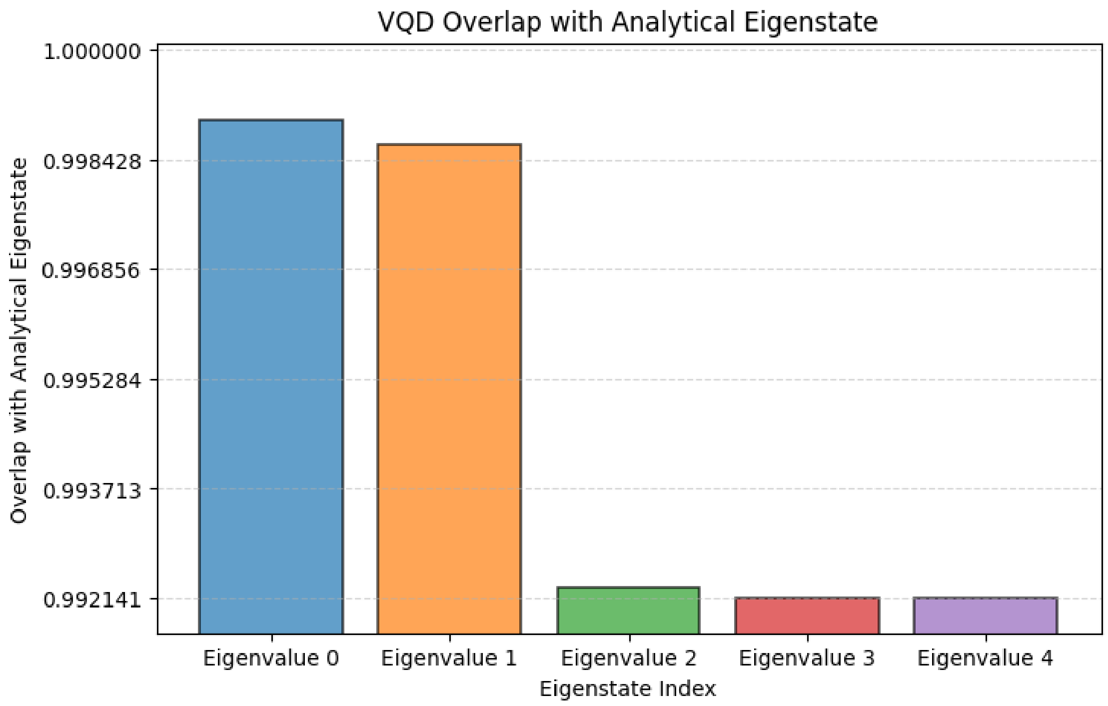


VQD

Results: VQD for H2O

- 6 ansatz layer
- Tolerance = 10^{-6}
- Penalty = 2 eV
- Max iterations = 200



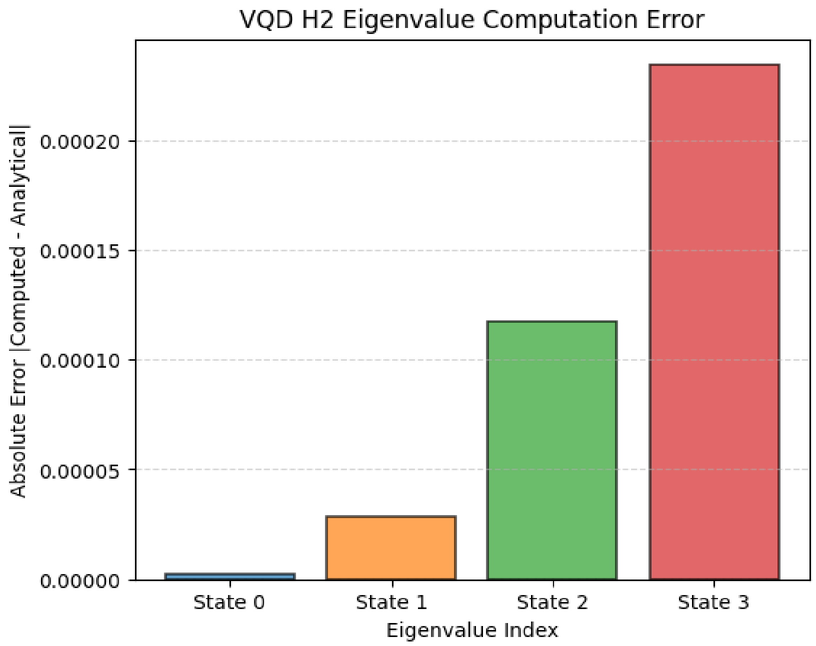


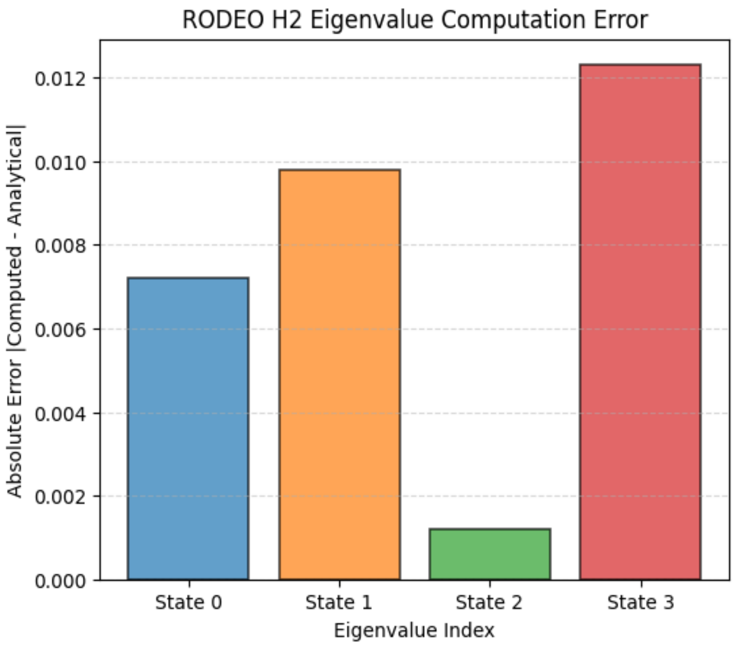
Comparison:
RODEO vs VQD

Comparison: RODEO vs VQD

H2 Eigenvalue Computation Error

- 6 ansatz layer.
- Tolerance $= 10^{-6}$
- Penalty = 2eV
- Iterations = 200



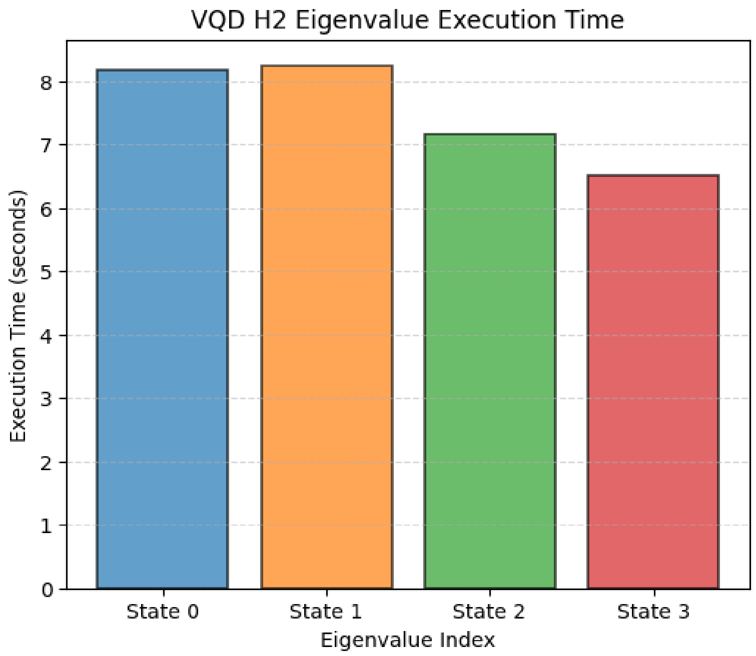


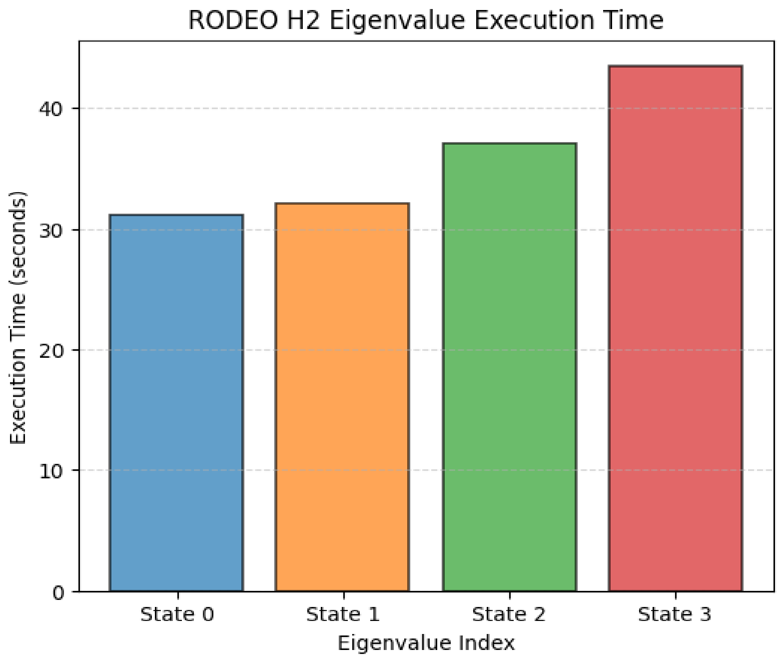
- $T_{RMS} = 7$
- N = 4
- Energy range taken from Gershgoring.
- 60 interrogation points for each range.

Comparison: RODEO vs VQD

H2 Eigenvalue Execution Time: VQD VS RODEO

- 6 ansatz layer.
- Tolerance $= 10^{-6}$
- Penalty = 2eV
- Iterations = 200



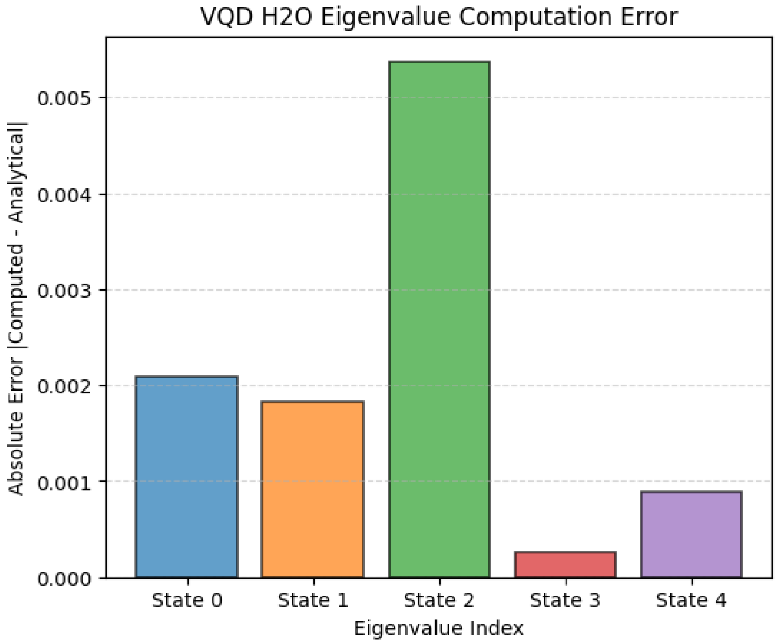


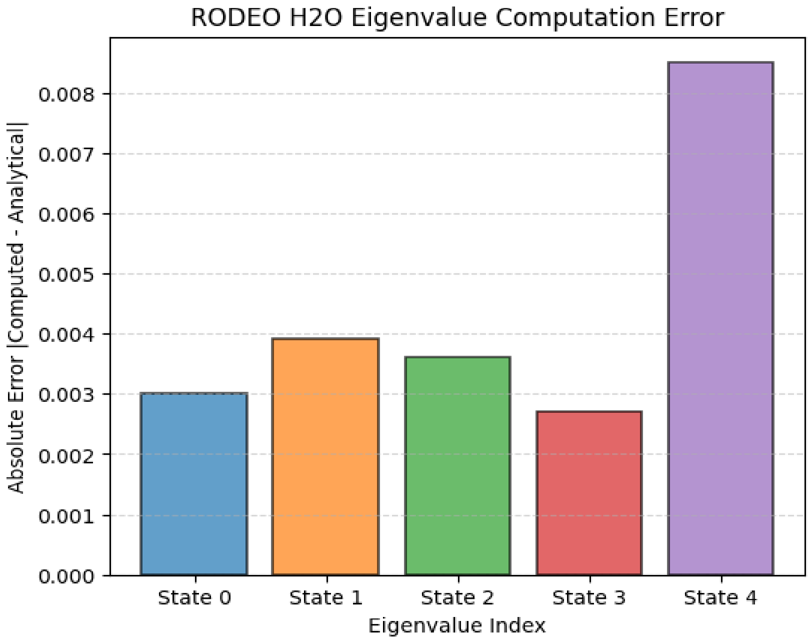
- $T_{RMS} = 7$
- N = 4
- Energy range taken from Gershgoring.
- 60 interrogation points for each range.

Comparison: RODEO vs VQD

H2O Eigenvalue Computation Error: VQD VS RODEO

- 6 ansatz layer.
- Tolerance $= 10^{-6}$
- Penalty = 2eV
- Iterations = 200



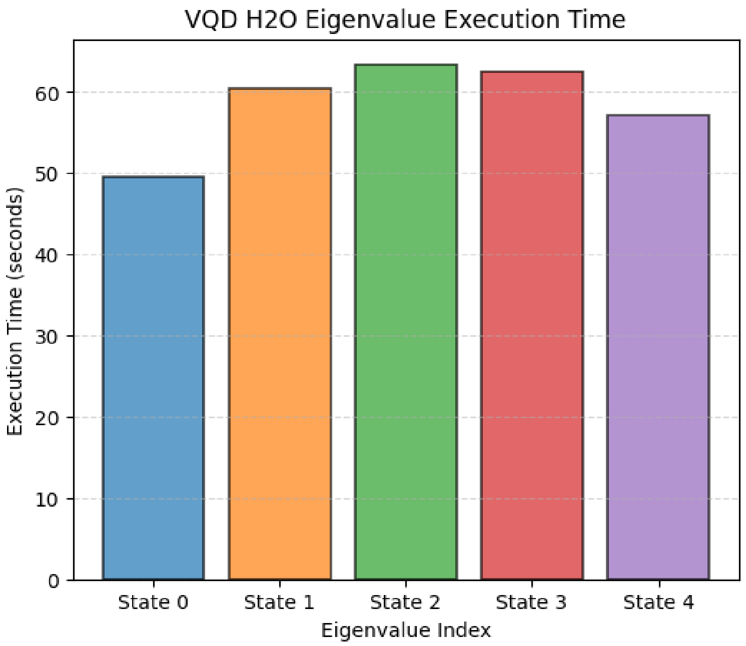


- $T_{RMS} = 7$
- N = 4
- Energy range taken from Gershgoring.
- 50 interrogation points for each range.

Comparison: RODEO vs VQD

H2O Eigenvalue Execution Time: VQD VS RODEO

- 6 ansatz layer.
- Tolerance $= 10^{-6}$
- Penalty = 2eV
- Iterations = 200



- $T_{RMS} = 7$
- N = 4
- Energy range taken from Gershgoring.
- 50 interrogation points for each range.

Conclusions

Politecnico di Torino

Conclusions

Politecnico di Torino

Summary

In this work we studied two NISQ quantum algorithms for molecular spectrum determination:

1- Rodeo algorithm, in combination with **Gershgoring Circle Theorem**, based on energy filtering through phase kickback

2- VQD, a modified VQE used to obtain also excited states' eigenenergies through penalties

We **compared** their performance in determining the first eigenvalues of H_2 and H_2O Hamiltonians

Result: VQD wins!

REFERENCES

[KR06]: Kempe, Julia, Alexei Kitaev, and Oded Regev. "The complexity of the local Hamiltonian problem." *Siam journal on computing*35.5 (2006): 1070-1097.

[CK21]: Choi, Kenneth, et al. "Rodeo algorithm for quantum computing." *Physical Review Letters* 127.4 (2021): 040505.

[HO19] Higgott, Oscar, Daochen Wang, and Stephen Brierley. "Variational quantum computation of excited states." *Quantum*3 (2019): 156.