Variational Quantum Algorithm

Kaiyan Shi, Haowei Deng

Concept Review

- A qubit can be described by 2-d vector:

$$\alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

- The state of an n-qubit system can be represented by 2^n-dim vector.
- The space of such states is a 2^n-dim Hilbert space.

- Unitary transformation $U: |\varphi\rangle \rightarrow U|\varphi\rangle$.

- How a state time-evolve under a Hamiltonian? Schrödinger's equation

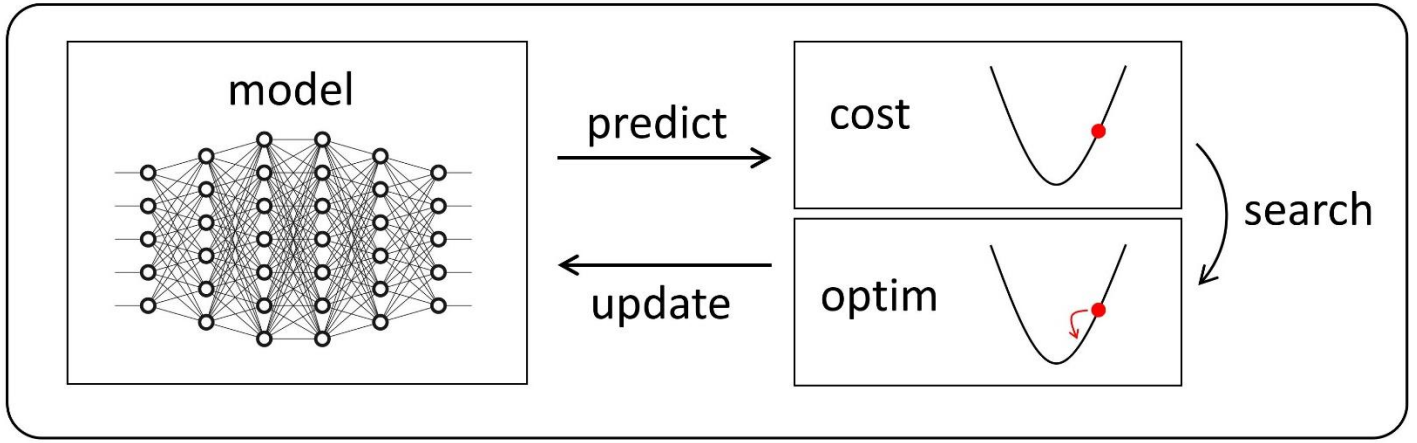
$$i\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = H(t)|\psi(t)\rangle.$$

Variational Quantum Algorithm (VQA)

In classical machine learning, we have:

1. data $\{(x_k, y_k)\}$
2. model $y = f(x; \theta)$
3. cost function $C(\theta) = \tilde{C}(f(x_k; \theta), y_k)$
4. optimizer $\theta_c = argmin\, C(\theta)$

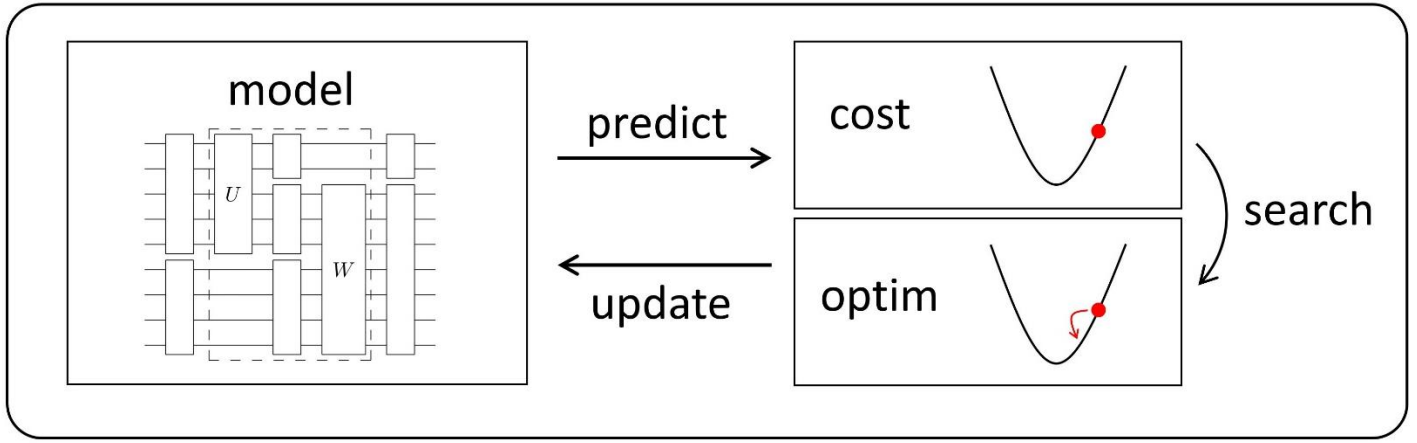
ML



The model in ML can be a neural network running on a classical computer.

In VQA, the model is a quantum circuit running on the quantum computer.

VQA



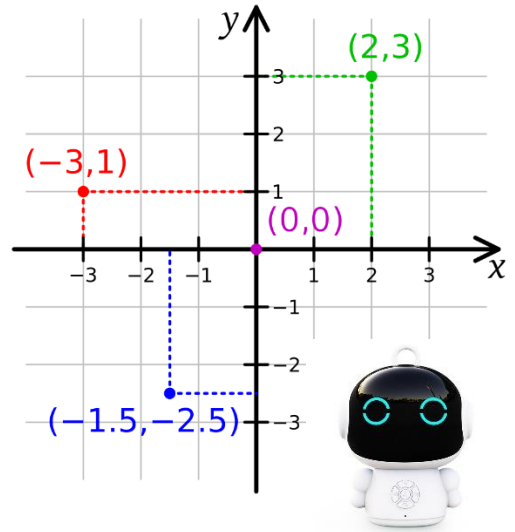
Variational Quantum Algorithm (VQA)

- A simple VQA example
- Suppose the n-qubits quantum system's initial state is $|\varphi_0\rangle$ and we want to construct a unitary circuit U which can transfer $|\varphi_0\rangle$ to target $|\varphi_1\rangle$. $|\varphi_0\rangle$ and $|\varphi_1\rangle$ can be represented by 2^n-dim vector. Let

$$|\varphi(\boldsymbol{\theta})\rangle = U(\theta_1, \dots, \theta_m)|\varphi_0\rangle.$$

- We need to figure out the parameters $\theta_1, \dots, \theta_m$ which make $|\varphi(\boldsymbol{\theta})\rangle = |\varphi_1\rangle$.

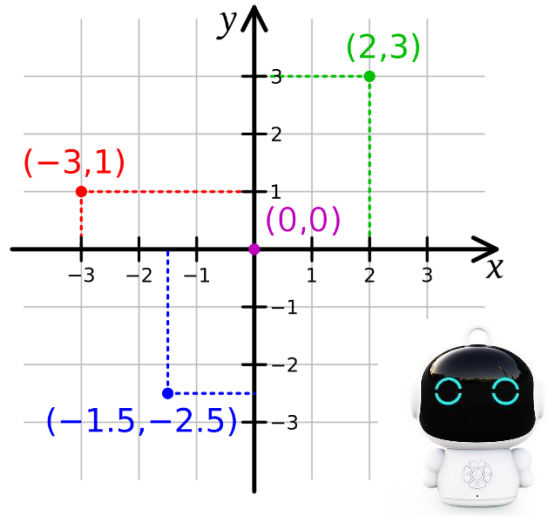
Anology: Consider a robot control problem. Suppose we have a robot at (0,0) and we want to drive it to a target position. The position of the robot can also be represented by a 2-dim vector.
Suppose the driver is parametrized by: The angle θ between the moving direction and the x-axis and the driving distance L. We need to figure out θ, L to move the robot to the target position.



Variational Quantum Algorithm (VQA)

- **Cost Function:** In robot control problem, when the robot is in (x, y), we can use its distance from the target position as the cost function which we want to minimize

$$D = \sqrt{(x_t - x)^2 + (y_t - y)^2}$$



- In VQA, when our circuit generate a state $|\varphi(\boldsymbol{\theta})\rangle$, we also want to evaluate $|\varphi(\boldsymbol{\theta})\rangle$'s distance from our target state $|\varphi_1\rangle$. The distance can be calculated as

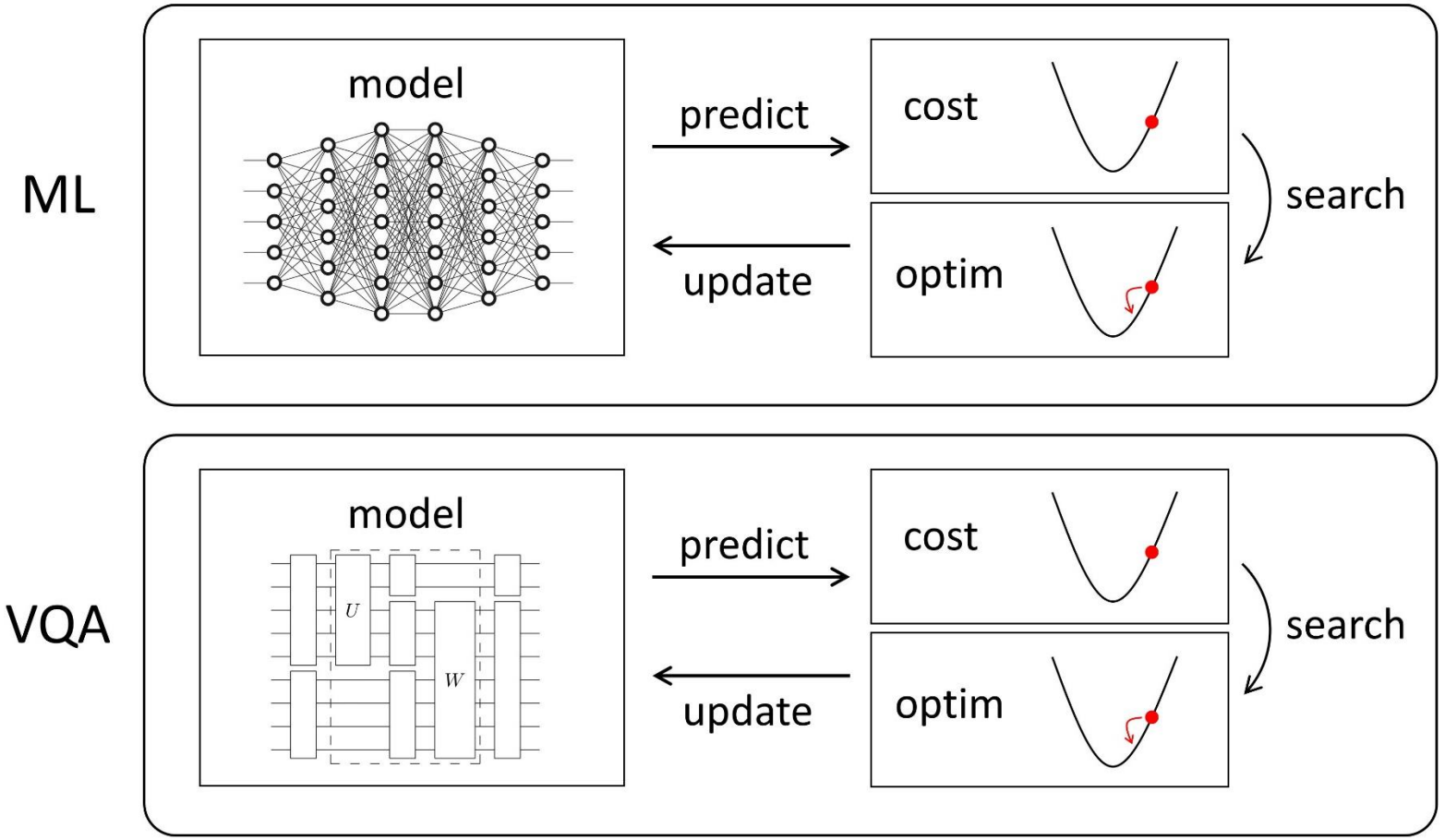
$$D' = 1 - \langle\psi_1|\varphi(\boldsymbol{\theta})\rangle$$

- In VQA, we need to tune $\theta_1, \ldots, \theta_m$ to minimize this distance D'

Variational Quantum Algorithm (VQA)

Steps for this example VQA:

1. Run $U(\theta_1, \dots, \theta_m)$ on quantum computer and get $|\varphi(\boldsymbol{\theta})\rangle$
2. Evaluate D', the distance between $|\varphi(\boldsymbol{\theta})\rangle$ and $|\varphi_1\rangle$.
3. Calculate the gradients of $\theta_1, \dots, \theta_m$ and tune them.



Variational Quantum Algorithm (VQA)

- Why we need quantum computer?
- In application (i.e. quantum chemical simulation), n can be very large. Using classical model to simulate the evolution of a 2^n-dim vector ($|\varphi_0\rangle, |\varphi_1\rangle, |\varphi(\boldsymbol{\theta})\rangle$) has $O(2^n)$ complexity. Quantum computer can do this in $O(\text{poly}(n))$.
- This example is exactly Variational Quantum Eigensolver (VQE), one of the most promising applications of quantum computing
  - Circuit – based VQE
  - Pulse – based VQE

Variational Quantum Eigensolver

Variational Quantum Eigensolver (VQE) [PA14]

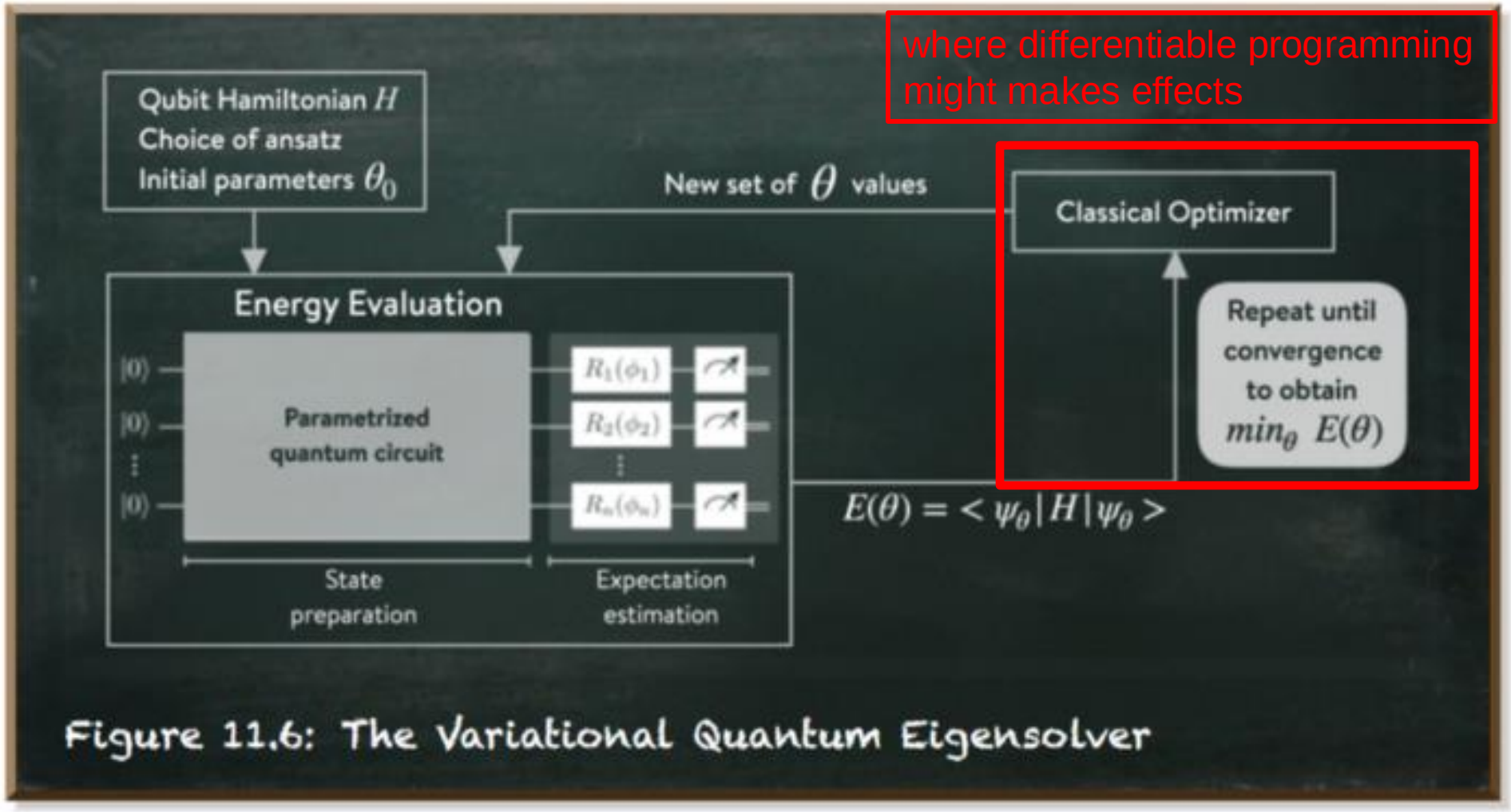
- What is VQE?
  - a hybrid quantum-classical algorithm
  - aim to find the lowest eigenvalue of a given Hamiltonian

- Why VQE?
  - beneficial for (quantum) systems whose dimension of the problem space grows **exponentially**
  - applicable on Noisy Intermediate-Scale Quantum (NISQ) devices
    - current devices have a lot of noise and have restriction in circuit depth
    - a lot of quantum algorithms are not applicable
  - usage in quantum chemistry, one of the most promising applications of quantum computing

- Connection with Differentiable Programming
  - optimizer requires gradient calculation

VQE Diagram Illustration

A hybrid quantum-classical algorithm



Variational Quantum Eigensolver (VQE)

- **Problem (Core Task):** Solve the ground state (energy) of any molecular Hamiltonian $\hat{H}$.
- **Input:** Some molecular Hamiltonian $\hat{H}$
- **Approach:**
  - Prepare a parameterized wave function ansatz $|\Psi(\boldsymbol{\theta})\rangle$ on a quantum computer.
  - Adopt classical optimization methods (e.g. gradient descent) to **adjust** $\boldsymbol{\theta}$ to minimize the expectation value
$$\langle\Psi(\boldsymbol{\theta})|\hat{H}|\Psi(\boldsymbol{\theta})\rangle$$
- **Output:**
  - Minimized expectation value, expected to be the ground state enegy
  - $|\Psi(\boldsymbol{\theta})\rangle$, expected to be the ground state

Theoretical foundation of VQE: **Rayleign-Ritz variational principle**

$$E_0 = \min_{\boldsymbol{\theta}} \langle\Psi(\boldsymbol{\theta})|\hat{H}|\Psi(\boldsymbol{\theta})\rangle$$

Variational Quantum Eigensolver (VQE)

- **Problem (Reduced Task):** Finding the smallest eigenvalue $\lambda_{\min}$ of a **discretized** Hamiltonian H and its corresponding eigenvector $|\Psi_0\rangle$.
- **Input:** Some discretized Hamiltonian H
- **Approach:**
  - Prepare a parameterized ansatz $|\Psi(\boldsymbol{\theta})\rangle$ on a quantum computer.
  - Adopt classical optimization methods (e.g. gradient descent) to **adjust** $\boldsymbol{\theta}$ to minimize the expectation value
$$\langle\Psi(\boldsymbol{\theta})|H|\Psi(\boldsymbol{\theta})\rangle$$
- **Output:**
  - Minimized expectation value, expected to be $\lambda_{\min}$
  - $|\Psi(\boldsymbol{\theta})\rangle$, ecpected to be $|\Psi_0\rangle$

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$
$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Tranformation from $\widehat{H}$ to H :

The corresponding Hamiltonian H of $\widehat{H}$ should be expressed as a weighted sum of Pauli spin operators $\{\sigma_x, \sigma_y, \sigma_z\}$ such that the information can be processed on a quantum computer,

$$H = \sum_k c_k \left(\bigotimes_{j=0}^{M-1} \sigma_j^{(k)} \right)$$

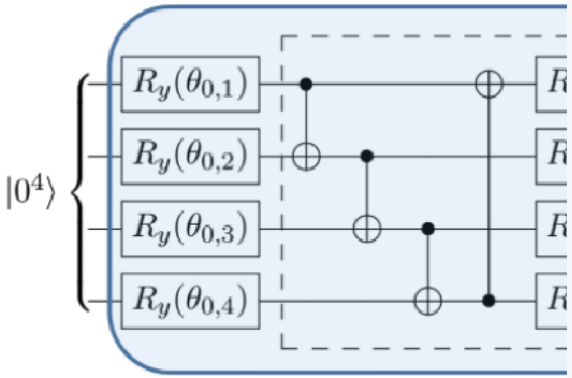
where $\sigma_j^{(k)} \in \{I, \sigma_x, \sigma_y, \sigma_z\}$ and M stands for qubit number.

VQE Example - Hydrogen Molecule H_2

1. Construct H for H

```
The generated h2 Hamiltonian is
 -0.09706626861762556 I
-0.04530261550868938 X0, X1, Y2, Y3
0.04530261550868938 X0, Y1, Y2, X3
0.04530261550868938 Y0, X1, X2, Y3
-0.04530261550868938 Y0, Y1, X2, X3
0.17141282639402394 Z0
0.16868898168693286 Z0, Z1
0.12062523481381837 Z0, Z2
0.16592785032250773 Z0, Z3
0.17141282639402394 Z1
0.16592785032250773 Z1, Z2
0.12062523481381837 Z1, Z3
-0.2234315367466399 Z2
0.17441287610651626 Z2, Z3
-0.2234315367466399 Z3
```

2. Build a quantum r tion ansatz $|\Psi(\boldsymbol{\theta})\rangle$.



$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

This is a 4-qubit quantum circuit emplate with D layers.
The dotted frame in the figure denotes a single layer.
Exemplary circuit, R_y can be eplaced with R_x, R_z etc.

3. Setting loss functi

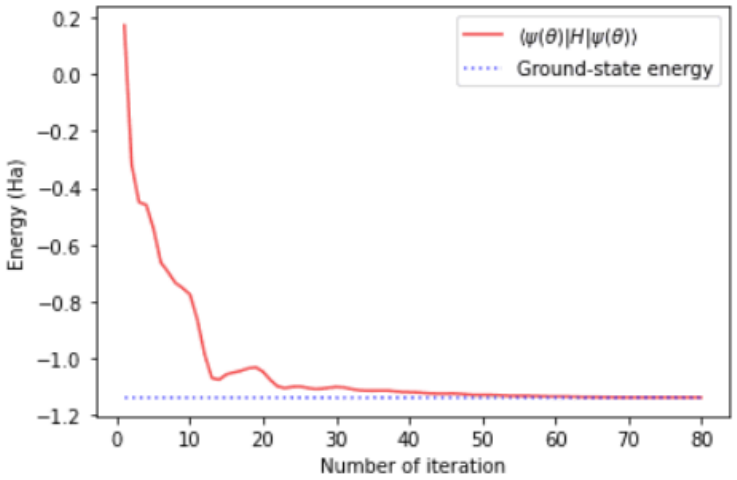
$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_k w_k \langle \Psi(\boldsymbol{\theta}) | \sigma_j^{(k)} | \Psi(\boldsymbol{\theta}) \rangle.$$

VQE Example - Hydrogen Molecule H_2

4. Set up the parameters of the model and start training the quantum neural network

```
iter:  20 loss: -1.0467
iter:  20 Ground state energy: -1.0467 Ha
iter:  40 loss: -1.1185
iter:  40 Ground state energy: -1.1185 Ha
iter:  60 loss: -1.1337
iter:  60 Ground state energy: -1.1337 Ha
iter:  80 loss: -1.1370
iter:  80 Ground state energy: -1.1370 Ha

Circuit after training:
--Ry(7.841)----*--------------x----Ry(7.851)----*--------------x----Ry(0.021)--
               |              |                 |              |
--Ry(1.407)----x----*---------|----Ry(4.485)----x----*---------|----Ry(5.357)--
                    |         |                      |         |
--Ry(6.923)---------x----*----|----Ry(1.570)---------x----*----|----Ry(6.209)--
                         |    |                           |    |
--Ry(1.648)--------------x----*----Ry(4.728)--------------x----*----Ry(-0.16)--
```



VQE and QAOA

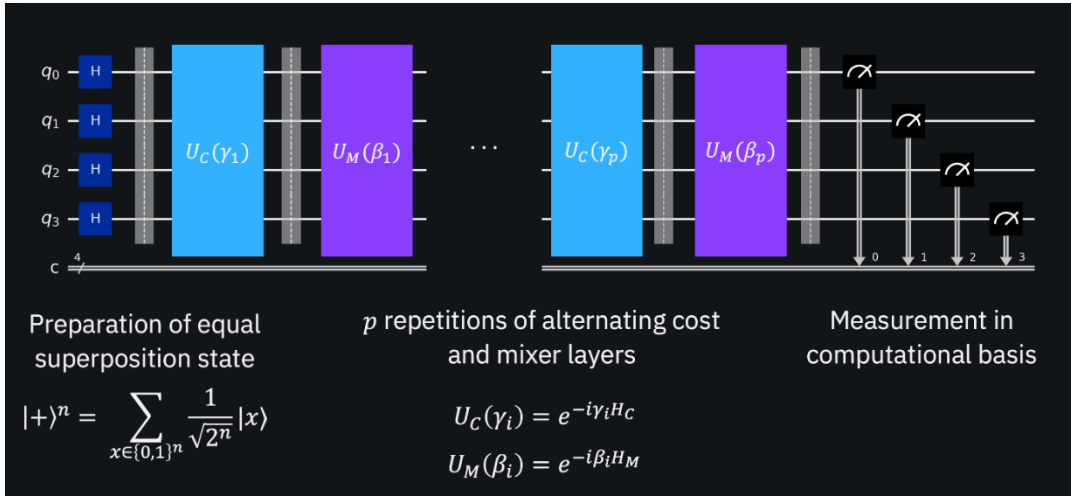
From my understanding, those Hamiltonians are interchangable.

- VQE, QAOA ∈ VQA

- Application
  - QAOA: combinatorial optimization $H = \sum_i C_i(z)$
  - VQE: quantum chemistry
    - ➢ target Hamiltonian comes from molecular Hamiltonian.

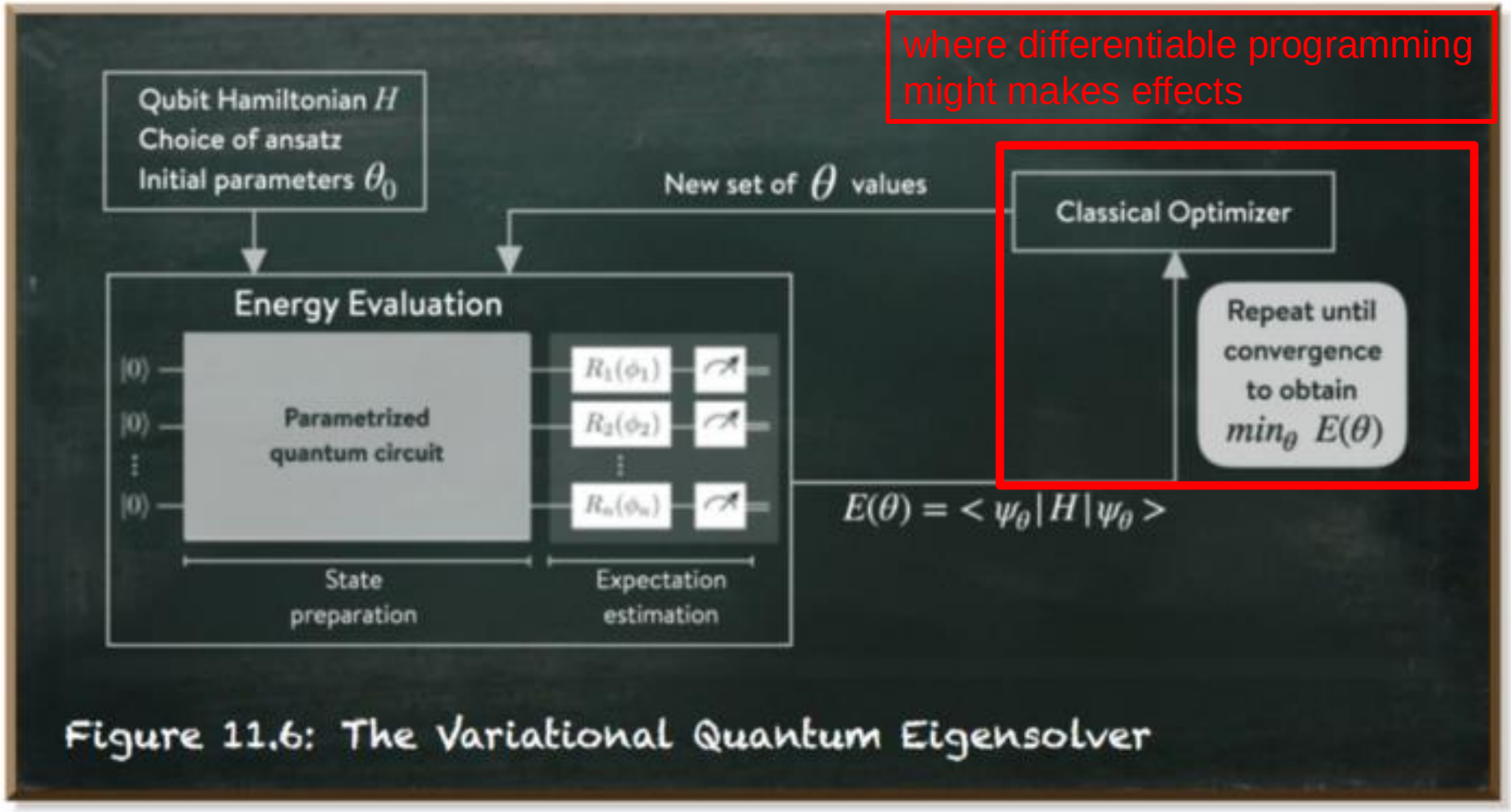
- Circuit
  - QAOA: related to the target Hamiltonian H
  - VQE: customized circuit, can be independent of H
    - ➢ QAOA circuit can also be used.

QAOA circuit



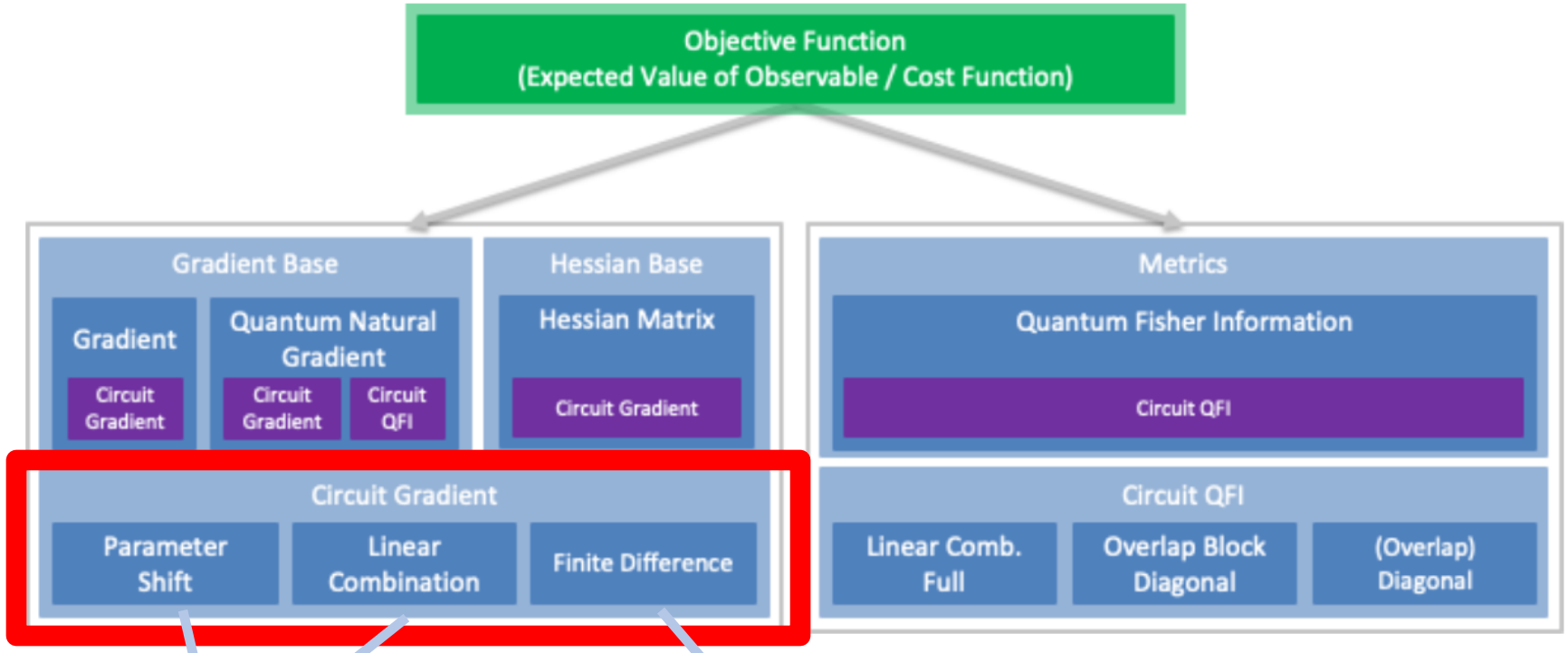
VQE Diagram Illustration

A hybrid quantum-classical algorithm



Calculating gradient using quantum circuits

Qiskit Gradient Framework



Automatic differentiation - Backpropagation

- Not clear how intermediate derivatives could be stored and reused inside of a quantum computation
- No computing graph

To get the gradient of a quantum program, the key idea is to construct some new programs to compute the gradients of the original one.

Automatic differentiation

Numerical Differentiation

Symbolic Differentiation

Calculating gradient using quantum circuits

- Loss function is

$$\mathcal{L}(\boldsymbol{\theta}) = \langle \Psi(\boldsymbol{\theta}) | H | \Psi(\boldsymbol{\theta}) \rangle = \langle 0 \ldots 0 | \, U(\boldsymbol{\theta})^{\dagger} H U(\boldsymbol{\theta}) \, | 0 \ldots 0 \rangle$$

  where $\boldsymbol{\theta} = [\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \ldots, \boldsymbol{\theta_n}]$ is a list of trainable parameters in the circuit.

- Goal is to find

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_n} \end{bmatrix}$$

Finite Difference Method

- Numerical Differentiation
- Main Idea
  - The error of the derivative of a function $f(x)$ tends to zero as h tends to zero:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

  - By choosing a sufficiently small h, we can get a good approximation of the derivative.
  - For example, for **the central finite difference method**, the gradient of the loss fnction is

$$\nabla \mathcal{L}(\boldsymbol{\theta}) \approx \frac{\mathcal{L}(\boldsymbol{\theta}+\delta) - \mathcal{L}(\boldsymbol{\theta}-\delta)}{2\delta} = \frac{\langle 0 \ldots 0 | U^\dagger(\boldsymbol{\theta}+\delta) H U(\boldsymbol{\theta}+\delta) | 0 \ldots 0 \rangle - \langle 0 \ldots 0 | U^\dagger(\boldsymbol{\theta}-\delta) H U(\boldsymbol{\theta}-\delta) | 0 \ldots 0 \rangle}{2\delta}$$

- Advantage
  - no need to build extra circuits or using extra qubits
- Disadvantage
  - only get an estimation of the gradient
  - high errors of near-term quantum devices

Parameter Shift Rules

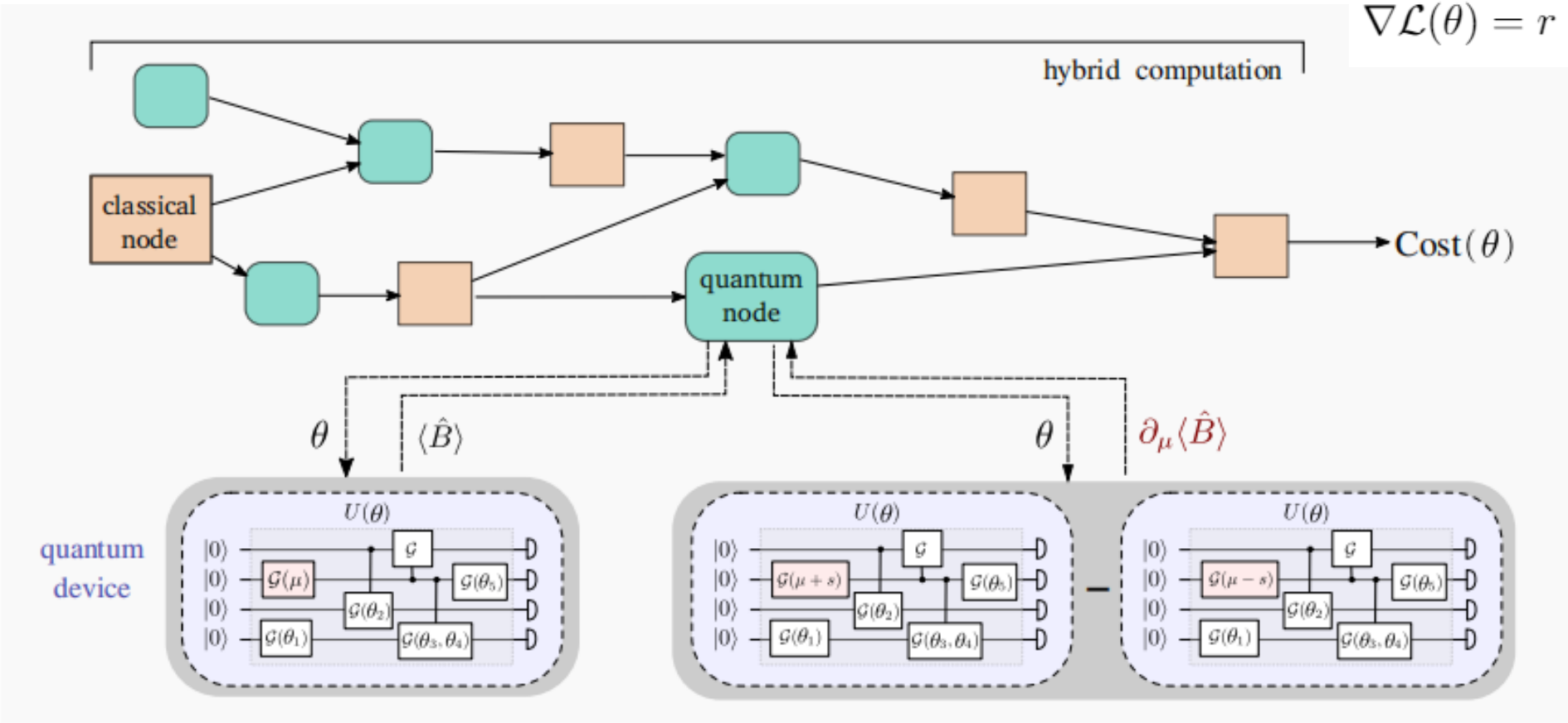
- Original parameter shift rule
  - **restriction:** $U(\theta)$ is a single-parameter gate.
  - **restriction:** $U(\theta)$ can be written as $e^{-ia\theta G}$, where G has two unique eigenvalues λ_1 and λ_2.
  - parameter-shift rule to find its gradient

$$\nabla \mathcal{L}(\theta) = r\left[\mathcal{L}\left(\theta + \tfrac{\pi}{4r}\right) - \mathcal{L}\left(\theta - \tfrac{\pi}{4r}\right)\right]$$

  where $r = \frac{a}{2}(\lambda_2 - \lambda_1)$.
- Advantage
  - no need to build extra circuits or using extra qubits as well
  - analytical gradient
- Disadvantage
  - restrictions

Parameter Shift Rules Diagram Illustration

$$\nabla \mathcal{L}(\theta) = r\left[\mathcal{L}\left(\theta + \frac{\pi}{4r}\right) - \mathcal{L}\left(\theta - \frac{\pi}{4r}\right)\right]$$

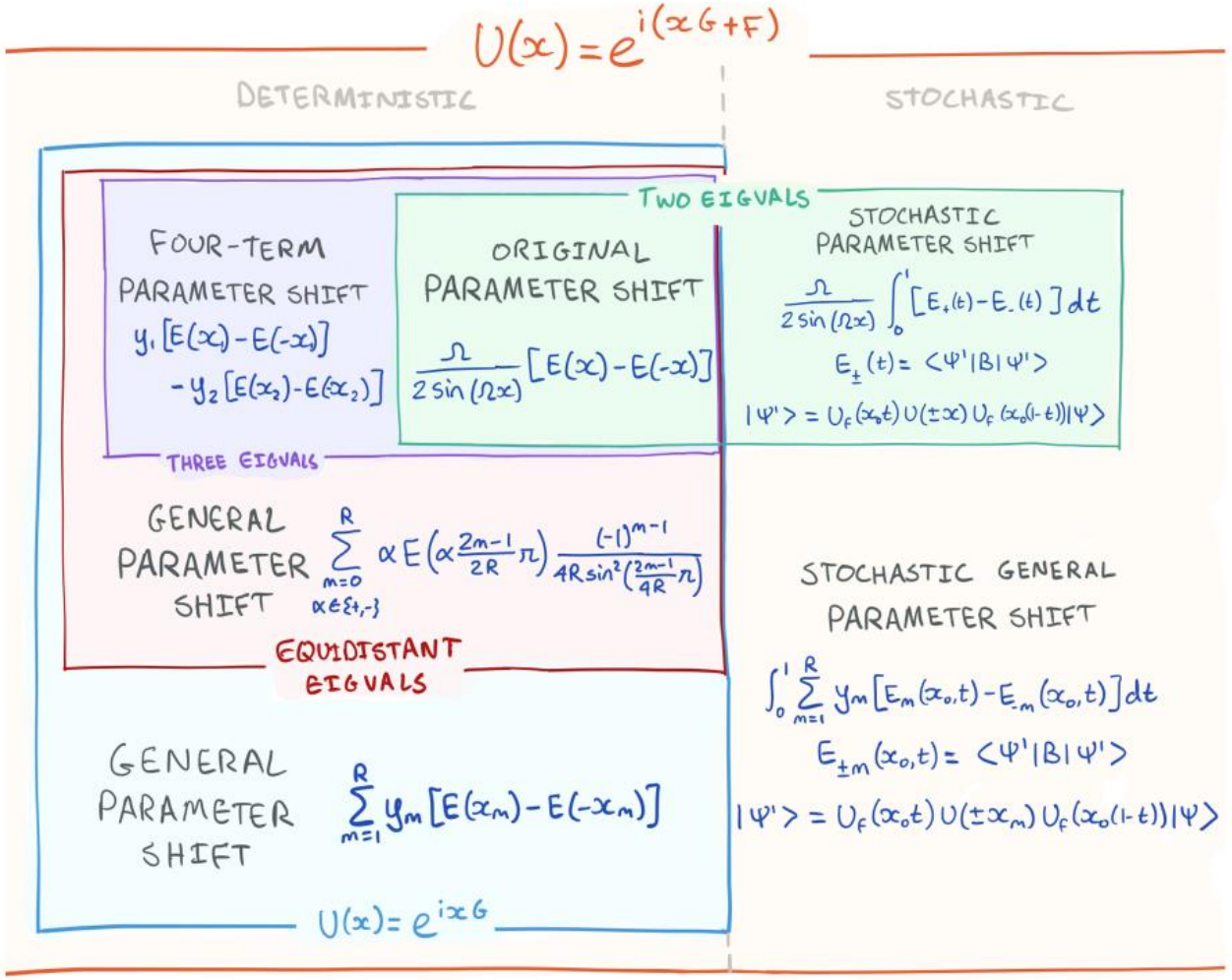


In the diagram

- $\mu = \theta$
- $s = \frac{\pi}{4r}$

The “parameter shift rule” in the larger context of hybrid optimization.
A quantum node can compute derivatives of its outputs with respect to gate parameters by running the original circuit twice, but with a shift in the parameter in question.

Parameter Shift Rules



Overview of existing and new parameter-shift rules for first-order univariate derivatives as Venn diagram on the space of quantum gates.[WD21]

History of Parameter-Shift Rules:

$$E(x) := \langle\psi|\, U^\dagger(x) B U(x)\,|\psi\rangle \quad U(x) = \exp(ixG)$$

$$E(x) = a_0 + \sum_{\ell=1}^{R} a_\ell \cos(\Omega_\ell x) + b_\ell \sin(\Omega_\ell x)$$

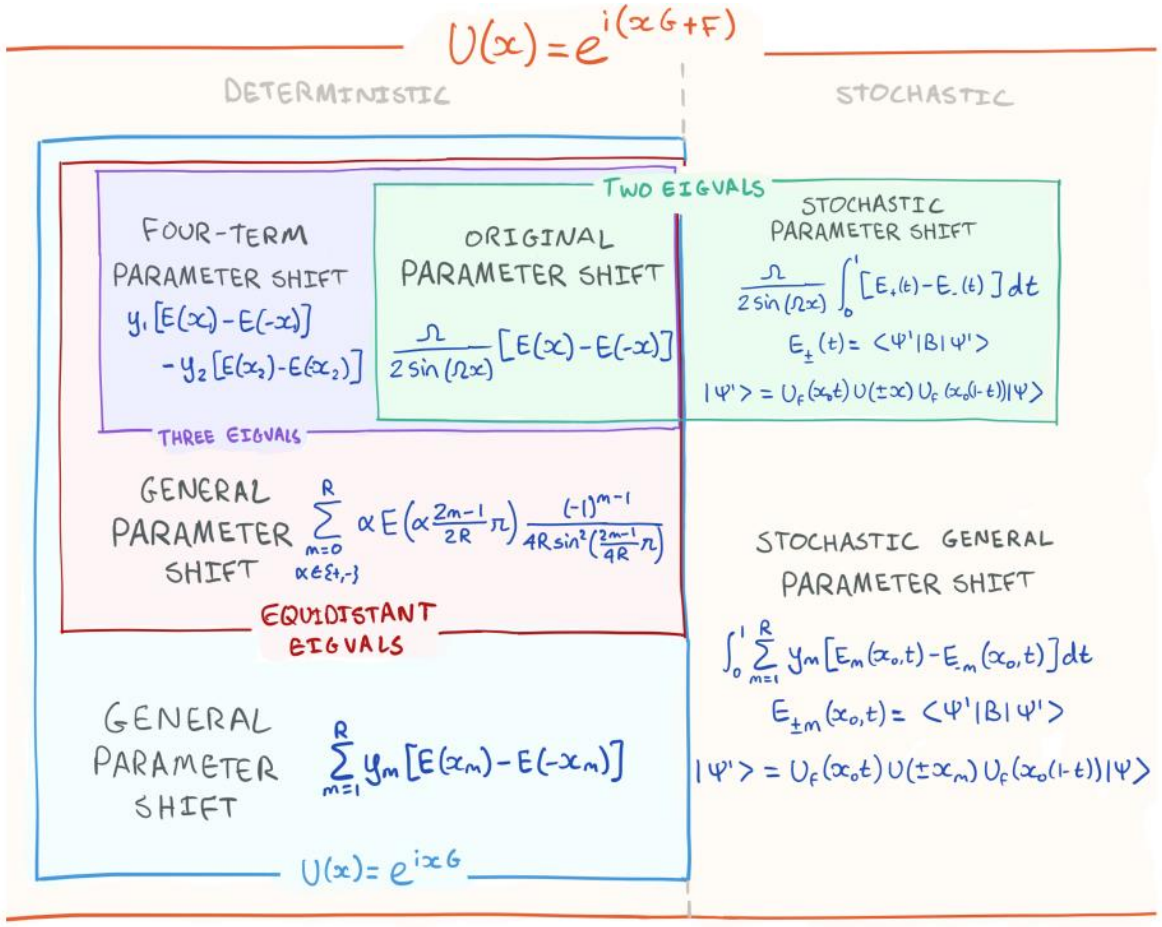
1. Two eigenvalues [LJ17, MK18, Schuld19]
2. Three eigenvalues [WD21, KJAA21]
3. More complicated gate structure - two eigenvalues

$$U_F(x) = \exp(i(xG + F))$$

   - stochastic parameter-shift rule [BLG21]

4. Equidistant eigenvalues [VJD18]
5. General parameter-shift rules [WD21, IART21]

Parameter Shift Rules



History of Parameter-Shift Rules:

- General parameter-shift rules [WD21, IART21]
  - [WD21] derives new, general parameter-shift rules for **single-parameter** quantum gates. It also combines the general rule with the stochastic parameter-shift rule, to extend the framework to **multi-parameter** quantum gates.

Overview of existing and new parameter-shift rules for first-order univariate derivatives as Venn diagram on the space of quantum gates.[WD21]

Linear Combiantion of Unitaries

- Write $U(\boldsymbol{\theta})$ as $U_1(\theta_1)U_2(\theta_2)\ldots U_m(\theta_m)$
  - where $U_i(\theta_i)$ is one of the one-qubit (e.x. R_z) and two-qubit gates (e.x. CR_z) and m is the total number of parameterized gates in the circuit $U(\boldsymbol{\theta})$
- Get gradient of inividual parameter

$$\frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} = U_1(\theta_1)\, U_2(\theta_2) \cdots \frac{\partial U_i(\theta_i)}{\partial \theta_i} \cdots U_m(\theta_m)$$

  - Single-qubit gate gradient, e.x. R_x, R_y, R_z

$$\frac{\partial R_x(\theta)}{\partial \theta} = -i\frac{1}{2} X R_x(\theta)$$

  - Single-qubit gate gradient, e.x. $U3(\theta, \Phi, \lambda)$
    - some symbolic differentiation

```
The circuit for gradient of Rx:
------------x----Rx(1.047)--
            |
--H---SDG---*--------H------
```

```
Circuits for gradient of u3:
-----------z----U--
           |
--H---SDG---*----H--

--Rz(1.746)---------y----Ry(2.101)----Rz(0.798)--
                    |
------H-------SDG---*--------H------------------

--Rz(1.746)----Ry(2.101)----z----Rz(0.798)--
                            |
------H-----------SDG-------*--------H------
```

Linear Combiantion of Unitaries

- Get gradient of inividual parameter
  - Two-qubit gate gradient - control rotation gates e.x. CR_x

$$CR_x(\theta) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R_x(\theta)$$

cannot be represented using one circuit

$$\frac{\partial CR_x(\theta)}{\partial \theta} = |1\rangle\langle 1| \otimes \frac{\partial R_x(\theta)}{\partial \theta} = -\frac{i}{2}|1\rangle\langle 1| \otimes X e^{-i\frac{\theta}{2}X}$$

$$= -\frac{i}{4}\left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R_x(\theta)\right) I \otimes X + \frac{i}{4}\left(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R_x(\theta)\right) Z \otimes X$$

```
The circuit for gradient of cry:
--*----Rzz(1.93)----y--------*------
  |        |        |        |
--U----Rzz(1.93)----|----Ry(2.687)--
                    |
--H-------SDG-------*--------H------

--*----Rzz(1.93)---------y--------*------
  |        |             |        |
--U----Rzz(1.93)----z----|----Ry(2.687)--
                    |    |
--H--------S--------*----*--------H------
```

  - Two-qubit gate gradient - rotation gates e.x. R_{xx}

$$R_{xx}(\theta) = e^{-i\frac{\theta}{2}X\otimes X}$$

$$\frac{\partial R_{xx}(\theta)}{\partial \theta} = -i\frac{1}{2}X \otimes X e^{-i\frac{\theta}{2}X\otimes X}$$

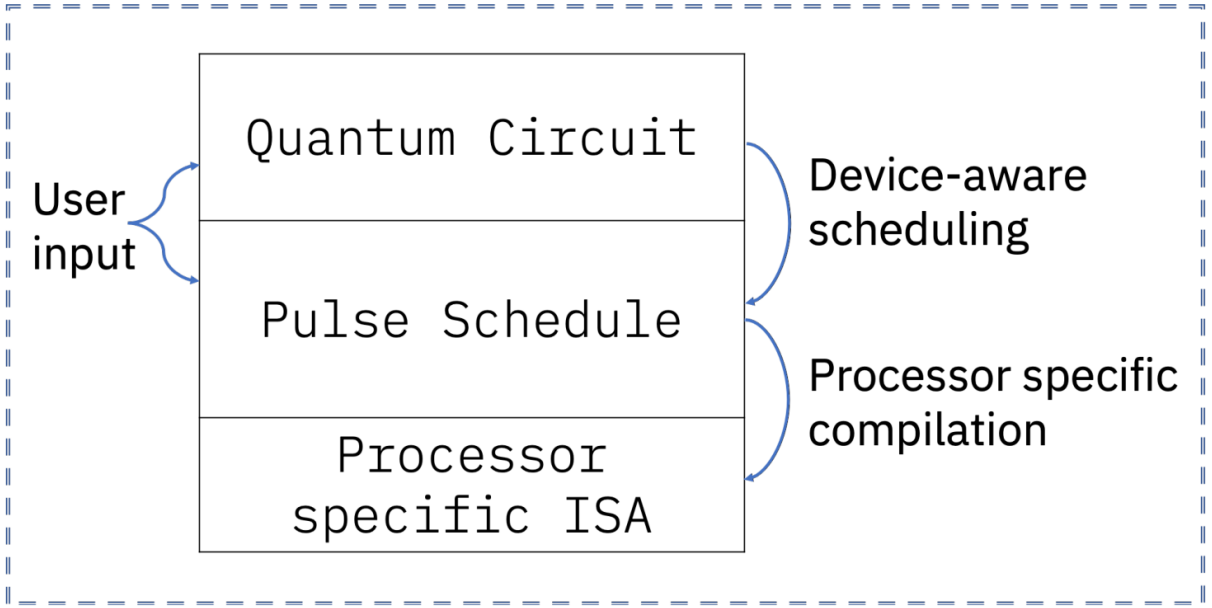
```
The circuit for gradient of rzz:
--*---------z---------Rzz(1.93)--------*------
  |         |             |            |
--U---------|----z----Rzz(1.93)----Ry(2.687)--
            |    |
--H---SDG---*----*--------H------------------
```

Gradient Review

- Numerical Differentiation
  - Finite Difference Method
    - pros: no need to build extra circuits or using extra qubits
    - cons: estimation of gradients, high errors
- Symbolic Differentiation
  - pros: analytic and exact & cons: exhaustive, not always computable
- Automatic Differentiation
  - Parameter-Shift Rule
    - pros: no need for extra circuits, exact and analytic & cons: ?
  - Linear Combinations of Unitaries
    - pros: the most general
    - cons: needs ancilla qubit, takes long time to run on complex circuits

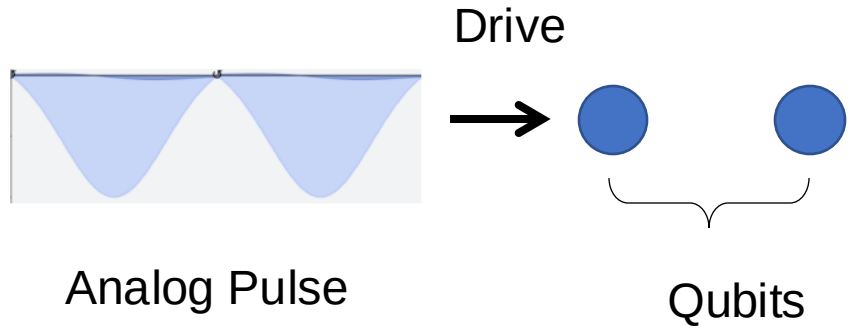
Pulse-Based Variational Quantum Eigensolver

Quantum Pulse Scedule



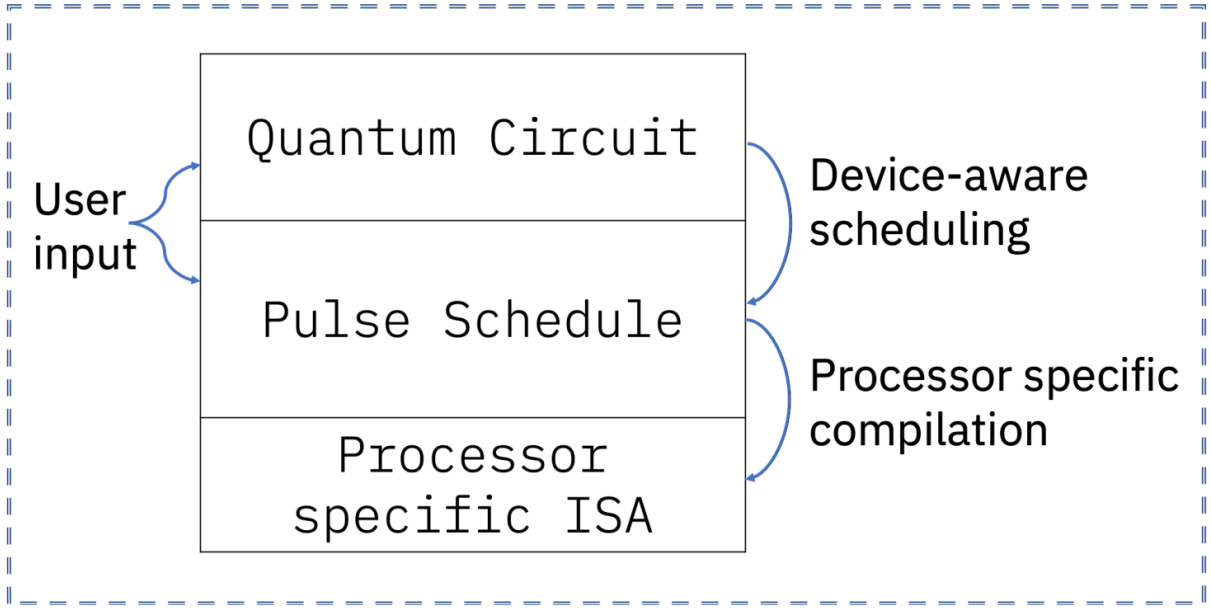
The envisioned quantum program representations and compilation procedures. Circuits are built and optimized first, and then are scheduled into pulse programs by using calibrated native gate definitions.

Recent framework enables advanced users to control the quantum system at the pulse level.

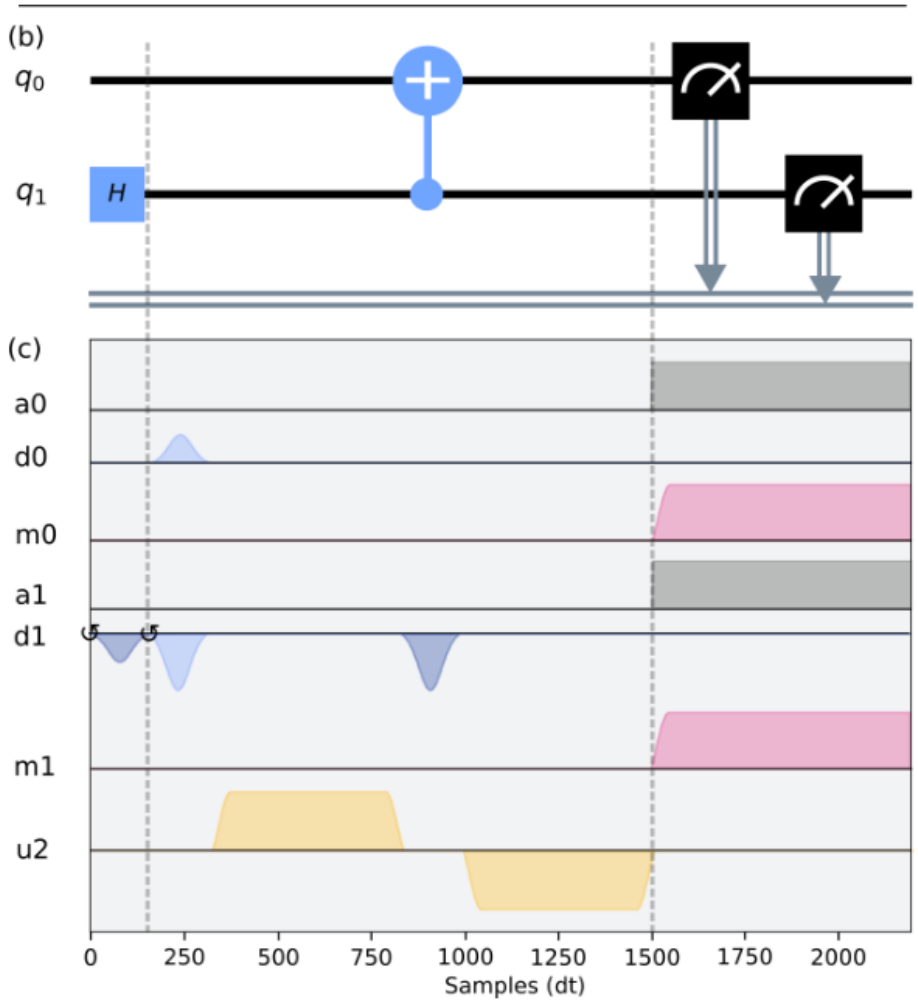


An analog pulse is a control signal that can control the state of qubits in a quantum computer. The effect of each pulse depends on its shape, frequency and amplitude.

Quantum Pulse Scedule

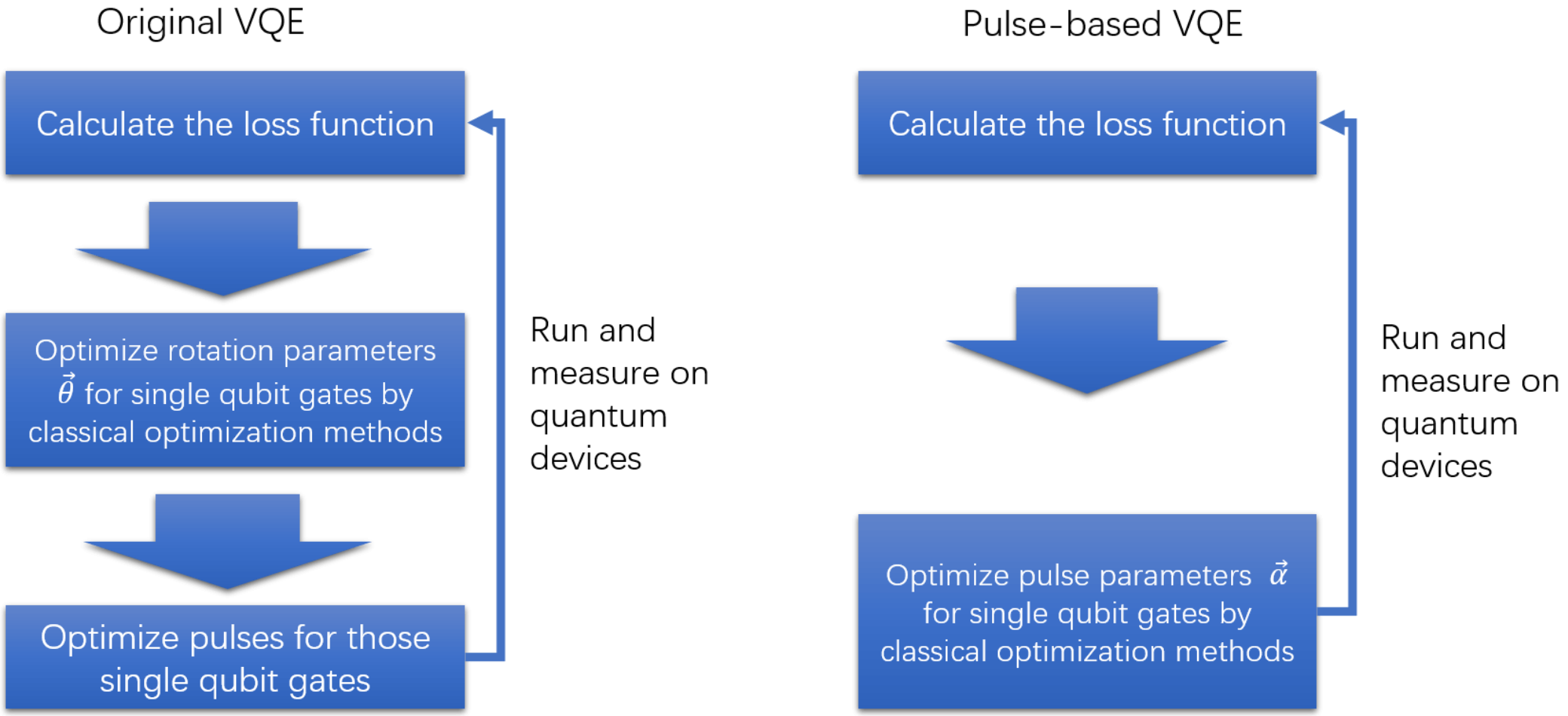


Gates in a quantum circuit are decomposed into several corresponding pulses.



i.e. Bell state circuit and its corresponding pulse sequence.

Pulse-based VQE



Unlike the classical VQE which optimizes the rotation parameters of the single-qubit gates from the logical quantum circuit, pulse-based VQE directly takes the pulse parameters as the optimization parameters to find the minimal loss value

Why Pulse-Based Optimization

- The quantum circuit model hides the underlying physical implementation of gates and measurements on a quantum computer.

- Pulse-Based optimization focuses on the program closer to the hardware than circuit-level optimization.

- Extracting the highest performance out of quantum hardware requires the ability to craft a pulse-level instruction schedule, which cannot be done within the standard circuit model [3].

- **Quantum Optimal Control:** Control the quantum system with highly efficient means (i.e. with analog pulse).

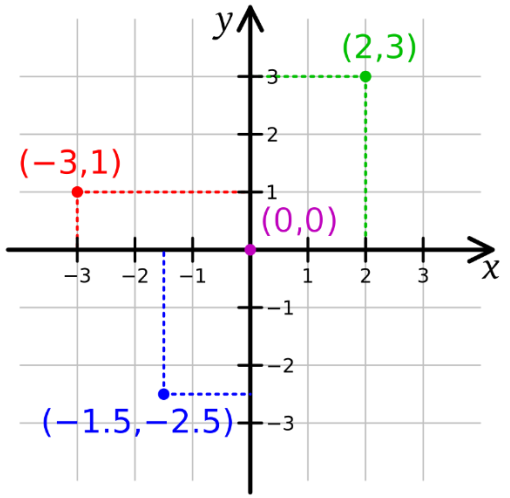
Quantum Optimal Control

- In quantum control we look to prepare some specific quantum state, effect some state-to-state transfer, or effect some transformation (or gate) on a quantum system. A quantum state can be represented by a vector in the 2^n-dim Hilbert space.
- i.e. $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

- Suppose we want to effect a state-to-state transfer

$$|\varphi_0\rangle \rightarrow |\varphi_1\rangle$$

- **Analogy:** Recall the robot control problem. We have a robot at (0,0) and we want to drive it to a target position. The position of the robot can also be represented by a 2-dim vector.



Quantum Optimal Control

- We can use the analog pulse as control signal to drive the state from $|\varphi_0\rangle$ to $|\varphi_1\rangle$.

- So this leads to a question: given a specific quantum system with known time-independent dynamics generator and set of externally controllable fields for which the interaction can be described by control dynamics generators:
    - What is the shape of the control pulse required to achieve?

- GRadient Ascent Pulse Engineering (GRAPE) has been proposed to solve this problem.

GRadient Ascent Pulse Engineering (GRAPE)

In robot control problem, we usually have some dynamics generators to make the robot move.

For quantum system, the dynamics generators of the system are Hamiltonians $H(t)$. The dynamics of the system are governed by *Schrödingers* equation

$$\frac{d}{dt}|\psi\rangle = -iH(t)\,|\psi\rangle$$

The combined Hamiltonian for the system is given by

$$H(t) = H_0 + \sum_{j=1} u_j(t) H_j$$

where H_0 is the drift Hamiltonian and the H_j are the control Hamiltonians. The u_j are time varying amplitude functions for the specific control.

Gradient ascent algorithm can be used to determine a set of u_j that will drive our system from $|\varphi_0\rangle$ to $|\varphi_1\rangle$.

GRadient Ascent Pulse Engineering (GRAPE)

- In the hardware implementation, Time allowed for the system to evolve T is split into M timeslots (typically these are of equal duration). The combined Hamiltonian can then be approximated as

$$H(t) \approx H(t_k) = H_0 + \sum_{j=1}^{N} u_{jk} H_j$$

k is a timeslot index and t_k is the evolution time at the start of the timeslot where t locates, and u_{jk} is the amplitude of control j throughout timeslot k. Then the time evolution operator X within timeslot k can be calculated as

$$X_k := e^{-iH(t_k)\Delta t_k}$$

The evolution up to timeslot t_k is

$$X(t_k) := X_k X_{k-1} \cdots X_1 X_0$$

If the objective is state-to-state transfer, then $\mathrm{X}_0 = |\varphi_0\rangle$ and $\mathrm{X}_{targ} = |\varphi_1\rangle$

GRadient Ascent Pulse Engineering (GRAPE)

A *figure of merit* or *fidelity* f_{PSU} is some measure of how close the evolution X(T) is to the target X_{targ} , based on the control amplitudes u_{jk} in the timeslots. It can be calculated by

$$f_{PSU} = \frac{1}{d}\left|\operatorname{tr}\{X_{targ}^{\dagger} X(T)\}\right|$$

Where d is the dimension of the system.

As there are now $N \times M$ variables (u_{jk} in $H(t) \approx H(t_k) = H_0 + \sum_{j=1}^{N} u_{jk} H_j$) and the fidelity f to maximize, then the problem becomes a finite multi-variable optimization problem, and we can solve this problem with gradient method.

How can we get the gradient for u_{jk} ?

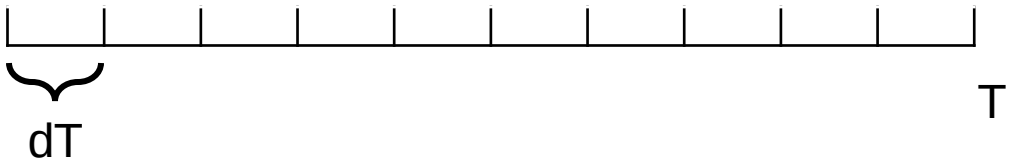
Evaluate the Gradient in Pulse-level Control

	Circuit-Level	**Pulse-Level**
Numerical Differentiation	High errors of near-term quantum devices can make it unfeasible	
Automatic Differentiation through backpropagation	Need to store and reuse the intermediate derivatives. Hard in quantum computing since measurement will impact the overall computation.	
New Strategy	Parameter-shift rule Linear combination rule	

Can we apply these rules to Pulse-level control directly?

Evaluating the Gradient in Pulse-level Control

Naive way: divide the evolution time T into lots of timeslot which have the same duration dT. The amplitude for each control Hamiltonian in each timeslot is constant and can be regarded as a quantum gate. Circuit-level techniques(i.e. parameter shift, Linear combination) can be applied directly.



Drawback: Not scalable. The number of timeslot dT which the evolution time T divided into is usually very large(much larger than a equivalent quantum circuit).

Need a better rule to calculate the pulse sequence's gradient.

Reference

- [WD21] Wierichs, David, et al. "General parameter-shift rules for quantum gradients." arXiv preprint arXiv:2107.12390 (2021).
- [VJD18] Vidal, Javier Gil, and Dirk Oliver Theis. "Calculus on parameterized quantum circuits." arXiv preprint arXiv:1812.06323 (2018).
- [IART21] Izmaylov, Artur F., Robert A. Lang, and Tzu-Ching Yen. "Analytic gradients in variational quantum algorithms: Algebraic extensions of the parameter-shift rule to general unitary transformations." arXiv preprint arXiv:2107.08131 (2021).
- [WD21] Wierichs, David, et al. "General parameter-shift rules for quantum gradients." arXiv preprint arXiv:2107.12390 (2021).
- [PA14] Peruzzo, Alberto, et al. "A variational eigenvalue solver on a photonic quantum processor." Nature communications 5.1 (2014): 1-7.
- [LJ17] Li, Jun, et al. "Hybrid quantum-classical approach to quantum optimal control." Physical review letters 118.15 (2017): 150503.
- [MK18] Mitarai, Kosuke, et al. "Quantum circuit learning." Physical Review A 98.3 (2018): 032309.
- [Schuld19] Schuld, Maria, et al. "Evaluating analytic gradients on quantum hardware." Physical Review A 99.3 (2019): 032331.

Reference

- [KJAA21] Kottmann, Jakob S., Abhinav Anand, and Alán Aspuru-Guzik. "A feasible approach for automatically differentiable unitary coupled-cluster on quantum computers." Chemical Science 12.10 (2021): 3497-3508.
- [BLG21] Banchi, Leonardo, and Gavin E. Crooks. "Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule." Quantum 5 (2021): 386.