Differentiable Pulse-based VQE

Haowei Deng, Kaiyan Shi

September 2021

1 Introduction

Variational Quantum Eigensolver (VQE) is a popular quantum algorithm for approximating the ground state energy of molecules on Noisy Intermediate-Scale Quantum (NISQ) computers. VQE is mainly used to estimate the minimum eigenvalue of a given Hamiltonian and find its ground state. For near-term quantum computers, critical gate errors, decoherence, and poor connectivity limit the depth of quantum circuits. However, the VQE algorithm can work with low-depth quantum circuits. Hence, the VQE algorithm is considered an ideal candidate to utilize NISQ devices to solve real-world problems.

In this project, we study the VQE algorithm at the pulse level, which we call 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 (i.e. ground state energy). The following figure shows the difference between the pulse-based VQE and the standard VQE:

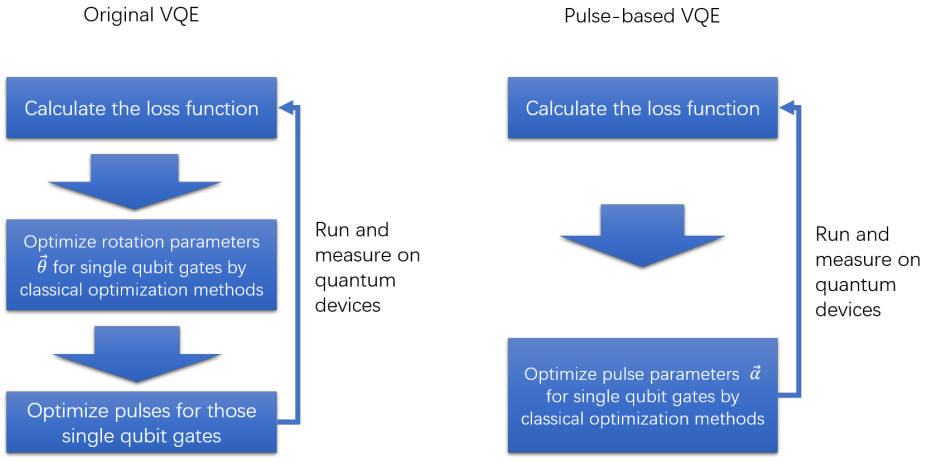

Figure 1: Difference between circuit-base VQE and pulse-base VQE.

ctrl-VQE [4] has shown the benefit of skipping the circuit step and optimizing the pulses to drive

the state directly. [1] also shows that Extracting the highest performance from quantum hardware requires the ability to craft a pulse-level instruction schedule, which cannot be done within the standard circuit model. A common way to implement a pulse-based VQE is to parametrize the pulse schedule and use gradient descent to train the schedule until it provides the desired result. The gradients for each parameter in the pulse schedule are needed to apply the gradient descent method. Previous works [2, 3, 6] calculate the gradients by using finite difference or analytical difference by simulating the quantum evolution with a classical simulator. On the one hand, the high errors of near-term quantum devices can make it unfeasible to use finite difference formulas to approximate the gradient of a circuit. On the other hand, simulating the quantum evolution with a classical simulator is not scalable since the complexity is exponentially related to the number of qubits.

To handle these shortcomings, we need to calculate the gradients for all parameters and ensure that we can implement this calculation process in a quantum computer. In this project, we figure out a new method to parametrize the pulse schedule for VQE. We also figure out a scalable method to calculate the exact gradients of the parameters in this control pulses parametrization under some assumptions.

2 Background

2.1 Basic Quantum Computation

As a classical bit has a state - either 0 or 1 - a qubit also has a state. Two possible sates for a qubit are the states $|0\rangle$ and $|1\rangle$. A qubit can also be represented in the form of a linear combination of states, often called superposition:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where α and β are complex numbers.

For multiple qubits, as one of the four postulates of quantum mechanics, we usually use tensor products to put them together.

Postulate 1. *[5] The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered* 1 *through* n*, and system number* i *is prepared in the state* $|\psi_i\rangle$*, then the joint state of the total system is* $|\psi_i\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$*.*

Another postulate describes how the quantum states of a closed quantum system at two different times are related.

Postulate 2. *[5] The evolution of a closed quantum system is described by a unitary transformation. That is, the state* $|\psi\rangle$ *of the system at time* t_1 *is related to the state* $|\psi'\rangle$ *of the system at time* t_2 *by a unitary operator* U *which depends only on the times* t_1 *and* t_2*,*

$$|\psi'\rangle = U\,|\psi\rangle\,.$$

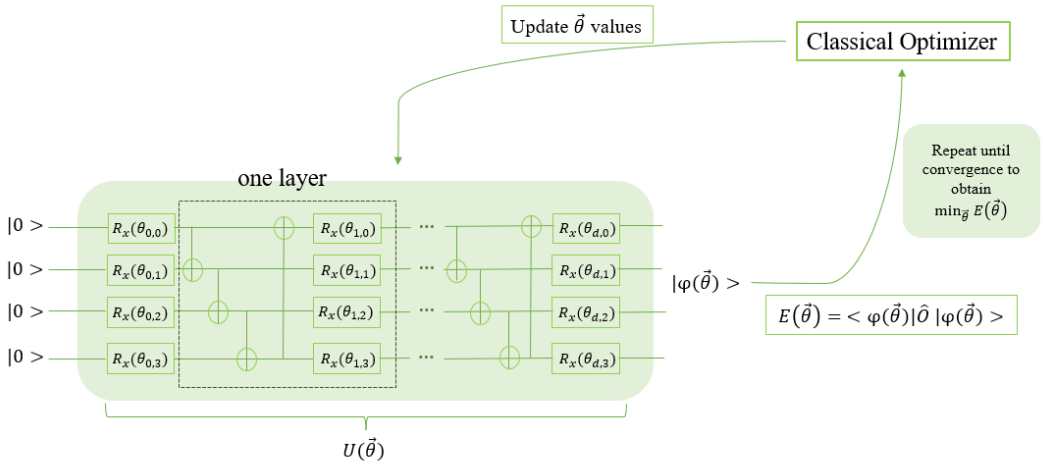

Figure 2: Variational Quantum Eigensolver Flow Chart

For more details on background knowledge of quantum computation, we refer readers to [5] as a good resource for quantum computation and quantum information.

2.2 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a flagship quantum/classical hybrid algorithm for near-term quantum computers. It aims to find the minimum eigenvalue of a given matrix and so is widely adopted in quantum chemistry to find the ground state of a given system.

Given a Hamiltonian $\hat{H}$ which describes a molecule system, and an parameterized circuit $U(\vec{\theta})$ for the problem which defines learnable parameters $(\vec{\theta})$ to optimize. VQE works as presented in Figure 2.2, and also as described in the following:

1. It first transforms the given Hamiltonian into a discretized Hamiltonian H consisting of Pauli spin operators. This procedure is usually performed under Jordan-Wigner transformation.

2. It then prepares the initial state, which is usually all 0 state, $|\varphi_0\rangle = |0\rangle^{\otimes n}$, where n represents the number of qubits needed for the molecule simulation. It also samples random initial parameters $\vec{\theta}$ to be plugged into the parameterized circuit. The output state would be
$$|\varphi\rangle = U(\vec{\theta})\,|\phi_0\rangle\,.$$

3. It uses current parameters $\vec{\theta}$, repeatedly execute the circuit, to minimize the objective function,
$$E(\vec{\theta}) = \langle\varphi(\vec{\theta})|\,H\,|\varphi(\vec{\theta})\rangle\,.$$

4. It checks convergence. If the computed $E(\vec{\theta})$ is optimized until convergence, exit and output the value. Otherwise, it updates $\vec{\theta}$ through classical gradient descent methods and repeats the previous step.

3 Related Work

The Variational Quantum Eigensolver was first proposed in [3], and became a hot area of research since then. The VQE is believed to be one of the most promising examples of Noisy Intermediate-Scale Quantum Computing (NISQ) algorithms as it is relatively robust against of noise and errors caused by current hardware. But it is still hard to implement VQE with deep circuits as ansatz for optimization due to finite coherence times and frequent gate errors. Detailed work on minimizing ansatz circuit depth, maximizing accuracy etc. can be found in [6].

Although there are numerous works on the traditional VQE, few of them study the construction when the gate circuit is removed. Pulse-based VQE with gates removed has been adopted in [4]. Under such construction, the parameterized gate circuit has been replaced by a prameterized laboratory-frame parameterized pulse sequence, to drive the initial state to the target. Both of their method and our method have used analytic gradient in the gradient descent optimization.

The parameterization method of ours is different than theirs. The parameterization method used in [4] is similar to the Gradient Ascent Pulse Engineering (GRAPE) algorithm [2], a widely used optimal control technique. They set the amplitude of each pulse as a parameter, while we assign a n-degree polynomial of time, i.e. n parameters as the amplitude for all pulses. So this gives a better parameter complexity of our algorithm.

Industry companies also has adopted pulse-based VQE. Quanlse from Baidu Inc. has adopted it using the same parameterization method as GRAPE and [4], but it uses finite difference method for gradient calculation, which is less accurate theoretically and slower than analytic gradients.

4 Method

4.1 GRadient Ascent Pulse Engineering (GRAPE)

GRAPE is a quantum optimal control algorithm that aims to optimize the control pulse sequences through by making a piecewise constant approximation to the pulse amplitudes.

Assume the total time allowed for the system to evolve is T and it is split into M timeslots, where it is assumed that the amplitude remains constant. So the Hamiltonian at time $t_k = k\Delta = kT/M$ can be written as:

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

where H_j represents the Hamiltonian of the controller, N is the total number of controllers used, and u_{jk} is the amplitude of the controller j during t_k. If we only have one controller, then the Hamiltonian becomes

$$H(t_k) = H_0 + u_k H_1. \tag{2}$$

The time evolution operator within t_k can be calculated as

$$X_k = e^{-iH(t_k)\Delta t} = e^{-i(u_k H_1 + H_0)\Delta t}. \tag{3}$$

Then the overall time evolution operators up to time T is

$$X(T) = X_{M-1} \dots X_1 X_0 = \prod_{k=M-1}^{0} X_k, \tag{4}$$

and the state after the evolution would be

$$|\varphi\rangle = X(T) |\varphi_0\rangle . \tag{5}$$

Suppose the observable is $\hat{O}$. Then the cost function would be

$$E = \langle\varphi| \hat{O} |\varphi\rangle \tag{6}$$

$$= \langle\varphi_0| \prod_{k=0}^{M-1} e^{iH(t_k)\Delta t} \hat{O} \prod_{k=M-1}^{0} e^{-iH(t_k)\Delta t} |\varphi_0\rangle . \tag{7}$$

Therefore, with only one controller, Q qubits and M timeslots, we would have the parameter complexity being $Q \cdot M$.

4.2 Poly-Parameterization

Given the initial state $|\varphi_0\rangle$, evolution time T, we divide T into M timeslot, with the duration of each timeslot being $\Delta t = \frac{T}{M}$. Suppose we have a polynomial with degree n:

$$p(t) = \sum_{j=0}^{n} a_j t^j \tag{8}$$

In the k^{th} timeslot, with $t_k = k\Delta t$, the Hamiltonian of the system with only one controller can be approximated as

$$H(t_k) = H_0 + p(t_k) H_1. \tag{9}$$

The corresponding time evolution operator can be calculated as

$$X_k = e^{-iH(t_k)\Delta t} = e^{-i(p(t_k)H_1 + H_0)\Delta t} \tag{10}$$

Then the evolution sequence up to time T and the cost function in the form of energy are represented the same as in Equation 4 and 6.

With the polynomial parameterization method, with only one controller, Q qubits and n-degree polynomial approximation of the pulse amplitude, the parameter complexity would be $Q \cdot n$, independent of the number of timeslots M. Normally the degree would smaller than the number of timeslots, and so we earn some constant advantage over traditional parameterization used in GRAPE.

4.3 Analytic Gradient Calculation

In this section, we calculate the corresponding analytic gradient of the energy with respect to each parameter a_j in the polynomial $p(t)$, based on our poly-parameterization method. The gradient can be represented as follows:

$$\frac{\partial E}{\partial a_j} = \langle \varphi_0 | \frac{\partial \left(\prod_{k=0}^{M-1} e^{iH(t_k)dt} \hat{O} \prod_{k=M-1}^{0} e^{-iH(t_k)dt} \right)}{\partial a_j} | \varphi_0 \rangle \tag{11}$$

$$= \langle \varphi_0 | \frac{\partial \left(\prod_{k=0}^{M-1} e^{i(H_0 + p(t_k)H_1)\Delta t} \right)}{\partial a_j} \hat{O} | \varphi \rangle + \langle \varphi | \hat{O} \frac{\partial \left(\prod_{k=M-1}^{0} e^{-i(H_0 + p(t_k)H_1)\Delta t} \right)}{\partial a_j} | \varphi_0 \rangle \tag{12}$$

We then first calculate the left hand side gradient,

$$\frac{\partial \left(\prod_{k=0}^{M-1} e^{i(H_0 + p(t_k)H_1)\Delta t} \right)}{\partial a_j} \tag{13}$$

$$= \sum_{l=0}^{M-1} \left(\Pi_{k=0}^{l-1} e^{i(H_0 + p(t_k)H_1)\Delta t} \left(\frac{\partial e^{i(H_0 + p(t_l)H_1)\Delta t}}{\partial a_j} \right) \Pi_{k=l+1}^{M-1} e^{i(H_0 + p(t_k)H_1)\Delta t} \right). \tag{14}$$

The middle gradient can be calculated as follows:

$$\frac{\partial e^{i(H_0 + p(t_l)H_1)\Delta t}}{\partial a_j} = i\Delta t \left(\frac{\partial p(t_l)}{\partial a_j} \right) \bar{H}_1 e^{i(H_0 + p(t_l)H_1)\Delta t}, \tag{15}$$

where $\bar{H}\Delta t = \int_0^{\Delta t} e^{i(H_0 + p(t_l)H_1)\tau} H_1 e^{i(H_0 + p(t_l)H_1)\tau} d\tau$. But for small $\Delta t \ll \|p(t_l)H_1 + H_0\|^{-1}$, $\bar{H}_1 \approx H_1$. So the gradient can be reduced to

$$\frac{\partial e^{i(H_0 + p(t_l)H_1)\Delta t}}{\partial a_j} = i\Delta t \left(\frac{\partial p(t_l)}{\partial a_j} \right) H_1 e^{i(H_0 + p(t_l)H_1)\Delta t}, \tag{16}$$

$$= i\Delta t \cdot t_l^j H_1 e^{i(H_0 + p(t_l)H_1)\Delta t}. \tag{17}$$

Then Equation 18 can be written as

$$\frac{\partial \left(\prod_{k=0}^{M-1} e^{i(H_0+p(t_k)H_1)\Delta t}\right)}{\partial a_j} \tag{18}$$

$$= \sum_{l=0}^{M-1} \left(\Pi_{k=0}^{l-1} e^{i(H_0+p(t_k)H_1)\Delta t} \left(i\Delta t \cdot t_l^j H_1 e^{i(H_0+p(t_l)H_1)\Delta t}\right) \Pi_{k=l+1}^{M-1} e^{i(H_0+p(t_k)H_1)\Delta t}\right) \tag{19}$$

$$= \sum_{l=0}^{M-1} \left(\Pi_{k=0}^{l-1} e^{i(H_0+p(t_k)H_1)\Delta t} H_1 \Pi_{k=l}^{M-1} e^{i(H_0+p(t_k)H_1)\Delta t} \cdot (i\Delta t \cdot t_l^j)\right) \tag{20}$$

Similar for the right hand side gradient in Equation 11,

$$\frac{\partial E}{\partial a_j} = \langle\varphi_0| \sum_{l=0}^{M-1} \left(\Pi_{k=0}^{l-1} e^{i(H_0+p(t_k)H_1)\Delta t} H_1 \Pi_{k=l}^{M-1} e^{i(H_0+p(t_k)H_1)\Delta t} \cdot (i\Delta t \cdot t_l^j)\right) \cdot \hat{O} |\varphi\rangle \tag{21}$$

$$+ \langle\varphi| \hat{O} \cdot \sum_{l=0}^{M-1} \left(\Pi_{k=l}^{M-1} e^{-i(H_0+p(t_k)H_1)\Delta t} H_1 \Pi_{k=0}^{l-1} e^{-i(H_0+p(t_k)H_1)\Delta t} \cdot (-i\Delta t \cdot t_l^j)\right) |\varphi_0\rangle \tag{22}$$

$$= \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \left(\langle\varphi_0| \Pi_{k=0}^{l-1} e^{iH(t_k)\Delta t} H_1 \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} \cdot \hat{O} |\varphi\rangle - \langle\varphi| \hat{O} \cdot \Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t} H_1 \Pi_{k=0}^{l-1} e^{-iH(t_k)\Delta t} |\varphi_0\rangle\right) \tag{23}$$

There are some commutator relations we could use,

$$H_1 \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} = \left[H_1, \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t}\right] + \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} H_1 \tag{24}$$

$$\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t} H_1 = \left[\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}, H_1\right] + H_1 \Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}. \tag{25}$$

As for the two commutators, if H_1 commute with H_0, i.e. $[H_1, H_0] = 0$, then $[H_1, H(t_k)] = 0$. It then naturally follows that

$$\left[H_1, \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t}\right] = \left[\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}, H_1\right] = 0, \tag{26}$$

based on the property of commutators that $[A, B] = 0$ implying $[A, e^B] = 0$.

The partial gradient 11 can be simplified further given the above commutator relation,

$$\frac{\partial E}{\partial a_j} = \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \left[\langle\varphi_0| \Pi_{k=0}^{l-1} e^{iH(t_k)\Delta t} \left(\left[H_1, \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} \right] + \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} H_1 \right) \cdot \hat{O} |\varphi\rangle \right. \tag{27}$$

$$\left. - \langle\varphi| \hat{O} \cdot \left(\left[\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}, H_1 \right] + H_1 \Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t} \right) \Pi_{k=0}^{l-1} e^{-iH(t_k)\Delta t} |\varphi_0\rangle \right] \tag{28}$$

$$= \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \left(\langle\varphi_0| \Pi_{k=0}^{M-1} e^{iH(t_k)\Delta t} H_1 \cdot \hat{O} |\varphi\rangle - \langle\varphi| \hat{O} \cdot H_1 \Pi_{k=0}^{M-1} e^{-iH(t_k)\Delta t} |\varphi_0\rangle \right) \tag{29}$$

$$+ \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \left(\langle\varphi_0| \Pi_{k=0}^{l-1} e^{iH(t_k)\Delta t} \left[H_1, \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} \right] \cdot \hat{O} |\varphi\rangle \right. \tag{30}$$

$$\left. - \langle\varphi| \hat{O} \cdot \left[\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}, H_1 \right] \Pi_{k=0}^{l-1} e^{-iH(t_k)\Delta t} |\varphi_0\rangle \right) \tag{31}$$

$$= \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \langle\varphi| [H_1, \hat{O}] |\varphi\rangle + \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \left(\langle\varphi_0| \Pi_{k=0}^{l-1} e^{iH(t_k)\Delta t} \left[H_1, \Pi_{k=l}^{M-1} e^{iH(t_k)\Delta t} \right] \cdot \hat{O} |\varphi\rangle \right. \tag{32}$$

$$\left. - \langle\varphi| \hat{O} \cdot \left[\Pi_{k=l}^{M-1} e^{-iH(t_k)\Delta t}, H_1 \right] \Pi_{k=0}^{l-1} e^{-iH(t_k)\Delta t} |\varphi_0\rangle \right) \tag{33}$$

If H_0 and H_1 commute, then based on Equation 26, we would have

$$\frac{\partial E}{\partial a_j} = \sum_{l=0}^{M-1} i\Delta t \cdot t_l^j \langle\varphi| [H_1, \hat{O}] |\varphi\rangle \tag{34}$$

$$= \langle\varphi| [H_1, \hat{O}] |\varphi\rangle \, i\Delta t \sum_{l=0}^{M-1} (l\Delta t)^j \tag{35}$$

$$= \langle\varphi| [H_1, \hat{O}] |\varphi\rangle \, i \, (\Delta t)^{j+1} \sum_{l=0}^{M-1} l^j, \tag{36}$$

which can be used as the gradient in the optimization step.

Compared to finite difference method, which needs to run the whole pulse sequence twice for gradient calculation with respect to each parameter, our gradient calculation only needs to run the sequence once for gradients w.r.t. all parameters. This would decrease the computational cost for gradient calculation from $O(2M)$ to $O(1)$.

However, such gradient calculation has some limitations in implementations. As described in Section 4.2, we have only considered the case of only one controller, i.e. H_1. Also to make the easy formula in Equation 36 work, we require H_0 and H_1 commute. Those limitations restrict applicable ansatz for pulse-based VQE to only a small range, like $H_0 = H_1 = \text{SigmaX}$.

5 Evaluation

We evaluate our differentiation method in Quanlse platform

Quanlse is a cloud-based platform for quantum control and supports the pulse generation for arbitrary single-qubit and two-qubit pulses. It provides toolkits for modeling real superconducting quantum chips, simulating noisy quantum devices and dynamical evolution.

The evaluation task is to find the ground state energy of hydrogen molecule H_2 with an atomic interval of d=74pm and we need to determine the parameters vector $\vec{\theta}$ that minimize the energy E_0. In specific, we construct an antsaz circuit to find the groud state energy of the target Hamiltonian

```
targetHam = [
    [-0.042078976477822, 'i0'],
    [ 0.177712874651399, 'z0'],
    [ 0.177712874651399, 'z1'],
    [-0.242742805131446, 'z2'],
    [-0.242742805131462, 'z3'],
    [ 0.170597383288005, 'z0,z1'],
    [ 0.044750144015351, 'y0,x1,x2,y3'],
    [-0.044750144015351, 'y0,y1,x2,x3'],
    [-0.044750144015351, 'x0,x1,y2,y3'],
    [ 0.044750144015351, 'x0,y1,y2,x3'],
    [ 0.122933050561837, 'z0,z2'],
    [ 0.167683194577189, 'z0,z3'],
    [ 0.167683194577189, 'z1,z2'],
    [ 0.122933050561837, 'z1,z3'],
    [ 0.176276408043195, 'z2,z3']
]
hMatrix = pauliStrToMatrix(targetHam, 4)
```

We evaluate three training methods:

1. Our differentiation method with suitable antsaz.
2. Numerical differentiation with the same antsaz.
3. Numerical differentiation with the best antsaz provided by Quanlse.

We run each method ten times with different random initial state and picks the average final result. Tab When using the same ansatz, our method can achieve the same final loss as the

Method	**Ave. Loss**	**Variance**	**Ave. Runtime**	**Ave. Iteration**
Method 1	-0.5784	0.09741	6.1 min	44
Method 2	-0.5789	0.08388	11 min	42
Method 3	-1.1321	0.0121	9 min	35

Table 1: Evaluation result for three training method. Lower average loss and average runtime is better.

numerical differentiation method in half the time. But the restriction of our method limits that we cannot use the best ansatz which can get the best result.

6 Conclusion and Further Work

We have proposed a different paramertization method "poly-parameterization", to run pulse-based VQE, i.e. parameterizing the pulse sequences via a polynomial. This parameterization with a n-degree polynomial has reduced the number of parameters used by a constant factor, from $M \cdot Q$ to $n \cdot Q$, where M is the number of time segments and Q is the number of qubits.

We also have derived an analytic gradient formula, which can be used to optimize the pulse sequence in the pulse-based VQE. This gradient calculation method only needs $O(1)$ computational cost for running the pulse sequence, while ordinary finite difference method needs $2M = O(M)$. However, this analytic gradient only works for a limited number of pulse sequence type, i.e. only one controller and this controller commutes with the drift controller.

Implementations are also provided, which verifies that our proposed pulse-based VQE with proposed analytic gradient is fast. But the results also show that we cannot reach the theoretical minimum due to pulse sequence ansatz choices, which are limited by our gradient calculation method.

For further work, we would like to try different parameterization methods which could further reduce the parameter complexity, and optimistically easy to compute gradients. Advanced numerical techniques can also be applied to approximate commutators in analytic gradient calculations for a wider range of ansatz. Also instead of analytic gradient, sampling and simulation techniques can also be considered in approximating gradients.

References

[1] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. Optimized quantum compilation for near-term algorithms with openpulse. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 186–200. IEEE, 2020.

[2] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrüggen, and Steffen J Glaser. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *Journal of magnetic resonance*, 172(2):296–305, 2005.

[3] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[4] Oinam Romesh Meitei, Bryan T Gard, George S Barron, David P Pappas, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. Gate-free state preparation for fast

variational quantum eigensolver simulations: ctrl-vqe. *arXiv preprint arXiv:2008.04302*, 2020.

[5] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[6] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth, et al. The variational quantum eigensolver: a review of methods and best practices. *arXiv preprint arXiv:2111.05176*, 2021.