The Variational Quantum Eigensolver: a review of recent research and best practices

A B[a]

[a]*Rahko Limited, N4 3JP London, United Kingdom*

ARTICLE INFO

Keywords:
Some very
~~cool~~
keywords

Abstract

AAA

Contents

1. Optimization strategies

The VQE is in essence an optimization problem, it aims at heuristically constructing an approximation of an electronic wavefunction through iterative learning of ansatz parameters. For the algorithm to be viable, it must be that it can learn a good enough approximation of the solution within a tractable number of learning steps. It was already demonstrated that optimization of the variational quantum ansätze is NP-hard [**?**], meaning that there exist at least some problems in which finding an exact solution for the VQE problem is intractable. As such, efficient optimization strategies that provide a well-approximated solution within an acceptable number of iterations are essential for any variational algorithms to be put into practice. Compared to the conventional numerical optimization problem, however, optimizing the expectation value of a variational quantum ansatz faces additional challenges:

- Sampling noise and gate noise on NISQ devices disturb the landscape of the objective function. Such noise can be detrimental to the convergence of optimization [**?** **?**], and could limit the scope for quantum advantage [**?**].
- While the precision of conventional numerical optimization is generally not considered a problem, the precision of the measured expectation value is limited by the sample shot number. The cost of optimization is heavily dependent on the precision required for optimization.
- Related to the point above, the landscape of the expectation value of variational ansatz may cause the vanishing of gradient very easily as a result of the barren plateau problem [**?**] (See Sec. **??** for further details).

On the flip side, studies from recent years show the landscape of expectation value has some analytical properties that are useful to extract information, such as evaluating gradients directly on quantum devices [**?** **?** **?** **?**]. In addition, the ansatz' landscape can be efficiently approximated to accelerate the convergence [**?** **?**]. Utilizing such prior knowledge helps to develop efficient optimization strategies for variational quantum algorithms. This section presents the most relevant optimizers used in the context of the VQE and recent studies of optimization strategies, focusing on the strategies that target fast convergence rates. Some optimizers are developed to reduce the shot numbers [**?** **?**], for which specific applications are discussed in section **??**. We also compare these proposed algorithms and discuss the current challenge facing the optimization of variational quantum algorithms.

ORCID(s):

1.1. Background and notation

The objective function of a variational algorithm is constructed conventionally based on the measurement outcome. Denote $\mathbf{O}(\boldsymbol{\theta}) = (\hat{O}_1(\boldsymbol{\theta}^{(1)}), \hat{O}_2^{(2)}(\boldsymbol{\theta}^{(2)}), \S, \hat{O}_a(\boldsymbol{\theta}^{(a)}))$ are the observables used to compose the objective function and a is the number of observables. The objective function is given by

$$\mathcal{L}(\boldsymbol{\theta}) = C(\mathbf{O}(\boldsymbol{\theta})) \tag{1}$$

where C is a function that maps the observed expectation value to the objective function, and usually have the simple linear form

$$C(\mathbf{X}) = \sum_i c_i X_i \tag{2}$$

where c_i is a constant defined by the problem as the coefficient of each measurement expectation value, X_i is the i-th component of $\mathbf{X}$. Note that such linear form preserves the analytical properties and it is essential for using the analytical methods to directly calculate the gradient or implement analytical gradient free optimization strategy [**? ? ? ? ? ?**].

A measurement expectation value is given by

$$\langle \hat{O}_k(\boldsymbol{\theta}^{(k)}) \rangle = \langle \psi_0 | \, U^{(k)\pounds}(\boldsymbol{\theta}^{(k)}) \hat{M}_k U^{(k)}(\boldsymbol{\theta}^{(k)}) \, | \psi_0 \rangle \,, \tag{3}$$

where $|\psi_0\rangle$ is the initial state on the quantum computer. $\hat{M}^{(k)}$ is a Hermitian measurement operator, usually chosen to be the tensor product of Pauli operators to match the physical measurement implementation on quantum hardware. $U_k(\boldsymbol{\theta}^{(k)})$ is the variational ansatz defined as

$$U^{(k)}(\boldsymbol{\theta}^{(k)}) = \prod_j U_j^{(k)}(\theta_j^{(k)}) \tag{4}$$

and each U_j is a quantum gate, which is generalized as

$$U_j^{(k)}(\boldsymbol{\theta}_j^{(k)}) = \exp\Big(i\theta_j^{(k)} P_j^{(k)} \Big) \tag{5}$$

where $P_j^{(k)}$ is a Hermitian matrix, usually is a tensor product of Pauli operators.

It is sometimes convenient to utilize the superoperator formalism and consider the noise into the optimization process, the expectation value can be written as

$$\langle \hat{O}_k(\boldsymbol{\theta}^{(k)}) \rangle = \mathrm{Tr}[\hat{M}_k \Phi^{(k)}(\boldsymbol{\theta}^{(k)}) \rho_0], \tag{6}$$

where ρ_0 is the initial state density operator and $\Phi^{(k)}(\boldsymbol{\theta}^{(k)})$ denote the transformation matrix, which is given by

$$\Phi^{(k)}(\boldsymbol{\theta}^{(k)}) = \prod_j \Phi_j^{(k)}(\theta_j^{(k)}). \tag{7}$$

For devices that support single shot readout, each sample the quantum device would yield a bit string $\mathbf{s}$. For each string $\mathbf{s}$ a measurement value could be calculated with $M_i(\mathbf{s})$, and the expectation value is the average of each $M_k(\mathbf{s})$.

$$\langle \hat{O}_k(\boldsymbol{\theta}^{(k)}) \rangle = \sum_j \mathrm{Prob}(\mathbf{s}(\boldsymbol{\theta}^{(k)})) = b_j M_k(b_j) \tag{8}$$

where b_j Ë B, B covers all possible single shot bit string (all binary numbers from 0 to 2^{n*1}) of the measurement outcome.

Due to the physical implementation from the quantum hardware, not all quantum computing systems support single-shot readout. Some systems can only yield expectation value by averaging the signal from the readout. For example, some NMR systems use an ensemble of molecules to implement quantum computing and cannot read the state of each single molecule [**? ?**]. In practice, the quantum hardware system may directly yield an expectation value, and the measurement approaches vary from different physical systems.

In the following discussion, the upper label "(k)" is simplified when there is no ambiguity.

1.2. Gradient evaluation

A significant amount of numerical optimization methods requires the knowledge of the gradient of the objective function at a given parameter value. Therefore the evaluation of gradient is rather important and we would cover this topic in a separate subsection.

1.2.1. Stochastic approximation methods

Stochastic approximation (SA) is a family of methods used to reconstruct the properties of the expectation value of a function that depends on some random variables. Instead of measuring the expectation values directly, SA recovers the properties of the expectation value with random sampling. SA has been widely used for big data and machine learning applications when the objective function is too costly to evaluate directly. In the context of variational quantum algorithms, the objective function is some function of the expectation value of the quantum state; evaluating the expectation value of a variational ansatz is expensive. Therefore SA naturally fits into the context of variational quantum algorithms and has been used to approximate the gradient of the expectation values.

Finite difference stochastic approximation (FDSA) One of the simplest methods to approximate the gradient is evaluating two function points and using its difference as the approximated gradient. Finite difference stochastic approximation (FDSA) [**?**] is given by

$$(g(\boldsymbol{\theta}_t))_j = \frac{\mathcal{L}(\boldsymbol{\theta} + c_t \mathbf{e}_j) * \mathcal{L}(\boldsymbol{\theta} * c_t \mathbf{e}_j)}{2c_t}, \tag{9}$$

where $(g(\boldsymbol{\theta}_t))_j$ is j-th component of the gradient at point $\boldsymbol{\theta}_t$. $\mathbf{e}_j$ is the unit vector of the j-th dimension, c_t is the displacement value that generally decrease with iteration step number t [**?**].

Simultaneous perturbation stochastic approximation (SPSA) SPSA algorithm approximates the gradient with only two measurements of the objective function. Instead of choosing the measurement points symmetrically, SPSA uses a small random vector to perturb the objective function. While FDSA calculates the gradient at one direction of the variable, SPSA calculates a direction composed of multiple variables.

$$(g(\boldsymbol{\theta}_t))_j = \frac{\mathcal{L}(\boldsymbol{\theta} + c_{t\ j}) * \mathcal{L}(\boldsymbol{\theta} * c_{t\ j})}{2c_t(\Delta_t)_j}, \tag{10}$$

where Δ_t is a random perturbation vector. The SPSA algorithm needs to measure two points per iteration, which is friendly to variational quantum algorithms. Also it can be applied to noisy optimization [**? ? ?**].

The SPSA methods can approximate the preconditioned gradient for second order optimization methods [**?**]. Explanation of second order optimization methods are covered in Sec. **??**. Consider the preconditioned gradient $\breve{g}(\boldsymbol{\theta}) = \boldsymbol{F}^{*1} g(\boldsymbol{\theta})$ where $\boldsymbol{F}$ is some metric with information from second order derivatives of function $f(\boldsymbol{\theta})$. Denote $\tilde{\boldsymbol{F}}$ as the approximated value for $\boldsymbol{F}$, we have

$$\tilde{\boldsymbol{F}}^{(k)} = \frac{\delta f}{2\epsilon^2} \frac{\Delta_1^{(k)} \Delta_2^{(k)T} + \Delta_2^{(k)} \Delta_1^{(k)T}}{2} \tag{11}$$

where

$$\begin{aligned}
\delta f^{(k)} = & f(\boldsymbol{\theta}^{(k)} + \epsilon\Delta_1^{(k)} + \epsilon\Delta_2^{(k)}) && (12)\\
& * f(\boldsymbol{\theta}^{(k)} + \epsilon\Delta_1^{(k)}) && (13)\\
& * f(\boldsymbol{\theta}^{(k)} * \epsilon\Delta_1^{(k)} + \epsilon\Delta_2^{(k)}) && (14)\\
& + f(\boldsymbol{\theta}^{(k)} * \epsilon\Delta_1^{(k)}) && (15)\\
& && (16)
\end{aligned}$$

The approximated value is estimated from all previously evaluated values with an exponentially smooth estimator

$$\bar{F}^{(k)} = \frac{k}{k+1}\bar{F}^{(k*1)} + \frac{1}{k+1}\tilde{F}^{(k)} \tag{17}$$

1.2.2. Analytical gradient calculation

The gradient of measurement observables can be evaluated on quantum computers directly by utilizing the analytical property of the ansatz. For ansätze made from a sequence of parametrized quantum gates, the partial derivative is

$$\frac{\partial \langle \hat{O}_k(\boldsymbol{\theta})\rangle}{\partial \theta_j} = 2\,\mathrm{Im}(\langle \boldsymbol{\phi}_0 | V_k^{j\text{£}}(\boldsymbol{\theta}) \hat{\boldsymbol{M}}_k U(\boldsymbol{\theta}) | \boldsymbol{\phi}_0 \rangle), \tag{18}$$

where the operator $V_k^j(\mathbf{t})$ is defined as inserting $\boldsymbol{P}_k^j$ between the $(j*1)$-th term and j-th term, which comes from the derivative of the j-th term:

$$V_k^j(\boldsymbol{\theta}) = e^{i\theta_{N_k} P_k^{N_k}} \cdots e^{i\theta_j P_k^j} \boldsymbol{P}_k^j e^{i\theta_{j*1} P_k^{j*1}} \cdots e^{i\theta^1 P_k^1}. \tag{19}$$

The difficulty of evaluating the gradient directly is that $V_k^{j\text{£}}(\boldsymbol{\theta})\hat{\boldsymbol{M}}_k U(\boldsymbol{\theta})$ is usually not Hermitian, therefore there is no known method that can directly convert it to a quantum circuit. Two methods to analytically measure the gradient have been developed, and it is worth noticing that these two methods are measuring the gradient with the exact same mechanism, but with direct and indirect measurements [**?**]. Here, direct measurement means the observed quantity is encoded into a single quantum observable, while indirect measurement calculates the quantity with multiple quantum observables.

Direct analytical gradient measurement The imaginary part of $\langle \boldsymbol{\phi}_0 | V_k^{j\text{£}}(\boldsymbol{\theta}) \hat{\boldsymbol{M}}_k U(\boldsymbol{\theta}) | \boldsymbol{\phi}_0 \rangle$ can be directly evaluate by introducing an ancillary qubit [**?**]. This is done by first prepare the ancillary qubit into the $1/(\sqrt{2})(|0\rangle + |1\rangle)$ state, and prepare the rest of quantum register into $|\boldsymbol{\phi}(\boldsymbol{\theta})\rangle$ state. Now the system has the state

$$|\boldsymbol{\psi}\rangle = \frac{\sqrt{2}}{2}(|0\rangle + |1\rangle) \otimes |\boldsymbol{\psi}(\boldsymbol{\theta})\rangle\,. \tag{20}$$

We can then apply the $\boldsymbol{P}_j$ gate controlled by the ancillary qubit and apply the remaining unitary. Finally we apply the measurement observable controlled by the ancillary qubit. This gives the new state

$$|\boldsymbol{\psi}\rangle = \frac{(|0\rangle \otimes \boldsymbol{U}(\boldsymbol{\theta})\,|\boldsymbol{\psi}_0\rangle + |1\rangle \otimes \hat{\boldsymbol{M}}_k V_k^j(\boldsymbol{\theta})\,|\boldsymbol{\phi}_0\rangle)}{\sqrt{2}}. \tag{21}$$

Now we apply the Hadamard gate to the ancillary qubit, which gives

$$|\boldsymbol{\psi}\rangle = \frac{|0\rangle \otimes (\boldsymbol{U}\,|\boldsymbol{\phi}_0\rangle + \hat{\boldsymbol{M}}_k V_k^j(\boldsymbol{\theta})\,|\boldsymbol{\phi}_0\rangle) + |1\rangle \otimes (\boldsymbol{U}\,|\boldsymbol{\phi}_0\rangle * \hat{\boldsymbol{M}}_k V_k^j(\boldsymbol{\theta})\,|\boldsymbol{\phi}_0\rangle)}{2}. \tag{22}$$

The imaginary part of $\langle \boldsymbol{\phi}_0 | V_k^{j\text{£}}(\boldsymbol{\theta}) \hat{\boldsymbol{O}}_i \boldsymbol{U}(\boldsymbol{\theta}) | \boldsymbol{\phi}_0 \rangle$ is now encoded in the ancillary qubit in the $\boldsymbol{Y}$-basis.

Indirect analytical gradient measurement To illustrate this method first we make some modifications to our notations. We now absorb the common term of $\boldsymbol{U}$ and $\boldsymbol{V}$ into the state and measurement operator, and define

$$\begin{aligned}
\boldsymbol{W}_{k,1}(\boldsymbol{\theta}) &= \exp\left(i\theta_{j*1}\boldsymbol{P}_k^{j*1}\right) \cdots \exp\left(i\theta^1 \boldsymbol{P}_k^1\right) \\
\boldsymbol{W}_{k,2}(\boldsymbol{\theta}) &= \exp\left(i\theta_n \boldsymbol{P}_k^n\right) \cdots \exp\left(i\theta_j \boldsymbol{P}_k^j\right) \\
|\boldsymbol{\phi}(\boldsymbol{\theta})\rangle &= \boldsymbol{W}_{k,1}(\boldsymbol{\theta})\,|\boldsymbol{\phi}_0\rangle \\
\boldsymbol{Q}_k(\boldsymbol{\theta}) &= \boldsymbol{W}_{k,2}^{\text{£}} \hat{\boldsymbol{M}}_i \boldsymbol{W}_{k,2}.
\end{aligned} \tag{23}$$

Then we have

$$\frac{\partial \langle \hat{O}_k(\boldsymbol{\theta}) \rangle}{\partial \theta_j} = 2\,\mathrm{Im}(\langle \phi(\boldsymbol{\theta}) | Q_k(\boldsymbol{\theta}) P_k | \phi(\boldsymbol{\theta}) \rangle). \tag{24}$$

We use following construct which is related to $\langle \phi(\boldsymbol{\theta}) |\, P_k Q_k(\boldsymbol{\theta})\, | \phi(\boldsymbol{\theta}) \rangle$:

$$\langle \phi(\boldsymbol{\theta}) | (\mathbb{1} - i P_k) Q_k(\boldsymbol{\theta}) (\mathbb{1}\,, i P_k | \phi(\boldsymbol{\theta}) \rangle = -\,\mathrm{Im}(\langle \phi(\boldsymbol{\theta}) | P_k Q_k(\boldsymbol{\theta}) | \phi(\boldsymbol{\theta}) \rangle) + \boldsymbol{C}, \tag{25}$$

where $\boldsymbol{C} = \langle \phi(\boldsymbol{\theta}) |\, Q_k(\boldsymbol{\theta})\, | \phi(\boldsymbol{\theta}) \rangle + \langle \phi(\boldsymbol{\theta}) |\, P_k Q_k(\boldsymbol{\theta}) P_k\, | \phi(\boldsymbol{\theta}) \rangle)$ is a constant value.

For single qubit gate given by $\boldsymbol{U}_j(\theta_j) = \exp(i\theta_j P_k), P_k$ Ë $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are Pauli matrices and with exactly two eigenvalues, we have [**?**]

$$\mathbb{1}\,, i P_k = \exp(-\frac{\pi}{4} i P_k), \tag{26}$$

therefore

$$\langle \phi(\boldsymbol{\theta}) | (\mathbb{1} - i P_k) Q_k(\boldsymbol{\theta}) (\mathbb{1}\,, i P_k | \phi(\boldsymbol{\theta}) \rangle \tag{27}$$

$$= \langle \phi(\boldsymbol{\theta}\,, \frac{\pi}{2} \mathbf{e}_j) |\, Q_k(\boldsymbol{\theta}\,, \frac{\pi}{2} \mathbf{e}_j)\, | \phi(\boldsymbol{\theta}\,, \frac{\pi}{2} \mathbf{e}_j) \rangle\,, \tag{28}$$

and

$$\frac{\partial \langle \hat{O}_k(\boldsymbol{\theta}) \rangle}{\partial \theta_j} = \langle \hat{O}_k(\boldsymbol{\theta} + \frac{\pi}{2} \mathbf{e}_j) \rangle * \langle \hat{O}_k(\boldsymbol{\theta} * \frac{\pi}{2} \mathbf{e}_j) \rangle\,. \tag{29}$$

The parameter shift rule can be generalized to a general parameter shift rule and give gradient g_{gPSR} to obtain higher-order derivatives [**? ?**].

$$g_{gPSR}(\theta, \gamma_1, \gamma_2) := r[L(\theta + \gamma_1) + L(\theta + \gamma_2)] \tag{30}$$

where $2r$ is the difference of the eigenvalue of the gate generator $\boldsymbol{G}$. For single-qubit gates with Pauli operators as the generator, $r = 1$.

$$g_{gPSR}(\theta, \gamma_1, \gamma_2) = \frac{sin(2r\gamma_1) + sin(2r\gamma_2)}{2} g^{(1)}(\theta) * \frac{cos(2r\gamma_1) * cos(2r\gamma_2)}{4r} g^{(2)}(\theta), \tag{31}$$

and because of the sinusoidal property of the expectation value, the k-th order derivative has a constant multiplier different to the $k * 2$-th order derivative.

$$g^{(k+2)} = *\frac{1}{4r} g^{(k)}. \tag{32}$$

So far the analytical gradient methods only apply to parametrized single-qubit gates. The relation presented in Eq. **??** holds only when P_k has exactly two different eigenvalues. To use the parameter shift rule for arbitrary two-qubit gates, one can decompose the multi-qubit gate into a sequence of single-qubit gate and product of the same Pauli matrices, which always has two different eigenvalues [**?**]. When the operators has 3 different eigenvalues, the gradient can be evaluated with a modified 4-value shift rule [**?**]. For other operators, the value $\boldsymbol{P}$ can be polynomially expanded into a linear combination of low-rank (2 or 3 eigenvalues) operators [**?**]. Alternatively, the objective function can be decomposed into trigonometric polynomials, and the gradient can be evaluated with trigonometric interpolation methods [**? ?**].

It is worth mentioning that for VQE, the parameter shift rule can be implemented before encoding the fermionic excitation to the qubits, which reduces the required measurement amount [**?**].

1.3. Gradient-based searching strategy

Gradient-based optimization strategies utilize information from cost function derivatives. First-order optimizers utilize only the first-order derivatives of the cost function, and second-order optimizers utilize both first-order and second-order derivatives, at the cost of computing the second-order derivatives. In this section, we discuss some of the popular gradient-based optimizers.

1.3.1. First order optimizers

First-order optimizers for variational ansatz are mostly borrowed from the deep learning communities. These methods have been widely used for the early studies of variational quantum algorithms.

Simple gradient descent Simple gradient descent is the simplest first-order method searching for local minimum with gradient [**?** **?**]. The algorithm takes a step towards the opposite direction of the gradient, and the step size is calculated based on the absolute value of the gradient and a meta-parameter η usually referred to as the learning rate. Almost all the gradient-based methods are developed based on the idea of simple gradient descent.

Algorithm 1: Simple gradient descent

η*: Learning rate.*
while *Not converged* **do**
  Calculate gradient $g_t = g(\theta_t)$;
  $\theta_{t+1} = \theta_t * \eta g_t$
end

Adaptive optimizers well developed from the deep learning community are adapted to the ansatz parameter optimization.

RMSProp RMSProp is an adaptive learning rate optimization method. The RMSProp divides the gradient by the weighted moving average of the root mean square of gradient value, which enhances the direction of the gradient and reduces the significance of the gradient magnitude.

Algorithm 2: RMSProp optimizer

γ*: Moving average parameter.*
η*: Learning rate.*
while *Not converged* **do**
  Calculate gradient $g_t = g(\theta_t)$;
  $E[g^2]_t = \gamma E[g^2]_{t*1} + (1 * \gamma) g_t^2$;
  $\theta_{t+1} = \theta_t * \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$
end

Adam optimizer Adaptive moment (Adam) optimizer [**?**] is a widely used optimizer developed from the deep learning community to solve the stochastic gradient-based optimization problem. The Adam optimizer performs efficient stochastic optimization with only first-order gradient. Adam optimizers utilize adaptive moment estimation as an extra on RMSProp. Instead of descent to the gradient direction, it jumps towards the momentum direction, the weighted moving average of the gradient.

Algorithm 3: Adam optimizer

β_1*: Moving average parameter for past gradients.*
β_2*: Moving average parameter for past squared gradients.*
while *Not converged* **do**
Calculate gradient $g_t = g(\theta_t)$
$m_t = \beta_1 m_{t*1} + (1 * \beta_1) g_t$;
$v_t = \beta_2 v_{t*1} + (1 * \beta_2) g_t^2$;
$\ddot{m_t} = \frac{m_t}{1 * \beta_1^t}$;
$\ddot{v_t} = \frac{v_t}{1 * \beta_2^t}$;
$\theta_{t+1} = \theta_t * \frac{\eta}{\sqrt{\ddot{v_t}} + \epsilon} \ddot{m_t}$;
end

Since the Adam optimizer utilizes moving average for both momentum and magnitude estimations, it can be applied to scenarios with noisy gradients.

1.3.2. Second order optimizers

Second-order optimization techniques make use of the second-order derivative of the objective function to determine the descent direction.

Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm The BFGS algorithm [**? ? ? ?**] is a gradient-based iterative approach to solve standard nonlinear optimization problems. The direction of each step is obtained by preconditioning the gradient with curvature information. Preconditioning the gradient in the context of optimization means it modifies the gradient before taking the descending step. The BFGS algorithm modifies the gradient based on the curvature information obtained by the Hessian. In practice, the Hessian is difficult to calculate precisely, even for conventional problems with large parameter space. A different variant of the BFGS method has been developed to approximate the Hessian, such as Limited-memory BFGS (L-BFGS) [**?**].

Algorithm 4: BFGS algorithm

$\boldsymbol{B}_k$*: Approximated Hessian at step* k*.*
$\mathcal{L}(\theta)$*: Objective function*
α_k*: stepsize at step* k
$\mathbf{p}_k$*: Descent direction*
Calculate $\boldsymbol{B}_0$ from initial guess x_0;
while *Not converge* **do**
Obtain $\mathbf{p}_k$ by solving $\boldsymbol{B}_k \mathbf{p}_k = *(\mathcal{L}(\theta_k)$;
Perform line search for $\alpha_k = \operatorname{argmin} \mathcal{L}(\theta_k + \alpha \mathbf{p}_k)$;
Set $\mathbf{s}_k = \alpha_k \mathbf{p}_k$ and update $\theta_{k+1} = \theta_k + \mathbf{s}_k$;
Set $\mathbf{y}_k = (\mathcal{L}(\theta_{k+1}) * (\mathcal{L}(\theta_k)$;
Set $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} * \frac{\boldsymbol{B}_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} \boldsymbol{B}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \boldsymbol{B}_k \mathbf{s}_k}$;
end

Although the Hessian of a variational ansatz landscape can be calculated with analytical method [**?**], it is still difficult to calculate the full Hessian. Therefore this method is only used in the very early age of simulating conventional algorithms or as a baseline of comparison for lately developed optimizers. The SPSA algorithm from section **??** can be used to approximate a Hessian.

Quantum natural gradient Natural gradient descent is a well-studied second-order optimization technique for numerical optimization. Natural gradient descent uses the steepest descent direction for the information geometry instead of taking each step with the gradient from parameter space [**?** **?** **?**]. A similar principle can be applied to optimizing the parameter of quantum variational ansatz, which is then referred to as quantum natural gradient descent. The idea of taking steps in another manifold can also be extended beyond information geometry [**?**].

Before diving into the details of quantum natural gradients, it is worth noting that the pure state quantum natural gradient descent is, in fact, equivalent to imaginary time evolution [**?**]. The imaginary time evolution is defined by

$$|\psi(\tau)\rangle = A(\tau)e^{*H\tau}\,|\psi(0)\rangle\,. \tag{33}$$

The state at $\tau \to \emptyset$ is the ground state of H. Instead of evolving the state directly, variational imaginary time evolution calculates the evolution direction with McLachlan's variational principle and simulates the evolution by variating the ansatz parameters [**?**].

The idea of natural gradient descent can be explained as follows. Consider the objective function $\mathcal{L}(\theta)$ is a likelihood function $\mathcal{L}(\theta|x)$ where x denote the measurement outcome of the distribution. For a general optimization, the likelihood function is given by the definition of the problem. While the traditional gradient descent algorithm takes one step towards the gradient direction in the parameter space, such direction may not necessarily be the direction that most significantly changes the difference of the distribution of $\mathcal{L}(\theta|x)$. From the statistical point of view, we would like each step to be taken in the direction that maximizes the distribution difference of the cost function. The natural gradient is then invented to present the gradient of the difference of the distribution function, and the corresponding geometry space is called information geometry.

The likelihood function's distribution difference from different θ can be characterized by the Kullback–Leibler (KL) divergence [**?**]. The KL divergence defines a distance measure between $\mathcal{L}(\theta|x)$, and when parameter θ is mapped to the likelihood function, it creates a Riemannian manifold with a metric F, where for small vector $d\theta$ in the parameter space, the distance ds on the manifold is

$$ds^2 = d\theta^T F d\theta. \tag{34}$$

Here F is the Fisher information metric, which is the Hessian of the KL divergence. The Riemannian manifold describes the information geometry of the problem. The natural gradient descent instead takes the gradient from the information geometry and maximizes the KL divergence for each step. The natural gradient is defined as

$$\tilde{g} = F^{*1}g = F^{*1}(\mathcal{L}(\theta). \tag{35}$$

So far our discussion about natural gradient descent are with real space. To implement natural gradient descent for variational quantum algorithms, we can use the Fubini-Study metric as the Fisher Information metric in Hilbert space [**?** **?**], which is given by [**?** **?** **?** **?**]

$$(F)_{ij} \coloneqq \mathrm{Re}(\langle\partial_i\psi|\partial_j\psi\rangle) * \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle\,, \tag{36}$$

where $|\partial_i\psi\rangle \coloneqq \frac{\partial U(\theta)}{\partial\theta_i}|\psi_0\rangle$. These quantities can be evaluated with similar techniques as the Hadamard test. More details of Hadamard test has been included in appendix **??**.

There are a few challenges for the quantum natural gradient. The first is the quantum natural gradient is defined on pure states, while in practice we need to consider the noise and non-unitary evolution. To solve this problem, the Fubini-Study metric can be generalized to a non-unitary circuit [**?**].

$$(F)_{ij} = \frac{1}{2}\,\mathrm{Tr}\big(\rho(\boldsymbol{\theta})(L_iL_j + L_jL_i)\big), \tag{37}$$

where L is the symmetric difference defined as

$$\partial_k\rho(\boldsymbol{\theta}) =: \frac{\partial\rho(\boldsymbol{\theta})}{\partial\theta_k} = \frac{1}{2}(L_k\rho(\boldsymbol{\theta}) + \rho(\boldsymbol{\theta})L_k). \tag{38}$$

Suppose the density matrix of mixed state has fidelity $f_{id} = 1 * \epsilon$, which means the eigenvalue $\lambda_1 = f_{id}$. Such density matrix is given by

$$\rho_\epsilon = (1 * \epsilon) |\psi_1\rangle \langle\psi_1| + \epsilon \sum_{k=2}^{d} \lambda_k |\psi_k\rangle \langle\psi_k| . \tag{39}$$

The Fisher information matrix on NISQ device can be approximated efficiently as

$$(F)_{ij} = 2 \operatorname{Tr}[\frac{(\partial_i \rho_\epsilon)(\partial_j \rho_\epsilon)}{f_{id}}] + \mathcal{O}(\frac{1 * f_{id}}{d}). \tag{40}$$

Term $\frac{1}{f_{id}}$ is ignored since it is only a scale factor. $\operatorname{Tr}[(\partial_i \rho_\epsilon)(\partial_j \rho_\epsilon)]$ is a Riemannian metric tensor and can be evaluated with SWAP tests.

Secondly, the quantum natural gradient can be ill-conditioned and might lead to unreasonably large updates due to very small eigenvalues of F. In practice, the learning rate for the initial steps needs to be chosen very small or one needs to use *Tikhonov* regularization to add a small constant to the diagonal of F before the inversion. Another method to get a stable gradient is by doing a "half-inversion", given by [**?**]

$$\tilde{g} = F^{*\alpha}(\mathcal{L}(\theta) \tag{41}$$

where α is considered a regularization parameter. When $\alpha = 0$, no precondition is applied to the original gradient, and when $\alpha = 1$ the formula gets back to the original natural gradient. In practice, α is often chosen to be 0.5.

The third challenge comes from the cost of constructing the Fisher information metric. To fully construct the Fisher information metric with p parameters, p^2 different expectation values needs to be estimated from the quantum computer. For a large p the cost of construct the Fisher information metric can be significant. Gacon *et al.* [**?**] proposed that quantum natural gradient can also be approximated with the SPSA methods previously discussed in Sec. **??**. Given the Fisher information metric $F(\theta_1, \theta_2) = |\langle\psi_0|U^T(\boldsymbol{\theta_1})U(\boldsymbol{\theta_2})|\psi_0\rangle|$, the estimated gradient $\tilde{g}$ is given by

$$\tilde{g}^{(k)} = \frac{\delta F}{2\epsilon^2} \frac{\Delta_1^{(k)} \Delta_2^{(k)T} + \Delta_2^{(k)} \Delta_1^{(k)T}}{2} \tag{42}$$

where

$$\begin{aligned} \delta F^{(k)} = {} & F(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)} + \epsilon\Delta_1^{(k)} + \epsilon\Delta_2^{(k)}) && (43)\\ & * F(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)} + \epsilon\Delta_1^{(k)}) && (44)\\ & * F(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)} * \epsilon\Delta_1^{(k)} + \epsilon\Delta_2^{(k)}) && (45)\\ & + F(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)} * \epsilon\Delta_1^{(k)}), && (46)\\ & && (47) \end{aligned}$$

and the exponentially smooth estimator

$$\bar{F}^{(k)} = \frac{k}{k+1} \bar{F}^{(k*1)} + \frac{1}{k+1} \tilde{F}^{(k)}. \tag{48}$$

The simulation results from [**?**] show that the stochastic approximated natural gradient does not perform as well as the original natural gradient, however, it still improves the convergence speed compared to the standard gradient descent.

1.4. Gradient-free searching strategy

The gradient-free search strategy is the other large category of optimization strategy. Because of the analytical property of the quantum ansatz, a few gradient-free searching strategies are developed for optimizing the variational ansatz.

1.4.1. Gradient-free optimizers

Nelder-Mead algorithm The Nelder-Mead algorithm [**?**] is a gradient-free heuristic search method based on a simplex. A simplex $\mathcal{S}$ in $\mathbb{R}^k$ is defined as the convex hull of $k+1$ vertices in $\mathbb{R}^k$. It can be considered a multidimensional version of triangles. For example, in $\mathbb{R}^2$ a simplex is a triangle, and in $\mathbb{R}^3$ a simplex is a tetrahedron.

The algorithm first generates a random simplex and then transforms the simplex $\mathcal{S}$ and decreases the function values at each vertex iteratively. In each iteration, the objective function value at one or more test points is measured. Then a new simplex is updated by replacing the vertices with one of the test points.

Algorithm 5: Nelder-Mead algorithm

α_r*: Reflection coefficient.*
α_e*: Expansion coefficient.*
α_c*: Contraction coefficient.*
α_s*: Shrink coefficient.*
while *Not converge* **do**
  Ordering the existing test points, so that $\mathcal{L}(\boldsymbol{\theta}_0) \leq \mathcal{L}(\boldsymbol{\theta}_1) ... \leq \mathcal{L}(\boldsymbol{\theta}_{n+1})$
  Calculate $\boldsymbol{\theta}_o$, the centroid of all points except $\boldsymbol{\theta}_{n+1}$;
  Calculate reflection point $\boldsymbol{\theta}_r = \boldsymbol{\theta}_o + \alpha_r(\boldsymbol{\theta}_o * \boldsymbol{\theta}_{n+1})$ with $\alpha_r > 0$.
  if *The reflection point the not the best estimation, but better than the second worst point* $\boldsymbol{\theta}_n$ **then**
    replace the worst point $\boldsymbol{\theta}_{n+1}$ with the reflected point $\boldsymbol{\theta}_r$;
    continue;
  end
  if *Reflected point* $\boldsymbol{\theta}_r$ *is the best point* **then**
    Calculate expansion point $\boldsymbol{\theta}_e = \boldsymbol{\theta}_o + \alpha_e(\theta_r * \theta_o)$ with $\alpha_e > 1$
    replace the worst point $\boldsymbol{\theta}_{n+1}$ with the best point from expansion point $\boldsymbol{\theta}_e$ or reflected point $\boldsymbol{\theta}_r$;
  end
  Calculate contraction point $\boldsymbol{\theta}_c = \boldsymbol{\theta}_o + \alpha_c(\boldsymbol{\theta}_{n+1} * \boldsymbol{\theta}_o)$ with $0 < \alpha_c \leq 0.5$.
  if *Contraction point* $\boldsymbol{\theta}_c$ *is better than the worst point* $\boldsymbol{\theta}_{n+1}$ **then**
    Replacing the worst point $\boldsymbol{\theta}_{n+1}$ with the contracted point $\boldsymbol{\theta}_c$;
    Continue ;
  end
  `// Shrink`
  Replace all points except the best point $\boldsymbol{\theta}_1$ with $\boldsymbol{\theta}_i = \boldsymbol{\theta}_1 + \alpha_s(\boldsymbol{\theta}_i * \boldsymbol{\theta}_1)$
end

The Nelder-Mead algorithm is gradient-free, and for each iteration, only very limited points need to be measured. Therefore it is very friendly to variational quantum algorithms. A result from [**? ?**] shows the Nelder-Mead algorithm performs well in a low-noise environment, but not as well when a noise model is included.

Powell's conjugate direction algorithm Powell's conjugate direction algorithm is a gradient-free optimization method. Powell's algorithm takes a starting point and two non-parallel vectors as inputs. The algorithm first search bidirectionally through the direction parallel to the first vector, and uses the minimal point as the direction, and search through the next direction parallel to the second vector. A conjugate direction can be defined as the displacement between the initial starting point and the optimal point of the second iteration. The algorithm repeats this process until it converges. The linear directional search can be implemented with Golden-section search or Brent's method [**?**].

Algorithm 6: Powell's algorithm

x_k: *Searched minimal point.* x_0 *is the starting point.*
$\mathcal{L}(\theta)$: *Objective function*
α_k: *Searching displacement*
$\mathbf{d}_k$: *Searching direction.* $\mathbf{d}_1$ *and* $\mathbf{d}_2$ are the starting direction.
while *Not converged* **do**
Search for α_k that minimize $\mathcal{L}(\mathbf{x}_{k^*1} + \alpha_k \mathbf{d}_k)$
Set $x_k = x_{k^*1} + \alpha_k d_k$.
Remove the initial vectors by assigning $\mathbf{d}_j = \mathbf{d}_{j+1}$ Set $\mathbf{d}_N = \mathbf{x}_N * \mathbf{x}_0$ Search for α_N that minimize $\mathcal{L}(\mathbf{x}_N + \alpha_N \mathbf{d}_N)$
Set $\mathbf{x}_0 = \mathbf{x}_0 + \alpha_N \mathbf{d}_N$
end

1.4.2. Analytical optimization

In this section, we review optimization methods that take advantage of the analytical property of the objective function landscape. Consider the ansatz circuit as a CPTP mapping as a product of individual quantum super-operators

$$\Phi(\boldsymbol{\theta}) = \Phi_k(\theta_k) \S \Phi_2(\theta_2)\Phi_1(\theta_1). \tag{49}$$

Here $\Phi_k(\theta_k)$ are k-th parameterised quantum gates where $\Phi_k(\theta_k)\rho := \boldsymbol{U}_k \rho \boldsymbol{U}_k^{\pounds}$ and $\boldsymbol{U}_k = \exp\left(*i\frac{\theta_k}{2}\boldsymbol{P}_k\right)$. Here $\boldsymbol{P}_k$ are tensor products of single-qubit Pauli operators as $\boldsymbol{P}_k$ Ë ^$\boldsymbol{I}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$‘$^{\otimes N}$. Each superoperator can be expanded around $\boldsymbol{\theta}_0$ as

$$\Phi_k(\theta_0 + \theta) = a(\theta)\Phi_{ak} + b(\theta)\Phi_{bk} + c(\theta)\Phi_{ck}, \tag{50}$$

where $a(\theta), b(\theta) = 1$, $\cos(\theta)$ and $c(\theta) = \frac{1}{2}\sin(\theta)$.

Now we only allow n parameters to be variables and fix all the others, the whole ansatz can be expanded into

$$\Phi(\boldsymbol{\theta}_0+\boldsymbol{\theta}) = \prod_{k=1}^{\nu}[a(\theta_k)\Phi_{ak} + b(\theta_k)\Phi_{bk} + c(\theta_k)\Phi_{ck}], \tag{51}$$

where $\boldsymbol{\theta}$ is the displacement around the reference point $\boldsymbol{\theta_0}$.

Sequential optimization with sinusoidal fitting (Rotosolve) A variety of quantum variational algorithms are dealing with linear objective functions. For example, the energy value for VQE is a linear objective function. For these problems, the objective function landscape of every single variable is a sinusoidal function. The full information of such function can be obtained by simply measuring three different points of the objective function. The sequential optimization method [**? ? ?**] utilizes this feature and calculates the minima of the sinusoidal function directly. By iteratively finding the minima of every single parameter while fixing the other parameters, a greedy method can be derived to efficiently optimize the parameters of the ansatz.

From Eq. (**??**) we can simplify $a(\theta), b(\theta)$ and $b(\theta)$ into a simpler form if we only allow one single parameter to change. When all the parameters are independent, the objective function can be transformed as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_0+\theta\cdot e_i) = \mathcal{L}_i(\theta)_{\boldsymbol{\theta}_0} &= \mathrm{Tr}[\boldsymbol{M}\Phi_i(\boldsymbol{\theta}_0+\theta\cdot e_i)] &(52)\\ &= \mathrm{Tr}[\boldsymbol{M}(a(\theta)\Phi_a + b(\theta)\Phi_b + c(\theta)\Phi_c)] &(53)\\ &= \boldsymbol{A}\sin(\theta+\boldsymbol{\phi}) + \boldsymbol{C}. &(54)\end{aligned}$$

The parameter can be evaluated analytically with following equation when the hermitian generator has exactly two eigenvalues.

$$\theta^{<} = \boldsymbol{\phi} * \tfrac{\pi}{2} * \arctan2\left(\boldsymbol{\phi}_1,\ \boldsymbol{\phi}_2\right) + 2\pi k, \tag{55}$$

where

$$\phi_1 = 2\mathcal{L}_i(\theta)_{\theta_0} * \mathcal{L}_i(\theta + \frac{\pi}{2})_{\theta_0} * \mathcal{L}_i(\theta * \frac{\pi}{2})_{\theta_0} \tag{56}$$

$$\phi_2 = \mathcal{L}_i(\theta + \frac{\pi}{2})_{\theta_0} * \mathcal{L}_i(\theta * \frac{\pi}{2})_{\theta_0}. \tag{57}$$

All the single qubit gates have Pauli operators as their generators, which satisfy the above equations. The cost function can also sum sinusoidal functions with different period [**?** **?**] when the generator has more than two distinct eigenvalues.

When several gates share the same parameter, the objective function can be transformed into similar form but with different oscillation period:

$$\mathcal{L}(\boldsymbol{\theta}_0 + \theta \cdot e_i) = A \sin(k\theta + \phi) + C, \tag{58}$$

where k is the number of appearances of θ_i in the ansatz. The same approach can be applied to find the analytical solution.

It is important to note that the conditions for validity of this optimization method implies that it cannot be applied to some ansätze (for instance in the case where a parameterized controlled unitary is used) In general terms, Rotosolve and its subsequent extension require that all parametrized gate subject to optimization have 2π-periodicity and full-rank for the Hermitian matrix generating the rotation. The work presented by Wierisch *et al.* [**?**], relying on alternative general parameter-shift rules, demonstrates a generalization of Rotosolve which allows for it to be used on all quantum gates with arbitrary frequencies.

Analytical Free-Axis Selection with fixed rotation angles (Fraxis) The analytical method we mentioned so far has fixed rotation axis in the ansatz, but with a flexible rotation angle. The Analytical Free-Axis Selection (Fraxis) method implemented an alternative version where angles are fixed but the axis can be flexible [**?**]. The method is shown to provide faster optimization than Rotosolve in some numerical tests. Wada *et al.* [**?**] further improved on this idea by optimizing the angle and the axis at the same time for time evolution simulations.

The Fraxis method considers each single qubit gate in the ansatz as

$$U_k(\theta_k) = e^{*i\frac{\theta_k}{2}\hat{n}_k \cdot \vec{P}} \tag{59}$$

where $U_k(\theta_k)$ is the k-th single qubit rotation, θ_k is the rotation angle, and is the $\hat{n}_k$ is the direction vector characterize the rotation direction of the hermitian generator, given by

$$\hat{n}_k \cdot \vec{P} = n_{k,x}X + n_{k,y}Y + n_{k,z}Z \text{ for } \hat{n}_k \text{ Ë } \mathcal{R}^3, |\hat{n}_k| = 1. \tag{60}$$

Here X,Y,Z are Pauli matrices.

With Lagrange multiplier method, the optimal $\hat{n}_k$ is found to satisfy following linear equations:

$$\sin^2(\frac{\theta_k}{2}\mathbf{R} * 2\lambda^{<}\hat{n}_{k<}) = *\alpha_{\theta_k}\mathbf{b} \tag{61}$$

where

$$\mathbf{b} = (tr(M[\rho, X]), tr(M[\rho, Y]), tr(M[\rho, Z]))^T \tag{62}$$

and

$$r_x \equiv \mathrm{tr}\,(MX\rho X), \tag{63}$$

$$r_y \equiv \mathrm{tr}\left(\boldsymbol{M}\boldsymbol{Y}\rho\boldsymbol{Y}\right), \tag{64}$$

$$r_z \equiv \mathrm{tr}\left(\boldsymbol{M}\boldsymbol{Z}\rho\boldsymbol{Z}\right), \tag{65}$$

$$r_{(x+y)} \equiv \mathrm{tr}\left(\boldsymbol{M}\left(\frac{X+Y}{\sqrt{2}}\right)\rho\left(\frac{X+Y}{\sqrt{2}}\right)\right), \tag{66}$$

$$r_{(x+z)} \equiv \mathrm{tr}\left(\boldsymbol{M}\left(\frac{X+Z}{\sqrt{2}}\right)\rho\left(\frac{X+Z}{\sqrt{2}}\right)\right), \tag{67}$$

$$r_{(y+z)} \equiv \mathrm{tr}\left(\boldsymbol{M}\left(\frac{Y+Z}{\sqrt{2}}\right)\rho\left(\frac{Y+Z}{\sqrt{2}}\right)\right). \tag{68}$$

$$\mathbf{R} \equiv \begin{pmatrix} 2r_x & 2r_{(x+y)} * r_x * r_y & 2r_{(x+z)} * r_x * r_z \\ 2r_{(x+y)} * r_x * r_y & 2r_y & 2r_{(y+z)} * r_y * r_z \\ 2r_{(x+z)} * r_x * r_z & 2r_{(y+z)} * r_y * r_z & 2r_z \end{pmatrix}. \tag{69}$$

The optimum value $\hat{n}_k^<$ can be solved by measuring all elements in $\mathbf{R}$ and solving the linear equation **??**. When $\theta_k = \pi$, the method can be further simplified and $\hat{n}_k^<$ becomes the eigenvector of $\mathbf{R}$.

Quantum analytical descent The full landscape of the expectation value could be too expensive to construct. However, it can be approximated. Previous works from Sung et.al [**?**] approximate the local region with a predefined model, such as a quadratic polynomial. The quantum analytical descent [**?**] approximates the landscape properly and utilizes a similar idea of sequential optimization with sinusoidal fitting. Instead of fitting the sinusoidal function for each parameter iteratively, the entire landscape of the linear objective function can be approximated efficiently with a quadratic number of parameters and quadratic number of measurements.

The expansion of the entire ansatz has 3^p terms, which cannot be efficiently evaluated. The full expansion of the ansatz can be approximated into

$$\begin{aligned}\Phi(\boldsymbol{\theta}) = A(\boldsymbol{\theta})\Phi^{(A)} &+ \sum_{k=1}^{p}[B_k(\boldsymbol{\theta})\Phi_k^{(B)} + C_k(\boldsymbol{\theta})\Phi_k^{(C)}] \\ &+ \sum_{l>k}^{p}[D_{kl}(\boldsymbol{\theta})\Phi_{kl}^{(D)}] + O(\sin^3\epsilon).\end{aligned} \tag{70}$$

Here $A, B_k, C_k, D_{kl} : \mathbb{R}^p \mapsto \mathbb{R}$ are products of simple univariate trigonometric functions. And the loss function can be constructed as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = A(\boldsymbol{\theta})\mathcal{L}^{(A)} &+ \sum_{k=1}^{p}[B_k(\boldsymbol{\theta})\mathcal{L}_k^{(B)} + C_k(\boldsymbol{\theta})\mathcal{L}_k^{(C)}] \\ &+ \sum_{l>k}^{p}[D_{kl}(\boldsymbol{\theta})\mathcal{L}_{kl}^{(D)}] + O(\sin^3\epsilon).\end{aligned} \tag{71}$$

Here $\mathcal{L}^{(A)}, \mathcal{L}_k^{(B)}, \mathcal{L}_k^{(C)}, \mathcal{L}_{kl}^{(D)}$ Ë ℝ can be calculated from the hardware by measuring expectation value at $1 + 2p^2 * 2p$ different points where p is the number of parameters.

After the objective function landscape has been approximated, the author uses natural gradient descent to find the minimum conventionally. The natural gradient descent requires the Fubini-Study metric tensor to find the natural gradient, which can be approximated as

$$[F_Q]_{pq} = F_{BB}F_{BB}(\boldsymbol{\theta}) + F_{AB}F_{AB}(\boldsymbol{\theta}) + \S\, O(\sin^2\epsilon), \tag{72}$$

where

$$F_{BB}(\boldsymbol{\theta}) := 2\frac{\partial B_p(\boldsymbol{\theta})}{\partial \theta_p}\frac{\partial B_q(\boldsymbol{\theta})}{\partial \theta_q} \tag{73}$$

$$F_{AB}(\boldsymbol{\theta}) := 2\frac{\partial B_p(\boldsymbol{\theta})}{\partial \theta_p}\frac{\partial A(\boldsymbol{\theta})}{\partial \theta_q} + 2\frac{\partial A(\boldsymbol{\theta})}{\partial \theta_p}\frac{\partial B_n(\boldsymbol{\theta})}{\partial \theta_q}. \tag{74}$$

Once the local minima of the approximated landscape have been found, the algorithm updates the best-guessed parameters in this iteration to be the local minima. Then repeat the approximation and conventional optimization steps until the algorithm converges.

Jacobi diagonalization and Anderson acceleration An analytical method inspired by Jacobi diagonalization and Anderson acceleration has been introduced in Ref. [**?**]. This method is an improved version of the sequential sinusoidal fitting. Instead of fitting the landscape one dimension at a time, the landscape can be accurately reconstructed by only allowing a small subset of parameters to vary. This technique is similar to the Jacobi diagonalization algorithm for large matrices by iteratively optimizing a random subset of parameters [**?**]. The Anderson/Pulay DIIS sequence acceleration is then introduced to produce a better estimation for each iteration.

This algorithm first constructs the analytical landscape with a few parameters and then iteratively chooses a random subset of parameters to fit the analytical landscape for the optimal value. This iteration approach is similar to the Jacobian diagonalization method.

Parrish *et al.* [**?**] then introduce the DIIS (Direct inversion of the iterative subspace) method to better estimate and improve the convergence speed. Suppose the optimized parameter after i-th iteration is θ^i, the error of the optimized parameter is ϵ^i, which is the difference between the optimal value and the optimized parameter θ^i. The DIIS algorithm gives a better estimation:

$$\theta^{i\ddot{}} = \sum_i c_i \theta^i, \tag{75}$$

where c_i is a real coefficient that minimizes the square of the 2-norm of ϵ,

$$O(c_i) = \sum_{ij} c_i c_j \epsilon^i \cdot \epsilon^j, \tag{76}$$

and subject to the normalization condition

$$\sum_i c_i = 1. \tag{77}$$

The DIIS algorithm utilizes the previous steps' historical value and extrapolates a better estimation for the next steps. Since the error ϵ^i is not accessible for each step, it can be approximated by

$$\epsilon^i \grave{u}\ \delta\theta^i = \theta^i * \theta^{i*1} \quad \text{(Anderson style)}, \tag{78}$$

or

$$\epsilon^i \grave{u}\ g(\theta^i) \quad \text{(Pulay style)}, \tag{79}$$

where $g(\theta^i)$ is the gradient of the cost function at θ^i. Here the Anderson style and Pulay style are different approaches to approximate the error value.

1.5. Engineering cost function

Collective optimization [**?**] In practice, when using VQE to solve eigenenergies of molecules, there are different Hamiltonians with varied bond lengths that can be studied. Since the two different solutions should be close when the Hamiltonian is only different in a small amount, all of the optimal solutions of the series of Hamiltonian should also be chained together in the parameter space, forming a snake-like shape. The entire

optimization process can be considered optimizing the arrangement of the whole snake instead of optimizing points separately.

The collective optimization algorithm redefines the cost function into

$$\mathcal{L}(\boldsymbol{\theta}(\lambda)) = \int_{\lambda_0}^{\lambda_1} [L(\boldsymbol{\theta}(\lambda)) + E(\boldsymbol{\theta}(\lambda))], \tag{80}$$

where $\boldsymbol{\theta}(\lambda)$ is the optimized ansatz parameter for Hamiltonian of bond distance λ, the $\boldsymbol{E}$ term is the energy of the molecules, while the $\boldsymbol{L}$ term defines the internal energy of the snake.

$$\boldsymbol{L}(\boldsymbol{\theta}(\lambda)) = \alpha|\frac{\partial\boldsymbol{\theta}(\lambda)}{\partial\lambda}|^2 + \beta|\frac{\partial^2\boldsymbol{\theta}(\lambda)}{\partial^2\lambda}|^2. \tag{81}$$

The first term refers to the energy that depends on the snake's length, and the second term refers to the curvature of the snake. α and β are meta parameters.

In practice the snake is discretized into a sequence of parameters at different bond distance $r_i = (\theta_i(\lambda_1), \theta_i(\lambda_2), \S, \theta_i(\lambda_K))$ where i is the i-th component for each $\theta_i(\lambda_k)$. Then the r_i can be solved iteratively by

$$r_i^t = (\eta\mathbf{A} + \mathbb{1})^{*1}(r_i^{t^{*1}} * \eta\frac{dE(r^{t^{*1}})}{dr_i}), \tag{82}$$

where η is the learning rate, $E(r) = \sum_{k=1}^{K} E(\theta(\lambda_j))$ and $\boldsymbol{A}$ is a pentadiagonal banded matrix with $A_{i^{*2},i} = A_{i,i^{*2}} = \beta$, $A_{i^{*1},i} = A_{i,i^{*1}} = {}^{*}\alpha * 4\beta$, $A_{i,i} = 2\alpha + 6\beta$.

The simulation result shows the collective optimization methods help to pull the parameters from local minimums.

Conditional Value-at-Risk as objective function Conditional Value-at-Risk (CVaR) is a measure that takes into account only the tail of the probability distribution. The CVaR of a random variable $\boldsymbol{X}$ for a confidence level α Ë $(0, 1]$ is defined as

$$\mathrm{CVaR}_\alpha(X) = E[X|X \leq F_X^{*1}(\alpha)], \tag{83}$$

where $\boldsymbol{F_X}$ is the cumulative density function of X. To illustrate the idea, consider the random variable $\boldsymbol{X}$ has been sampled $\boldsymbol{N}$ times. The CVaR with confidence α can be calculated by selecting $\alpha\boldsymbol{N}$ samples with the lowest value and evaluating the average. When $\alpha = 1$, CVaR equals the expectation value. For variational quantum algorithms, consider the result of each sampling to be a random variable. Then the CVaR can be used as an objective function and replace the expectation value [**?**].

$$\mathcal{L}(\theta) = \mathrm{CVaR}_\alpha(X(\theta)), \tag{84}$$

where $\boldsymbol{X}$ is the random variable for each sample result.

The CVaR could give a reasonable benefit. Suppose $|\psi_0\rangle$ is the ground state, and $|\psi_1\rangle$,$|\psi_2\rangle$,$|\psi_3\rangle$ are first, second and third excited state. Define $|\psi_A\rangle = (|\psi_0\rangle + |\psi_3\rangle)\sqrt{2}$ and $|\psi_B\rangle = (|\psi_1\rangle + |\psi_2\rangle)\sqrt{2}$. Suppose the energy level are equally separated, then $\langle\psi_A|\,\boldsymbol{H}\,|\psi_A\rangle = \langle\psi_B|\,\boldsymbol{H}\,|\psi_B\rangle$, therefore the optimizer will encounter a gradient plateau. However for our purpose, we would like to obtain the ground state, therefore $|\boldsymbol{A}\rangle$ is better than $|\boldsymbol{B}\rangle$ in practice. The CVaR could help with this problem by emphasizing the distribution $|\boldsymbol{A}\rangle$. The CVaR emphasizes the best-observed samples and leads to a smooth objective function without introducing local minimum [**?**]. Also, the implementation of CVaR is relatively straightforward. Since CVaR throws away some of the samples, the accuracy of the estimation decreases. In order to get the same accuracy, the sampling number needs to be increased. The same amount of samples should be involved in the calculation.

Symmetry preserving cost function adjustments: One means to maintain electron number conservation in the output wavefunction of VQE is to impose a constraint on the cost function [**? ?**]. Namely, one can

include a penalty term corresponding to violation of symmetries. The VQE cost function can therefore be re-written as

$$E(\theta,\mu) = \langle\psi(\theta)|\,\hat{H}\,|\psi(\theta)\rangle + \sum_i \mu\left(\langle\psi(\theta)|\,\hat{O}_i\,|\psi(\theta)\rangle * O_i\right)^2, \tag{85}$$

where $\hat{O}_i$ represents the symmetry operators that can be independently measured, for instance, the electron number operator, or the spin operator (square of total spin). O_i is the target expectation value for each of these operators, and μ is a Lagrangian parameter to determine the strength of the constraints. This method, referred to as Constrained VQE in Ref. [**?**], was shown to also eliminate 'kinks' appearing in the VQE implementation that had been shown to appear in Ref. [**?**]. It is worth pointing out (as done in Refs. [**? ?**]) that the Pauli strings used for operators representing the electron number or the total spin already need to be computed for the Hamiltonian thereby the method does not require additional Quantum costs with respect to the computation of the energy function. One caveat to this is that, as is the case for any constrained optimization problem, the optimization landscape becomes more complex with the addition of constraints. This, in addition to the management of the hyper-parameter μ, could result in additional optimization costs and risks of local minima. The use of this method, and further considerations regarding optimal application have been discussed in Ref. [**?**].

1.6. Discussion

In this Section, we have reviewed some of the latest optimization strategies adapted to optimize variational quantum ansätze. Some methods are adapted from the traditional numerical optimization, while some new methods are developed to provide some essential features specifically for variational quantum ansatz optimization.

The first key feature to consider is the speedup of the convergence for variational quantum ansatz. From the convergence speed per iteration, the analytical method is much faster than the gradient descent strategies. However, the analytical methods would require taking more measurement points to finish a single iteration, which is unfair to directly compare the convergence speed per iteration between optimization strategies. Numerical studies have nonetheless shown in multiple occasions that Rotosolve (and by association its extensions) indeed reach convergence significantly faster than other methods [**? ? ?**].

Several studies have been performed to compare the relative strength of convergence of different optimizers. Mihlikov *et al.* [**?**] show as part of a case study on Hydrogen that SPSA presents a clear advantage on the Nelder-Mead and the Powell optimizers. Bonet-Monroig *et al.* [**?**] test four different optimizers on a variety of small molecular systems and find that SPSA performs slightly better than all others. Beyond these studies, several methods allow for improved convergence on almost any optimizer: the covariance functions between the Hamiltonian and operator can also be used to increase the convergence speed when the optimization is almost converged [**?**]. Stochastic methods are also suggested as add-ons to reduce the impact of hardware noise, accelerating convergence speed [**? ? ?**].

Another key feature is the resilience of the barren plateau problem. The natural-gradient-based strategy is considered resilient to barren plateau, and its absolute value of gradient has a lower bound $1/(2^{2N+1})$ [**? ?**]. The analytical method is gradient-free and directly jumps into an optimal or approximately optimal position in each iteration, which can prevent entering a barren plateau through optimization dependent on a learning rate [**? ? ? ?**]. However, this is at the cost of increasing the measurement points and conventional computation.

From the discussion above, no optimization strategy outperforms other strategies in every aspect; the convergence needs to trade with the implementation complexity and measurement points. In fact, multiple strategies can be applied together to improve the overall performance of the optimizer. In practice, the sequential analytical fitting method has been mainly used when a non-traditional optimizer optimizes the variational ansatz. Quantum natural gradient-based methods are more popular in simulation-based studies.

A final comment is worth raising on the optimization of VQE ansätze: it does not need to be done entirely using samples from observable obtained on the quantum computer. In particular Okada *et al.* [**?**] have shown that one can use efficient circuit simulation to optimize local order parameters and subsequently use quantum devices post-optimization to measure the global quantities.

Table 1
Comparison of optimization strategies mentioned in this section. C_M denotes the number of different measurement expectation values per iteration that need to be evaluated from the quantum computer. C_C denotes the complexity of the classical algorithm for each iteration. Since the gradient can be evaluated or approximated with different methods, one can use $g^{(1)}$ to denote the cost of evaluating first order gradients and $g^{(2)}$ to denote the cost of evaluation second order gradients. S denote the required sample shot number. k denote the number of Hamiltonians with different bond distance being optimized simultaneously. p denote the number of parameters in the ansatz.

Strategies	Type	Meta parameters	C_M	C_C	References
Simple gradient descent	First order	η:Learning rate.	$Sg^{(1)}$	$\mathcal{O}(p)$	[? ?]
RMSProp	First order	γ: Moving average parameter. η: Learning rate.	$Sg^{(1)}$	$\mathcal{O}(p)$	[?]
Adam	First order	β_1, β_2: Moving average parameters. η: Learning rate.	$Sg^{(1)}$	$\mathcal{O}(p)$	[?]
BFGS	Second order	η:Learning rate.	$S(g^{(1)} + g^{(2)})$	$\mathcal{O}(p^3)$	[? ? ?]
Quantum natural gradient decent	Second order	η:Learning rate.	$S(g^{(1)} + g^{(2)})$	$\mathcal{O}(p^3)$	[? ? ?]
Nelder-Mead	Gradient free	α_r: Reflection coefficient. α_e: Expansion coefficient. α_c: Contraction coefficient. α_s: Shrink coefficient.	S	$\mathcal{O}(p)$	[? ?]
Powell	Gradient free	α_k: Searching displacement. $\mathbf{d}_k$:Searching direction.	Depends on linear search method.	$\mathcal{O}(1)$	[?]
Rotosolve	Gradient free	None	$\mathcal{O}(3n)$	$\mathcal{O}(1)$	[? ?]
Fraxis	Gradient free	None	$\mathcal{O}(6n)$	$\mathcal{O}(1)$	[? ?]
Quantum analytical descent	Gradient free	η:Learning rate.	$S(1 + 2p^2 * 2p)$	NP	[?]
Anderson acceleration	Addon	None	No extra costs	$\mathcal{O}(p)$	[?]
Collective optimization	Addon	α: Coefficient for snake length β:Coefficient for snake curvature.	No extra costs	$\mathcal{O}(k)$	[?]
CVaR	Addon	α:Confident level	Multiply factor $1/\alpha$	$\mathcal{O}(1)$	[?]

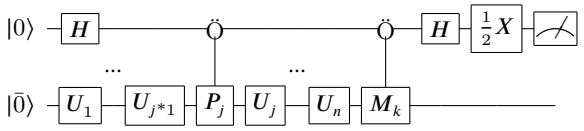

Figure 1: Quantum circuit that evaluates $\mathrm{Im}(\langle\phi_0|V_k^{j\text{£}}(\boldsymbol{\theta})\hat{M}_k U(\boldsymbol{\theta})|\phi_0\rangle)$.

[grouping-prefix]symboldescription

[g]$\hat{H}$Hamiltonian of the quantum mechanical system. [g]$\hat{P}_a$Any Pauli operator or tensor of Pauli operators (Pauli string) [c]nNumber of basis functions used for in the Hamiltonian. It can also refer to, the number of fermionic modes, or the number of site on a lattice [c]mThe number of electrons in the system of interest [c]NRefers to the number of qubits used to model the wavefunction of the system of interest [c]ϵA given level of precision, i.e. the maximum error in estimation. A variant of this symbol, ε, is used to represent the level of noise in the quantum computer. [l]$\mathcal{O}(f(n))$Order of function $f(n)$. [g]$\mathbb{1}$The identity matrix of an appropriate dimension. [g]IThe 2 by 2 identity matrix. [g]$X/Y/Z$The three 2 by 2 Pauli X, Pauli-Y and Pauli-Z matrices. [g]$\mathcal{P}$The number of Pauli strings in a Hamiltonian. [g]$\hat{P}$Any Pauli string. [l]$\phi_i(.)$, $|\phi_i\rangle$A spin-orbital function [l]$\psi(.)$, $|\psi\rangle$An electronic wavefunction