ALGORITHMS FOR NOISY QUANTUM COMPUTERS AND TECHNIQUES FOR ERROR MITIGATION

By

Ryan LaRose

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computational Mathematics, Science and Engineering – Doctor of Philosophy
Physics – Dual Major

2022

ABSTRACT

ALGORITHMS FOR NOISY QUANTUM COMPUTERS AND
TECHNIQUES FOR ERROR MITIGATION

By

Ryan LaRose

Quantum computation will likely provide significant advantages relative to classical architectures for certain computational problems in number theory and physics, and potentially in other areas such as optimization and machine learning. While some key theoretical and engineering problems remain to be solved, experimental advances in recent years have demonstrated the first beyond-classical quantum computation as well as the first experiments in error-corrected quantum computation. In this thesis, we focus on quantum computers with around one hundred qubits that can implement around one thousand operations, the so-called noisy-intermediate scale quantum (NISQ) regime or kilo-scale quantum (KSQ) regime, and develop algorithms tailored to these devices as well as techniques for error mitigation that require significantly less overhead than fault-tolerant quantum computation. In the first part, we develop quantum algorithms for diagonalizing quantum states (density matrices) and compiling quantum circuits. These algorithms use a quantum computer to evaluate a cost function which is classically hard to compute and a classical computer to adjust parameters of an ansatz circuit, similar to the variational principle in quantum mechanics and other variational quantum algorithms for chemistry and optimization. In the second part, we extend an error mitigation technique known as zero-noise extrapolation and introduce a new framework for error mitigation which we call logical shadow tomography. In particular, we adapt zero-noise extrapolation (ZNE) to the gate model and introduce new methods for noise scaling and (adaptive) extrapolation. Further, we analyze ZNE in the presence of time-correlated noise and experimentally show ZNE increases the effective quantum volume of various quantum computers. Finally, we develop a simple framework for error mitigation that enables (the composition of) several error mitigation techniques with significantly fewer resources than prior methods, and numerically show the advantages of our framework.

To my wonderful, inspiring, encouraging K-12 teachers.
John Barber Jr, Laurie Blom, Jan Boswell, Don Dziuk, Todd Green,
David Kirsten, Rachel Kleinke, Kim Latona, Jessica Norton, Catherine Nutter,
Bradley Porter, Philip Ricci, Pamela Ruggiero, David Stumpf, Janet Nellis-Trubiano,
Mark Van Hecke, Paula VanHeusden, Nate Williams

ACKNOWLEDGEMENTS

This thesis is the result of working with many collaborators at many institutions. It would not be possible without the support of my advisor Matthew Hirn to whom I owe the most thanks. At MSU I'd like to thank Dean Lee, Morten Hjorth-Jensen, Johannes Pollanen, Huey-Wen Lin, Alexei Bazavov, Justin Lane, Ben Hall, Jacob Watkins, Joe Kitzman, Niyaz Beysengulov, and Camille Mikolas for scientific discussions and help with organizing various quantum computing conferences, courses, seminars, and events. At LANL I'd like to thank Patrick Coles, Lukasz Cincio, Andrew Sornberger, and Yigit Subasi for collaborations, and everyone in the inaugural LANL quantum computing summer school for a fun summer of research and friendship. At IBM I'd like to thank Jennifer Glick, Antonio Mezzacapo, Travis Scholten, and Sam Slezak. At NASA I'd like to thank Eleanor Rieffel, Davide Venturelli, Zhihui Wong, Hong-Ye Hu, and the entire NASA QuAIL team for collaboration throughout my PhD. Although my work at Alphabet X is outside the scope of this thesis, I'd like to thank Guifre Vidal for the scientific training and many fun discussions. At Unitary Fund I'd like to thank Will Zeng for helping me multiple times at various stages throughout my PhD as well as the entire Unitary Fund technical team, especially Andrea Mari for many discussions. At Google Quantum AI I'd like to thank Alan Ho as well as all of the people I got the chance to work with including Matthew Harrigan, Wojtek Mruczkiewicz, Nick Rubin, Zhang Jiang, Jarrod McClean, Tanuj Khattar, Orion Martin, Doug Strain, Pedram Rousham, and Xiao Mi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

PRELIMINARIES

1.1 Notation

$\|\cdot\rangle$	A column vector labeled by $\cdot$
$\dagger$	Conjugate transpose operator (as superscript)
$\langle\cdot\|$	The row vector $\|\cdot\rangle^\dagger$
$\otimes$	Tensor product
$\oplus$	Addition modulo 2
$\{0,1\}^n$	Length n bitstrings. E.g., $\{0,1\}^2$ consists of 00, 01, 10, and 11
$\mathcal{U}(d^n)$	The unitary group of dimension d^n

Table 1.1: Notation.

This thesis follows the Feynman-Twain principle in that no attempt is made at mathematical rigor and persons attempting to find mathematical rigor will be shot. We are most interested in the Hilbert space

$$\underbrace{\mathbb{C}^d \otimes \cdots \otimes \mathbb{C}^d}_{n \text{ terms}} \cong \mathbb{C}^{d^n} \tag{1.1}$$

(where $d = 2$ almost always in this thesis), i.e. the space of n quantum bits (*qubits*). Common bases for a single qubit include the *computational* (standard) basis

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.2}$$

and the *Hadamard basis* $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$. For n qubits and a bitstring $z = z_1 \cdots z_n \in \{0,1\}^n$ we write basis elements as $|z_1\rangle \otimes \cdots \otimes |z_n\rangle$ or simply $|z_1 \cdots z_n\rangle \equiv |z\rangle$ for short, so a general state may be written

$$|\psi\rangle = \sum_{z \in \{0,1\}^n} \alpha_z |z\rangle \tag{1.3}$$

with $\alpha_z \in \mathbb{C}$ and $\sum_z |\alpha_z|^2 = 1$.

I	The identity gate/element $I\|z\rangle = \|z\rangle$
$X \equiv \sigma_x$	Pauli X. $X\|z\rangle = \|z \oplus 1\rangle$
$Z \equiv \sigma_z$	Pauli Z. $Z\|z\rangle = (-1)^z\|z\rangle$
$Y \equiv \sigma_y$	Pauli Y. $Y = iXZ$.
H	The Hadamard gate $H\|z\rangle = (\|0\rangle + (-1)^z\|1\rangle)/\sqrt{2}$
CNOT	CNOT$\|a\rangle\|b\rangle = \|a\rangle\|a \oplus b\rangle$
CZ	CZ$\|a\rangle\|b\rangle = (-1)^{ab}\|a\rangle\|b\rangle$

Table 1.2: Some common single- and two-qubit gates. Here, $a, b, z \in \{0, 1\}$.

Quantum operations (*gates*) are elements in $\mathcal{U}(2^n)$. Some common single-qubit and two-qubit gates are defined in Table 1.2. A *quantum circuit* is a series of operations acting on an initial state with one or more terminal measurements. An example is shown below.

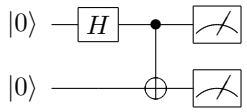


Figure 1.1: An example quantum circuit.

We read this circuit left-to-right as follows:

$$\begin{aligned} |0\rangle \otimes |0\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \\ &\xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \end{aligned}$$

This is the final quantum state before measurement. Born's rule tells us that we measure 00 with probability $1/2$ and 11 with probability $1/2$.

1.2 Quantum algorithms

A *quantum algorithm* is a quantum circuit (potentially with classical pre- and/or post-processing) for performing some computational task. As an example, consider the following computational task: Given a single-bit function $f : \{0, 1\} \to \{0, 1\}$ and the ability to query the function, determine if $f(0) = f(1)$. There are four such functions, two of which satisfy $f(0) = f(1)$ and two of which do not. Since $f(0)$ and $f(1)$ are independent, the best classical algorithm takes at least two queries. Interestingly, a quantum algorithm exists which takes only one query. This algorithm

tells us whether $f(0) = f(1)$ but does not tell us the value of either $f(0)$ or $f(1)$. The circuit for performing this algorithm is shown in Fig. 1.2.

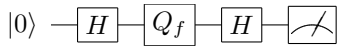


Figure 1.2: Quantum circuit for the Deutsch-Jozsa algorithm.

Here, the operation Q_f is a *phase query* $Q_f|z\rangle = (-1)^z|z\rangle$. One can show the final state of this circuit before the measurement is

$$|\psi\rangle = \left[(-1)^{f(0)} + (-1)^{f(1)}\right]|0\rangle + \left[(-1)^{f(0)} - (-1)^{f(1)}\right]|1\rangle. \tag{1.4}$$

Thus, we measure 0 with probability 1 if $f(0) = f(1)$, otherwise we measure 1. Notice how constructive and destructive interference are used in preparing the solution. While this problem is artificial, the same underlying principle is used for more realistic algorithms, e.g. Shor's algorithm [10] and others, which have rigorous performance and scaling guarantees but prohibitively large overhead for (near-term) implementations. In Chapter 2 and Chapter 3, we develop quantum algorithms with lower overhead which are still, in a well-defined manner, classically hard, but have heuristic elements and less general performance guarantees.

1.3 Open quantum systems

A *closed quantum system* — one which is completely isolated from its environment — is primarily a convenient mathematical abstraction. An *open quantum system* — one which interacts with its environment — more accurately describes a quantum computer.

A noisy quantum state is described by an ensemble $\{p_i, |\psi_i\rangle\}_i$ where p_i forms a probability distribution and each $|\psi_i\rangle$ is a wavefunction. Physically, this means we do not know with certainty which wavefunction $|\psi_i\rangle$ we possess. Mathematically, we work with the *density operator (matrix)* of an ensemble

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \tag{1.5}$$

a Hermitian, positive semi-definite operator with unit trace which generalizes classical probability distributions (diagonal ρ).

Letting ρ denote the quantum state of interest and ρ_{env} the environment, noise can be characterized physically by the process

$$\rho \mapsto \text{Tr}_{\text{env}} \left[U \left(\rho \otimes \rho_{\text{env}} \right) U^\dagger \right]$$

where U is a unitary on the composite Hilbert space and Tr_{env} denotes the partial trace over the environment. (Given $\rho_{AB} \in \mathcal{H}_A \otimes \mathcal{H}_B$, the partial trace over $\mathcal{H}_B$ is defined by

$$\text{Tr}_A \rho_{AB} := \sum_{j=1}^{\dim(\mathcal{H}_B)} \left(I \otimes \langle j | \right) \rho_{AB} \left(I \otimes |j\rangle \right) \tag{1.6}$$

where $|j\rangle$ form a basis for $\mathcal{H}_B$. Similarly for the partial trace over $\mathcal{H}_A$.) This can be written [11] in the equivalent, often more convenient, operator-sum representation

$$\rho \mapsto \sum_{k=1}^{K} E_k \rho E_k^\dagger \tag{1.7}$$

where the Kraus operators E_k satisfy the completeness relation

$$\sum_{k=1}^{K} E_k^\dagger E_k = I.$$

Equation (1.7) is known as a *quantum operation* or *quantum channel*. Physically, it can be interpreted as randomly replacing the state ρ by the (properly normalized) state $E_k \rho E_k^\dagger$ with probability $\text{Tr}[E_k \rho E_k^\dagger]$. Mathematically, it is a completely positive, trace preserving map. *Coherent errors* are noisy channels defined by $K = 1$ unitary Kraus operators whereas *incoherent errors* are defined by $K > 1$ Kraus operators.

We often model noise in devices with channels used in theoretical work. One commonly used noise model is the Pauli channel.

Definition 1. The Pauli channel maps a single qubit state ρ to $\mathcal{E}_{\mathbf{p}}^{\text{P}}(\rho)$ defined by

$$\mathcal{E}_{\mathbf{p}}^{\text{P}}(\rho) := p_I \rho + p_X X \rho X + p_Y Y \rho Y + p_Z Z \rho Z \tag{1.8}$$

where $p_I + p_X + p_Y + p_Z = 1$.

While the Pauli channel acts on a single qubit, it can be generalized to a d-dimensional Hilbert space via the *Weyl channel*

$$\mathcal{E}_p^{\mathrm{W}}(\rho) := \sum_{k,l=0}^{d-1} p_{kl} W_{kl} \rho W_{kl}^{\dagger} \tag{1.9}$$

where p_{kl} are probabilities and the Weyl operators are

$$W_{kl} := \sum_{m=0}^{d-1} e^{2\pi i m k/d} |m\rangle\langle m+1|.$$

For $d = 2$, Eqn. (1.9) reduces to Eqn. (1.8).

Two special cases of the Pauli channel are the bit-flip and phase-flip (dephasing) channel.

Definition 2. The bit-flip channel maps a single qubit state ρ to $\mathcal{E}_p^{\mathrm{BF}}(\rho)$ defined by

$$\mathcal{E}_p^{\mathrm{BF}}(\rho) := (1-p)\rho + pX\rho X \tag{1.10}$$

where $0 \leq p \leq 1$.

While a bit-flip channel flips the computational basis state with probability p, the phase-flip channel introduces a relative phase with probability p.

Definition 3. The phase-flip (dephasing) channel maps a single qubit state ρ to $\mathcal{E}_p^{\mathrm{deph}}(\rho)$ defined by

$$\mathcal{E}_p^{\mathrm{deph}}(\rho) := (1-p)\rho + pZ\rho Z \tag{1.11}$$

where $0 \leq p \leq 1$.

Another special case of the Pauli channel is the depolarizing channel which occurs when each Pauli is equiprobable $p_X = p_Y = p_Z = p$ and $p_I = 1 - 3p$. This channel can be equivalently thought of as replacing the state ρ by the maximally mixed state $I/2$ with probability p.

Definition 4. The depolarizing channel maps a single qubit state ρ to $\mathcal{E}_p^{\mathrm{depo}}(\rho)$ defined by

$$\mathcal{E}_p^{\mathrm{depo}}(\rho) := (1-p)\rho + pI/2 \tag{1.12}$$

where $0 \leq p \leq 1$.

The $d = 2^n$-dimensional generalization of Def. 4 is straightforward:

Definition 5. The global depolarizing channel maps an n-qubit state ρ to $\mathcal{E}_p^{\mathrm{GD}}(\rho)$ defined by

$$\mathcal{E}_p^{\mathrm{GD}}(\rho) := (1-p)\rho + pI/d \tag{1.13}$$

where $0 \leq p \leq 1$, $d = 2^n$, and $I \equiv I_d$ is the d-dimensional identity.

We use this general description of noisy quantum systems, as well as the particular channels we have defined, both for analyzing algorithms in the presence of noise in Chapter 2 and Chapter 3 and for developing general techniques for error mitigation in Chapter 4 and Chapter 5.

1.4 Quantum error correction

Because real quantum computers are open quantum systems, it is rather unlikely we will be able to run circuits at the scale needed for, say, Shor's algorithm without some solution for dealing with errors. The primary long-term solution is error correction and fault-tolerance. Some of our ideas for error mitigation in this thesis stem from error correction, so we briefly review this now.

Suppose a state $|\psi\rangle := \alpha|0\rangle + \beta|1\rangle$ incurs a phase error $E|\psi\rangle = \alpha|0\rangle + e^{i\delta}\beta|1\rangle$. In principle, $\delta \in \mathbb{R}$ could in principle be infinitesimal, in which case the task may appear hopeless from the start. We can expand this error in the Pauli basis

$$E = e_0 I + e_1 X + e_2 Y + e_3 Z \tag{1.14}$$

to get a finite set of terms, but each $e_i \in \mathbb{R}$ could still in principle be infinitesimal. The almost magical trick is that performing a measurement $\{M_i\}$ on $E|\psi\rangle$ returns $M_i E|\psi\rangle/\sqrt{p_i}$ with probability p_i, i.e., *some* term $\eta_i\sigma_i|\psi\rangle$ where $\eta_i \in \mathbb{R}$ and $\sigma_i \in \{I, X, Y, Z\}$. The σ_i can be removed by applying σ_i, and although we still have a potentially infinitesimal η_i, it is now a global phase and has no influence on measurement statistics. In other words, we can say that *causing* an error to occur is a crucial step of quantum error correction.

In a bit more detail, a *stabilizer quantum error correction code* (*stabilizer code*) is specified by any subgroup $\mathcal{G}$ with $-I \notin \mathcal{G}$ of the n-qubit Pauli group

$$\mathcal{P}_n := \{p\sigma_1 \otimes \cdots \otimes \sigma_n : p \in \{\pm 1, \pm i\}, \sigma_i \in \{I, X, Y, Z\rangle\}. \tag{1.15}$$

The group $\mathcal{G}$ is called the *gauge group*. The center of $\mathcal{G}$ in $\mathcal{P}_n$ is called the *stabilizer group* $\mathcal{S} := Z(\mathcal{G}) \cap \mathcal{G}$. Note that, by construction, $\mathcal{S}$ is abelian and does not contain $-I$. We desire these conditions to define codewords from $\mathcal{S} := \langle S_1, ..., S_r \rangle$. A *codeword* is a state $|\psi\rangle$ such that

$$S|\psi\rangle = |\psi\rangle \quad \forall\, S \in \{S_1, ..., S_r\}. \tag{1.16}$$

The *codespace* is the span of codewords. It's easy to show that the codespace is trivial if $\mathcal{S}$ is not abelian or $-I \in \mathcal{S}$. If $\mathcal{S} = \mathcal{G}$, the code is called a *subspace code*, otherwise the code is called a *subsystem code*.

As an example, the three-qubit repetition code is a subspace code specified by $\mathcal{S} = \langle Z_1Z_2, Z_2Z_3 \rangle$. One can verify that $|000\rangle =: |\bar{0}\rangle$ and $|111\rangle =: |\bar{1}\rangle$ are codewords. This thus defines a two-dimensional codespace

$$\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle = \alpha|000\rangle + \beta|111\rangle \tag{1.17}$$

which we identify as a *logical qubit*. The word *logical* is used to distinguish from *physical* qubits: this logical qubit (1.17) is formed by defining a two-dimensional subspace of $\mathbb{C}^{2^3}$ which we formed out of three physical qubits. We use notation $[[n, k]]$ to describe a code with n physical qubits encoding k logical qubits. The relationship to the number of stabilizer generators r for such a code is $r = n - k$.

In the repetition code example, one can check that the operator Z_1 satisfies $Z_1|\bar{0}\rangle = |\bar{0}\rangle$ and $Z_1|\bar{1}\rangle = -|\bar{1}\rangle$. It thus behaves as the Pauli Z operator on the logical qubit — i.e., the *logical operator* $\bar{Z}$. Similarly, one can check that $XXX|\bar{0}\rangle = |\bar{1}\rangle$ and $XXX|\bar{1}\rangle = |\bar{0}\rangle$. It thus behaves as the Pauli X operator on the logical qubit — i.e., the *logical operator* $\bar{X}$. In general, for an $[[n, k]]$ code it is always possible to find logical operators $\bar{Z}_1, ..., \bar{Z}_k, \bar{X}_1, ..., \bar{X}_k$ with the expected commutation relations. Specifically, logical operations are elements of $\mathcal{L} := N(\mathcal{S}) - \mathcal{S}$ where N denotes the normalizer. Logical operations are not necessarily unique: e.g., for the three qubit repetition code, Z_2, Z_3, and $Z_1Z_2Z_3$ also behave as $\bar{Z}$. Note that, in this example, if *any* single-qubit phase flip error occurs, $|\bar{0}\rangle$ gets mapped to $|\bar{1}\rangle$. It would be better if this took many single-qubit errors since, in reasonable noise models, many single-qubit errors occurring is much less likely than any one of

the errors occurring. This property of "how far" codewords are from each other is referred to as the *distance* of the code and can be formulated as

$$d := \min_{L \in \mathcal{L}} w(L) \tag{1.18}$$

where w is the *weight* (number of non-identity terms) of the Pauli L. A code with distance d can correct errors on up to $t = (d-1)/2$ qubits. We often augment the $[[n, k]]$ notation with the distance d as $[[n, k, d]]$. We can thus describe the three-qubit repetition code as a $[[3, 1, 1]]$ code.

A correctable error commutes with all logical operators and all but one stabilizer generator. Specifically, correctable errors are elements of the abelian group $T := N(\mathcal{L}) - \mathcal{S}$. Because of the commutation relations, measuring each stabilizer generator reveals whether the error commutes or anti-commutes with each stabilizer generator. This information, called a *syndrome*, can be used to infer which error occurred. To see this, consider again the three-qubit repetition code example. As $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$, the product ZZ can be written $ZZ = |00\rangle\langle 00| + |11\rangle\langle 11| - (|01\rangle\langle 01| + |10\rangle\langle 10|)$. In other words, states in which the two bits agree are in the $+1$ eigenspace and states in which the two bits disagree are in the -1 eigenspace. So, measuring the stabilizer generator $Z_1 Z_2$ tells us if the first two bits agree or disagree. Similarly for measuring the other stabilizer generator $Z_2 Z_3$. If the error X_1 occurs, we would measure the syndrome $[-1, 1]$ as X_1 anticommutes with $Z_1 Z_2$ and commutes with $Z_2 Z_3$. Of course in practice we only get the syndrome and have to infer which error occurred — this process is known as *decoding*.

The general pattern of error correction is to encode the state with an $[[n, k, d]]$ code, measure stabilizers to obtain a syndrome, decode the syndrome to infer which error occurred, then correct the error. We typically assume we can do some of these operations perfectly — e.g., we prepare states and measure stabilizers perfectly, and errors only occur elsewhere during the computation. This is of course not realistic in practice but serves as the first step towards the theory of fault-tolerant quantum computation in which all elements (state preparation, measurement, etc.) are treated as noisy or unreliable. This background is sufficient for the purposes of this thesis, however. We use the ideas of error correction for the purpose of error mitigation in Chapter 5.

1.5 The Gottesman-Knill theorem

The problem of simulating quantum circuits with n qubits and depth d is important for verifying the output of quantum computers as well as ultimately understanding why, and in what sense, quantum computers are more powerful than classical computers. Correspondingly, many methods have been developed to classically simulate quantum systems. We use the term *quantum simulator* or just *simulator* to denote a classical algorithm which inputs a quantum circuit and outputs a quantity of interest. To truly mimic a quantum computer, this "quantity of interest" should only be a set of bitstrings $z \in \{0,1\}^n$ as this is the return type of a real experimental (qubit) quantum information processing system. However, simulators work by manipulating some classical representation of quantum information, so it is generally possible to return additional values, for example a classical representation of the wavefunction, a reduced density matrix on one or more qubits, a single amplitude of the wavefunction, or an expectation value of a given observable.

The Gottesman-Knill theorem presents an algorithm for efficiently simulating a certain class of quantum circuits which, remarkably, contains circuits with very large numbers of qubits, very large depth, *and* large entanglement. This class of circuits is known as *Clifford circuits*, the defining characteristic being the types of gates (*Clifford gates*) appearing in the circuit. For general circuits, the resources of this simulation strategy grow exponentially in the number of non-Clifford gates.

The Gottesman-Knill theorem [12] (or algorithm / simulator) works by updating operators instead of updating the state in the same spirit as the Heisenberg picture vs. the Schrödinger picture. (It was originally presented this way [13], though the terminology is no longer as standard.) In the Schrödinger picture, we think of operators being fixed and the state evolving over time. Applying an operator U to a state $V|\psi\rangle$, we say that the new state is $UV|\psi\rangle$. However, we may equivalently write this as $(UVU^\dagger)U|\psi\rangle$ and say that

$$V \mapsto UVU^\dagger. \tag{1.19}$$

See Table 1.3 for a summary.

If we keep track of (1.19) for a basis $\{P_1, ..., P_k\}$, then we are able to reconstruct the evolution

	Schrödinger picture	Heisenberg picture
V	—	$UVU^\dagger$
$\|\psi\rangle$	$UV\|\psi\rangle$	$U\|\psi\rangle$

Table 1.3: How each term in the quantity $V|\psi\rangle$ is updated after application of U in the Schrödinger vs. Heisenberg picture. The answer is always $UV|\psi\rangle$ (the product of each column).

of any operator $V = \sum_i \alpha_i P_i$ since

$$UVU^\dagger = \sum_i \alpha_i (UP_iU^\dagger) \tag{1.20}$$

by linearity. Furthermore, the map (1.19) is a group homomorphism since

$$VW \mapsto UVWU^\dagger = UVU^\dagger UWU^\dagger. \tag{1.21}$$

Therefore it suffices to track the evolution of a generating set. If we take the Pauli basis as our basis, then a convenient generating set is $\{X_1, ..., X_n, Z_1, ..., Z_n\}$. For a general operator of $\mathcal{U}(2^n)$, we thus need to keep track of only $2n$ single qubit operators.

So far this presentation is completely general with respect to what the operators (gates) are. For arbitrary operators, keeping track of how the generating set transforms will grow exponentially. However, if we only allow operators which preserve Paulis under conjugation, the size of the description does not grow. This class of operators is precisely the normalizer of the Pauli group $\mathcal{P}_n$ in $\mathcal{U}(2^n)$, also called the Clifford group, and is denoted $N(\mathcal{P}_n)$ or C.

The Clifford group is generated by $\{H, S, \text{CNOT}_{ij}\}$ between arbitrary pairs of qubits $i, j \in [n]$. One can verify for the single-qubit gates that

$$HXH^\dagger = Z \qquad HZH^\dagger = X \tag{1.22}$$

$$SXS^\dagger = Y \qquad SZS^\dagger = Z \tag{1.23}$$

and for the two-qubit CNOT that

$$\text{CNOT}(XI)\text{CNOT}^\dagger = XX \tag{1.24}$$

$$\text{CNOT}(IX)\text{CNOT}^\dagger = IX \tag{1.25}$$

$$\text{CNOT}(ZI)\text{CNOT}^\dagger = ZI \tag{1.26}$$

$$\text{CNOT}(IZ)\text{CNOT}^\dagger = ZZ \tag{1.27}$$

Assuming without loss of generality a Clifford circuit is compiled into this gateset, the Gottesman-Knill algorithm works by iterating through the circuit and updating the generating set at each step. As is typical with $\mathcal{P}_n$, in software one represents elements using symplectic notation and updates the so-called *tableau* of the generators. For clarity we proceed by example with the two-qubit circuit in Fig. 1.1 that performs $|00\rangle \mapsto (|00\rangle + |11\rangle)/\sqrt{2}$ and show how the tableau is updated:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{H_0} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{CNOT}_{01}} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{1.28}$$

In other words, $X_0 \mapsto Z_0$, $X_1 \mapsto X_1$, $Z_0 \mapsto X_0X_1$, and $Z_1 \mapsto Z_0Z_1$. From the final stabilizer tableau, one can sample bitstrings or compute expectation values using algorithms described in [12]. For our purposes, we are primarily concerned with the ability to efficiently store and manipulate stabilizer states with classical resources as described above, a task which will be crucial for our error mitigation strategy in Chapter 5.

1.6 Outline of thesis

The remainder of this thesis is split into two parts, with this first part developing algorithms for quantum computers in the "NISQ" (noisy intermediate-scale quantum) [14] or "KSQ" (kilo-scale quantum, $\sim 10^3$ qubits $\times$ operations) regime, and the second part developing error mitigation techniques for such computers. Both parts are complementary towards the goal of useful quantum computing. In the first part, we develop algorithms for diagonalizing quantum states (density matrices) in Chapter 2 and for compiling quantum circuits in Chapter 3. In the second part, we analyze and extend an existing error mitigation technique, zero-noise extrapolation, in Chapter 4, and develop a new resource-efficient procedure for implementing the composition of several error mitigation techniques in Chapter 5.

The following paragraphs are a short preface to Part 1 (chapters two and three). It would be useful to format this as a part, but I'm not allowed to have parts in my thesis because someone in the graduate school gets paid to look at the format of theses and tell you that you can't have colored hyperlinks, you have to have a table of algorithms with the word "Algorithm" before each entry, and you can't have parts unless they are formatted like chapters and simultaneously do and do not appear in the table of contents (like chapters). So consider the following paragraphs to be a preface to chapters two and three which one may logically format as a higher-level abstraction than a chapter if one had the ability to do so. I would offer a link to a usefully formatted thesis on my website or something but I'm sure my friend in the graduate school would inform me that's a violation of university policy and I can no longer graduate. (And that the link can't be colored.) I'd usually include this as a footnote but I'm sure if I did I'd have to include a list of footnotes where each entry has to start with the word "Footnote" and the list of footnotes has to appear in the table of contents (where the parts should and shouldn't be, formatted like chapters). So I'm writing this as plain text, not sectioned or chaptered or otherwise numbered, and absolutely not in a part. As a result it makes nearly no sense, in accordance with university policy. Will my friend in the graduate school still notice, and subsequently examine every part of my thesis with a microscope to ensure I don't graduate? I'm almost certain the state-of-the-art PDF diff tool MSU bought from the RAND Corporation in 1852 will pick this up. But I leave it to fate whether this is received with a smile or a frown in the hope that someone reading this document out of interest will be better oriented by this remark. A similar remark will appear before Part 2 (chapters four and five).

The future applications of quantum computers, assuming that large-scale, fault-tolerant versions will eventually be realized, are manifold. From a mathematical perspective, applications include number theory [15], linear algebra [16, 17, 18], differential equations [19, 20], and optimization [21]. From a physical perspective, applications include electronic structure determination [22, 23] for molecules and materials and real-time simulation of quantum dynamical processes [24] such as protein folding and photo-excitation events. Naturally, some of these applications are more long-term than others. Factoring and solving linear systems of equations are typically viewed as

longer term applications due to their high resource requirements. On the other hand, approximate optimization and the determination of electronic structure may be nearer term applications, and could even serve as demonstrations of quantum supremacy in the near future [25, 26].

A major aspect of quantum algorithms research is to make applications of interest more near term by reducing quantum resource requirements including qubit count, circuit depth, numbers of gates, and numbers of measurements. A powerful strategy for this purpose is algorithm hybridization, where a fully quantum algorithm is turned into a hybrid quantum-classical algorithm [27]. The benefit of hybridization is two-fold, both reducing the resources (hence allowing implementation on smaller hardware) as well as increasing accuracy (by outsourcing calculations to "error-free" classical computers).

Variational hybrid algorithms are a class of quantum-classical algorithms that involve minimizing a cost function that depends on the parameters of a quantum gate sequence. Cost evaluation occurs on the quantum computer, with speedup over classical cost evaluation, and the classical computer uses this cost information to adjust the parameters of the gate sequence. Variational hybrid algorithms have been proposed for Hamiltonian ground state and excited state preparation [22, 28, 29], approximate optimization [21], error correction [30], quantum data compression [31, 32], and quantum simulation [33, 34]. A key feature of such algorithms is their near-term relevance, since only the subroutine of cost evaluation occurs on the quantum computer, while the optimization procedure is entirely classical, and hence standard classical optimization tools can be employed.

CHAPTER 2

VARIATIONAL QUANTUM STATE DIAGONALIZATION

2.1 Introduction

In this chapter, we consider the application of diagonalizing quantum states. In condensed matter physics, diagonalizing states is useful for identifying properties of topological quantum phases—a field known as entanglement spectroscopy [35]. In data science and machine learning, diagonalizing the covariance matrix (which could be encoded in a quantum state [36, 16]) is frequently employed for principal component analysis (PCA). PCA identifies features that capture the largest variance in one's data and hence allows for dimensionality reduction [37].

Classical methods for diagonalization typically scale polynomially in the matrix dimension [38]. Similarly, the number of measurements required for quantum state tomography—a general method for fully characterizing a quantum state—scales polynomially in the dimension. Interestingly, Lloyd et al. proposed a quantum algorithm for diagonalizing quantum states that can potentially perform exponentially faster than these methods [16]. Namely, their algorithm, called quantum principal component analysis (qPCA), gives an exponential speedup for low-rank matrices. qPCA employs quantum phase estimation combined with density matrix exponentiation. These subroutines require a significant number of qubits and gates, making qPCA difficult to implement in the near term, despite its long-term promise.

Here, we propose a variational hybrid algorithm for quantum state diagonalization. For a given state ρ, our algorithm is composed of three steps: (i) Train the parameters $\vec{\alpha}$ of a gate sequence $\mathcal{U}$ such that $\tilde{\rho} = U_p(\vec{\alpha}_{\text{opt}})\rho U_p(\vec{\alpha}_{\text{opt}})^\dagger$ is approximately diagonal, where $\vec{\alpha}_{\text{opt}}$ is the optimal value of $\vec{\alpha}$ obtained (ii) Read out the largest eigenvalues of ρ by measuring in the eigenbasis (i.e., by measuring $\tilde{\rho}$ in the standard basis), and (iii) Prepare the eigenvectors associated with the largest eigenvalues. We call this the variational quantum state diagonalization (VQSD) algorithm. VQSD is a near-term algorithm with the same practical benefits as other variational hybrid algorithms.

Employing a layered ansatz for $U_p(\vec{\alpha})$ (where p is the number of layers) allows one to obtain a hierarchy of approximations for the eigevalues and eigenvectors. We therefore think of VQSD as an approximate diagonalization algorithm.

We carefully choose our cost function C to have the following properties: (i) C is faithful (i.e, it vanishes if and only if $\tilde{\rho}$ is diagonal), (ii) C is efficiently computable on a quantum computer, (iii) C has operational meanings such that it upper bounds the eigenvalue and eigenvector error (see Sec. 2.2.1), and (iv) C scales well for training purposes in the sense that its gradient does not vanish exponentially in the number of qubits. The precise definition of C is given in Sec. 2.2.1 and involves a difference of purities for different states. To compute C, we introduce novel short-depth quantum circuits that likely have applications outside the context of VQSD.

To illustrate our method, we implement VQSD on Rigetti's 8-qubit quantum computer. We successfully diagonalize one-qubit pure states using this quantum computer. To highlight future applications (when larger quantum computers are made available), we implement VQSD on a simulator to perform entanglement spectroscopy on the ground state of the one-dimensional (1D) Heisenberg model composed of 12 spins.

Our paper is organized as follows. Section 2.2 outlines the VQSD algorithm and presents its implementation. In Sec. 2.3, we give a comparison to the qPCA algorithm, and we elaborate on future applications. Section 2.4 presents our methods for quantifying diagonalization and for optimizing our cost function.

2.2 Results

2.2.1 The VQSD Algorithm

2.2.1.1 Overall structure

Figure 2.1 shows the structure of the VQSD algorithm. The goal of VQSD is to take, as its input, an n-qubit density matrix ρ given as a quantum state and then output approximations of the m-largest eigenvalues and their associated eigenvectors. Here, m will typically be much less than 2^n, the

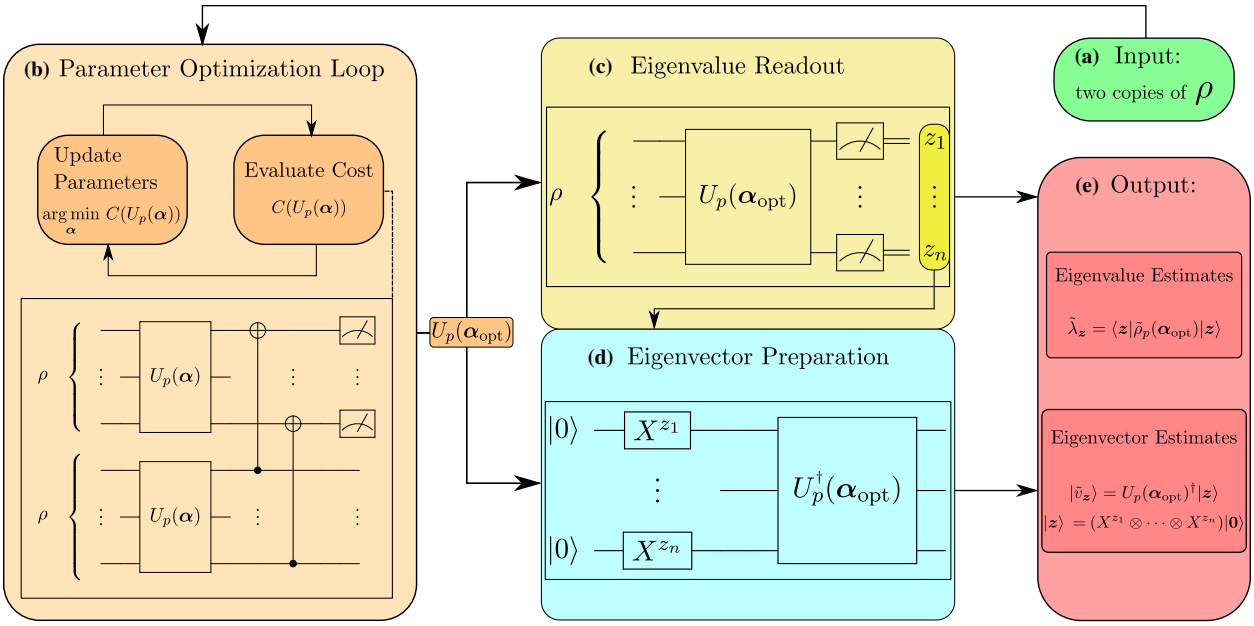


Figure 2.1: Schematic diagram showing the steps of the VQSD algorithm. (a) Two copies of quantum state ρ are provided as an input. These states are sent to the parameter optimization loop (b) where a hybrid quantum-classical variational algorithm approximates the diagonalizing unitary $U_p(\vec{\alpha}_{\text{opt}})$. Here, p is a hyperparameter that dictates the quality of solution found. This optimal unitary is sent to the eigenvalue readout circuit (c) to obtain bitstrings $\vec{z}$, the frequencies of which provide estimates of the eigenvalues of ρ. Along with the optimal unitary $U_p(\vec{\alpha}_{\text{opt}})$, these bitstrings are sent to the eigenvector preparation circuit (c) to prepare the eigenstates of ρ on a quantum computer. Both the eigenvalues and eigenvectors are the outputs (d) of the VQSD algorithm.

matrix dimension of ρ, although the user is free to increase m with increased algorithmic complexity (discussed below). The outputted eigenvalues will be in classical form, i.e., will be stored on a classical computer. In contrast, the outputted eigenvectors will be in quantum form, i.e., will be prepared on a quantum computer. This is necessary because the eigenvectors would have 2^n entries if they were stored on a classical computer, which is intractable for large n. Nevertheless, one can characterize important aspects of these eigenvectors with a polynomial number of measurements on the quantum computer.

Similar to classical eigensolvers, the VQSD algorithm is an approximate or iterative diagonalization algorithm. Classical eigenvalue algorithms are necessarily iterative, not exact [39]. Iterative algorithms are useful in that they allow for a trade-off between run-time and accuracy. Higher degrees of accuracy can be achieved at the cost of more iterations (equivalently, longer run-time), or

short run-time can be achieved at the cost of lower accuracy. This flexibility is desirable in that it allows the user of the algorithm to dictate the quality of the solutions found.

The iterative feature of VQSD arises via a layered ansatz for the diagonalizing unitary. This idea similarly appears in other variational hybrid algorithms, such as the Quantum Approximate Optimization Algorithm [21]. Specifically, VQSD diagonalizes ρ by variationally updating a parameterized unitary $U_p(\vec{\alpha})$ such that

$$\tilde{\rho}_p(\vec{\alpha}) := U_p(\vec{\alpha})\rho U_p^\dagger(\vec{\alpha}) \tag{2.1}$$

is (approximately) diagonal at the optimal value $\vec{\alpha}_{\text{opt}}$. (For brevity we often write $\tilde{\rho}$ for $\tilde{\rho}_p(\vec{\alpha})$.) We assume a layered ansatz of the form

$$U_p(\vec{\alpha}) = L_1(\vec{\alpha}_1)L_2(\vec{\alpha}_2)\cdots L_p(\vec{\alpha}_p)\,. \tag{2.2}$$

Here, p is a hyperparameter that sets the number of layers $L_i(\vec{\alpha}_i)$, and each $\vec{\alpha}_i$ is a set of optimization parameters that corresponds to internal gate angles within the layer. The parameter $\vec{\alpha}$ in (2.1) refers to the collection of all $\vec{\alpha}_i$ for $i = 1, ..., p$. Once the optimization procedure is finished and returns the optimal parameters $\vec{\alpha}_{\text{opt}}$, one can then run a particular quantum circuit (shown in Fig. 2.1(c) and discussed below) N_{readout} times to approximately determine the eigenvalues of ρ. The precision (i.e, the number of significant digits) of each eigenvalue increases with N_{readout} and with the eigenvalue's magnitude. Hence for small N_{readout} only the largest eigenvalues of ρ will be precisely characterized, so there is a connection between N_{readout} and how many eigenvalues, m, are determined. The hyperparameter p is a refinement parameter, meaning that the accuracy of the eigensystem (eigenvalues and eigenvectors) typically increases as p increases. We formalize this argument as follows.

Let C denote our cost function, defined below in (2.10), which we are trying to minimize. In general, the cost C will be non-increasing (i.e., will either decrease or stay constant) in p. One can ensure that this is true by taking the optimal parameters learned for p layers as the starting point for the optimization of $p + 1$ layers and by setting $\vec{\alpha}_{p+1}$ such that $L_{p+1}(\vec{\alpha}_{p+1})$ is an identity. This

strategy also avoids barren plateaus [40, 41] and helps to mitigate the problem of local minima, as we discuss in Section 2.8.

Next, we argue that C is closely connected to the accuracy of the eigensystem. Specifically, it gives an upper bound on the eigensystem error. Hence, one obtains an increasingly tighter upper bound on the eigensystem error as C decreases (equivalently, as p increases). To quantify eigenvalue error, we define

$$\Delta_\lambda := \sum_{i=1}^{d} (\lambda_i - \tilde{\lambda}_i)^2 \,, \tag{2.3}$$

where $d = 2^n$, and $\{\lambda_i\}$ and $\{\tilde{\lambda}_i\}$ are the true and inferred eigenvalues, respectively. Here, i is an index that orders the eigenvalues in decreasing order, i.e., $\lambda_i \geq \lambda_{i+1}$ and $\tilde{\lambda}_i \geq \tilde{\lambda}_{i+1}$ for all $i \in \{1, ..., d-1\}$. To quantify eigenvector error, we define

$$\Delta_v := \sum_{i=1}^{d} \langle \delta_i | \delta_i \rangle \,, \quad \text{with } |\delta_i\rangle = \rho|\tilde{v}_i\rangle - \tilde{\lambda}_i|\tilde{v}_i\rangle = \Pi_i^\perp \rho |\tilde{v}_i\rangle \,. \tag{2.4}$$

Here, $|\tilde{v}_i\rangle$ is the inferred eigenvector associated with $\tilde{\lambda}_i$, and $\Pi_i^\perp = I - |\tilde{v}_i\rangle\langle\tilde{v}_i|$ is the projector onto the subspace orthogonal to $|\tilde{v}_i\rangle$. Hence, $|\delta_i\rangle$ is a vector whose norm quantifies the component of $\rho|\tilde{v}_i\rangle$ that is orthogonal to $|\tilde{v}_i\rangle$, or in other words, how far $|\tilde{v}_i\rangle$ is from being an eigenvector of ρ.

As proven in Sec. 2.4.1, our cost function upper bounds the eigenvalue and eigenvector error up to a proportionality factor β,

$$\Delta_\lambda \leq \beta C \,, \quad \text{and} \quad \Delta_v \leq \beta C \,. \tag{2.5}$$

Because C is non-increasing in p, the upper bound in (2.5) is non-increasing in p and goes to zero if C goes to zero.

We remark that Δ_v can be interpreted as a weighted eigenvector error, where eigenvectors with larger eigenvalues are weighted more heavily in the sum. This is a useful feature since it implies that lowering the cost C will force the eigenvectors with the largest eigenvalues to be highly accurate. In many applications, such eigenvectors are precisely the ones of interest. (See Sec. 2.2.2.2 for an illustration of this feature.)

The various steps in the VQSD algorithm are shown schematically in Fig. 2.1. There are essentially three main steps: (1) an optimization loop that minimizes the cost C via back-and-forth

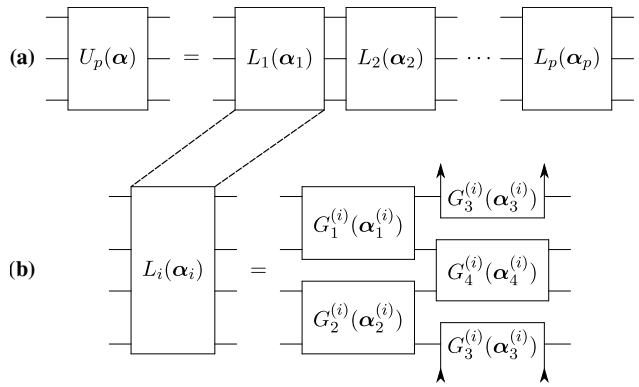


Figure 2.2: **(a)** Layered ansatz for the diagonalizing unitary $U_p(\vec{\alpha})$. Each layer L_i, $i = 1, ..., p$, consists of a set of optimization parameters $\vec{\alpha}_i$. **(b)** The two-qubit gate ansatz for the ith layer, shown on four qubits. Here we impose periodic boundary conditions on the top/bottom edge of the circuit so that G_3 wraps around from top to bottom. Section 2.7 discusses an alternative approach to the construction of $U_p(\vec{\alpha})$, in which the ansatz is modified during the optimization process.

communication between a classical and quantum computer, where the former adjusts $\vec{\alpha}$ and the latter computes C for $U_p(\vec{\alpha})$, (2) a readout procedure for approximations of the m largest eigenvalues, which involves running a quantum circuit and then classically analyzing the statistics, and (3) a preparation procedure to prepare approximations of the eigenvectors associated with the m largest eigenvalues. In the following subsections, we elaborate on each of these procedures.

2.2.1.2 Parameter optimization loop

Naturally, there are many ways to parameterize $U_p(\vec{\alpha})$. Ideally one would like the number of parameters to grow at most polynomially in both n and p. Figure 2.2 presents an example ansatz that satisfies this condition. Each layer L_i is broken down into layers of two-body gates that can be performed in parallel. These two-body gates can be further broken down into parameterized one-body gates, for example, with the construction in Ref. [5]. We discuss a different approach to parameterize $U_p(\vec{\alpha})$ in Section 2.7.

For a given ansatz, such as the one in Fig. 2.2, parameter optimization involves evaluating the cost C on a quantum computer for an initial choice of parameters and then modifying the parameters

on a classical computer in an iterative feedback loop. The goal is to find

$$\vec{\alpha}_{\text{opt}} := \arg\min_{\vec{\alpha}} C(U_p(\vec{\alpha}))\,. \tag{2.6}$$

The classical optimization routine used for updating the parameters can involve either gradient-free or gradient-based methods. In Sec. 2.4.2, we explore this further and discuss our optimization methods.

In Eq. (2.6), $C(U_p(\vec{\alpha}))$ quantifies how far the state $\tilde{\rho}_p(\vec{\alpha})$ is from being diagonal. There are many ways to define such a cost function, and in fact there is an entire field of research on *coherence measures* that has introduced various such quantities [42]. We aim for a cost that is efficiently computable with a quantum-classical system, and hence we consider a cost that can be expressed in terms of purities. (It is well known that a quantum computer can find the purity $\text{Tr}(\sigma^2)$ of an n-qubit state σ with complexity scaling only linearly in n, an exponential speedup over classical computation [43, 44].) Two such cost functions, whose individual merits we discuss in Sec. 2.4.1, are

$$C_1(U_p(\vec{\alpha})) = \text{Tr}(\rho^2) - \text{Tr}(\mathcal{Z}(\tilde{\rho})^2)\,, \tag{2.7}$$

$$C_2(U_p(\vec{\alpha})) = \text{Tr}(\rho^2) - \frac{1}{n}\sum_{j=1}^{n} \text{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)\,. \tag{2.8}$$

Here, $\mathcal{Z}$ and $\mathcal{Z}_j$ are quantum channels that dephase (i.e., destroy the off-diagonal elements) in the global standard basis and in the local standard basis on qubit j, respectively. Importantly, the two functions vanish under the same conditions:

$$C_1(U_p(\vec{\alpha})) = 0 \iff C_2(U_p(\vec{\alpha})) = 0 \iff \tilde{\rho} = \mathcal{Z}(\tilde{\rho})\,. \tag{2.9}$$

So the global minima of C_1 and C_2 coincide and correspond precisely to unitaries $U_p(\vec{\alpha})$ that diagonalize ρ (i.e., unitaries such that $\tilde{\rho}$ is diagonal).

As elaborated in Sec. 2.4.1, C_1 has operational meanings: it bounds our eigenvalue error, $C_1 \geq \Delta_\lambda$, and it is equivalent to our eigenvector error, $C_1 = \Delta_v$. However, its landscape tends to be insensitive to changes in $U_p(\vec{\alpha})$ for large n. In contrast, we are not aware of a direct operational

meaning for C_2, aside from its bound on C_1 given by $C_2 \geq (1/n)C_1$. However, the landscape for C_2 is more sensitive to changes in $U_p(\vec{\alpha})$, making it useful for training $U_p(\vec{\alpha})$ when n is large. Due to these contrasting merits of C_1 and C_2, we define our overall cost function C as a weighted average of these two functions

$$C(U_p(\vec{\alpha})) = qC_1(U_p(\vec{\alpha})) + (1-q)C_2(U_p(\vec{\alpha}))\,, \tag{2.10}$$

where $q \in [0,1]$ is a free parameter that allows one to tailor the VQSD method to the scale of one's problem. For small n, one can set $q \approx 1$ since the landscape for C_1 is not too flat for small n, and, as noted above, C_1 is an operationally relevant quantity. For large n, one can set q to be small since the landscape for C_2 will provide the gradient needed to train $U_p(\vec{\alpha})$. The overall cost maintains the operational meaning in (2.5) with

$$\beta = n/(1+q(n-1))\,. \tag{2.11}$$

Section 2.9 illustrates the advantages of training with different values of q.

Computing C amounts to evaluating the purities of various quantum states on a quantum computer and then doing some simple classical post-processing that scales linearly in n. This can be seen from Eqns. (2.7) and (2.8). The first term, $\mathrm{Tr}(\rho^2)$, in C_1 and C_2 is independent of $U_p(\vec{\alpha})$. Hence, $\mathrm{Tr}(\rho^2)$ can be evaluated outside of the optimization loop in Fig. 2.1 using the Destructive Swap Test (see Sec. 2.4.1 for the circuit diagram). Inside the loop, we only need to compute $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ and $\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ for all j. Each of these terms are computed by first preparing two copies of $\tilde{\rho}$ and then implementing quantum circuits whose depths are constant in n. For example, the circuit for computing $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ is shown in Fig. 2.1(b), and surprisingly it has a depth of only one gate. We call it the Diagonalized Inner Product (DIP) Test. The circuit for computing $\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ is similar, and we call it the Partially Diagonalized Inner Product (PDIP) Test. We elaborate on both of these circuits in Sec. 2.4.1.

2.2.1.3 Eigenvalue readout

After finding the optimal diagonalizing unitary $U_p(\vec{\alpha}_{\text{opt}})$, one can use it to readout approximations of the eigenvalues of ρ. Figure 2.1(c) shows the circuit for this readout. One prepares a single copy of ρ and then acts with $U_p(\vec{\alpha}_{\text{opt}})$ to prepare $\tilde{\rho}_p(\vec{\alpha}_{\text{opt}})$. Measuring in the standard basis $\{|\vec{z}\rangle\}$, where $\vec{z} = z_1 z_2 ... z_n$ is a bitstring of length n, gives a set of probabilities $\{\tilde{\lambda}_{\vec{z}}\}$ with

$$\tilde{\lambda}_{\vec{z}} = \langle \vec{z} | \tilde{\rho}_p(\vec{\alpha}_{\text{opt}}) | \vec{z} \rangle \,. \tag{2.12}$$

We take the $\tilde{\lambda}_{\vec{z}}$ as the inferred eigenvalues of ρ. We emphasize that the $\tilde{\lambda}_{\vec{z}}$ are the diagonal elements, not the eigenvalues, of $\tilde{\rho}_p(\vec{\alpha}_{\text{opt}})$.

Each run of the circuit in Fig. 2.1(c) generates a bitstring $\vec{z}$ corresponding to the measurement outcomes. If one obtains $\vec{z}$ with frequency $f_{\vec{z}}$ for N_{readout} total runs, then

$$\tilde{\lambda}^{\text{est}}_{\vec{z}} = f_{\vec{z}} / N_{\text{readout}} \tag{2.13}$$

gives an estimate for $\tilde{\lambda}_{\vec{z}}$. The statistical deviation of $\tilde{\lambda}^{\text{est}}_{\vec{z}}$ from $\tilde{\lambda}_{\vec{z}}$ goes with $1/\sqrt{N_{\text{readout}}}$. The relative error $\epsilon_{\vec{z}}$ (i.e., the ratio of the statistical error on $\tilde{\lambda}^{\text{est}}_{\vec{z}}$ to the value of $\tilde{\lambda}^{\text{est}}_{\vec{z}}$) then goes as

$$\epsilon_{\vec{z}} = \frac{1}{\sqrt{N_{\text{readout}}} \tilde{\lambda}^{\text{est}}_{\vec{z}}} = \frac{\sqrt{N_{\text{readout}}}}{f_{\vec{z}}} \,. \tag{2.14}$$

This implies that events $\vec{z}$ with higher frequency $f_{\vec{z}}$ have lower relative error. In other words, the larger the inferred eigenvalue $\tilde{\lambda}_{\vec{z}}$, the lower the relative error, and hence the more precisely it is determined from the experiment. When running VQSD, one can pre-decide on the desired values of N_{readout} and a threshold for the relative error, denoted ϵ_{max}. This error threshold ϵ_{max} will then determine m, i.e., how many of the largest eigenvalues that get precisely characterized. So $m = m(N_{\text{readout}}, \epsilon_{\text{max}}, \{\tilde{\lambda}_{\vec{z}}\})$ is a function of N_{readout}, ϵ_{max}, and the set of inferred eigenvalues $\{\tilde{\lambda}_{\vec{z}}\}$. Precisely, we take $m = |\vec{\tilde{\lambda}}^{\text{est}}|$ as the cardinality of the following set:

$$\vec{\tilde{\lambda}}^{\text{est}} = \{\tilde{\lambda}^{\text{est}}_{\vec{z}} : \epsilon_{\vec{z}} \leq \epsilon_{\text{max}}\} \,, \tag{2.15}$$

which is the set of inferred eigenvalues that were estimated with the desired precision.

2.2.1.4 Eigenvector preparation

The final step of VQSD is to prepare the eigenvectors associated with the m-largest eigenvalues, i.e., the eigenvalues in the set in Eq. (2.15). Let $\vec{Z} = \{\vec{z} : \tilde{\lambda}^{\text{est}}_{\vec{z}} \in \vec{\tilde{\lambda}}^{\text{est}}\}$ be the set of bitstrings $\vec{z}$ associated with the eigenvalues in $\vec{\tilde{\lambda}}^{\text{est}}$. (Note that these bitstrings are obtained directly from the measurement outcomes of the circuit in Fig. 2.1(c), i.e., the outcomes become the bitstring $\vec{z}$.) For each $\vec{z} \in \vec{Z}$, one can prepare the following state, which we take as the inferred eigenvector associated with our estimate of the inferred eigenvalue $\tilde{\lambda}^{\text{est}}_{\vec{z}}$,

$$|\tilde{v}_{\vec{z}}\rangle = U_p(\vec{\alpha}_{\text{opt}})^\dagger |\vec{z}\rangle \tag{2.16}$$

$$= U_p(\vec{\alpha}_{\text{opt}})^\dagger (X^{z_1} \otimes \cdots \otimes X^{z_n})|\vec{0}\rangle \,. \tag{2.17}$$

The circuit for preparing this state is shown in Fig. 2.1(d). As noted in (2.17), one first prepares $|\vec{z}\rangle$ by acting with X operators raised to the appropriate powers, and then one acts with $U_p(\vec{\alpha}_{\text{opt}})^\dagger$ to rotate from the standard basis to the inferred eigenbasis.

Once they are prepared on the quantum computer, each inferred eigenvector $|\tilde{v}_{\vec{z}}\rangle$ can be characterized by measuring expectation values of interest. That is, important physical features such as energy or entanglement (e.g., entanglement witnesses) are associated with some Hermitian observable M, and one can evaluate the expectation value $\langle \tilde{v}_{\vec{z}}|M|\tilde{v}_{\vec{z}}\rangle$ to learn about these features.

2.2.2 Implementations

Here we present our implementations of VQSD, first for a one-qubit state on a cloud quantum computer to show that it is amenable to currently available hardware. Then, to illustrate the scaling to larger, more interesting problems, we implement VQSD on a simulator for the 12-spin ground state of the Heisenberg model. See Sections 2.6 and 2.7 for further details. The code used to generate some of the examples presented here can be accessed from [45].

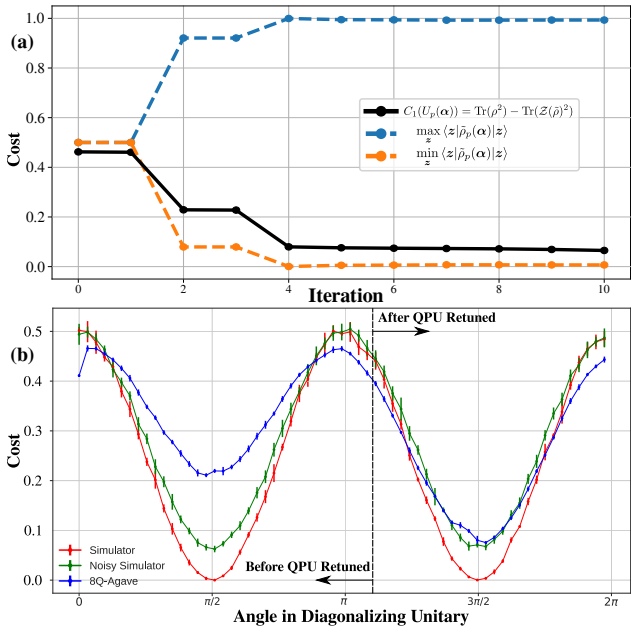


Figure 2.3: The VQSD algorithm run on Rigetti's 8Q-Agave quantum computer for $\rho = |+\rangle\langle+|$. (a) A representative run of the parameter optimization loop, using the Powell optimization algorithm (see Sec. 2.4.2 for details and Section 2.6 for data from additional runs). Cost versus iteration is shown by the black solid line. The dotted lines show the two inferred eigenvalues. After four iterations, the inferred eigenvalues approach $\{0, 1\}$, as required for a pure state. (b) The cost landscape on a noiseless simulator, Rigetti's noisy simulator, and Rigetti's quantum computer. Error bars show the standard deviation (due to finite sampling) of multiple runs. The local minima occur roughly at the theoretically predicted values of $\pi/2$ and $3\pi/2$. During data collection for this plot, the 8Q-Agave quantum computer retuned, after which its cost landscape closely matched that of the noisy simulator.

2.2.2.1 One-qubit state

We now discuss the results of applying VQSD to the one-qubit plus state $\rho = |+\rangle\langle+|$ on the 8Q-Agave quantum computer provided by Rigetti [46]. Because the problem size is small ($n = 1$), we set $q = 1$ in the cost function (2.10). Since ρ is a pure state, the cost function is

$$C(U_p(\vec{\alpha})) = C_1(U_p(\vec{\alpha})) = 1 - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2). \tag{2.18}$$

For $U_p(\vec{\alpha})$, we take $p = 1$, for which the layered ansatz becomes an arbitrary single qubit rotation.

The results of VQSD for this state are shown in Fig. 2.3. In Fig. 2.3(a), the solid curve shows the cost versus the number of iterations in the parameter optimization loop, and the dashed curves show the inferred eigenvalues of ρ at each iteration. Here we used the Powell optimization algorithm,

see Section 2.4.2 for more details. As can be seen, the cost decreases to a small value near zero and the eigenvalue estimates simultaneously converge to the correct values of zero and one. Hence, VQSD successfully diagonalized this state.

Figure 2.3(b) shows the landscape of the optimization problem on Rigetti's 8Q-Agave quantum computer, Rigetti's noisy simulator, and a noiseless simulator. Here, we varied the angle α in the diagonalizing unitary $U(\alpha) = R_x(\pi/2)R_z(\alpha)$ and computed the cost at each value of this angle. The landscape on the quantum computer has local minima near the optimal angles $\alpha = \pi/2, 3\pi/2$ but the cost is not zero. This explains why we obtain the correct eigenvalues even though the cost is nonzero in Fig. 2.3(a). The nonzero cost can be due to a combination of decoherence, gate infidelity, and measurement error. As shown in Fig. 2.3(b), the 8Q-Agave quantum computer retuned during our data collection, and after this retuning, the landscape of the quantum computer matched that of the noisy simulator significantly better.

2.2.2.2 Heisenberg model ground state

While current noise levels of quantum hardware limit our implementations of VQSD to small problem sizes, we can explore larger problem sizes on a simulator. An important application of VQSD is to study the entanglement in condensed matter systems, and we highlight this application in the following example.

Let us consider the ground state of the 1D Heisenberg model, the Hamiltonian of which is

$$H = \sum_{j=1}^{2n} \vec{S}^{(j)} \cdot \vec{S}^{(j+1)}, \tag{2.19}$$

with $\vec{S}^{(j)} = (1/2)(\sigma_x^{(j)}\hat{x} + \sigma_y^{(j)}\hat{y} + \sigma_z^{(j)}\hat{z})$ and periodic boundary conditions, $\vec{S}^{(2n+1)} = \vec{S}^{(1)}$. Performing entanglement spectroscopy on the ground state $|\psi\rangle_{AB}$ involves diagonalizing the reduced state $\rho = \mathrm{Tr}_B(|\psi\rangle\langle\psi|_{AB})$. Here we consider a total of 8 spins ($2n = 8$). We take A to be a subset of 4 nearest-neighbor spins, and B is the complement of A.

The results of applying VQSD to the 4-spin reduced state ρ via a simulator are shown in Fig. 2.4. Panel (a) plots the inferred eigenvalues versus the number of layers p in our ansatz (see Fig. 2.2).

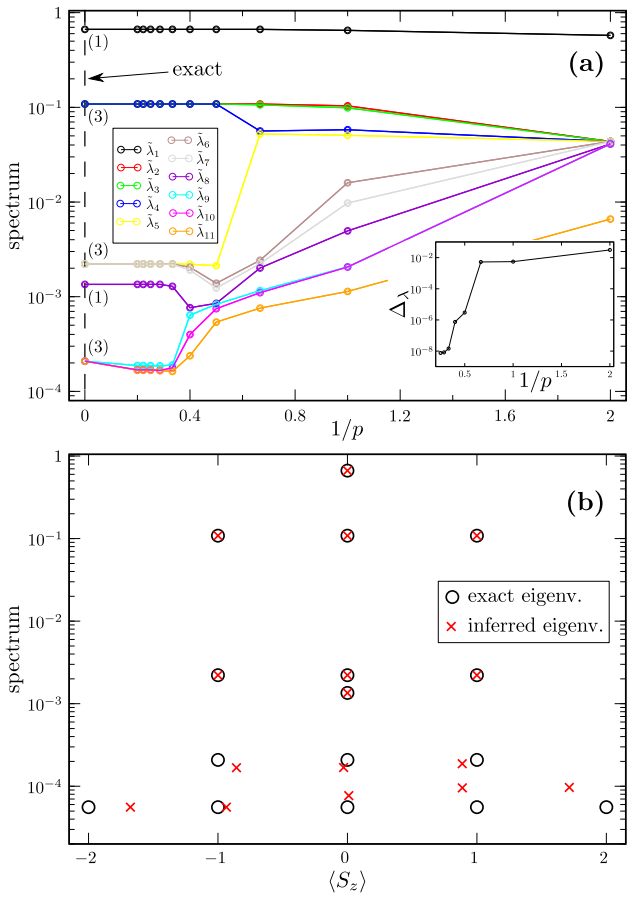


Figure 2.4: Implementing VQSD with a simulator for the ground state of the 1D Heisenberg model, diagonalizing a 4-spin subsystem of a chain of 8 spins. We chose $q = 1$ for the cost in (2.10) and employed a gradient-based method to find $\vec{\alpha}_{\text{opt}}$. (a) Largest inferred eigenvalues $\tilde{\lambda}_j$ versus $1/p$, where p is the number of layers in our ansatz, which in this example takes half-integer values corresponding to fractions of layers shown in Fig. 2.2. The exact eigenvalues are shown on the y-axis (along $1/p = 0$ line) with their degeneracy indicated in parentheses. One can see the largest eigenvalues converge to their correct values, including the correct degeneracies. Inset: overall eigenvalue error Δ_λ versus $1/p$. (b) Largest inferred eigenvalues resolved by the inferred $\langle S_z \rangle$ quantum number of their associated eigenvector, for $p = 5$. The inferred data points (red X's) roughly agree with the theoretical values (black circles), particularly for the largest eigenvalues. Section 2.7 discusses Heisenberg chain of 12 spins.

One can see that the inferred eigenvalues converge to their theoretical values as p increases. Panel (b) plots the inferred eigenvalues resolved by their associated quantum numbers (z-component of total spin). This plot illustrates the feature we noted previously that minimizing our cost will first result in minimizing the eigenvector error for those eigenvectors with the largest eigenvalues. Overall our VQSD implementation returned roughly the correct values for both the eigenvalues

and their quantum numbers. Resolving not only the eigenvalues but also their quantum numbers is important for entanglement spectroscopy [35], and clearly VQSD can do this.

In Section 2.7 we discuss an alternative approach employing a variable ansatz for $U_p(\vec{\alpha})$, and we present results of applying this approach to a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model.

2.3 Discussion

We emphasize that VQSD is meant for states ρ that have either low rank or possibly high rank but low entropy $H(\rho) = -\mathrm{Tr}(\rho \log \rho)$. This is because the eigenvalue readout step of VQSD would be exponentially complex for states with high entropy. In other words, for high entropy states, if one efficiently implemented the eigenvalue readout step (with N_{readout} polynomial in n), then very few eigenvalues would get characterized with the desired precision. In Section 2.11 we discuss the complexity of VQSD for particular example states.

Examples of states for which VQSD is expected to be efficient include density matrices computed from ground states of 1D, local, gapped Hamiltonians. Also, thermal states of some 1D systems in a many-body localized phase at low enough temperature are expected to be diagonalizable by VQSD. These states have rapidly decaying spectra and are eigendecomposed into states obeying a 1D area law [47, 48, 49]. This means that every eigenstate can be prepared by a constant depth circuit in alternating ansatz form [48], and hence VQSD will be able to diagonalize it.

2.3.1 Comparison to literature

Diagonalizing quantum states with classical methods would require exponentially large memory to store the density matrix, and the matrix operations needed for diagonalization would be exponentially costly. VQSD avoids both of these scaling issues.

Another quantum algorithm that extracts the eigenvalues and eigenvectors of a quantum state is qPCA [16]. Similar to VQSD, qPCA has the potential for exponential speedup over classical diagonalization for particular classes of quantum states. Like VQSD, the speedup in qPCA is

contingent on ρ being a low-entropy state.

We performed a simple implementation of qPCA to get a sense for how it compares to VQSD, see Section 2.12 for details. In particular, just like we did for Fig. 2.3, we considered the one-qubit plus state $\rho = |+\rangle\langle+|$. We implemented qPCA for this state on Rigetti's noisy simulator (whose noise is meant to mimic that of their 8Q-Agave quantum computer). The circuit that we implemented applied one controlled-exponential-swap gate (in order to approximately exponentiate ρ, as discussed in [16]). We employed a machine-learning approach [50] to compile the controlled-exponential-swap gate into a novel short-depth gate sequence (see Section 2.12. With this circuit we inferred the two eigenvalues of ρ to be approximately 0.8 and 0.2. Hence, for this simple example, it appears that qPCA gave eigenvalues that were slightly off from the true values of 1 and 0, while VQSD was able to obtain the correct eigenvalues, as discussed in Fig. 2.3.

2.3.2 Future applications

Finally we discuss various applications of VQSD.

As noted in Ref. [16], one application of quantum state diagonalization is benchmarking of quantum noise processes, i.e., quantum process tomography. Here one prepares the Choi state by sending half of a maximally entangled state through the process of interest. One can apply VQSD to the resulting Choi state to learn about the noise process, which may be particular useful for benchmarking near-term quantum computers.

A special case of VQSD is variational state preparation. That is, if one applies VQSD to a pure state $\rho = |\psi\rangle\langle\psi|$, then one can learn the unitary $U(\vec{\alpha})$ that maps $|\psi\rangle$ to a standard basis state. Inverting this unitary allows one to map a standard basis state (and hence the state $|0\rangle^{\otimes n}$) to the state $|\psi\rangle$, which is known as state preparation. Hence, if one is given $|\psi\rangle$ in quantum form, then VQSD can potentially find a short-depth circuit that approximately prepares $|\psi\rangle$. Variational quantum compiling algorithms that were very recently proposed [51, 52] may also be used for this same purpose, and hence it would be interesting to compare VQSD to these algorithms for this special case. Additionally, in this special case one could use VQSD and these other algorithms as an error

mitigation tool, i.e., to find a short-depth state preparation that achieves higher accuracy than the original state preparation.

In machine learning, PCA is a subroutine in supervised and unsupervised learning algorithms and also has many direct applications. PCA inputs a data matrix X and finds a new basis such that the variance is maximal along the new basis vectors. One can show that this amounts to finding the eigenvectors of the covariance matrix $E[XX^T]$ with the largest eigenvalues, where E denotes expectation value. Thus PCA involves diagonalizing a positive-semidefinite matrix, $E[XX^T]$. Hence VQSD can perform this task provided one has access to QRAM [36] to prepare the covariance matrix as a quantum state. PCA can reduce the dimension of X as well as filter out noise in data. In addition, nonlinear (kernel) PCA can be used on data that is not linearly separable. Very recent work by Tang [53] suggests that classical algorithms could be improved for PCA of low-rank matrices, and potentially obtain similar scaling as qPCA and VQSD. Hence future work is needed to compare these different approaches to PCA.

Perhaps the most important near-term application of VQSD is to study condensed matter physics. In particular, we propose that one can apply the variational quantum eigensolver [22] to prepare the ground state of a many-body system, and then one can follow this with the VQSD algorithm to characterize the entanglement in this state. Ultimately this approach could elucidate key properties of condensed matter phases. In particular, VQSD allows for entanglement spectroscopy, which has direct application to the identification of topological order [54]. Extracting both the eigenvalues and eigenvectors is useful for entanglement spectroscopy [54], and we illustrated this capability of VQSD in Fig. 2.4. Finally, an interesting future research direction is to check how the discrepancies in preparation of multiple copies affect the performance of the diagonalization.

2.4 Methods

2.4.1 Diagonalization test circuits

Here we elaborate on the cost functions C_1 and C_2 and present short-depth quantum circuits to compute them.

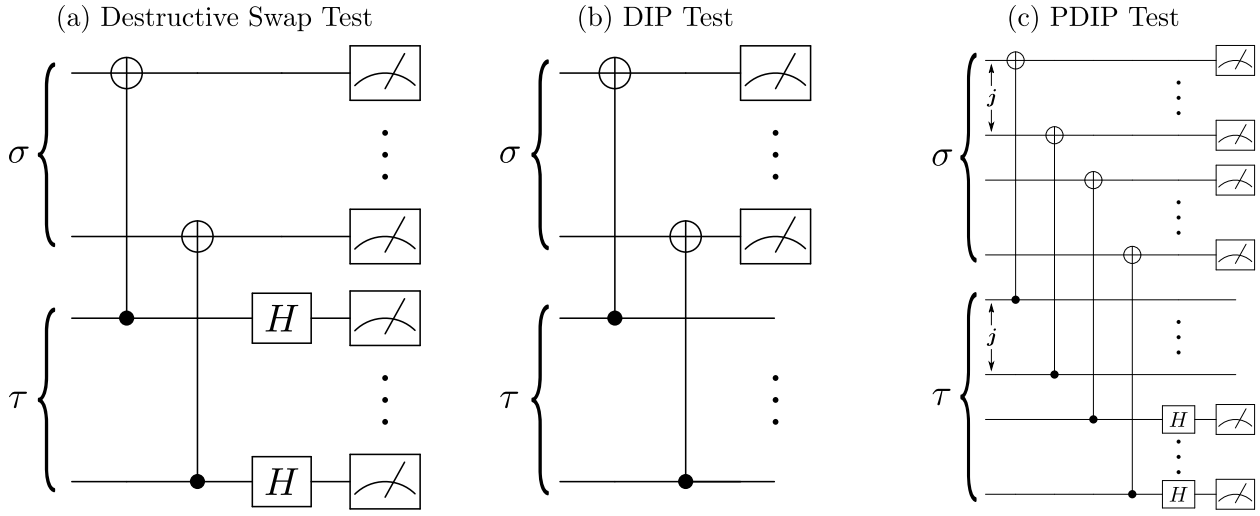


Figure 2.5: Diagonalization test circuits used in VQSD. (a) The Destructive Swap Test computes $\mathrm{Tr}(\sigma\tau)$ via a depth-two circuit. (b) The Diagonalized Inner Product (DIP) Test computes $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ via a depth-one circuit. (c) The Partially Diagonalized Inner Product (PDIP) Test computes $\mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\sigma)\mathcal{Z}_{\vec{j}}(\tau))$ via a depth-two circuit, for a particular set of qubits $\vec{j}$. While the DIP test requires no postprocessing, the postprocessing for the Destructive Swap Test and the Partial DIP Test scales linearly in n.

2.4.1.1 C_1 and the DIP Test

The function C_1 defined in (2.7) has several intuitive interpretations. These interpretations make it clear that C_1 quantifies how far a state is from being diagonal. In particular, let $D_{\mathrm{HS}}(A, B) := \mathrm{Tr}\left((A-B)^\dagger(A-B)\right)$ denote the Hilbert-Schmidt distance. Then we can write

$$C_1 = \min_{\sigma\in\mathcal{D}} D_{\mathrm{HS}}(\tilde{\rho}, \sigma) \tag{2.20}$$

$$= D_{\mathrm{HS}}(\tilde{\rho}, \mathcal{Z}(\tilde{\rho})) \tag{2.21}$$

$$= \sum_{\vec{z},\vec{z}'\neq\vec{z}} |\langle\vec{z}|\tilde{\rho}|\vec{z}'\rangle|^2 . \tag{2.22}$$

In other words, C_1 is (1) the minimum distance between $\tilde{\rho}$ and the set of diagonal states $\mathcal{D}$, (2) the distance from $\tilde{\rho}$ to $\mathcal{Z}(\tilde{\rho})$, and (3) the sum of the absolute squares of the off-diagonal elements of $\tilde{\rho}$.

C_1 can also be written as the eigenvector error in (2.4) as follows. For an inferred eigenvector

$|\tilde{v}_{\vec{z}}\rangle$, we define $|\delta_{\vec{z}}\rangle = \rho|\tilde{v}_{\vec{z}}\rangle - \tilde{\lambda}_{\vec{z}}|\tilde{v}_{\vec{z}}\rangle$ and write the eigenvector error as

$$\langle\delta_{\vec{z}}|\delta_{\vec{z}}\rangle = \langle\tilde{v}_{\vec{z}}|\rho^2|\tilde{v}_{\vec{z}}\rangle + \tilde{\lambda}^2_{\vec{z}} - 2\tilde{\lambda}_{\vec{z}}\langle\tilde{v}_{\vec{z}}|\rho|\tilde{v}_{\vec{z}}\rangle \tag{2.23}$$

$$= \langle\tilde{v}_{\vec{z}}|\rho^2|\tilde{v}_{\vec{z}}\rangle - \tilde{\lambda}^2_{\vec{z}}\,, \tag{2.24}$$

since $\langle\tilde{v}_{\vec{z}}|\rho|\tilde{v}_{\vec{z}}\rangle = \tilde{\lambda}_{\vec{z}}$. Summing over all $\vec{z}$ gives

$$\Delta_v = \sum_{\vec{z}}\langle\delta_{\vec{z}}|\delta_{\vec{z}}\rangle = \sum_{\vec{z}}\langle\tilde{v}_{\vec{z}}|\rho^2|\tilde{v}_{\vec{z}}\rangle - \tilde{\lambda}^2_{\vec{z}} \tag{2.25}$$

$$= \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2) = C_1\,, \tag{2.26}$$

which proves the bound in (2.5) for $q = 1$.

In addition, C_1 bounds the eigenvalue error defined in (2.3). Let $\vec{\tilde{\lambda}} = (\tilde{\lambda}_1, ..., \tilde{\lambda}_d)$ and $\vec{\lambda} = (\lambda_1, ..., \lambda_d)$ denote the inferred and actual eigenvalues of ρ, respectively, both arranged in decreasing order. In this notation we have

$$\Delta_\lambda = \vec{\lambda}\cdot\vec{\lambda} + \vec{\tilde{\lambda}}\cdot\vec{\tilde{\lambda}} - 2\vec{\lambda}\cdot\vec{\tilde{\lambda}} \tag{2.27}$$

$$C_1 = \vec{\lambda}\cdot\vec{\lambda} - \vec{\tilde{\lambda}}\cdot\vec{\tilde{\lambda}} \tag{2.28}$$

$$= \Delta_\lambda + 2(\vec{\lambda}\cdot\vec{\tilde{\lambda}} - \vec{\tilde{\lambda}}\cdot\vec{\tilde{\lambda}})\,. \tag{2.29}$$

Since the eigenvalues of a density matrix majorize its diagonal elements, $\vec{\lambda} \succ \vec{\tilde{\lambda}}$, and the dot product with an ordered vector is a Schur convex function, we have

$$\vec{\lambda}\cdot\vec{\tilde{\lambda}} \geq \vec{\tilde{\lambda}}\cdot\vec{\tilde{\lambda}}\,. \tag{2.30}$$

Hence from (2.29) and (2.30) we obtain the bound

$$\Delta_\lambda \leq C_1\,, \tag{2.31}$$

which corresponds to the bound in (2.5) for the special case of $q = 1$.

For computational purposes, we use the difference of purities interpretation of C_1 given in (2.7). The $\mathrm{Tr}(\rho^2)$ term is independent of $U_p(\vec{\alpha})$. Hence it only needs to be evaluated once, outside of the

parameter optimization loop. It can be computed via the expectation value of the swap operator S on two copies of ρ, using the identity

$$\mathrm{Tr}(\rho^2) = \mathrm{Tr}((\rho \otimes \rho)S) \,. \tag{2.32}$$

This expectation value is found with a depth-two quantum circuit that essentially corresponds to a Bell-basis measurement, with classical post-processing that scales linearly in the number of qubits [55, 50]. This is shown in Fig. 2.5(a). We call this procedure the Destructive Swap Test, since it is like the Swap Test, but the measurement occurs on the original systems instead of on an ancilla.

Similarly, the $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ term could be evaluated by first dephasing $\tilde{\rho}$ and then performing the Destructive Swap Test, which would involve a depth-three quantum circuit with linear classical post-processing. This approach was noted in Ref. [56]. However, there exists a simpler circuit, which we call the Diagonalized Inner Product (DIP) Test. The DIP Test involves a depth-one quantum circuit with no classical post-processing. An abstract version of this circuit is shown in Fig. 2.5(b), for two states σ and τ. The proof that this circuit computes $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ is given in Section 2.13 For our application we will set $\sigma = \tau = \tilde{\rho}$, for which this circuit gives $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$.

In summary, C_1 is efficiently computed by using the Destructive Swap Test for the $\mathrm{Tr}(\rho^2)$ term and the DIP Test for the $\mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$ term.

2.4.1.2 C_2 and the PDIP test

Like C_1, C_2 can also be rewritten in terms of of the Hilbert-Schmidt distance. Namely, C_2 is the average distance of $\tilde{\rho}$ to each locally-dephased state $\mathcal{Z}_j(\tilde{\rho})$:

$$C_2 = \frac{1}{n}\sum_{j=1}^{n} D_{\mathrm{HS}}(\tilde{\rho}, \mathcal{Z}_j(\tilde{\rho})) \,. \tag{2.33}$$

where $\mathcal{Z}_j(\cdot) = \sum_z (|z\rangle\langle z|_j \otimes I_{k\neq j})(\cdot)(|z\rangle\langle z|_j \otimes I_{k\neq j})$. Naturally, one would expect that $C_2 \leq C_1$, since $\tilde{\rho}$ should be closer to each locally dephased state than to the fully dephased state. Indeed this is true and can be seen from:

$$C_2 = C_1 - \frac{1}{n}\sum_{j=1}^{n} \min_{\sigma\in\mathcal{D}} D_{\mathrm{HS}}(\mathcal{Z}_j(\tilde{\rho}), \sigma) \,. \tag{2.34}$$

However, C_1 and C_2 vanish under precisely the same conditions, as noted in Eq. (2.9). One can see this by noting that C_2 also upper bounds $(1/n)C_1$ and hence we have

$$C_2 \leq C_1 \leq nC_2 \,. \tag{2.35}$$

Combining the upper bound in (2.35) with the relations in (2.26) and (2.31) gives the bounds in (2.5) with β defined in (2.11). The upper bound in (2.35) is proved as follows. Let $\vec{z} = z_1 \ldots z_n$ and $\vec{z'} = z'_1 \ldots z'_n$ be n-dimensional bitstrings. Let $\mathcal{S}$ be the set of all pairs $(\vec{z}, \vec{z'})$ such that $\vec{z} \neq \vec{z'}$, and let $\mathcal{S}_j$ be the set of all pairs $(\vec{z}, \vec{z'})$ such that $z_j \neq z'_j$. Then we have $C_1 = \sum_{(\vec{z},\vec{z'})\in\mathcal{S}} |\langle\vec{z}|\tilde{\rho}|\vec{z'}\rangle|^2$, and

$$nC_2 = \sum_{j=1}^{n} \sum_{(\vec{z},\vec{z'})\in\mathcal{S}_j} |\langle\vec{z}|\tilde{\rho}|\vec{z'}\rangle|^2 \tag{2.36}$$

$$\geq \sum_{(\vec{z},\vec{z'})\in\mathcal{S}^U} |\langle\vec{z}|\tilde{\rho}|\vec{z'}\rangle|^2 = C_1 \,, \tag{2.37}$$

where $\mathcal{S}^U = \bigcup_{j=1}^{n} \mathcal{S}_j$ is the union of all the $\mathcal{S}_j$ sets. The inequality in (2.37) arises from the fact that the $\mathcal{S}_j$ sets have non-trivial intersection with each other, and hence we throw some terms away when only considering the union $\mathcal{S}^U$. The last equality follows from the fact that $\mathcal{S}^U = \mathcal{S}$, i.e, the set of all bitstring pairs that differ from each other ($\mathcal{S}$) corresponds to the set of all bitstring pairs that differ for at least one element ($\mathcal{S}^U$).

Writing C_2 in terms of purities, as in (2.8), shows how it can be computed on a quantum computer. As in the case of C_1, the first term in (2.8) is computed with the Destructive Swap Test. For the second term in (2.8), each purity $\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)$ could also be evaluated with the Destructive Swap Test, by first locally dephasing the appropriate qubit. However, we present a slightly improved circuit to compute these purities that we call the Partially Diagonalized Inner Product (PDIP) Test. The PDIP Test is shown in Fig. 2.5(c) for the general case of feeding in two distinct states σ and τ with the goal of computing the inner product between $\mathcal{Z}_{\vec{j}}(\sigma)$ and $\mathcal{Z}_{\vec{j}}(\tau)$. For generality we let l, with $0 \leq l \leq n$, denote the number of qubits being locally dephased for this computation. If $l > 0$, we define $\vec{j} = (j_1, \ldots, j_l)$ as a vector of indices that indicates which qubits are being locally dephased. The PDIP Test is a hybrid of the Destructive Swap Test and the DIP Test, corresponding to the former when $l = 0$ and the latter when $l = n$. Hence, it generalizes both the Destructive Swap

Test and the DIP Test. Namely, the PDIP Test performs the DIP Test on the qubits appearing in $\vec{j}$ and performs the Destructive Swap Test on the qubits not appearing in $\vec{j}$. The proof that the PDIP Test computes $\mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\sigma)\mathcal{Z}_{\vec{j}}(\tau))$, and hence $\mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\tilde{\rho})^2)$ when $\sigma = \tau = \tilde{\rho}$, is given in Section 2.13.

2.4.1.3 C_1 versus C_2

Here we discuss the contrasting merits of the functions C_1 and C_2, hence motivating our cost definition in (2.10).

As noted previously, C_2 does not have an operational meaning like C_1. In addition, the circuit for computing C_1 is more efficient than that for C_2. The circuit in Fig. 2.5(b) for computing the second term in C_1 has a gate depth of one, with n CNOT gates, n measurements, and no classical post-processing. The circuit in Fig. 2.5(c) for computing the second term in C_2 has a gate depth of two, with n CNOT gates, $n-1$ Hadamard gates, $2n-1$ measurements, and classical post-processing whose complexity scales linearly in n. So in every aspect, the circuit for computing C_1 is less complex than that for C_2. This implies that C_1 can be computed with greater accuracy than C_2 on a noisy quantum computer.

On the other hand, consider how the landscape for C_1 and C_2 scale with n. As a simple example, suppose $\rho = |0\rangle\langle 0| \otimes \cdots \otimes |0\rangle\langle 0|$. Suppose one takes a single parameter ansatz for U, such that $U(\theta) = R_X(\theta) \otimes \cdots \otimes R_X(\theta)$, where $R_X(\theta)$ is a rotation about the X-axis of the Bloch sphere by angle θ. For this example,

$$C_1(\theta) = 1 - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2) = 1 - x(\theta)^n \tag{2.38}$$

where $x(\theta) = \mathrm{Tr}(\mathcal{Z}(R_X(\theta)|0\rangle\langle 0|R_X(\theta)^\dagger)^2) = (1+\cos^2\theta)/2$. If θ is not an integer multiple of π, then $x(\theta) < 1$, and $x(\theta)^n$ will be exponentially suppressed for large n. In other words, for large n, the landscape for $x(\theta)^n$ becomes similar to that of a delta function: it is zero for all θ except for multiples of π. Hence, for large n, it becomes difficult to train the unitary $U(\theta)$ because the gradient vanishes for most θ. This is just an illustrative example, but this issue is general. Generally speaking, for large n, the function C_1 has a sharp gradient near its global minima, and the gradient

vanishes when one is far away from these minima. Ultimately this limits C_1's utility as a training function for large n.

In contrast, C_2 does not suffer from this issue. For the example in the previous paragraph,

$$C_2(\theta) = 1 - x(\theta)\,, \tag{2.39}$$

which is independent of n. So for this example the gradient of C_2 does not vanish as n increases, and hence C_2 can be used to train θ. More generally, the landscape of C_2 is less barren than that of C_1 for large n. We can argue this, particularly, for states ρ that have low rank or low entropy. The second term in (2.8), which is the term that provides the variability with $\vec{\alpha}$, does not vanish even for large n, since (as shown in Section 2.14):

$$\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) \geq 2^{-H(\rho)-1} \geq \frac{1}{2r}\,. \tag{2.40}$$

Here, $H(\rho) = -\mathrm{Tr}(\rho \log_2 \rho)$ is the von Neumann entropy, and r is the rank of ρ. So as long as ρ is low entropy or low rank, then the second term in C_2 will not vanish. Note that a similar bound does not exist for second term in C_1, which does tend to vanish for large n.

2.4.2 Optimization methods

Finding $\vec{\alpha}_{\mathrm{opt}}$ in (2.6) is a major component of VQSD. While many works have benchmarked classical optimization algorithms (e.g., Ref. [57]), the particular case of optimization for variational hybrid algorithms [58] is limited and needs further work [59]. Both gradient-based and gradient-free methods are possible, but gradient-based methods may not work as well with noisy data. Additionally, Ref. [40] notes that gradients of a large class of circuit ansatze vanish when the number of parameters becomes large. These and other issues (e.g., sensitivity to initial conditions, number of function evaluations) should be considered when choosing an optimization method.

In our preliminary numerical analyses (see Section 2.10), we found that the Powell optimization algorithm [60] performed the best on both quantum computer and simulator implementations of VQSD. This derivative-free algorithm uses a bi-directional search along each parameter using

Brent's method. Our studies showed that Powell's method performed the best in terms of convergence, sensitivity to initial conditions, and number of correct solutions found. The implementation of Powell's algorithm used in this paper can be found in the open-source Python package SciPy Optimize [61]. Finally, Section 2.8 shows how our layered ansatz for $U_p(\vec{\alpha})$ as well as proper initialization of $U_p(\vec{\alpha})$ helps in mitigating the problem of local minima.

2.5 Code availability

The code used to generate some of the examples presented here can be accessed from [45].

2.6 Details on VQSD implementations

Here we provide further details on our implementations of VQSD in Sec. 2.2.2. This includes further details about the optimization parameters as well as additional statistics for our runs on the quantum computer.

2.6.1 Optimization parameters

First, we discuss our implementation on a quantum computer (data shown in Fig. 2.3). Figure 2.6 displays the circuit used for this implementation. This circuit is logically divided into three sections. First, we prepare two copies of the plus state $\rho = |+\rangle\langle+| = H|0\rangle\langle 0|H$ by doing a Hadamard gate H on each qubit. Next, we implement one layer of a unitary ansatz, namely $U(\theta) = R_x(\pi/2)R_z(\theta)$. This ansatz was chosen because each gate can be natively implemented on Rigetti's quantum computer. To simplify the search space, we restricted to one parameter instead of a universal one-qubit unitary. Last, we implement the DIP Test circuit, described in Fig. 2.5, which here consists of only one CNOT gate and one measurement.

For the parameter optimization loop, we used the Powell algorithm mentioned in Sec. 2.4.2. This algorithm found the minimum cost in less than ten objective function evaluations on average. Each objective function evaluation (i.e., call to the quantum computer) sampled from 10,000 runs of the circuit in Fig. 2.6. As can be seen in Fig. 2.3(b), 10,000 runs was sufficient to accurately

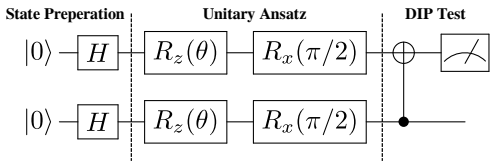


Figure 2.6: Circuit used to implement VQSD for $\rho = |+\rangle\langle+|$ on Rigetti's 8Q-Agave quantum computer. Vertical dashed lines separate the circuit into logical components.

estimate the cost function (2.10) with small variance. Because the problem size was small, we took $q = 1$ in (2.10), which provided adequate variability in the cost landscape.

Because of the noise levels in current quantum computers, we limited VQSD implementations on quantum hardware to only one-qubit states. Noise affects the computation in multiple areas. For example, in state preparation, qubit-specific errors can cause the two copies of ρ to actually be different states. Subsequent gate errors (notably two-qubit gates), decoherence, and measurement errors prevent the cost from reaching zero even though the optimal value of θ is obtained. The effect of these various noise sources, and in particular the effect of discrepancies in preparation of two copies of ρ, will be important to study in future work.

Next, we discuss our VQSD implementation on a simulator (data shown in Fig. 2.4). For this implementation we again chose $q = 1$ in our cost function. Because of the larger problem size (diagonalizing a 4-qubit state), we employed multiple layers in our ansatz, up to $p = 5$. The simulator directly calculated the measurement probability distribution in the DIP Test, as opposed to determining the desired probability via sampling. This allowed us to use a gradient-based method to optimize our cost function, reducing the overall runtime of the optimization. Hence, our simulator implementation for the Heisenberg model demonstrated a future application of VQSD while alleviating the optimization bottleneck that is present for all variational quantum algorithms on large problem sizes, an area that needs further research [59]. We explore optimization methods further in Section 2.10.

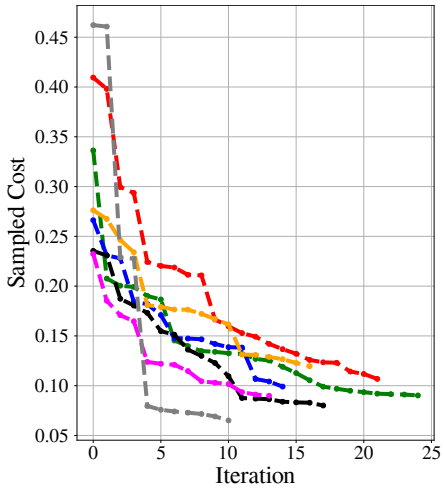


Figure 2.7: Cost vs iteration for all attempts of VQSD on Rigetti's 8Q-Agave computer for diagonalizing the plus state $\rho = |+\rangle\langle+|$. Each of the seven curves represents a different independent run. Each run starts at a random initial angle and uses the Powell optimization algorithm to minimize the cost.

2.6.2 Additional statistics for the quantum computer implementation

Here, we present statistics for several runs of the VQSD implementation run on Rigetti's 8Q-Agave quantum computer. One example plot of cost vs. iteration for diagonalizing the plus state $\rho = |+\rangle\langle+|$ is shown in Figure 2.3(a). Here, we present all data collected for this implementation of VQSD, shown in Figure 2.7. The following table displays the final costs achieved as well the associated inferred eigenvalues.

VQSD Run	$\min(C)$	$\min(\tilde{\lambda}^{\text{est}}_{\vec{z}})$	$\max(\tilde{\lambda}^{\text{est}}_{\vec{z}})$
1	0.107	0.000	1.000
2	0.090	0.142	0.858
3	0.099	0.054	0.946
4	0.120	0.079	0.921
5	0.080	0.061	0.939
6	0.090	0.210	0.790
7	0.65	0.001	0.999
Avg.	0.093	**0.078**	**0.922**
Std.	0.016	0.070	0.070

Table 2.1: Minimum cost and eigenvalues achieved after performing the parameter optimization loop for seven independent runs of VQSD for the example discussed in Sec. 2.2.2. The final two rows show average values and standard deviation across all runs.

2.7 Alternative ansatz and the Heisenberg model ground state

In this Section, we describe a modification of the layered ansatz discussed in Section 2.2.1. Figure 2.2 in the main text shows an example of a layered ansatz in which every layer has the same, fixed structure consisting of alternating two-qubit gates acting on nearest-neighbor qubits. The modified approach presented here may be useful in situations where there is no natural choice of the structure of the layered ansatz.

Here, instead of working with a fixed structure for the diagonalizing unitary $U(\vec{\alpha})$, we allow it to vary during the optimization process. The algorithm used to update the structure of $U(\vec{\alpha})$ is probabilistic and resembles the one presented in [50].

In the examples studied here, the initial $U(\vec{\alpha})$ consists of a small number of random two-qubit gates with random supports (i.e. the qubits on which a gate acts). An optimization step involves minimizing the cost function by changing parameters $\vec{\alpha}$ as well as a small random change to the structure of $U(\vec{\alpha})$. This change to the structure typically amounts to a random modification of support for a limited number of gates. The new structure is accepted or rejected following the usual simulated annealing schemes. We refer the reader to Section II D of [50] for further details on the optimization method.

The gate sequence representing $U(\vec{\alpha})$ is allowed to grow. If the algorithm described above cannot minimize the cost function for a specified number of iterations, an identity gate (spanned by new variational parameters) is randomly added to $U(\vec{\alpha})$. This step is similar in spirit to adding a layer to $U(\vec{\alpha})$ as discussed in Section 2.2.1 of the main text.

We compared the current method with the one based on the layered ansatz and found that it produced diagonalizing circuits involving significantly fewer gates. Figure 2.8 shows the eigenvalue error Δ_λ, defined in Eq. (2.3), as a function of $1/D$, where D is the total number of gates of $U(\vec{\alpha})$. Here, VQSD is used to diagonalize a 4-qubit reduced state of the ground state of the one-dimensional Heisenberg model defined on 8 qubits, see Eq. (2.19). For every number of gates D, the current algorithm outperforms the one based on the fixed, layered ansatz. It finds a sequence of gates that results in a smaller eigenvalue error Δ_λ.

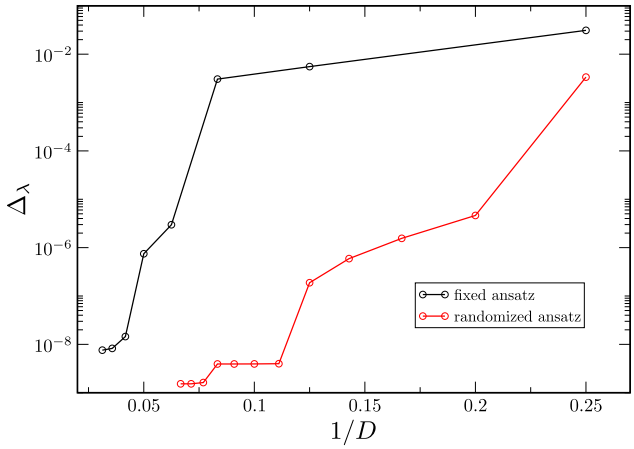


Figure 2.8: Comparison of two approaches to obtaining the diagonalizing unitary $U(\vec{\alpha})$: (i) based on a fixed layered ansatz shown in Fig. 2.2 in the main text (black line) and (ii) based on random updates to the structure of $U(\vec{\alpha})$ (red line). The plot shows eigenvalue error Δ_λ versus $1/D$, where D is the number of gates in $U(\vec{\alpha})$. For the same D, the second approach found a more optimal gate sequence.

Finally, we use the current optimization approach to find the spectrum of a 6-qubit reduced state ρ of the 12-qubit ground state of a one-dimensional Heisenberg model. The results of performing VQSD on ρ are shown in Fig. 2.9. Panel (a) shows the convergence of the 11 largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ to their exact values. We can see that the quality of the inferred eigenvalues increases quickly with the number of gates D used in the diagonalizing unitary $U(\vec{\alpha})$. In panel (b), we show the dominant part of the spectrum of ρ resolved in the z-component of the total spin. The results show that VQSD could be used to accurately obtain the dominant part of the spectrum of the density matrix together with the associated quantum numbers.

2.8 Optimization and local minima

In this Section we describe a strategy to avoid local minima that is used in the optimization algorithms throughout the paper and detailed in Section 2.7. We adapt the optimization involved in the diagonalization of the 6-qubit density matrix described in Section 2.7 as an illustrative example.

We note that the classical optimization problem associated with VQSD is potentially very difficult one. In the example studied in Section 2.7 the diagonalizing unitary consisted of 150 two-qubit gates. This means that in order to find that unitary one has to optimize over at least

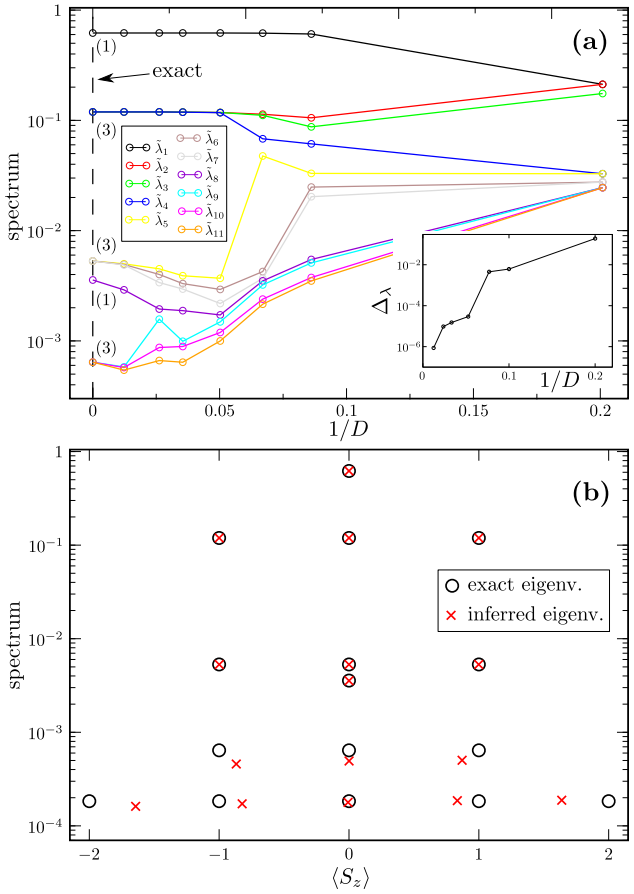


Figure 2.9: VQSD applied to the ground state of the Heisenberg model. Here we consider a 6-qubit reduced state ρ of the 12-qubit ground state. (a) Largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ as a function of $1/D$, where D is the total number of gates in the diagonalizing unitary $U(\vec{\alpha})$. The inferred eigenvalues converge to their exact values shown along the $1/D = 0$ line recovering the correct degeneracy. Inset: Eigenvalue error Δ_λ as a function of $1/D$. (b) The largest inferred eigenvalues $\tilde{\lambda}_j$ of ρ resolved in the $\langle S_z \rangle$ quantum number. We find very good agreement between the inferred eigenvalues (red crosses) and the exact ones (black circles), especially for large eigenvalues. The data was obtained for $D = 150$ gates.

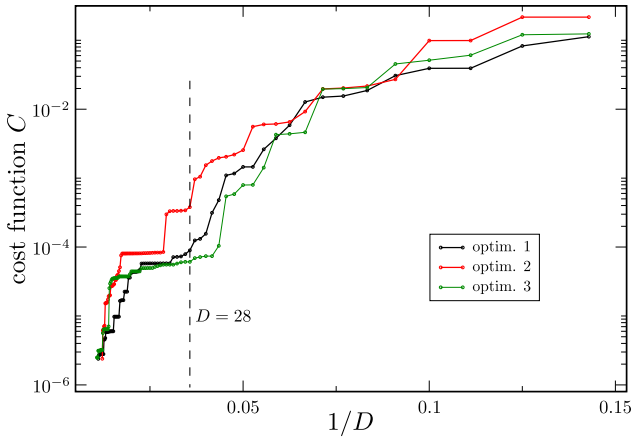


Figure 2.10: Cost function C versus $1/D$ for three independent optimization runs. Here, D is the total number of gates in the diagonalizing unitary $U_D(\vec{\alpha})$. Every optimization run got stuck at local minimum at some point during the minimization but thanks to the growth of the ansatz for $U_D(\vec{\alpha})$ described in the text, the predefined small value of C was eventually attained. The data was obtained for a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model.

$150 \cdot 13$ continuous parameters (every two-qubit gate is spanned by 15 parameters, but there is some reduction in the total number of parameters when two consecutive gates have overlapping supports). Initiated randomly, off-the-shelf techniques will most likely return suboptimal solution due to the presence of multiple local minima and the rough cost function landscape.

Let $U_D(\vec{\alpha})$ denote a diagonalizing unitary that is built by D two-qubit gates parameterized by $\vec{\alpha}$. Our optimization method begins with a shallow circuit consisting of few gates only. Since there is only a small number of variational parameters, the local minimum is quickly attained. After this initial step, the circuit that implements the unitary $U_D(\vec{\alpha})$ is grown by adding an identity gate (either randomly as discussed in this Section or by means of a layer of identity gates as presented in the main text). This additional gate contains new variational parameters that are initiated such that the unitary $U_D(\vec{\alpha}) = U_{D+1}(\vec{\alpha})$ and hence the value of the cost function are not changed. After the gate was added, the unitary $U_{D+1}(\vec{\alpha})$ contains more variational parameters which allows for further minimization of the cost function. In summary, the optimization of a deeper circuit $U_{D+1}(\vec{\alpha})$ is initialized by previously obtained $U_D(\vec{\alpha})$ as opposed to random initialization. What is more, even if the unitary $U_D(\vec{\alpha})$ was not the most optimal one for a given D, the growth of the circuit allows the algorithm to escape the local minimum and eventually find the global one, as illustrated by an

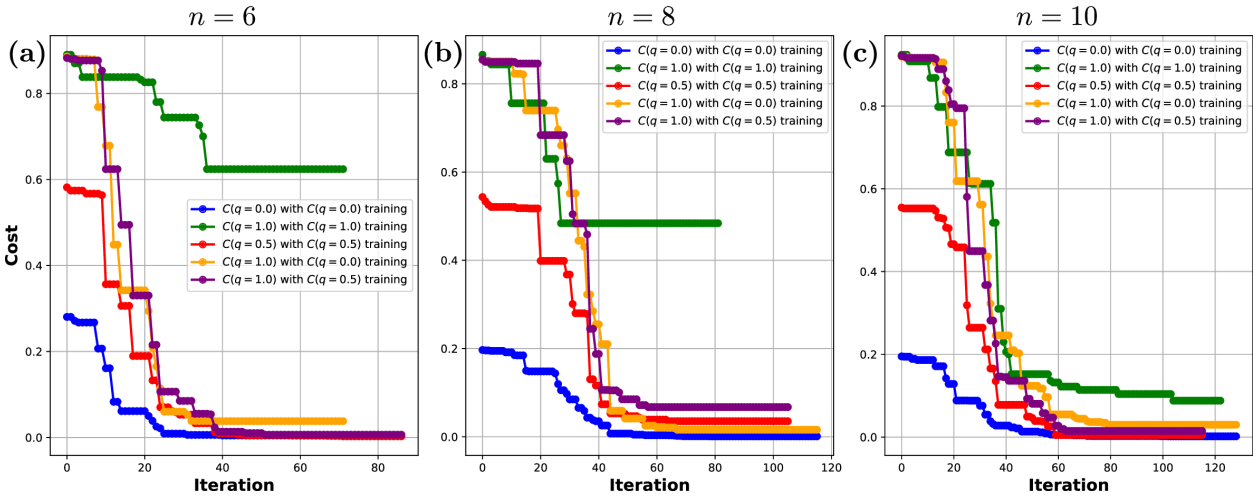


Figure 2.11: Cost versus iteration for different values of q, when ρ is a tensor product of pure states on n qubits. Here we consider (a) $n = 6$, (b) $n = 8$, and (a) $n = 10$. We employed the COBYLA optimization method for training (see Section 2.10 for discussion of this method). For each call to the quantum simulator (i.e., classical simulator of a quantum computer), we took 500 shots for statistics. The green, red, and blue curves respectively correspond to directly training the cost with $q = 1$, $q = 0.5$, and $q = 0$. The purple and yellow curves respectively correspond to evaluating the $q = 1$ cost for the angles $\vec{\alpha}$ obtained by training the $q = 0.5$ and $q = 0$ costs.

example below and shown in Fig. 2.10. For a similar discussion, see [41].

To clarify the above analysis, let us consider an example of diagonalizing a 6-qubit reduced state of the 12-qubit ground state of the Heisenberg model, see Sec. 2.7 for comparison. Figure 2.10 shows the value of the cost function C as a function of $1/D$ for three independent optimization runs. Each optimization was initialized randomly and we applied the same optimization scheme described above to each of them. We see that despite getting stuck in local minima, every optimization run managed to minimize the cost function to the predefined small value (which was set to $2 \cdot 10^{-6}$ in this example). For instance, at $D = 28$, optimization run no. 2 clearly returns suboptimal solution (optimization run no. 3 gives lower cost function by a factor of 6) but after adding several identity gates, it manages to escape the local minimum and continue towards the global one.

2.9 Optimization runs with various q values

In this Section we present some numerical results for training our overall cost function for various values of q. Recall from Eq. (2.10) that q is the weighting parameter that weights the contributions

of C_1 and C_2 in the overall cost, as follows:

$$C(U_p(\vec{\alpha})) = qC_1(U_p(\vec{\alpha})) + (1-q)C_2(U_p(\vec{\alpha}))\,, \tag{2.41}$$

where

$$C_1(U_p(\vec{\alpha})) = \mathrm{Tr}(\rho^2) - \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)\,, \tag{2.42}$$

$$C_2(U_p(\vec{\alpha})) = \mathrm{Tr}(\rho^2) - \frac{1}{n}\sum_{j=1}^{n}\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2)\,. \tag{2.43}$$

As argued in Section 2.2, C_1 is operationally meaningful, while C_2 has a landscape that is more amendable to training when n is large. In particular, one expects that for large n, the gradient of C_1 is sharp near the global minima but vanishes exponentially in n away from these minima. In contrast, the gradient of C_2 is not expected to exponentially vanish as n increases, even away from the minima.

Here, we numerically study the performance for different q values for a simple example where ρ is a tensor product of qubit pure states. Namely, we choose $\rho = \bigotimes_{j=1}^{n} V_j|0\rangle\langle 0|V_j^\dagger$, where $V_j = R_X(\theta_j)$ with θ_j randomly chosen. Such tensor product states are diagonalizable by a single layer ansatz: $U(\vec{\alpha}) = \bigotimes_{j=1}^{n} R_X(\alpha_j)$. We consider three different problem sizes: $n = 6$, 8, and 10. Figure 2.11 shows our numerical results.

Directly training the C_1 cost (corresponding to $q = 1$) sometimes fails to find the global minimum. One can see this in Fig. 2.11, where the green curve fails to fully reach zero cost. In contrast, the red and blue curves in Fig. 2.11, which correspond to $q = 0.5$ and $q = 0$ respectively, approximately go to zero for large iterations.

Even more interesting are the purple and yellow curves, which respectively correspond to evaluating the C_1 cost at the angles $\vec{\alpha}$ obtained from training the $q = 0.5$ and $q = 0$ costs. It is remarkable that both the purple and yellow curves perform better (i.e., achieve lower values) than the green curve. This implies that one can indirectly train the C_1 cost by training the $q = 0.5$ or $q = 0$ costs, and this indirect training performs better than directly training C_1. Since C_1 is operationally meaningful, this indirect training with $q < 1$ is performing better in an operationally meaningful way.

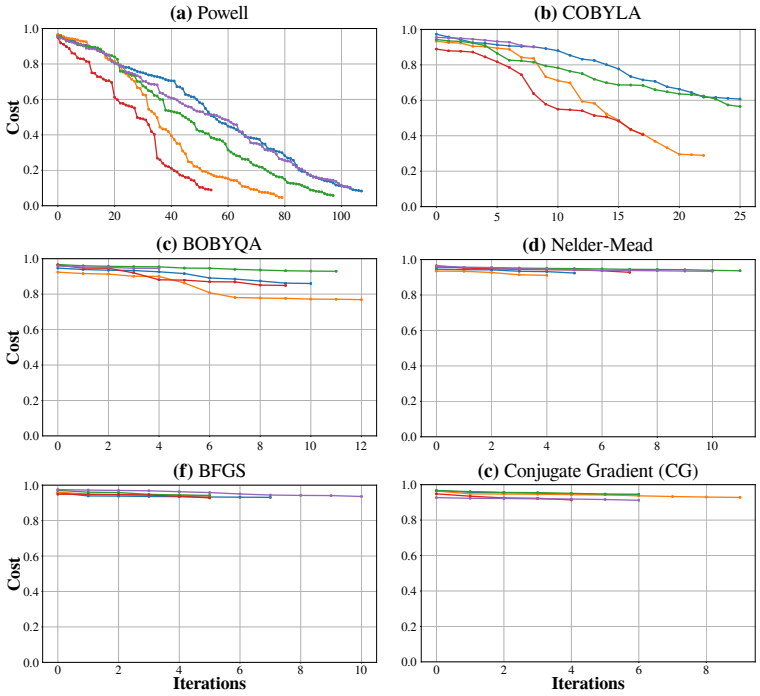


Figure 2.12: Optimization tests on six-qubit product states in the VQSD algorithm. Each plot shows a different optimization algorithm (described in main text) and curves on each plot show optimization attempts with different (random) initial conditions. Cost refers to the C_1 cost function ($q = 1$ in (2.10)), and each iteration is defined by a decrease in the cost function. As can be seen, the Powell algorithm is the most robust to initial conditions and provides the largest number of solved problem instances.

We expect that direct training of C_1 will perform worse as n increases, due to the exponential vanishing of the gradient of C_1. The particular runs shown in Fig. 2.11 do not show this trend, although this can be explained by the fact that the gradient of C_1 depends significantly on the initial values of the $\vec{\alpha}$, and indeed we saw large variability in the performance of the green curve even for a fixed n. Nevertheless, it is worth noting that we were always able to directly train C_1 (i.e., to make the green curve go to zero) for $n < 6$, which is consistent with our expectations.

Overall, Fig. 2.11 provides numerical justification of the definition of our cost function as a weighted average, as in Eq. (2.10). Namely, it shows that there is an advantage to choosing $q < 1$.

2.10 Comparison of optimization methods

As emphasized previously, numerical optimization plays a key role in all variational hybrid algorithms, and further research in optimization methods is needed. In VQSD, the accuracy of

the inferred eigenvalues are closely tied to the performance of the optimization algorithm used in the parameter optimization loop. This issue becomes increasingly important as one goes to large problem sizes (large n), where the number of parameters in the diagonalizing unitary becomes large.

Here, we compare the performance of six different optimization algorithms when used inside the parameter optimization loop of VQSD. These include Powell's algorithm [60], Constrained Optimization BY Linear Approximation (COBYLA) [62], Bound Optimization BY Quadratic Approximation (BOBYQA) [63], Nelder-Mead [64], Broyden-Fletcher-Goldfarb-Shanno (BFGS) [65], and conjugate gradient (CG) [65]. As mentioned in the main text, Powell's algorithm is a derivative-free optimizer that uses a bi-directional search along each parameter. The COBYLA and BOBYQA algorithms are both *trust region* or *restricted-step* methods, which approximate the objective function by a model function. The region where this model function is a good approximation is known as the trust region, and at each step the optimizer attempts to expand the trust region. The Nelder-Mead algorithm is a simplex method useful for derivative-free optimization with smooth objective functions. Lastly, the BFGS and CG algorithms are both gradient-based. The BFGS method is a quasi-Newton method that uses first derivatives only, and the CG method uses a nonlinear conjugate gradient descent. The implementations used in our study can be found in the open-source Python package SciPy Optimize [61] and in Ref. [66].

For this study, we take the input state ρ to be a six-qubit pure product state:

$$\rho = \bigotimes_{j=1}^{6} |\psi_j\rangle\langle\psi_j| \,, \quad \text{where} \quad |\psi_j\rangle = V_j|0\rangle \,. \tag{2.44}$$

Here, the state preparation unitary is

$$V_j = R_x(\alpha_x^{(j)}) R_y(\alpha_y^{(j)}) R_z(\alpha_z^{(j)}) \tag{2.45}$$

where the angles $(\alpha_x^{(j)}, \alpha_y^{(j)}, \alpha_z^{(j)})$ are randomly chosen.

Using each algorithm, we attempt to minimize the cost by adjusting 36 parameters in one layer of the unitary ansatz in Fig. 2.2. For fairness of comparison, only the objective function and initial

Alg.	Powell	COBYLA	BOBYQA	Nelder-Mead	BFGS	CG
r.r.	13.20	1	2.32	23.65	3.83	2.89
f.ev.	4474	341	518	7212	1016	1045

Table 2.2: Relative average run-times (r.r.) and absolute number of function evaluations (f.ev.) of each optimization algorithm (Alg.) used for the data obtained in Fig. 2.12. For example, BOBYQA took 2.32 times as long to run on average than COBYLA, which took the least time to run out of all algorithms. Absolute run-times depend on a variety of factors and computer performance. For reference, the COBYLA algorithm takes approximately one minute for this problem on a laptop computer. The number of cost function evaluations used (related to run-time but also dependent on the method used by the optimizer) is shown in the second row.

starting point were input to each algorithm, i.e., no special options such as constraints, bounds, or other information was provided. The results of this study are shown in Fig. 2.12 and Table 2.2.

Figure 2.12 shows cost versus iteration for each of the six algorithms. Here, we define one iteration by a call to the objective function in which the cost decreases. In particular, the number of iterations is different than the number of cost function evaluations (see Table 2.2), which is not set a priori but rather determined by the optimizer. Plotting cost per each function evaluation would essentially produce a noisy curve since the optimizer is trying many values for the parameters. Instead, we only plot the cost for each parameter update in which the cost decreases. Both the number of iterations, function evaluations, and overall runtime are important features of the optimizer.

In this study, as well as others, we found that the Powell optimization algorithm provides the best performance in terms of lowest minimum cost achieved, sensitivity to initial conditions, and fraction of correct solutions found. The trust-region algorithms COBYLA and BOBYQA were the next best methods. In particular, although the Powell algorithm consistently obtained lower minimum costs, the COBYLA method ran thirteen times faster on average (see Table 2.2). Indeed, both trust region methods provided the shortest runtime. The gradient-based methods BFGS and CG had comparable run-times but were unable to find any minima. Similarly, the Nelder-Mead simplex algorithm was unable to find any minima. This method also had the longest average run-time of all algorithms tested.

This preliminary analysis suggests that the Powell algorithm is the best method for VQSD.

For other variational quantum algorithms, this may not necessarily be the case. In particular, we emphasize that the optimization landscape is determined by both the unitary ansatz and the cost function definition, which may vary drastically in different algorithms. While we found that gradient-based methods did not perform well for VQSD, they may work well for other applications. Additionally, optimizers that we have not considered here may also provide better performance. We leave these questions to further work.

2.11 Complexity for particular examples

2.11.1 General complexity remarks

In what follows we discuss some simple examples of states to which one might apply VQSD. There are several aspects of complexity to keep in mind when considering these examples, including:

(C1) The gate complexity of the unitary that diagonalizes ρ. (It is worth remarking that approximate diagonalization might be achieved with a less complex unitary than exact diagonalization.)

(C2) The complexity of searching through the search space to find the diagonalizing unitary.

(C3) The statistical complexity associated with reading out the eigenvalues.

Naturally, (C1) is related to (C2). However, being efficient with respect to (C1) does not guarantee that (C2) is efficient.

2.11.2 Example states

In the simplest case, suppose $\rho = |\psi_1\rangle\langle\psi_1| \otimes \cdots \otimes |\psi_n\rangle\langle\psi_n|$ is a tensor product of pure states. This state can be diagonalized by a depth-one circuit $U = U_1 \otimes \cdots \otimes U_n$ composed of n one-qubit gates (all done in parallel). Each U_j diagonalizes the associated $|\psi_j\rangle\langle\psi_j|$ state. Searching for this unitary within our ansatz can be done by setting $p = 1$, i.e., with a single layer L_1 shown in Fig. 2.2. A single layer is enough to find the unitary that exactly diagonalizes ρ in this case. Hence, for this example, both complexities (C1) and (C2) are efficient. Finally, note that the eigenvalue readout, (C3), is efficient because there is only one non-zero eigenvalue. Hence, $\tilde{\lambda}_{\vec{z}}^{\text{est}} \approx 1$ and $\epsilon_{\vec{z}} \approx 1/\sqrt{N_{\text{readout}}}$ for

this eigenvalue. This implies that N_{readout} can be chosen to be constant, independent of n, in order to accurately characterize this eigenvalue.

A generalization of product states are classically correlated states, which have the form

$$\rho = \sum_{\vec{z}} p_{\vec{z}} |b_{z_1}^{(1)}\rangle\langle b_{z_1}^{(1)}| \otimes \cdots \otimes |b_{z_n}^{(n)}\rangle\langle b_{z_n}^{(n)}| \tag{2.46}$$

where $\{|b_0^{(j)}\rangle, |b_1^{(j)}\rangle\}$ form an orthonormal basis for qubit j. Like product states, classically correlated states can be diagonalized with a depth-one circuit composed of one-body unitaries. Hence (C1) and (C2) are efficient for such states. However, the complexity of eigenvalue readout depends on the $\{p_{\vec{z}}\}$ distribution; if it is high entropy then eigenvalue readout can scale exponentially.

Finally, we consider pure states of the form $\rho = |\psi\rangle\langle\psi|$. For such states, eigenvalue readout (C3) is efficient because N_{readout} can be chosen to be independent of n, as we noted earlier for the example of pure product states.

Next we argue that the gate complexity of the diagonalizing unitary, (C1), is efficient. The argument is simply that VQSD takes the state ρ as its input, and ρ must have been prepared on a quantum computer. Let V be the unitary that was used to prepare $|\psi\rangle = V|\vec{0}\rangle$ on the quantum computer. For large n, V must have been efficient to implement, otherwise the state $|\psi\rangle$ could not have been prepared. Note that $V^\dagger$, which is constructed from V by reversing the order of the gates and adjointing each gate, can be used to diagonalize ρ. Because V is efficiently implementable, then $V^\dagger$ is also efficiently implementable. Hence, ρ can be efficiently diagonalized. A subtlety is that one must compile $V^\dagger$ into one's ansatz, such as the ansatz in Fig. 2.2. Fortunately, the overhead needed to compile $V^\dagger$ into our ansatz grows (at worst) only linearly in n. An explicit construction for compiling $V^\dagger$ into our ansatz is as follows. Any one-qubit gate directly translates without overhead into our ansatz, while any two-qubit gate can be compiled using a linear number of swap gates to make the qubits of interest to be nearest neighbors, then performing the desired two-qubit gate, and finally using a linear number of swap gates to move the qubits back to their original positions.

Let us now consider the complexity (C2) of searching for U. Since there are a linear number of parameters in each layer, and p needs only to grow polynomially in n, then the total number

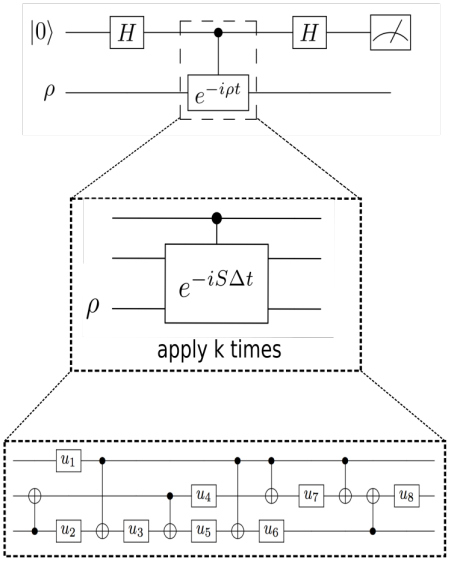


Figure 2.13: Circuit for our qPCA implementation. Here, the eigenvalues of a one-qubit pure state ρ are estimated to a single digit of precision. We use k copies of ρ to approximate $C_{V(t)}$ by applying the controlled-exponential-swap operator k times for a time period $\Delta t = t/k$. The bottom panel shows our compilation of the controlled-exponential-swap gate into one- and two-qubit gates.

of parameters grows only polynomially in n. But this does not guarantee that we can efficiently minimize the cost function, since the landscape is non-convex. In general, search complexity for problems such as this remains an open problem. Hence, we cannot make a general statement about (C2) for pure states.

2.12 Implementation of qPCA

In the main text we compared VQSD to the qPCA algorithm. Here we give further details on our implementation of qPCA. Let us first give an overview of qPCA.

2.12.1 Overview of qPCA

The qPCA algorithm exploits two primitives: quantum phase estimation and density matrix exponentiation. Combining these two primitives allows one to estimate the eigenvalues and prepare the eigenvectors of a state ρ.

Density matrix exponentiation refers to generating the unitary $V(t) = e^{-i\rho t}$ for a given state

ρ and arbitrary time t. For qPCA, one actually needs to apply the controlled-$V(t)$ gate ($C_{V(t)}$). Namely, in qPCA, the $C_{V(t)}$ gate must be applied for a set of times, $\{t, 2t, 2^2t, ..., 2^xt\}$, as part of the phase-estimation algorithm. Here we define $t_{\max} := 2^x t$.

Ref. [16] noted that $V(t)$ can be approximated with a sequence of k exponential swap operations between a target state σ and k copies of ρ. That is, let S_{JK} be the swap operator between systems J and K, and let σ and $\rho^{\otimes k}$ be states on systems A and $B = B_1...B_r$, respectively. Then one performs the transformation

$$\tau_{AB} = \sigma \otimes (\rho^{\otimes k}) \quad \rightarrow \quad \tau'_{AB} = W(\sigma \otimes (\rho^{\otimes k}))W^\dagger \,, \tag{2.47}$$

where

$$W = U_{AB_k} \cdots U_{AB_1}, \quad \text{and} \quad U_{JK} = e^{-iS_{JK}\Delta t} \,. \tag{2.48}$$

The resulting reduced state is

$$\tau'_A = \mathrm{Tr}_B(\tau'_{AB}) \approx V(t)\rho V(t)^\dagger \tag{2.49}$$

where $t = k\Delta t$. Finally, by turning each U_{JK} in (2.48) into a controlled operation:

$$C_{U_{JK}} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{-iS_{JK}\Delta t} \,, \tag{2.50}$$

and hence making W controlled, one can then construct an approximation of $C_{V(t)}$.

If one chooses the input state for quantum phase estimation to be $\rho = \sum_{\vec{z}} \lambda_{\vec{z}} |v_{\vec{z}}\rangle\langle v_{\vec{z}}|$ itself, then the final state becomes

$$\sum_{\vec{z}} \lambda_{\vec{z}} |v_{\vec{z}}\rangle\langle v_{\vec{z}}| \otimes |\hat{\lambda}_{\vec{z}}\rangle\langle \hat{\lambda}_{\vec{z}}| \tag{2.51}$$

where $\hat{\lambda}_{\vec{z}}$ is a binary representation of an estimate of the corresponding eigenvalue $\lambda_{\vec{z}}$. One can then sample from the state in (2.51) to characterize the eigenvalues and eigenvectors.

The approximation of $V(t)$ in (2.49) can be done with accuracy ϵ provided that one uses $O(t^2\epsilon^{-1})$ copies of ρ. The time $t_{\max}$ needed for quantum phase estimation to achieve accuracy ϵ is $t_{\max} = O(\epsilon^{-1})$. Hence, with qPCA, the eigenvalues and eigenvectors can be obtained with accuracy ϵ provided that one uses $O(\epsilon^{-3})$ copies of ρ.

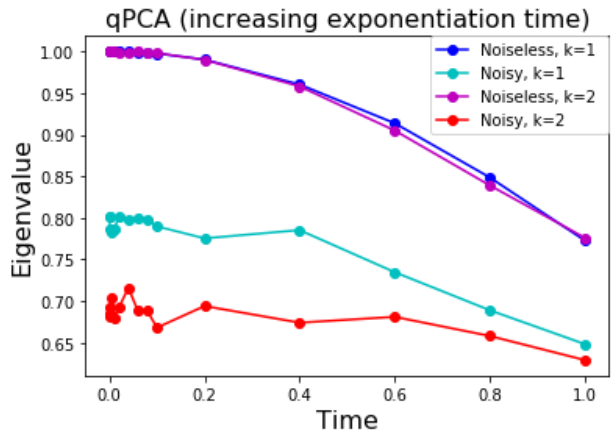


Figure 2.14: The largest inferred eigenvalue for the one-qubit pure state $\rho = |+\rangle\langle+|$ versus application time of unitary $e^{-i\rho t}$, for our implementation of qPCA on Rigetti's noisy and noiseless QVMs. Curves are shown for $k = 1$ and $k = 2$, where k indicates the number of controlled-exponential-swap operators applied.

2.12.2 Our implementation of qPCA

Figure 2.13 shows our strategy for implementing qPCA on an arbitary one-qubit state ρ. The circuit shown corresponds to the quantum phase estimation algorithm with one bit of precision (i.e., one ancilla qubit). A Hadamard gate is applied to the ancilla qubit, which then acts as the control system for the $C_{V(t)}$ gate, and finally the ancilla is measured in the x-basis. The $C_{V(t)}$ is approximated (as discussed above) with k applications of the controlled-exponential-swap gate.

To implement qPCA, the controlled-exponential-swap gate in (2.50) must be compiled into one- and two-body gates. For this purpose, we used the machine-learning approach from Ref. [50] to obtain a short-depth gate sequence for controlled-exponential-swap. The gate sequence we obtained is shown in Fig. 2.13 and involves 7 CNOTs and 8 one-qubit gates. Most of the one-qubit gates are z-rotations and hence are error-free (implemented via a clock change), including the following

gates:

$$u_1 = u_5 = u_7 = R_z(-(\pi + \Delta t)/2) \tag{2.52}$$

$$u_3 = R_z((\pi - \Delta t)/2) \tag{2.53}$$

$$u_4 = R_z(\Delta t/2) \tag{2.54}$$

$$u_8 = R_z(\pi/2)\,. \tag{2.55}$$

The one-qubit gates that are not z-rotations are:

$$u_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ e^{-i(\pi-\Delta t)/2} & e^{i(\pi+\Delta t)/2} \end{pmatrix} \tag{2.56}$$

$$u_6 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-i(\pi+\Delta t)/2} \\ -i & e^{-i\Delta t/2} \end{pmatrix}. \tag{2.57}$$

We implemented the circuit in Fig. 2.13 using both Rigetti's noiseless simulator, known as the Quantum Virtual Machine (QVM), as well as their noisy QVM that utilizes a noise model of their 8Q-Agave chip. Because the latter is meant to mimic the noise in the 8Q-Agave chip, our qPCA results on the noisy QVM can be compared to our VQSD results on the 8Q-Agave chip in Fig. 2.3. (We remark that lack of availability prevented us from implementing qPCA on the actual 8Q-Agave chip.)

For our implementation, we chose the one-qubit plus state, $\rho = |+\rangle\langle+|$. Implementations were carried out using both one and two controlled-exponential-swap gates, corresponding to $k = 1$ and $k = 2$. The time t for which the unitary $e^{-i\rho t}$ was applied was increased.

Figure 2.14 shows the raw data, i.e., the largest inferred eigenvalue versus t. In each case, small values of t gave more accurate eigenvalues. In the noiseless case, the eigenvalues of $\rho = |+\rangle\langle+|$ were correctly estimated to be $\approx \{1, 0\}$ already for $k = 1$ and consequently also for $k = 2$. In the noisy case, the eigenvalues were estimated to be $\approx \{0.8, 0.2\}$ for $k = 1$ and $\approx \{0.7, 0.3\}$ for $k = 2$, where we have taken the values for small t. Table 2.3 summarizes the different cases.

Already for the case of $k = 1$, the required resources of qPCA (3 qubits + 7 CNOT gates) for estimating the eigenvalue of an arbitary pure one-qubit state ρ are higher than those of the DIP test

QVM	$k = 1$	$k = 2$
noiseless	$\approx \{1, 0\}$	$\approx \{1, 0\}$
noisy	$\approx \{0.8, 0.2\}$	$\approx \{0.7, 0.3\}$

Table 2.3: Estimated eigenvalues for the $\rho = |+\rangle\langle+|$ state using qPCA on both the noiseless and the noisy QVMs of Rigetti.

(2 qubits + 1 CNOT gate) for the same task. Consequently, the DIP test yields more accurate results as can be observed by comparing Fig. 2.3 to Fig. 2.14. Increasing the number of copies to $k = 2$ only decreases the accuracy of the estimation, since the $C_{V(t)}$ gate is already well approximated for short application times t when $k = 1$ in the noiseless case. Thus, increasing the number of copies does not offer any improvement in the noiseless case, but instead leads to poorer estimation performance in the noisy case. This can be seen for the $k = 2$ case (see Fig. 2.14 and Table 2.3), due to the doubled number of required CNOT gates relative to $k = 1$.

2.13 Circuit derivation

2.13.1 DIP test

Here we prove that the circuit in Fig. 2.15(a) computes $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$ for any two density matrices σ and τ.

Let σ and τ be states on the n-qubit systems A and B, respectively. Let $\omega_{AB} = \sigma \otimes \tau$ denote the initial state. The action of the CNOTs in Fig. 2.15(a) gives the state

$$\omega'_{AB} = \sum_{\vec{z},\vec{z'}} X^{\vec{z}} \sigma X^{\vec{z'}} \otimes |\vec{z}\rangle\langle\vec{z}|\tau|\vec{z'}\rangle\langle\vec{z'}|, \tag{2.58}$$

where the notation $X^{\vec{z}}$ means $X^{z_1} \otimes X^{z_2} \otimes \cdots \otimes X^{z_n}$. Partially tracing over the B system gives

$$\omega'_A = \sum_{\vec{z}} \tau_{\vec{z},\vec{z}} X^{\vec{z}} \sigma X^{\vec{z}}, \tag{2.59}$$

where $\tau_{\vec{z},\vec{z}} = \langle\vec{z}|\tau|\vec{z}\rangle$. The probability for the all-zeros outcome is then

$$\langle\vec{0}|\omega'_A|\vec{0}\rangle = \sum_{\vec{z}} \tau_{\vec{z},\vec{z}} \langle\vec{0}|X^{\vec{z}} \sigma X^{\vec{z}}|\vec{0}\rangle = \sum_{\vec{z}} \tau_{\vec{z},\vec{z}} \sigma_{\vec{z},\vec{z}}, \tag{2.60}$$

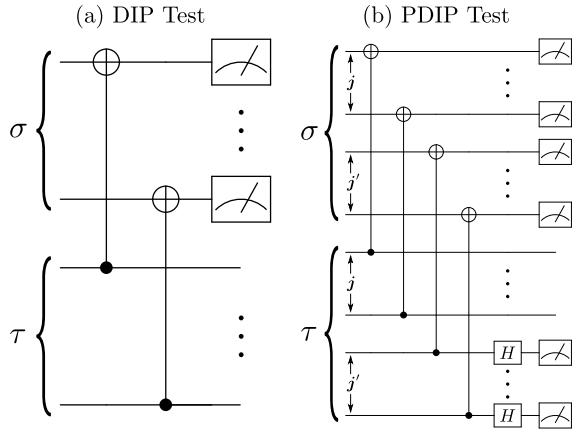


Figure 2.15: Test circuits used to compute the cost function in VQSD. (a) DIP test (b) PDIP test. (These circuits appear in Fig. 2.5 and are also shown here for the reader's convenience.)

which follows because $X^{\vec{z}}|\vec{0}\rangle = |\vec{z}\rangle$. Hence the probability for the all-zeros outcome is precisely the diagonalized inner product, $\mathrm{Tr}(\mathcal{Z}(\sigma)\mathcal{Z}(\tau))$. Note that in the special case where $\sigma = \tau = \tilde{\rho}$, we obtain the sum of the squares of the diagonal elements, $\sum_{\vec{z}} \tilde{\rho}^2_{\vec{z},\vec{z}} = \mathrm{Tr}(\mathcal{Z}(\tilde{\rho})^2)$.

2.13.2 PDIP test

We prove that the circuit in Fig. 2.15(b) computes $\mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\sigma)\mathcal{Z}_{\vec{j}}(\tau))$ for a given set of qubits $\vec{j}$.

Let $\vec{j}'$ denote the complement of $\vec{j}$. Let σ and τ, respectively, be states on the n-qubit systems $A = A_{\vec{j}\vec{j}'}$ and $B = B_{\vec{j}\vec{j}'}$. The initial state $\omega_{AB} = \sigma \otimes \tau$ evolves, under the action of the CNOTs associated with the DIP Test and then tracing over the control systems, to

$$\omega'_{AB_{\vec{j}'}} = \sum_{\vec{z}} (X^{\vec{z}} \otimes I)\sigma(X^{\vec{z}} \otimes I) \otimes \mathrm{Tr}_{B_{\vec{j}}}((|\vec{z}\rangle\langle\vec{z}| \otimes I)\tau), \tag{2.61}$$

where $X^{\vec{z}}$ and $|\vec{z}\rangle\langle\vec{z}|$ act non-trivially only on the $\vec{j}$ subsystems of A and B, respectively. Measuring system $A_{\vec{j}}$ and obtaining the all-zeros outcome would leave systems $A_{\vec{j}'}B_{\vec{j}'}$ in the (unnormalized) conditional state:

$$\mathrm{Tr}_{A_{\vec{j}}}((|\vec{0}\rangle\langle\vec{0}| \otimes I)\omega'_{AB_{\vec{j}'}}) = \sum_{\vec{z}} \sigma^{\vec{z}}_{\vec{j}'} \otimes \tau^{\vec{z}}_{\vec{j}'}, \tag{2.62}$$

where $\sigma^{\vec{z}}_{\vec{j}'} := \mathrm{Tr}_{A_{\vec{j}}}((|\vec{z}\rangle\langle\vec{z}| \otimes I)\sigma)$ and $\tau^{\vec{z}}_{\vec{j}'} := \mathrm{Tr}_{B_{\vec{j}}}((|\vec{z}\rangle\langle\vec{z}| \otimes I)\tau)$. Finally, computing the expectation

value for the swap operator (via the Destructive Swap Test) on the state in (2.62) gives

$$\sum_{\vec{z}} \mathrm{Tr}((\sigma^{\vec{z}}_{\vec{j}'} \otimes \tau^{\vec{z}}_{\vec{j}'})S) = \sum_{\vec{z}} \mathrm{Tr}(\sigma^{\vec{z}}_{\vec{j}'}\tau^{\vec{z}}_{\vec{j}'}) = \mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\sigma)\mathcal{Z}_{\vec{j}}(\tau))\,. \tag{2.63}$$

The last equality can be verified by noting that $\mathcal{Z}_{\vec{j}}(\sigma) = \sum_{\vec{z}} |\vec{z}\rangle\langle\vec{z}| \otimes \sigma^{\vec{z}}_{\vec{j}'}$ and $\mathcal{Z}_{\vec{j}}(\tau) = \sum_{\vec{z}} |\vec{z}\rangle\langle\vec{z}| \otimes \tau^{\vec{z}}_{\vec{j}'}$. Specializing (2.63) to $\sigma = \tau = \tilde{\rho}$ gives the quantity $\mathrm{Tr}(\mathcal{Z}_{\vec{j}}(\tilde{\rho})^2)$.

2.14 Proof of local dephasing channel bound

In this section we prove Eq. (2.40). Let $H_2(\sigma) = -\log_2[\mathrm{Tr}(\sigma^2)]$ be the Renyi entropy of order two. Then, noting that $H_2(\sigma) \leq H(\sigma)$, we have

$$\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) = 2^{-H_2(\mathcal{Z}_j(\tilde{\rho}))} \geq 2^{-H(\mathcal{Z}_j(\tilde{\rho}))}\,. \tag{2.64}$$

Next, let A denote qubit j, and let B denote all the other qubits. This allows us to write $\rho = \rho_{AB}$ and $\tilde{\rho} = \tilde{\rho}_{AB}$. Let C be a purifying system such that ρ_{ABC} and $\tilde{\rho}_{ABC}$ are both pure states. Then we have

$$H(\mathcal{Z}_j(\tilde{\rho})) = H(\mathcal{Z}_j(\tilde{\rho}_{AB})) \tag{2.65}$$

$$= H(\mathcal{Z}_j(\tilde{\rho}_{AC})) \tag{2.66}$$

$$\leq H(\mathcal{Z}_j(\tilde{\rho}_A)) + H(\tilde{\rho}_C) \tag{2.67}$$

where the inequality in (2.67) used the subadditivity of von Neumann entropy. Finally, note that

$$H(\tilde{\rho}_C) = H(\tilde{\rho}_{AB}) = H(\rho_{AB}) = H(\rho) \tag{2.68}$$

and $H(\mathcal{Z}_j(\tilde{\rho}_A)) \leq 1$, which gives

$$H(\mathcal{Z}_j(\tilde{\rho})) \leq 1 + H(\rho)\,. \tag{2.69}$$

Substituting (2.69) into (2.64) gives

$$\mathrm{Tr}(\mathcal{Z}_j(\tilde{\rho})^2) \geq 2^{-1-H(\rho)}\,, \tag{2.70}$$

and (2.40) follows from $H(\rho) \leq \log_2 r$.

CHAPTER 3

QUANTUM-ASSISTED QUANTUM COMPILING

3.1 Introduction

In classical computing, a compiler is a program that converts instructions into assembly language so that they can be read and executed by a computer. Similarly, a quantum compiler would take a high-level algorithm and convert it into a lower-level form that could be executed on a NISQ device. Already, a large body of literature exists on classical approaches for quantum compiling, e.g., using temporal planning [67, 68], machine learning [50], and other techniques [69, 70, 71, 72, 73, 74, 75, 76].

A recent exciting idea is to use quantum computers themselves to train parametrized quantum circuits, as proposed in Refs. [77, 78, 30, 79, 80, 81, 82, 83, 84]. The cost function to be minimized essentially defines the application. For example, in the variational quantum eigensolver (VQE) [78] and the quantum approximate optimization algorithm (QAOA) [77], the application is ground state preparation, and hence the cost is the expectation value of the associated Hamiltonian. Another example is training error-correcting codes [30], where the cost is the average code fidelity. In light of these works, it is natural to ask: what is the relevant cost function for the application of quantum compiling?

In this chapter, we introduce quantum-assisted quantum compiling (QAQC). The goal of QAQC is to compile a (possibly unknown) target unitary to a trainable quantum gate sequence. A key feature of QAQC is the fact that the cost is computed directly on the quantum computer. This leads to an exponential speedup (in the number of qubits involved in the gate sequence) over classical methods to compute the cost, since classical simulation of quantum dynamics is exponentially slower than quantum simulation. Consequently, one should be able to optimally compile larger-scale gate sequences using QAQC, whereas classical approaches to optimal quantum compiling

will be limited to smaller gate sequences.[1]

We carefully define a cost function for QAQC that satisfies the following criteria:

1. It is faithful (vanishing if and only if the compilation is exact);

2. It is efficient to compute on a quantum computer;

3. It has an operational meaning;

4. It scales well with the size of the problem.

A potential candidate for a cost function satisfying these criteria is the Hilbert-Schmidt inner product between a target unitary U and a trainable unitary V:

$$\langle V, U\rangle = \mathrm{Tr}(V^\dagger U). \tag{3.1}$$

It turns out, however, that this cost function does not satisfy the last criterion. We thus use Eq. (3.1) only for small-scale problems. For general, large-scale problems, we define a cost function satisfying all criteria. This cost involves a weighted average of the global overlap in (3.1) with localized overlaps, which quantify the overlap between U and V with respect to individual qubits.

We prove that computing our cost function is DQC1-hard, where DQC1 is the class of problems that can be efficiently solved in the one-clean-qubit model of computation [7]. Since DQC1 is classically hard to simulate [85], this implies that no classical algorithm can efficiently compute our cost function. We remark that an alternative cost function might be a worst-case distance measure (such as diamond distance), but such measures are known to be QIP-complete [86] and hence would violate criterion 2 in our list above. In this sense, our cost function appears to be ideal.

Furthermore, we present novel short-depth quantum circuits for efficiently computing the terms in our cost function. Our circuits achieve short depth by avoiding implementing controlled versions

[1] We note that classical compilers may be applied to large-scale quantum algorithms, but they are limited to local compiling. We thus emphasize the distinction between translating the algorithm to the native alphabet with simple, local compiling and optimal compiling. Local compiling may reach partial optimization but in order to discover the shortest circuit one may need to use a holistic approach, where the entire algorithm is considered, which requires a quantum computer for compiling.

of U and V, and by implementing U and V in parallel. We also present, in Section 3.15, circuits that compute the gradient of our cost function. One such circuit is a generalization of the well-known Power of One Qubit [7] that we call the Power of Two Qubits.

As a proof-of-principle, we implement QAQC on both IBM's and Rigetti's quantum computers, and we compile various one-qubit gates to the native gate alphabets used by these hardwares. To our knowledge, this is the first compilation of a target unitary with cost evaluation on actual NISQ hardware. In addition, we successfully implement QAQC on both a noiseless and noisy simulator for problems as large as 9-qubit unitaries. These larger scale implementations illustrate the scalability of our cost function, and in the case of the noisy simulator, show a somewhat surprising resilience to noise.

In what follows, we first discuss several applications of interest for QAQC. Section 3.3 provides a general outline of the QAQC algorithm. Section 3.4 presents our short-depth circuits for cost evaluation on a quantum computer. Section 3.5 states that our cost function is classically hard to simulate. Sections 3.6 and 3.7, respectively, present small-scale and larger-scale implementations of QAQC.

3.2 Applications of QAQC

Figure 3.1 illustrates four potential applications of QAQC. Suppose that there exists a quantum algorithm to perform some task, but its associated gate sequence is longer than desired. As shown in Fig. 3.1(a), it is possible to use QAQC to shorten the gate sequence by accounting for the NISQ constraints of the specific computer. This depth compression goes beyond the capabilities of classical compilers.

As a simple example, consider the quantum Fourier transform on n qubits. Its textbook algorithm is written in terms of Hadamard gates and controlled-rotation gates [11], which may need to be compiled into the native gate alphabet. The number of gates in the textbook algorithm is $O(n^2)$, so one could use a classical compiler to locally compile each gate. But this could lead to a sub-optimal depth since the compilation starts from the textbook structure. In contrast, QAQC is unbiased with

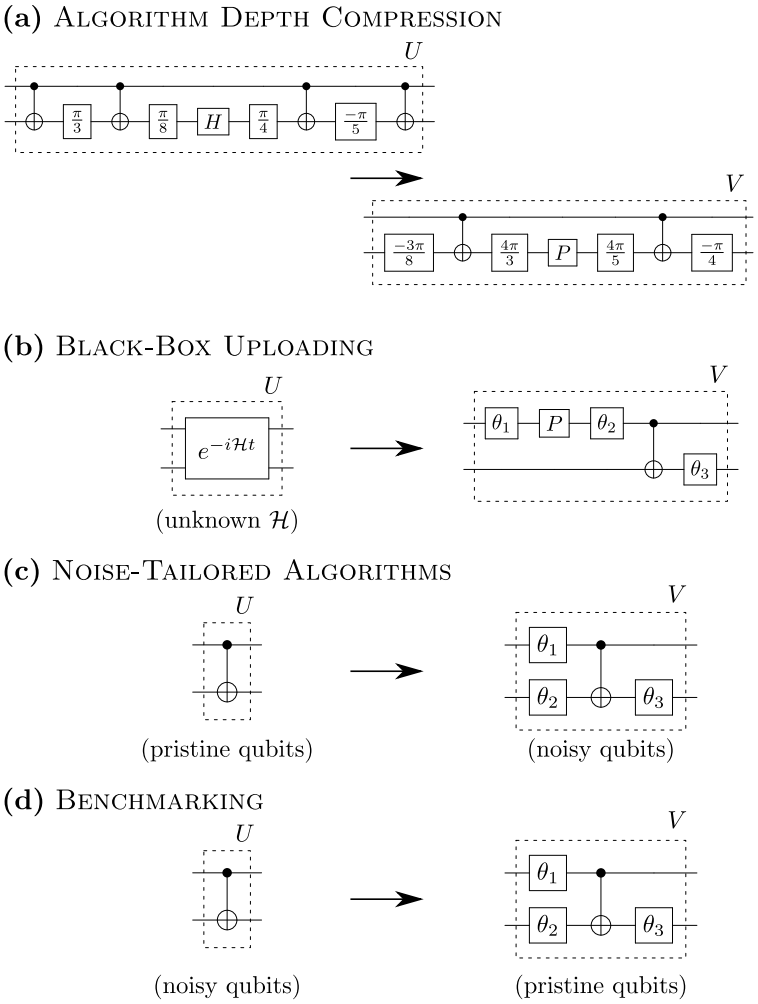


Figure 3.1: Potential applications of QAQC. Here, $-\boxed{\theta}-$ denotes the z-rotation gate $R_z(\theta)$, while $-\boxed{P}-$ represents the $\pi/2$-pulse given by the x-rotation gate $R_x(\pi/2)$. Both gates are natively implemented on commercial hardware [2, 3]. (a) Compressing the depth of a given gate sequence U to a shorter-depth gate sequence V in terms of native hardware gates. (b) Uploading a black-box unitary. The black box could be an analog unitary $U = e^{-i\mathcal{H}t}$, for an unknown Hamiltonian $\mathcal{H}$, that one wishes to convert into a gate sequence to be run on a gate-based quantum computer. (c) Training algorithms in the presence of noise to learn noise-resilient algorithms (e.g., via gates that counteract the noise). Here, the unitary U is performed on high-quality, pristine qubits and V is performed on noisy ones. (d) Benchmarking a quantum computer by compiling a unitary U on noisy qubits and learning the gate sequence V on high-quality qubits.

respect to the structure of the gate sequence, taking a holistic approach to compiling as opposed to a local one. Hence, in principle, it can learn the optimal gate sequence for given hardware. Note that classical compilers cannot take this holistic approach for large n due to the exponential scaling of the matrix representations of the gates.

Alternatively, consider the problem of simulating the dynamics of a given quantum system with an unknown Hamiltonian $\mathcal{H}$ (via $e^{-i\mathcal{H}t}$) on a quantum computer. We call this problem black-box uploading because by simulating the black-box, i.e., the unitary $e^{-i\mathcal{H}t}$, we are "uploading" the unitary onto the quantum computer. This scenario is depicted in Fig. 3.1(b). QAQC could be used to convert an analog black-box unitary into a gate sequence on a digital quantum computer.

Finally, we highlight two additional applications that are the opposites of each other. These two applications can be exploited when the quantum computer has some pristine qubits (qubits with low noise) and some noisy qubits. We emphasize that, in this context, "noisy qubits" refers to coherent noise such as systematic gate biases, where the gate rotation angles are biased in a particular direction. In contrast, we consider incoherent noise (e.g., T_1 and T_2 noise) later in this article, see Section 3.7.2.

Consider Fig. 3.1(c). Here, the goal is to implement a CNOT gate on two noisy qubits. Due to the noise, to actually implement a true CNOT, one has to physically implement a dressed CNOT, i.e., a CNOT surrounded by one-qubit unitaries. QAQC can be used to learn the parameters in these one-qubit unitaries. By choosing the target unitary U to be a CNOT on a pristine (i.e., noiseless) pair of qubits, it is possible to learn the unitary V that needs to be applied to the noisy qubits in order to effectively implement a CNOT. We call this application noise-tailored algorithms, since the learned algorithms are robust to the noise process on the noisy qubits.

Figure 3.1(d) depicts the opposite process, which is benchmarking. Here, the unitary U acts on a noisy set of qubits, and the goal is to determine what the equivalent unitary V would be if it were implemented on a pristine set of qubits. This essentially corresponds to learning the noise model, i.e., benchmarking the noisy qubits.

3.3 The QAQC algorithm

3.3.1 Approximate compiling

The goal of QAQC is to take a (possibly unknown) unitary U and return a gate sequence V, executable on a quantum computer, that has approximately the same action as U on any given input state (up

to possibly a global phase factor). The notion of approximate compiling [87, 88, 89, 90, 91, 92] requires an operational figure-of-merit that quantifies how close the compilation is to exact. A natural candidate is the probability for the evolution under V to mimic the evolution under U. Hence, consider the overlap between $|\psi(U)\rangle \coloneqq U|\psi\rangle$ and $|\psi(V)\rangle \coloneqq V|\psi\rangle$, averaged over all input states $|\psi\rangle$. This is the fidelity averaged over the Haar distribution,

$$\overline{F}(U,V) \coloneqq \int_{\psi} |\langle\psi(V)|\psi(U)\rangle|^2 \, \mathrm{d}\psi \, . \tag{3.2}$$

We call V an *exact compilation of* U if $\overline{F}(U,V) = 1$. If $\overline{F}(U,V) \geq 1-\varepsilon$, where $\varepsilon \in [0,1]$, then we call V an *ε-approximate compilation of* U, or simply an *approximate compilation of* U.

As we will see, the quantity $\overline{F}(U,V)$ has a connection to our cost function, defined below, and hence our cost function has operational relevance to approximate compiling. Minimizing our cost function is related to maximizing $\overline{F}(U,V)$, and thus is related to compiling to a better approximation.

QAQC achieves approximate compiling by training a gate sequence V of a fixed length L, which may even be shorter than the length required to exactly compile U. As one increases L, one can further minimize our cost function. The length L can therefore be regarded as a parameter that can be tuned to obtain arbitrarily good approximate compilations of U.

3.3.2 Discrete and continuous parameters

The gate sequence V should be expressed in terms of the native gates of the quantum computer being used. Consider an alphabet $\mathcal{A} = \{G_k(\alpha)\}_k$ of gates $G_k(\alpha)$ that are native to the quantum computer of interest. Here, $\alpha \in \mathbb{R}$ is a continuous parameter, and k is a discrete parameter that identifies the type of gate and which qubits it acts on. For a given quantum computer, the problem of compiling U to a gate sequence of length L is to determine

$$(\vec{\alpha}_{\mathrm{opt}}, \vec{k}_{\mathrm{opt}}) \coloneqq \arg\min_{(\vec{\alpha},\vec{k})} C(U, V_{\vec{k}}(\vec{\alpha})), \tag{3.3}$$

where

$$V_{\vec{k}}(\vec{\alpha}) = G_{k_L}(\alpha_L) G_{k_{L-1}}(\alpha_{L-1}) \cdots G_{k_1}(\alpha_1) \tag{3.4}$$

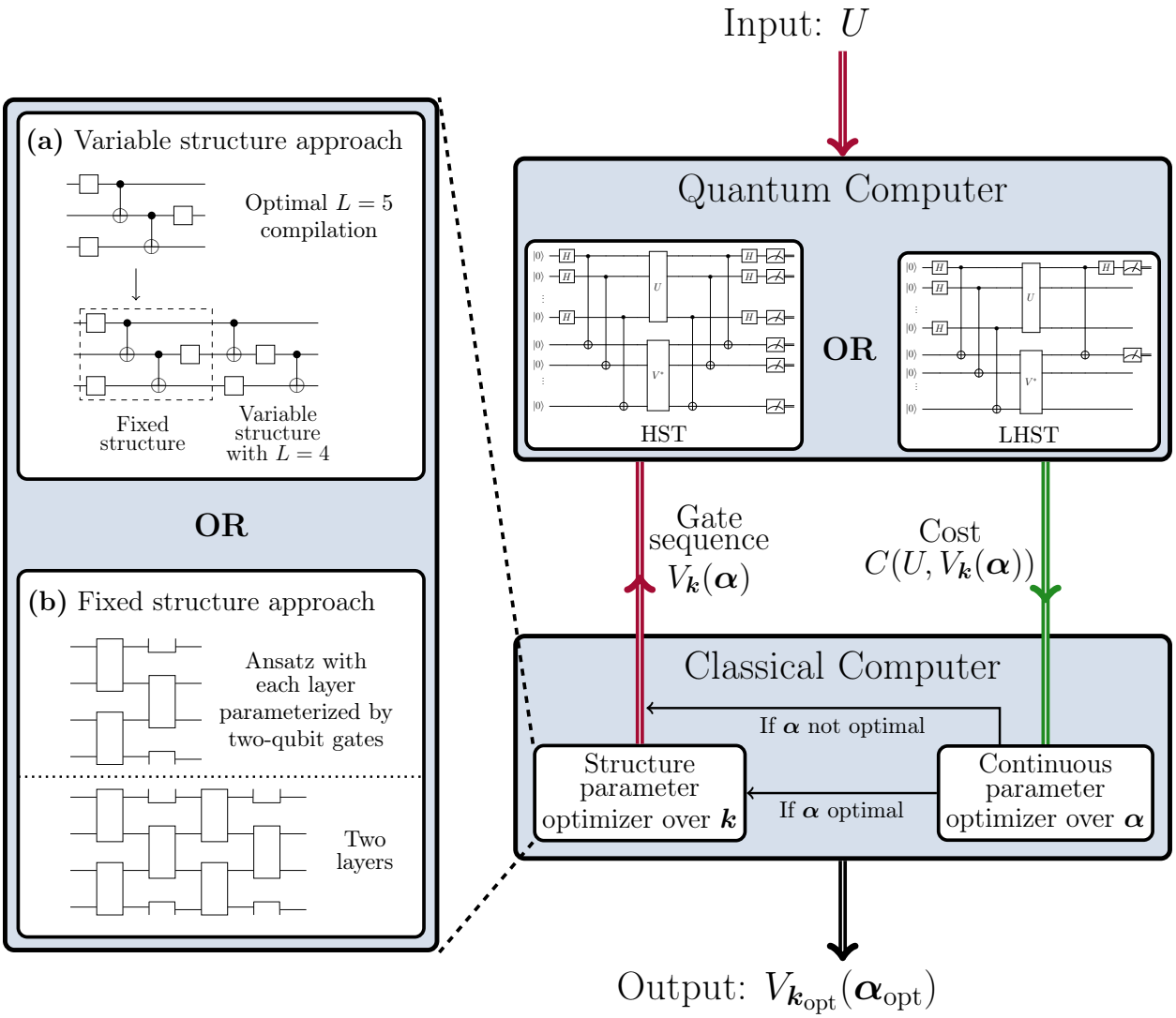


Figure 3.2: Outline of our variational hybrid quantum-classical algorithm, in which we optimize over gate structures and continuous gate parameters in order to perform QAQC for a given input unitary U. We take two approaches towards structure optimization: **(a)** For small problem sizes, we allow the gate structure to vary for a given gate sequence length L, which in general leads to an approximate compilation of U. To obtain a better approximate compilation, the best structure obtained can be concatenated with a new sequence of a possibly different length, whose structure can vary. For each iteration of the continuous parameter optimization, we calculate the cost using the Hilbert-Schmidt Test (HST); see Sec. 3.4.1. **(b)** For large problem sizes, we fix the gate structure using an ansatz consisting of layers of two-qubit gates. By increasing the number of layers, we can obtain better approximate compilations of U. For each iteration of the continuous parameter optimization, we calculate the cost using the Local Hilbert-Schmidt Test (LHST); see Sec. 3.4.2.

is the trainable unitary. Here, $V_{\vec{k}}(\vec{\alpha})$ is a function of the sequence $\vec{k} = (k_1, \ldots, k_L)$ of parameters describing which gates from the native gate set are used and of the continuous parameters $\vec{\alpha} = (\alpha_1, \ldots, \alpha_L)$ associated with each gate. The function $C(U, V_{\vec{k}}(\vec{\alpha}))$ is the cost, which quantifies how close the trained unitary is to the target unitary. We define the cost below to have the properties: $0 \leq C(U, V) \leq 1$ for all unitaries U and V, and $C(U, V) = 0$ if and only if $U = V$ (possibly up to a global phase factor).

The optimization in (3.3) contains two parts: discrete optimization over the finite set of gate structures parameterized by $\vec{k}$, and continuous optimization over the parameters $\vec{\alpha}$ characterizing the gates within the structure. Our quantum-classical hybrid strategy to perform the optimization in (3.3) is illustrated in Fig. 3.2. In the next subsection, we present a general, ansatz-free approach to optimizing our cost function, which may be useful for systems with a small number of qubits. In the subsection following that, we present an ansatz-based approach that would allow the extension to larger system sizes. In each case, we perform the continuous parameter optimization using gradient-free methods as described in Section 3.14. We also discuss a method for gradient-based continuous parameter optimization in Section 3.15.

3.3.3 Small problem sizes

Suppose U and V act on a d-dimensional space of n qubits, so that $d = 2^n$. To perform the continuous parameter optimization in (3.3), we define the cost function

$$\begin{aligned} C_{\mathrm{HST}}(U, V) &:= 1 - \frac{1}{d^2} |\langle V, U \rangle|^2 \\ &= 1 - \frac{1}{d^2} |\mathrm{Tr}(V^\dagger U)|^2, \end{aligned} \tag{3.5}$$

where HST stands for "Hilbert-Schmidt Test" and refers to the circuit used to evaluate the cost, which we introduce in Sec. 3.4.1. Note that the quantity $\frac{1}{d^2} |\langle V, U \rangle|^2$ is simply the fidelity between the pure states obtained by applying U and V to one half of a maximally entangled state. Consequently, it has an operational meaning in terms of $\overline{F}(U, V)$. Indeed, it can be shown [93, 94] that

$$C_{\mathrm{HST}}(U, V) = \frac{d+1}{d} \left(1 - \overline{F}(U, V)\right) . \tag{3.6}$$

Also note that for any two unitaries U and V, $C_{\mathrm{HST}}(U, V) = 0$ if and only if U and V differ by a global phase factor, i.e., $V = e^{i\varphi}U$ for some $\varphi \in \mathbb{R}$. By minimizing C_{HST}, we thus learn an equivalent unitary V up to a global phase.

Now, to perform the optimization over gate structures in (3.3), one strategy is to search over all possible gate structures for a gate sequence length L, which can be allowed to vary during the optimization. As the set of gate structures grows exponentially with the number of gates L, such a brute force search over all gate structures in order to obtain the best one is intractable in general. To efficiently search through this exponentially large space, we adopt an approach based on simulated annealing. (An alternative approach is genetic optimization, which has been implemented previously to classically optimize quantum gate sequences [95].)

Our simulated annealing approach starts with a random gate structure, then performs continuous optimization over the parameters $\vec{\alpha}$ that characterize the gates in order to minimize the cost function. We then perform a structure update that involves randomly replacing a subset of gates in the sequence with new gates (which can be done in a way such that the sequence length can increase or decrease) and re-optimizing the cost function over the continuous parameters $\vec{\alpha}$. If this structure change produces a lower cost, then we accept the change. If the cost increases, then we accept the change with probability decreasing exponentially in the magnitude of the cost difference. We iterate this procedure until the cost converges or until a maximum number of iterations is reached.

With a fixed gate sequence length L, the approach outlined above will in general lead to an approximate compilation of U, which in many cases is sufficient. One strategy for obtaining better and better approximate compilations of U is a layered approach illustrated in Fig. 3.2(a). In this approach, we consider a particular gate sequence length L and perform the full structure optimization, as outlined above, to obtain an (approximate) length-L compilation of U. The optimal gate sequence structure thus obtained can then be concatenated with a new sequence of a possibly different (but fixed) length, whose structure can vary. By performing the continuous parameter optimization over the entire longer gate sequence, and performing the structure optimization over the new additional segment of the gate sequence, we can obtain a better approximate compilation

of U. Iterating this procedure can then lead to increasingly better approximate compilations of U.

3.3.4 Large problem sizes

We emphasize two potential issues with scaling the above approach to large problem sizes.

First, one may want a guarantee that there exists an exact compilation of U within a polynomial size search space for V. When performing full structure optimization, as above, the search space size grows exponentially in the length L of the gate sequence. This implies that the search space size grows exponentially in n, if one chooses L to grow polynomially in n. Indeed, one would typically require L to grow polynomially in n if one is interested in exact compilation, since the number of gates in U itself grows polynomially in n for many applications. (Note that this issue arises if one insists on exact, instead of approximate, compiling.)

Second, and arguably more importantly, the cost $C_{\mathrm{HST}}(U, V)$ is exponentially fragile. The inner product between U and V will be exponentially suppressed for random choices of V, which means that $C_{\mathrm{HST}}(U, V)$ will be very close to one for most unitaries V. Hence, for random unitaries V, the number of calls to the quantum computer needed to resolve differences in the cost $C_{\mathrm{HST}}(U, V)$ to a given precision will grow exponentially.

The first issue can be addressed with an efficiently parameterized ansatz for V. With an ansatz, only the continuous parameters $\vec{\alpha}$ need to be optimized in V. The $\vec{k}$ parameters are fixed, which means that structure updates are not required. This fixed structure approach is depicted in Fig. 3.2(b). One can choose an ansatz such that the number of parameters needed to represent the target unitary U is only a polynomial function of n. Hence, one should allow the ansatz $A(U)$ to be application specific, i.e., to be a function of U. As an example, if $U = e^{-i\mathcal{H}t}$ for a local Hamiltonian $\mathcal{H}$, one could choose the ansatz to involve a polynomial number of local interactions. Due to the application-specific nature of the ansatz, the problem is a complex one, hence we leave the issue of finding efficient ansatzes for future work.

Nevertheless, we show a concrete example of a potential ansatz for V in Fig. 3.3. The ansatz is defined by a number ℓ of layers, with each layer being a gate sequence of depth two consisting

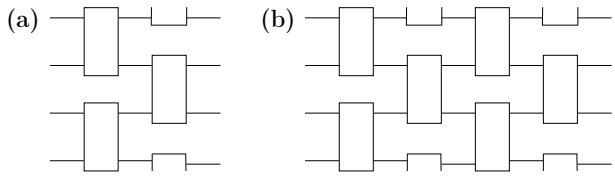


Figure 3.3: **(a)** One layer of the ansatz for the trainable unitary V in the case of four qubits. The gate sequence in the layer consists of a two-qubit gate acting on the first and second qubits, the third and fourth qubits, the second and third qubits, and the first and fourth qubits. **(b)** The full ansatz defining the trainable unitary V consists of a particular number ℓ of the layer in (a). Shown is two layers in the case of four qubits.

of two-qubit gates acting on neighboring qubits. Consider the following argument. In QAQC, the unitary U to be compiled is executed on the quantum computer, so it must be efficiently implementable, i.e., the gate count is polynomial in n. Next, note that the gate sequence used to implement U can be compiled into in the ansatz in Fig. 3.3 with only polynomial overhead. This implies that the ansatz in Fig. 3.3 could exactly describe U in only a polynomial number of layers and would hence eliminate the need to search through an exponentially large space. We remark that the ansatz in Fig. 3.3 may be particularly useful for applications involving compiling quantum simulations of physically relevant systems, as the structure resembles that of the Suzuki-Trotter decomposition [96] for nearest-neighbor Hamiltonians.

Let us now consider the second issue mentioned above: the exponentially suppressed inner product between U and V for large n. To address this, we propose an alternative cost function involving a weighted average between the function in (3.5) and a "local" cost function:

$$C_q(U,V) \coloneqq qC_{\mathrm{HST}}(U,V) + (1-q)C_{\mathrm{LHST}}(U,V), \tag{3.7}$$

where $0 \leq q \leq 1$ and

$$C_{\mathrm{LHST}}(U,V) \coloneqq \frac{1}{n}\sum_{j=1}^{n} C^{(j)}_{\mathrm{LHST}}(U,V) = 1 - \overline{F_e}. \tag{3.8}$$

Here, LHST stands for "Local Hilbert-Schmidt Test", referring to the circuit discussed in Sec. 3.4.2 that is used to compute this function. Also, $\overline{F_e} \coloneqq \frac{1}{n}\sum_{j=1}^{n} F_e^{(j)}$, where the quantities $F_e^{(j)}$ are entanglement fidelities (hence the notation F_e) of local quantum channels $\mathcal{E}_j$ defined in Sec. 3.4.2. Hence, $C_{\mathrm{LHST}}(U,V)$ is a sum of local costs, where each local cost is written as a local entanglement fidelity: $C^{(j)}_{\mathrm{LHST}}(U,V) = 1 - F_e^{(j)}$. Expressing the overall cost as sum of local costs is analogous

to what is done in the variational quantum eigensolver [78], where the overall energy is expressed as a sum of local energies. The functions $C^{(j)}_{\mathrm{LHST}}$ are local in the sense that only two qubits need to be measured in order to calculate each one of them. This is unlike the function C_{HST}, whose calculation requires the simultaneous measurement of $2n$ qubits.

The cost function C_q in (3.7) is a weighted average between the "global" cost function C_{HST} and the local cost function C_{LHST}, with q representing the weight given to the global cost function. The weight q can be chosen according to the size of the problem: for a relatively small number of qubits, we would let $q = 1$. As the number of qubits increases, we would slowly decrease q to mitigate the suppression of the inner product between U and V.

To see why C_{LHST} can be expected to deal with the issue of an exponentially suppressed inner product for large n, consider the following example. Suppose the unitary U to be compiled is the tensor product $U = U_1 \otimes U_2 \otimes \cdots \otimes U_n$ of unitaries U_j acting on qubit j, and suppose we take the tensor product $V = V_1 \otimes V_2 \otimes \cdots \otimes V_n$ as the trainable unitary. We get that $C_{\mathrm{HST}}(U, V) = 1 - \prod_{j=1}^{n} r_j$, where $r_j = (1/4)|\mathrm{Tr}(V_j^\dagger U_j)|^2$. Since each r_j will likely be less than one for a random choice of V_j, then their product will be small for large n. Consequently a very large portion of the cost landscape will have $C_{\mathrm{HST}}(U, V) \approx 1$ and hence will have a vanishing gradient. However, the cost function C_{LHST} is defined such that $C_{\mathrm{LHST}}(U, V) = 1 - \frac{1}{n}\sum_{j=1}^{n} r_j$, so that we obtain an average of the r_j quantities rather than a product. Taking the average instead of the product leads to a gradient that is not suppressed for large n.

More generally, for any U and V, the quantity $\overline{F_e}$, which is responsible for the variability in C_{LHST}, can be made non-vanishing by adding local unitaries to V. In particular, for a given U and V, it is straightforward to show that for all $j \in \{1, 2, \ldots, n\}$ there exists a unitary V_j acting on qubit j such that $F_e^{(j)} \geq \frac{1}{4}$ for the gate sequence given by $V' = V_j V$. In other words, there exists a local unitary V_j that can be added to the trainable gate sequence V such that $C^{(j)}_{\mathrm{LHST}}(U, V_j V) \leq \frac{3}{4}$. This implies that, with the appropriate local unitary applied to each qubit at the end of the trainable gate sequence, the local cost function C_{LHST} can always be decreased to no greater than $\frac{3}{4}$. Note that local unitaries cannot be used in this way to decrease the global cost function C_{HST}, i.e., to make

the second term in (3.5) non-vanishing.

Finally, one can show (See Section 3.12) that $C_{\mathrm{LHST}} \geq (1/n) C_{\mathrm{HST}}$. Combining this with Eq. (3.6) gives

$$C_q(U,V) \geq \left(\frac{1-q+nq}{n}\right)\left(\frac{d+1}{d}\right)\left(1-\overline{F}(U,V)\right), \tag{3.9}$$

which implies that

$$\overline{F}(U,V) \geq 1-\left(\frac{n}{1-q+nq}\right)\left(\frac{d}{d+1}\right) C_q(U,V). \tag{3.10}$$

Hence, the cost function C_q retains an operational meaning for the task of approximate compiling, since it provides a bound on the average fidelity between U and V.

3.3.5 Special case of a fixed input state

An important special case of quantum compiling is when the target unitary U happens to appear at the beginning of one's quantum algorithm, and hence the state that one inputs to U is fixed. For many quantum computers, this input state is $|\psi_0\rangle = |0\rangle^{\otimes n}$. We emphasize that many use cases of QAQC do not fall under this special case, since one is often interested in compiling unitaries that do not appear at the beginning of one's algorithm. For example, one may be interested in the optimal compiliation of a controlled-unitary, but such a unitary would never appear at the beginning of an algorithm since its action would be trivial. Nevertheless we highlight this special case because QAQC can potentially be simplified in this case. In addition, this special case was very recently explored in Ref. [52] after the completion of our article.

In this special scenario, a natural cost function would be

$$C_{\text{fixed input}} = 1 - |\langle\psi_0|UV^\dagger|\psi_0\rangle|^2 . \tag{3.11}$$

This could be evaluated on a quantum computer in two possible ways. One way is to apply U and then $V^\dagger$ to the $|\psi_0\rangle$ state and then measure the probability to be in the $|\psi_0\rangle$ state. Another way is to apply U to one copy of $|\psi_0\rangle$ and V to another copy of $|\psi_0\rangle$, and then measure the overlap [50, 55] between these two states.

However, this cost function would not scale well for the same reason discussed above that our $C_{\rm HST}$ cost does not scale well, i.e., its gradient can vanish exponentially. Again, one can fix this issue with a local cost function. Assuming $|\psi_0\rangle = |0\rangle^{\otimes n}$, this local cost can take the form:

$$C^{\rm local}_{\rm fixed\ input} = 1 - \frac{1}{n}\sum_{j=1}^{n} p_0^{(j)}\,, \tag{3.12}$$

where

$$p_0^{(j)} = {\rm Tr}[(|0\rangle\langle 0|_j \otimes I)V^\dagger U|\psi_0\rangle\langle\psi_0|U^\dagger V] \tag{3.13}$$

is the probability to obtain the zero measurement outcome on qubit j for the state $V^\dagger U|\psi_0\rangle$.

We remark that the two cost functions in (3.11) and (3.12) can each be evaluated with quantum circuits on only n qubits. This is in contrast to $C_{\rm HST}$ and $C_{\rm LHST}$, whose evaluation involves quantum circuits with $2n$ qubits (see the next section for the circuits). This reduction in resource requirements is the main reason why we highlight this special case.

3.4 Cost evaluation circuits

In this section, we present short-depth circuits for evaluating the functions in (3.5) and (3.8) and hence for evaluating the overall cost in (3.7). We note that these circuits are also interesting outside of the scope of QAQC, and they likely have applications in other areas.

In addition, in Section 3.15, we present circuits for computing the gradient of the cost function, including a generalization of the Power-of-one-qubit circuit [7] that computes both the real and imaginary parts of $\langle U, V\rangle$.

3.4.1 Hilbert-Schmidt Test

Consider the circuit in Fig. 3.4(a). Below we show that this circuit computes $|{\rm Tr}(V^\dagger U)|^2$, where U and V are n-qubit unitaries. The circuit involves $2n$ qubits, where we call the first (second) n-qubit system A (B).

The first step in the circuit is to create a maximally entangled state between A and B, namely, the state

$$|\Phi^+\rangle_{AB} = \frac{1}{\sqrt{d}} \sum_{\vec{j}} |\vec{j}\rangle_A \otimes |\vec{j}\rangle_B \,, \tag{3.14}$$

where $\vec{j} = (j_1, j_2, ..., j_n)$ is a vector index in which each component j_k is chosen from $\{0, 1\}$. The first two gates in Fig. 3.4(a)—the Hadamard gates and the CNOT gates (which are performed in parallel when acting on distinct qubits)—create the $|\Phi^+\rangle$ state.

The second step is to act with U on system A and with V^* on system B. (V^* is the complex conjugate of V, where the complex conjugate is taken in the standard basis.) Note that these two gates are performed in parallel. This gives the state

$$(U \otimes V^*)|\Phi^+\rangle_{AB} = \frac{1}{\sqrt{d}} \sum_{\vec{j}} U|\vec{j}\rangle_A \otimes V^*|\vec{j}\rangle_B \,. \tag{3.15}$$

We emphasize that the unitary V^* is implemented on the quantum computer, not V itself. (See Section 3.10 for elaboration on this point.)

The third and final step is to measure in the Bell basis. This corresponds to undoing the unitaries (the CNOTs and Hadamards) used to prepare $|\Phi^+\rangle$ and then measuring in the standard basis. At the end, we are only interested in estimating a single probability: the probability for the Bell-basis measurement to give the $|\Phi^+\rangle$ outcome, which corresponds to the all-zeros outcome in the standard basis. The amplitude associated with this probability is

$$\langle\Phi^+|U \otimes V^*|\Phi^+\rangle = \langle\Phi^+|UV^\dagger \otimes I|\Phi^+\rangle \tag{3.16}$$

$$= \frac{1}{d}\mathrm{Tr}(V^\dagger U) \,. \tag{3.17}$$

To obtain the first equality we used the ricochet property:

$$I \otimes X|\Phi^+\rangle = X^T \otimes I|\Phi^+\rangle, \tag{3.18}$$

which holds for any operator X acting on a d-dimensional space. The probability of the $|\Phi^+\rangle$ outcome is then the absolute square of the amplitude, i.e., $(1/d^2)|\mathrm{Tr}(V^\dagger U)|^2$. Hence, this probability

gives us the absolute value of the Hilbert-Schmidt inner product between U and V. We therefore call the circuit in Fig. 3.4(a) the Hilbert-Schmidt Test (HST).

Consider the depth of this circuit. Let $D(G)$ denote the depth of a gate sequence G for a fully-connected quantum computer whose native gate alphabet includes the CNOT gate and the set of all one-qubit gates. Then, for the HST, we have

$$D(\text{HST}) = 4 + \max\{D(U), D(V^*)\}\,. \tag{3.19}$$

The first term of 4 is associated with the Hadamards and CNOTs in Fig. 3.4(a), and this term is negligible when the depth of U or V^* is large. The second term results from the fact that U and V^* are performed in parallel. Hence, whichever unitary, U or V^*, has the larger depth will determine the overall depth of the HST.

3.4.2 Local Hilbert-Schmidt Test

Let us now consider a slightly modified form of the HST, shown in Fig. 3.4(b). We call this the Local Hilbert-Schmidt Test (LHST) because, unlike the HST in Fig. 3.4(a), only two of the total number $2n$ of qubits are measured: one qubit from system A, say A_j, and the corresponding qubit B_j from system B, where $j \in \{1, 2, \ldots, n\}$.

The state of systems A and B before the measurements is given by Eq. (3.15). Using the ricochet property in (3.18) as before, we obtain

$$(U \otimes V^*)|\Phi^+\rangle_{AB} = (UV^\dagger \otimes I)|\Phi^+\rangle_{AB} \tag{3.20}$$

$$= (W \otimes I)|\Phi^+\rangle_{AB}, \tag{3.21}$$

where $W := UV^\dagger$. Let $\bar{A}_j$ denote all systems A_k except for A_j, and let $\bar{B}_j$ denote all systems B_k except for B_j. Taking the partial trace over $\bar{A}_j$ and $\bar{B}_j$ on the state in (3.21) gives us the following state on the qubits A_j and B_j that are being measured:

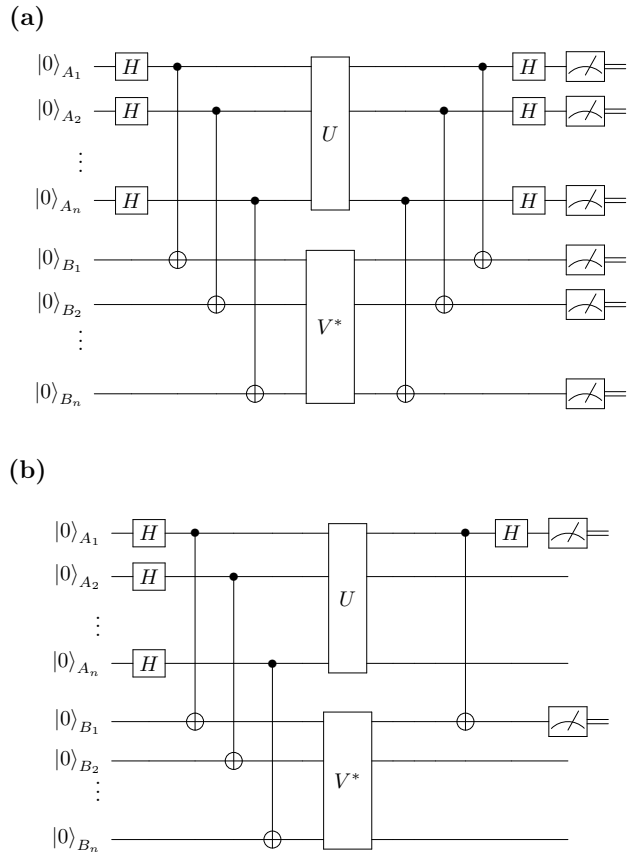


Figure 3.4: **(a)** The Hilbert-Schmidt Test. For this circuit, the probability to obtain the measurement outcome in which all $2n$ qubits are in the $|0\rangle$ state is equal to $(1/d^2)|\mathrm{Tr}(V^\dagger U)|^2$. Hence, this circuit computes the magnitude of the Hilbert-Schmidt inner product, $|\langle V, U\rangle|$, between U and V. **(b)** The Local Hilbert-Schmidt Test, which is the same as the Hilbert-Schmidt Test except that only two of the $2n$ qubits are measured at the end. Shown is the measurement of the qubits A_1 and B_1, and the probability that both qubits are in the state $|0\rangle$ is given by (3.25) with $j = 1$.

$$\begin{aligned}
&\mathrm{Tr}_{\bar{A}_j\bar{B}_j}((W_A \otimes I_B)|\Phi^+\rangle\langle\Phi^+|_{AB}(W_A^\dagger \otimes I_B)) \\
&\quad = \mathrm{Tr}_{\bar{A}_j}\left((W_A \otimes I_{B_j})\left(|\Phi^+\rangle\langle\Phi^+|_{A_jB_j} \otimes \frac{I_{\bar{A}_j}}{2^{n-1}}\right)\right. \\
&\qquad\qquad \left.\times(W_A^\dagger \otimes I_{B_j})\right) \qquad (3.22) \\
&\quad = (\mathcal{E}_j \otimes \mathcal{I}_{B_j})(|\Phi^+\rangle\langle\Phi^+|_{A_jB_j})\,. \qquad (3.23)
\end{aligned}$$

In (3.22), $|\Phi^+\rangle_{A_jB_j}$ is a 2-qubit maximally entangled state of the form in (3.14). In (3.23), we have

defined the channel $\mathcal{E}_j$ by

$$\mathcal{E}_j(\rho_{A_j}) \coloneqq \mathrm{Tr}_{\bar{A}_j}\left(W_A\left(\rho_{A_j}\otimes\frac{I_{\bar{A}_j}}{2^{n-1}}\right)W_A^\dagger\right). \tag{3.24}$$

The probability of obtaining the $(0,0)$ outcome in the measurement of A_j and B_j is the overlap of the state in (3.23) with the $|\Phi^+\rangle_{A_jB_j}$ state, given by

$$F_e^{(j)} := \mathrm{Tr}\left(|\Phi^+\rangle\langle\Phi^+|_{A_jB_j}(\mathcal{E}_j\otimes\mathcal{I}_{B_j})(|\Phi^+\rangle\langle\Phi^+|_{A_jB_j})\right). \tag{3.25}$$

Note that this is the entanglement fidelity of the channel $\mathcal{E}_j$. We use these entanglement fidelities (for each j) to define the local cost function $C_{\mathrm{LHST}}(U,V)$ as

$$C_{\mathrm{LHST}}(U,V) = \frac{1}{n}\sum_{j=1}^{n} C_{\mathrm{LHST}}^{(j)}(U,V), \tag{3.26}$$

where

$$C_{\mathrm{LHST}}^{(j)}(U,V) \coloneqq 1 - F_e^{(j)}. \tag{3.27}$$

Note that for all $j\in\{1,2,\ldots,n\}$, the maximum value of $F_e^{(j)}$ is one, which occurs when $\mathcal{E}_j$ is the identity channel. This means that the minimum value of $C_{\mathrm{LHST}}(U,V)$ is zero. In Section 3.11, we show that C_{LHST} is indeed a faithful cost function:

Proposition 1: For all unitaries U and V, it holds that $C_{\mathrm{LHST}}(U,V)=0$ if and only if $U=V$ (up to a global phase).

The cost function C_{LHST} is simply the average of the probabilities that the two qubits A_jB_j are not in the $|00\rangle$ state, while the cost function C_{HST} is the probability that all qubits are not in the $|0\rangle^{\otimes 2n}$ state. Since the probability of an intersection of events is never greater than the average of the probabilities of the individual events, we find that

$$C_{\mathrm{LHST}}(U,V) \leq C_{\mathrm{HST}}(U,V) \tag{3.28}$$

for all unitaries U and V. Furthermore, we can also formulate a bound in the reverse direction

$$nC_{\mathrm{LHST}}(U,V) \geq C_{\mathrm{HST}}(U,V). \tag{3.29}$$

In Section 3.12, we offer a proof for the above bounds.

Proposition 2: Let U and V be $2^n \times 2^n$ unitaries. Then,

$$C_{\mathrm{LHST}}(U, V) \leq C_{\mathrm{HST}}(U, V) \leq n C_{\mathrm{LHST}}(U, V) \,.$$

The depth of the circuit in Fig. 3.4(b) used to compute the cost function C_{LHST} is the same as the depth of the circuit in Fig. 3.4(a) used to compute C_{HST}, namely,

$$D(\mathrm{LHST}) = 4 + \max\{D(U), D(V^*)\}. \tag{3.30}$$

3.5 Computational complexity of cost evaluation

In this section, we state impossibility results for the efficient classical evaluation of both of our costs, C_{HST} and C_{LHST}. To show this, we analyze our circuits in the framework of deterministic quantum computation with one clean qubit (DQC1) [7]. We then make use of known hardness results for the class DQC1, and establish that the efficient classical approximation of our cost functions is impossible under reasonable complexity assumptions.

3.5.1 One-clean-qubit model of computation

The complexity class DQC1 consists of all problems that can be efficiently solved with bounded error in the one-clean-qubit model of computation. Inspired by the early implementations of NMR quantum computing [7], in the one-clean-qubit model of computation the input is specified by a single "clean qubit", together with a maximally mixed state on n qubits:

$$\rho = |0\rangle\langle 0| \otimes (I/2)^{\otimes n}. \tag{3.31}$$

A computation is then realized by applying a poly(n)-sized quantum circuit Q to the input. We then measure the clean qubit in the standard basis and consider the probability of obtaining the outcome "0", i.e.,

$$\mathrm{Tr}[(|0\rangle\langle 0| \otimes I^{\otimes n}) Q \rho Q^\dagger]. \tag{3.32}$$

The DQC1 model of computation has been widely studied, and several natural problems have been found to be complete for DQC1. Most notably, Shor and Jordan [4] showed that the problem of trace estimation for $2^n \times 2^n$ unitary matrices that specify poly(n)-sized quantum circuits is DQC1-complete. Moreover, Fujii et al. [85] showed that classical simulation of DQC1 is impossible, unless the polynomial hierarchy collapses to the second level. Specifically, it is shown that an efficient classical algorithm that is capable of weakly simulating the output probability distribution of any DQC1 computation would imply a collapse of the polynomial hierarchy to the class of Arthur-Merlin protocols, which is not believed to be true. Rather, it is commonly believed that the class DQC1 is strictly contained in BQP, and thus provides a sub-universal model of quantum computation that is hard to simulate classically. Finally, we point out that the complexity class DQC1 is known to give rise to average-case distance measures, whereas worst-case distance measures (such as the diamond distance) are much harder to approximate, and known to be QIP-complete [86]. Currently, it is not known whether there exists a distance measure that lies between the average-case and worst-case measures in DQC1 and QIP, respectively. However, we conjecture that only average-case distance measures are feasible for practical purposes. We leave the task of finding a distance measure whose approximation is complete for the class BQP as an interesting open problem.

Our contributions are the following. We adapt the proofs in [4, 85] and show that the problem of approximating our cost functions, C_{HST} or C_{LHST}, up to inverse polynomial precision is DQC1-hard. Our results build on the fact that evaluating either of our cost functions is, in some sense, as hard as trace estimation. Using the results from [85], it then immediately follows that no classical algorithm can efficiently approximate our cost functions under certain complexity assumptions.

3.5.2 Approximating C_{HST} is DQC1-hard

In Section 3.13, we prove the following:

Theorem 6: Let U and V be poly(n)-sized quantum circuits specified by $2^n \times 2^n$ unitary matrices, and let $\epsilon = O(1/\mathrm{poly}(n))$. Then, the problem of approximating $C_{\mathsf{HST}}(U, V)$ up to ϵ-precision is DQC1-hard.

3.5.3 Approximating C_{LHST} is DQC1-hard

In Sec. 3.13, we also prove the following:

Theorem 7: Let U and V be poly(n)-sized quantum circuits specified by $2^n \times 2^n$ unitary matrices, and let $\epsilon = O(1/\mathrm{poly}(n))$. Then, the problem of approximating $C_{\mathrm{LHST}}(U, V)$ up to ϵ-precision is DQC1-hard.

As a consequence of these results, it then follows from [85] that there is no classical algorithm to efficiently approximate our cost functions, C_{HST} or C_{LHST}, with inverse polynomial precision, unless the polynomial hierarchy collapses to the second level.

3.6 Small-scale implementations

This section presents the results of implementing QAQC, as described in Sec. 3.3, for well-known one- and two-qubit unitaries. Some of these implementations were done on actual quantum hardware, while others were on a simulator. In each case, we performed gradient-free continuous parameter optimization in order to minimize the cost function C_{HST} in (3.5), evaluating this cost function using the circuit in Fig. 3.4(a). For full details on the optimization procedure, see Section 3.14.

3.6.1 Quantum hardware

We implement QAQC on both IBM's and Rigetti's quantum computers. In what follows, the depth of a gate sequence is defined relative to the native gate alphabet of the quantum computer used.

3.6.1.1 IBM's quantum computers

Here, we consider the 5-qubit IBMQX4 and the 16-qubit IBMQX5. For these quantum computers, the native gate set is

$$\mathcal{A}_{\mathrm{IBM}} = \{R_x(\pi/2), R_z(\theta), \mathrm{CNOT}_{ij}\} \tag{3.33}$$

where the single-qubit gates $R_x(\pi/2)$ and $R_z(\theta)$ can be performed on any qubit and the two-qubit CNOT gate can be performed between any two qubits allowed in the topology; see [97] for the topology of IBMQX4 and [98] for the topology of IBMQX5.

To compile a given unitary U, we use the general procedure outlined in Sec. 3.3.3. Specifically, our initial gate structure, given by $V_{\vec{k}}(\vec{\alpha})$, is selected at random from the gate alphabet in (3.33). We then calculate the cost $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ by executing the HST shown in Fig. 3.4(a) on the quantum computer. To perform the continuous parameter optimization over the angles θ of the R_z gates, we make use of Algorithm 2 outlined in Section 3.14.1. This method is designed to limit the number of objective function calls to the quantum computer, which is an important consideration when using queue-based quantum computers like IBMQX4 and IBMQX5 since these can entail a significant amount of idle time in the queue.

In essence, our method in Algorithm 2 discretizes the continuous parameter space of angles θ to perform the continuous optimization. These angles are selected uniformly over the unit circle and the grid spacing between them decreases in the number of iterations. See Section 3.14.1 for full details. If the cost of the new sequence is less than the cost of the previous sequence, then we accept the change. Otherwise, we accept the change with a probability that decreases exponentially in the magnitude of the difference in cost. This change in cost defines one iteration.

In Fig. 3.5(a), we show results for compiling single-qubit gates on IBMQX4. All gates (I, T, X, and H) converge to a cost below 0.1, but no gate achieves a cost below our tolerance of 10^{-2}. As elaborated upon in Sec. 3.8, this is due to a combination of finite sampling, gate fidelity, decoherence, and readout error on the device. The single-qubit gates compile to the following gate sequences:

1. I gate: $R_z(\theta)$, with $\theta \approx 0.01\pi$.

2. T gate: $R_z(\theta)$, with $\theta \approx 0.30\pi$.

3. X gate: $R_x(\pi/2)R_x(\pi/2)$.

4. H gate: $R_z(\theta_1)R_x(\pi/2)R_z(\theta_2)$, with $\theta_1 = \theta_2 = 0.50\pi$.

Figure 3.5(b) shows results for compiling the same single-qubit gates as above on IBMQX5. The gate sequences have the same structure as listed above for IBMQX4. The optimal angles achieved are $\theta = -0.03\pi$ for the I gate and $\theta = 0.23\pi$ for the T gate. The X gate compiles to $R_x(\pi/2)R_x(\pi/2)$, and the Hadamard gate H compiles to $R_x(\pi/2)R_z(\pi/2)R_x(\pi/2)$.

In our data collection, we performed on the order of 10 independent optimization runs for each target gate above. The standard deviations of the angles θ were on the order of 0.05π, and this can be viewed as the error bars on the average values quoted above.

3.6.1.2 Rigetti's quantum computer

The native gate set of Rigetti's 8Q-Agave 8-qubit quantum computer is

$$\mathcal{A}_{\text{Rigetti}} = \{R_x(\pm\pi/2), R_z(\theta), \text{CZ}_{ij}\} \tag{3.34}$$

where the single-qubit gates $R_x(\pm\pi/2)$ and $R_z(\theta)$ can be performed on any qubit and the two-qubit CZ gate can be performed between any two qubits allowed in the topology; see [99] for the topology of the 8Q-Agave quantum computer.

As with the implementation on IBM's quantum computers, for the implementation on Rigetti's quantum computer we make use of the general procedure outlined in Sec. 3.3.3. Specifically, we perform random updates to the gate structure followed by continuous optimization over the parameters θ of the R_z gates using the gradient-free stochastic optimization technique described in Algorithm 1 in Section 3.14. In this optimization algorithm, we use fifty cost function evaluations to perform the continuous optimization over parameters. (That is, each iteration in Fig. 3.5(c) and Fig. 3.6 uses fifty cost function evaluations, and each cost function evaluation uses $10,000$ calls to the quantum computer for finite sampling.) We take the cost error tolerance (the parameter ε' in Algorithm 1) to be 10^{-2}, and for each run of the Hilbert-Schmidt Test, we take $10,000$ samples in order to estimate the cost. Our results are shown in Fig. 3.5(c). As described in Algorithm 1, we define an iteration to be one accepted update in gate structure followed by a continuous optimization over the internal gate parameters.

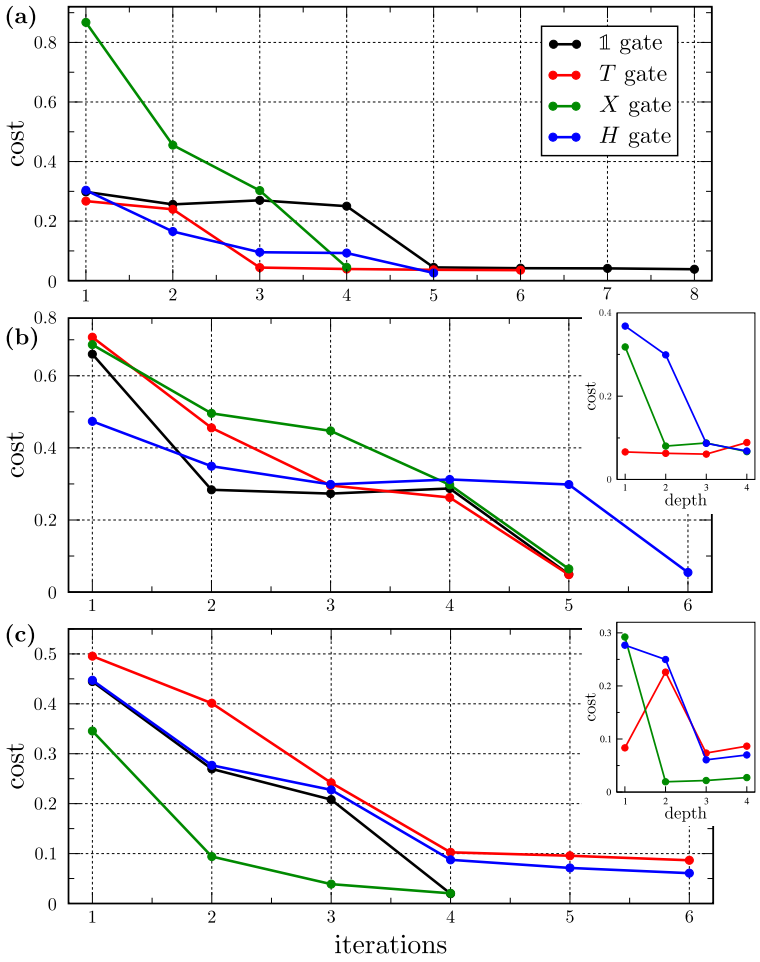


Figure 3.5: Compiling the one-qubit gates I, X, H, and T using the gradient-free optimization technique described in Section 3.14. The plots show the cost C_{HST} as a function of the number of iterations, where an iteration is defined by an accepted update to the gate structure; see Sec. 3.3.3 for a description of the procedure. The insets display the minimum cost achieved by optimizing over gate sequences with a fixed depth, where the depth is defined relative to the native gate alphabet of the quantum computer used. (a) Compiling on the IBMQX4 quantum computer, in which we took $8,000$ samples to evaluate the cost for each run of the Hilbert-Schmidt Test. (b) Compiling on the IBMQX5 quantum computer, in which we again took $8,000$ samples to evaluate the cost for each run of the Hilbert-Schmidt Test. (c) Compiling on Rigetti's 8Q-Agave quantum computer. In the plot, each iteration uses 50 cost function evaluations to perform the continuous optimization. For each run of the Hilbert-Schmidt Test to evaluate the cost, we took $10,000$ samples (calls to the quantum computer).

The gates compiled in Fig. 3.5(c) have the following optimal decompositions. The same decompositions also achieve the lowest cost in the cost vs. depth plot in the inset.

1. I gate: $R_z(\theta)$, with $\theta \approx 0$.

2. T gate: $R_z(\theta)$, with $\theta \approx 0.342\pi$.

3. X gate: $R_x(-\pi/2)R_x(-\pi/2)$.

4. H gate: $R_z(\theta_1)R_x(\pi/2)R_z(\theta_2)$, with $\theta_1 \approx 0.50\pi$ and $\theta_2 \approx 0.49\pi$.

As with the results on IBM's quantum computers, none of the gates achieve a cost less than 10^{-2}, due to factors such as finite sampling, gate fidelity, decoherence, and readout error. In addition, similar to the IBM results, the standard deviations of the angles θ here were on the order of 0.05π, which can be viewed as the error bars on the average values (over 10 independent runs) quoted above.

3.6.2 Quantum simulator

We now present our results on executing QAQC for single-qubit and two-qubit gates using a simulator. We use the gate alphabet

$$\mathcal{A} = \{R_x(\pi/2), R_z(\theta), \mathrm{CNOT}_{ij}\}, \tag{3.35}$$

which is the gate alphabet defined in Eq. (3.33) except with full connectivity between the qubits. We again use the gradient-free optimization method outlined in Section 3.14 to perform the continuous parameter optimization. The simulations are performed assuming perfect connectivity between the qubits, no gate errors, and no decoherence.

Using Rigetti's quantum virtual machine [2], we compile the controlled-Hadamard (CH) gate, the CZ gate, the SWAP gate, and the two-qubit quantum Fourier transform QFT_2 by adopting the gradient-free continuous optimization procedure in Algorithm 1. We also compile the single-qubit gates X and H. For each run of the Hilbert-Schmidt Test to determine the cost, we took $20,000$

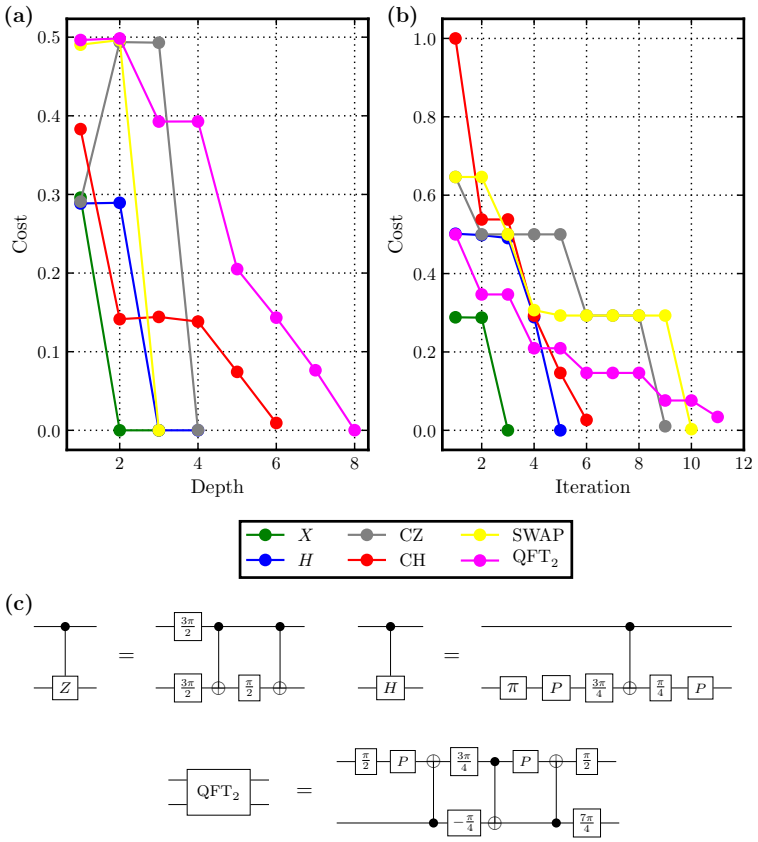


Figure 3.6: Compiling one- and two-qubit gates on Rigetti's quantum virtual machine with the gate alphabet in (3.35) using the gradient-free optimization technique described in Algorithm 1 in Section 3.14. (a) The minimum cost achieved by optimizing over gate sequences with a fixed depth. (b) The cost as a function of the number of iterations of the full gate structure and continuous parameter optimization; see Sec. 3.3.3 for a description of the procedure. Note that each iteration uses 50 cost function evaluations, and each cost function evaluation uses 10,000 samples (calls to the quantum computer). (c) Shortest-depth decompositions of the two-qubit controlled-Z, controlled-Hadamard, and quantum Fourier transform gates as determined by the compilation procedure. The equalities indicated are true up to a global phase factor. Here, $-\boxed{\theta}-$ denotes the rotation gate $R_z(\theta)$, while $-\boxed{P}-$ represents the rotation gate $R_x(\pi/2)$.

samples. Our results are shown in Fig. 3.6. For the SWAP gate, we find that circuits of depth one and two cannot achieve zero cost, but there exists a circuit with depth three for which the cost vanishes. The circuit achieving this zero cost is the well-known decomposition of the SWAP gate into three CNOT gates. While our compilation procedure reproduces the known decomposition of the SWAP gate, it discovers a decomposition of both the CZ and the QFT_2 gates that differs from their conventional "textbook" decompositions, as shown in Fig. 3.6(c). In particular, these decompositions have shorter depths than the conventional decompositions when written in terms of the gate alphabet in (3.35).

In Section 3.15, we likewise implement QAQC for one- and two-qubit gates on a simulator, but instead using a gradient-based continuous parameter optimization method outlined therein.

3.7 Larger-scale implementations

While in the previous section we considered one- and two-qubit unitaries, in this section we explore larger unitaries, up to nine qubits. The purpose of this section is to see how QAQC scales, and in particular, to study the performance of our C_{HST} and C_{LHST} cost functions as the problem size increases. We consider two different examples.

Example 1: In the first example, we let U be a tensor product of one-qubit unitaries. Namely we consider

$$U = \bigotimes_{j=1}^{n} R_z(\theta_j) \tag{3.36}$$

where the θ_j are randomly chosen, and $R_z(\theta)$ is a rotation about the z-axis of the Bloch sphere by angle θ. Similarly, our ansatz for V is of the same form,

$$V = \bigotimes_{j=1}^{n} R_z(\phi_j) \tag{3.37}$$

where the initial values of the angles ϕ_j are randomnly chosen.

Example 2: In the second example, we go beyond the tensor-product situation and explore a unitary that entangles all the qubits. The target unitary has the form $U = U_4(\vec{\theta}')U_3U_2U_1(\vec{\theta})$, with

$$U_1(\vec{\theta}) = \bigotimes_{j=1}^{n} R_z(\theta_j), \quad U_2 = ...\text{CNOT}_{34}\text{CNOT}_{12} \tag{3.38}$$

$$U_3 = ...\text{CNOT}_{45}\text{CNOT}_{23}, \quad U_4(\vec{\theta}') = \bigotimes_{j=1}^{n} R_z(\theta'_j)\,. \tag{3.39}$$

Here, CNOT_{kl} denotes a CNOT with qubit k the control and qubit l the target, while $\vec{\theta} = \{\theta_j\}$ and $\vec{\theta}' = \{\theta'_j\}$ are n-dimensional vectors of angles. Hence U_2 and U_3 are layers of CNOTs where the CNOTs in U_3 are shifted down by one qubit relative to those in U_2. Our ansatz for the trainable unitary V has the same form as U but with different angles, i.e., $V = U_4(\vec{\phi}')U_3U_2U_1(\vec{\phi})$ where $\vec{\phi}$ and $\vec{\phi}'$ are randomly initialized.

In what follows we discuss our implementations of QAQC for these two examples. We first discuss the implementation on a simulator without noise, and then we move onto the implementation on a simulator with a noise model.

3.7.1 Noiseless implementations

We implemented Examples 1 and 2 on a noiseless simulator. In each case, starting with the ansatz for V at a randomly chosen set of angles, we performed the continuous parameter optimization over the angles using a gradient-based approach. We made use of Algorithm 4 in Section 3.15.3, which is a gradient descent algorithm that explicitly evaluates the gradient using the formulas provided in Section 3.15.3. For each run of the HST and LHST, we took 1000 samples in order to estimate the value of the cost function. The results of this implementation are shown in Figs. 3.7 and 3.8.

In the case of Example 1 (Fig. 3.7), both the C_{HST} and C_{LHST} cost functions converge to the desired global minimum up to 5 qubits. However, for n = 6, 7, 8, and 9 qubits, we find cases in which the C_{HST} cost function does not converge to the global minimum but the C_{LHST} cost function does. Specifically, the cost C_{HST} stays very close to one, with a gradient value smaller than the pre-set threshold of 10^{-3} for four consecutive iterations, causing the gradient descent algorithm to

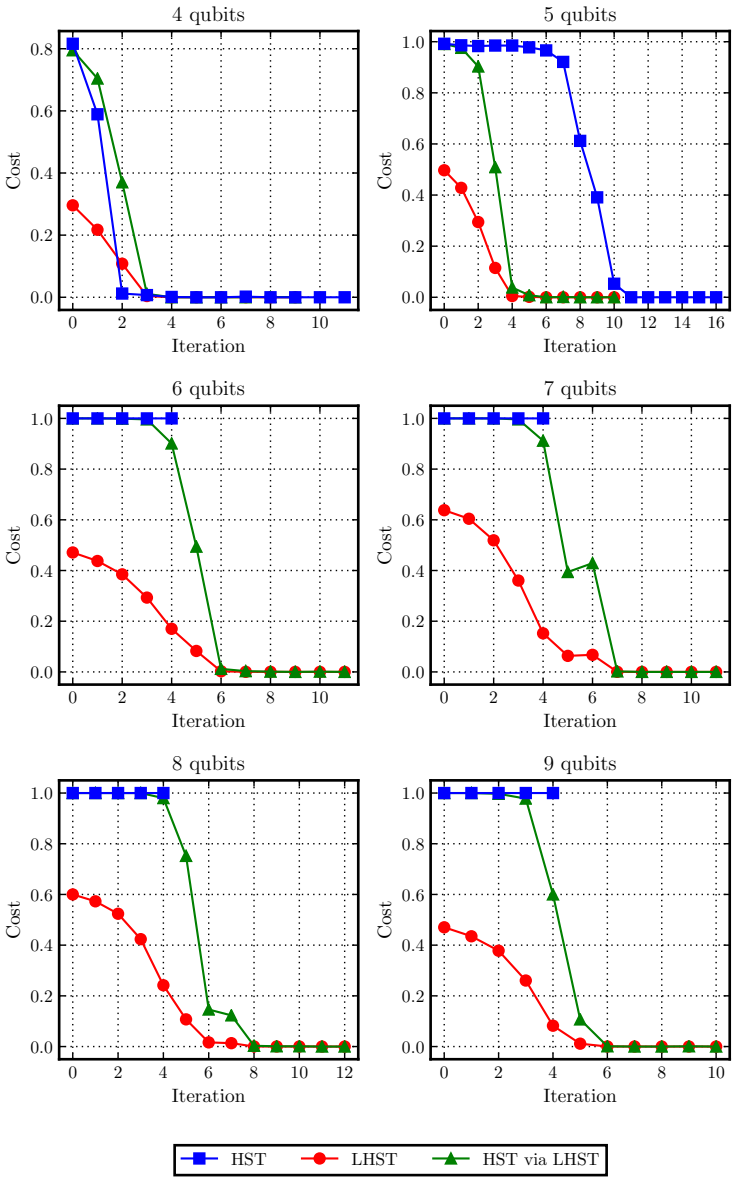


Figure 3.7: Results of performing continuous parameter optimization using the HST and the LHST for the scenario described in Example 1. We make use of the gradient-based optimization algorithm given by Algorithm 4 in Section 3.15. The curves "HST via LHST" are given by evaluating C_{HST} using the angles obtained during the optimization iterations of C_{LHST}. For each run of the HST and LHST, we use 1000 samples to estimate the cost function.

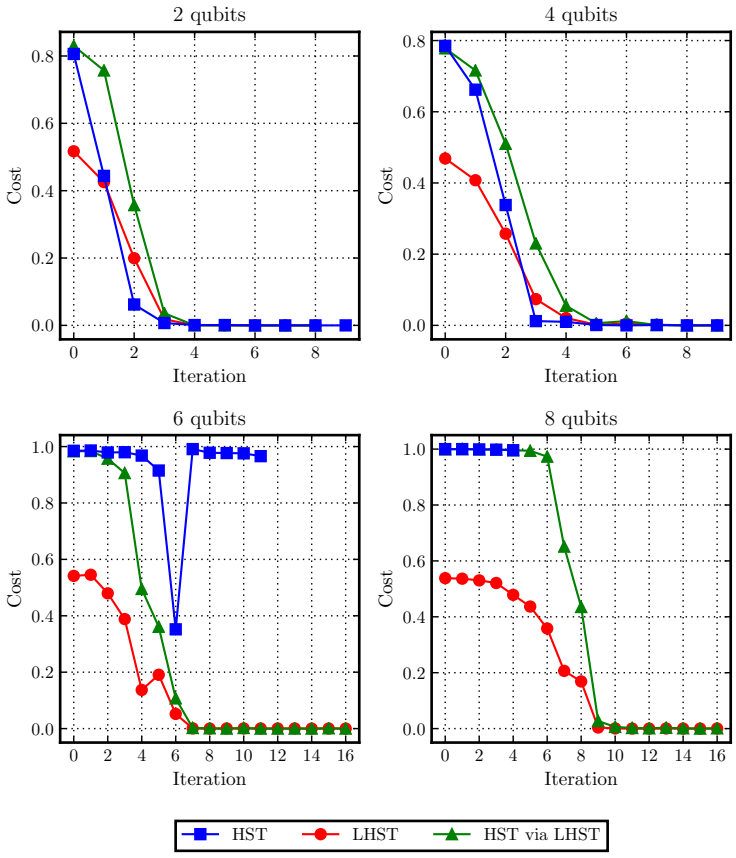


Figure 3.8: Results of performing continuous parameter optimization using the HST and the LHST for the scenario described in Example 2. We make use of the gradient-based optimization algorithm given by Algorithm 4 in Section 3.15, in which each iteration can involve several calls to the quantum computer. The curves "HST via LHST" are given by evaluating C_{HST} using the angles obtained during the optimization iterations of C_{LHST}. For each run of the HST and LHST, we use 1000 samples to estimate the cost function.

declare convergence. Interestingly, even in the cases that the C_{HST} cost does not converge to the global minimum, training with the C_{LHST} cost allows us to fully minimize the C_{HST} cost. (See the green curves labelled "HST via LHST" in Fig. 3.7, in which we evaluate the C_{HST} cost at the angles obtained during the optimization of the C_{LHST} cost.) This fascinating feature implies that, for $n \geq 6$ qubits in Example 1, training our C_{LHST} cost is better at minimizing the C_{HST} cost than is directly attempting to train the C_{HST} cost.

We find very similar behavior for Example 2 (Fig. 3.8). In particular, for $n \geq 6$ qubits, we were unable to directly train the C_{HST} cost. However, the C_{LHST} cost converges to the global minimum for $n = 6$ and 8 qubits. Furthermore, as with Example 1, we find that minimizing the C_{LHST} cost

also minimizes the C_{HST} cost.

3.7.2 Noisy implementations

We implemented Examples 1 and 2 on IBM's noisy simulator, where the noise model matches that of the 16-qubit IBMQX5 quantum computer. This noise model accounts for T_1 noise, T_2 noise, gate errors, and measurement errors. We emphasize that these are realistic noise parameters since they simulate the noise on currently available quantum hardware. (Note that when our implementations required more than 16 qubits, we applied similar noise parameters to the additional qubits as those for the 16 qubits of the IBMQX5.) We used the same training algorithm as the one we used in the noiseless case above. The results of these implementations are shown in Figs. 3.9 and 3.10.

Similar to the noiseless case, for Example 1 (Fig. 3.9) and for Example 2 (Fig. 3.10), we find that both the C_{HST} and C_{LHST} cost functions converge up to a problem size of 5 qubits. Due to the noise, as expected, both cost functions converge to a value greater than zero. For $n \geq 6$ qubits, however, we find that the C_{HST} cost function does not converge to a local minimum. Specifically, this cost stays very close to one with a gradient value smaller than the pre-set threshold of 10^{-3} for four consecutive iterations, causing the gradient descent algorithm to declare convergence. The local cost, on the other hand, converges to a local minimum in every case.

Remarkably, despite the noise in the simulation, we find that the angles obtained during the iterations of the C_{LHST} optimization correspond to the optimal angles in the noiseless case. This result is indicated by the green curves labeled "Noiseless HST via LHST". One can see that the green curves go to zero for the local minima found by training the noisy C_{LHST} cost function. Hence, in these examples, training the noisy C_{LHST} cost function can be used to minimize the noiseless C_{HST} cost function to the global minimum. This intriguing behavior suggests that the noise has not affected the location (i.e., the value for the angles) of the global minimum. We thus find evidence of the robustness of QAQC to the kind of noise present in actual devices. We elaborate on this point in the next section.

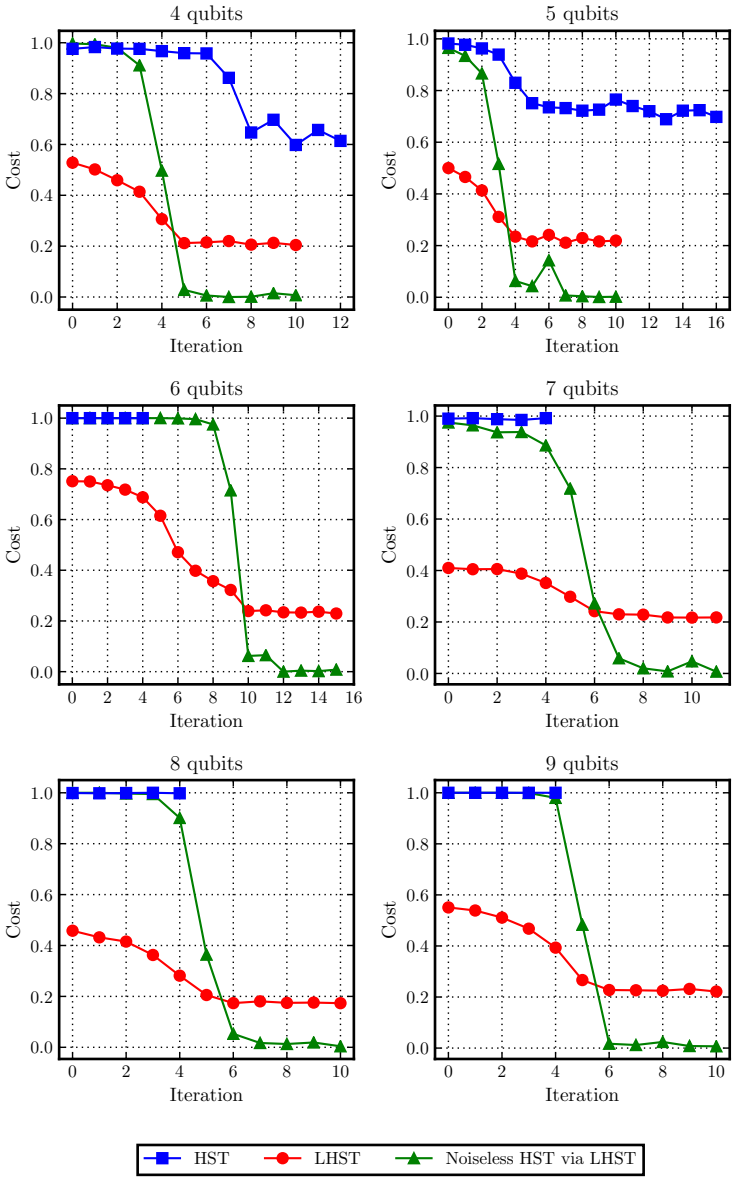


Figure 3.9: Results of performing continuous parameter optimization using the HST and the LHST, in the presence of noise, for the scenario described in Example 1. The noise model used matches that of the IBMQX5 quantum computer. We make use of the gradient-based optimization algorithm given by Algorithm 4 in Section 3.15. The curves "Noiseless HST via LHST" are given by evaluating C_{HST} (without noise) using the angles obtained during the optimization iterations of C_{LHST}. For each run of the HST and LHST, we use 1000 samples to estimate the cost function.

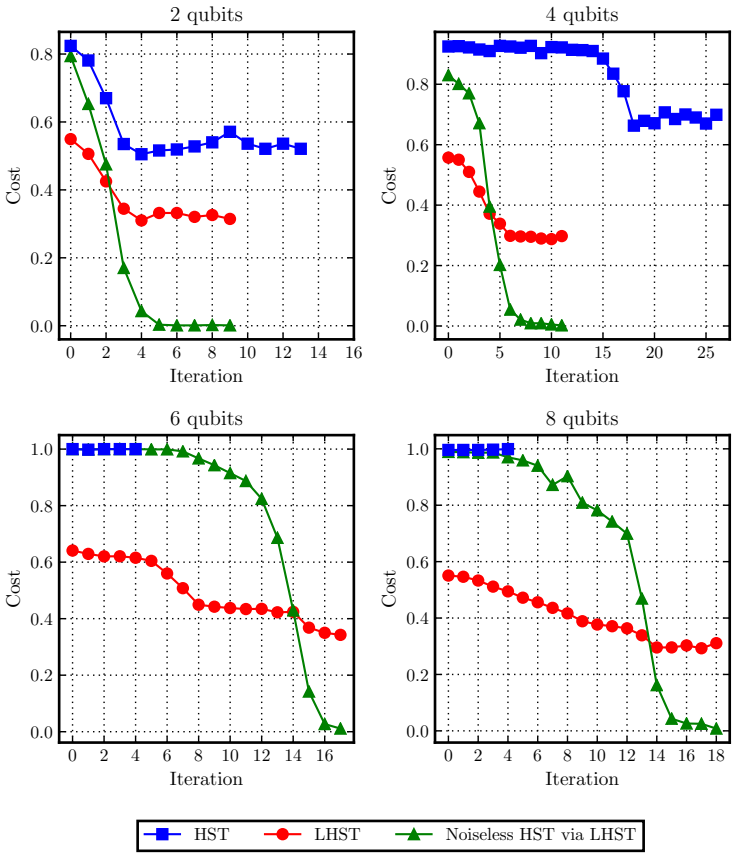


Figure 3.10: Results of performing continuous parameter optimization using the HST and the LHST, in the presence of noise, for the scenario described in Example 2. The noise model used matches that of the IBMQX5 quantum computer. We make use of the gradient-based optimization algorithm given by Algorithm 4 in Section 3.15. The curves "Noiseless HST via LHST" are given by evaluating C_{HST} (without noise) using the angles obtained during the optimization iterations of C_{LHST}. For each run of the HST and LHST, we use 1000 samples to estimate the cost function.

3.8 Discussion

On both IBM's and Rigetti's quantum hardware, we were able to successfully compile one-qubit gates with no *a priori* assumptions about gate structure or gate parameters. We also successfully implemented QAQC for simple 9-qubit gates on both a noiseless and noisy simulator. These implementations highlighted two important issues, (1) barren plateuas in cost landscape and (2) the effect of hardware noise, which we discuss further now.

3.8.1 Barren plateaus

Recent results [40, 100] on gradient-based optimization with random quantum circuits suggest that the probability of observing non-zero gradients tends to become exponentially small as a function of the number of qubits. That work showed that a hardware-efficient ansatz leads to vanishing gradients as the ansatz's depth becomes deeper (and hence begins to look more like a random unitary). This is an important issue for many variational hybrid algorithms, including QAQC, and motivates the need to avoid a deep, random ansatz. Strategies to address this "barren plateau" issue for QAQC include restricting to a short-depth ansatz, or alternatively employing an application-specific ansatz that takes into account some information about the target unitary U. We intend to explore application-specific ansatze in future work to address this issue. There may be other strategies based on the fact that similar issues have been identified in classical deep learning [101]. For instance, recent work [102] shows that gradient descent with momentum (GDM) using an adaptive (multiplicative) integration step update, called resilient backpropagation (rProp), can help with convergence. But, this is an active research area and will likely be important to the success of variational hybrid algorithms.

Interestingly, in this work, we identified another barren plateau issue that is completely independent and distinct from the issue raised in Refs. [40, 100]. Namely, we found that our operationally meaningful cost function, C_{HST}, can have barren plateaus even when the ansatz is a depth-one circuit. The gradient of C_{HST} can vanish exponentially in n even when the ansatz has only a single parameter. This issue became apparent in our implementations (see Figs. 3.7 through 3.10), where we were unable to directly train the C_{HST} cost for $n \geq 6$ qubits. Fortunately, we fixed this issue by introducing the C_{LHST} cost, which successfully trained in all cases we attemped (we attempted up to $n = 9$ qubits). Although C_{LHST} is not directly operationally meaningful, it is indirectly related to C_{HST} via Eqs. (3.28) and (3.29). Hence it can be used to indirectly train C_{HST}, as shown in Figs. 3.7 through 3.10. We believe this barren plateau issue will show up in other variational hybrid algorithms. For example, we encountered the same issue in a recently introduced variational algorithm for state diagonalization [103].

3.8.2 Effect of hardware noise

The impact of hardware noise, such as decoherence, gate infidelity, and readout error, is important to consider. This is especially true since QAQC is aimed at being a useful algorithm in the era of NISQ computers, although we remark that QAQC may also be useful for fault-tolerant quantum computing.

On the one hand, we intuitively expect noise to significantly affect the HST and LHST cost evaluation circuits. On the other hand, we see empirical evidence of noise resilience in Figs. 3.9 and 3.10. Let us elaborate on both our intuition and our empirical observations now.

A qualitative noise analysis of the HST circuit in Fig. 3.4(a) is as follows. To compile a unitary U acting on n qubits, a circuit with $2n$ qubits is needed. Preparing the maximally-entangled state $|\Phi^+\rangle$ in the first portion of the circuit requires n CNOT gates, which are significantly noisier than one-qubit gates and propagate errors to other qubits through entanglement. In principle, all Hadamard and CNOT gates can be implemented in parallel, but on near-term devices this may not be the case. Additionally, due to limited connectivity of NISQ devices, it is generally not possible to directly implement CNOTs between arbitrary qubits. Instead, the CNOTs need to be "chained" between qubits that are connected, a procedure that can significantly increase the depth of the circuit.

The next level of the circuit involves implementing U in the top n-qubit register and V^* in the bottom n-qubit register. Here, the noise of the computer on V^* is not necessarily undesirable since it could allow us to compile noise-tailored algorithms that counteract the noise of the specific computer, as described in Sec. 3.2. Nevertheless, the depth of V^* and/or of U essentially determines the overall circuit depth as noted in (3.19), and quantum coherence decays exponentially with the circuit depth. Hence, compiling larger gate sequences involves additional loss of coherence on NISQ computers.

The final level of the HST circuit involves making a Bell measurement on all qubits and is the reverse of the first part of the circuit. As such, the same noise analysis of the first portion of the circuit applies here. Readout errors can be significant on NISQ devices [104], and our HST circuit

involves a number of measurements that scales linearly in the number of qubits. Hence, compiling larger unitaries can increase overall readout error.

A similar qualitative noise analysis holds for the LHST circuit in Fig. 3.4(b), except we note that to calculate the functions $C^{(j)}_{\mathrm{LHST}}$ in (3.26) we require only one CNOT gate in the last portion of the LHST circuit before the measurement. Furthermore, we measure only two qubits regardless of the total number of qubits.

With that said, we observed a (somewhat surprising) noise resilience in Figs. 3.9 and 3.10. In these implementations, we imported the noise model of the IBMQX5 quantum computer, which is a currently available cloud quantum computer. Hence, we considered realistic noise parameters for decoherence, gate infedility, and readout error. This noise affected all circuit elements of the LHST circuit in Fig. 3.4(b). Yet we still obtained the correct unitary V via QAQC, as shown by the green curves going to zero in Figs. 3.9 and 3.10.

Naturally, we plan to investigate this noise resilience in full detail in future work. But it is worth emphasizing the following point here. *The value of the cost could be significantly affected by noise without shifting the location of the global minimum in parameter space.* In fact, one can see in Figs. 3.9 and 3.10 that the value of the C_{LHST} cost is significantly affected by noise. Namely, note that the red curves in these plots do not go to zero for larger iterations. However, the green curves do go to zero, which means that QAQC found the correct parameters for V despite the noisy cost values.

We could speculate reasons for why the global minimum appears not shift in parameter space with noise. For example, it could be due to the nature of our cost functions. These cost functions can be thought of as entanglement fidelities and hence are related to Hilbert-space averages of input-output fidelities, see Eq. (3.6). By averaging the input-output fidelity over the whole Hilbert space, the effect of noise could essentially be averaged away. This is just speculation at this point, and we will perform a detailed analysis of the effect of noise in future work. Regardless, our preliminary results in Figs. 3.9 and 3.10 suggest that QAQC may indeed be useful in the NISQ era.

3.9 Conclusions

Quantum compiling is crucial in the era of NISQ devices, where constraints on NISQ computers (such as limited connectivity, limited circuit depth, etc.) place severe restrictions on the quantum algorithms that can be implemented in practice. In this work, we presented a methodology for quantum compilation called quantum-assisted quantum compiling (QAQC), whereby a quantum computer provides an exponential speedup in evaluating the cost of a gate sequence, i.e., how well the gate sequence matches the target. In principle, QAQC should allow for the compiling of larger algorithms than standard classical methods for quantum compiling due to this exponential speedup. As a proof-of-principle, we implemented QAQC on IBM's and Rigetti's quantum computers to compile various one-qubit gates to their native gate alphabets. To our knowledge, this is the first time NISQ hardware has been used to compile a target unitary. In addition, we successfully implemented QAQC on a noiseless and noisy simulator for simple 9-qubit unitaries.

Our main technical results were the following. First, we carefully chose a cost function (which involved global and local overlaps between a target unitary U and a trainable unitary V) and proved that it satisfied four criteria: it is faithful, it is efficient to compute on a quantum computer, it has an operational meaning, and it scales well with the size of the problem. Second, we presented short-depth circuits (see Sections 3.4.1 and 3.4.2) for computing our cost function. Third, we proved that evaluating our cost function is DQC1-hard, and hence no classical algorithm can efficiently evaluate our cost function, under reasonable complexity assumptions. This established a rigorous proof for the difficulty of classically simulating QAQC. We also remark that, in the Section, we detailed our gradient-free and gradient-based methods for optimizing our cost function. This included a circuit for gradient computation that generalizes the famous Power of One Qubit [7] and hence is likely of interest to a broader community.

As elaborated in the Discussion section, our noisy implementations of QAQC showed a surprising resilience to noise. While simulating realistic noise parameters based on a currently available cloud quantum computer (IBMQX5), we were able to run QAQC on a 9-qubit unitary and obtain the correct parameters for V. We plan to investigate this intriguing noise resilience in future work.

QAQC is a novel variational hybrid algorithm, similar to other well-known variational hybrid algorithms such as VQE [78] and QAOA [77]. Variational hybrid algorithms are likely to provide some of the first real applications of quantum computers in the NISQ era. In the case of QAQC, it is an algorithm that makes other algorithms more efficient to implement, via algorithm depth compression. We note that the ability to compress algorithm depth will also be useful (to reduce the run-time of quantum circuits) in the era of fault-tolerant quantum computing. The central application of QAQC is thus to make quantum computers more useful.

3.10 Remark on implementation of V^*

As mentioned in Sec. 3.4, a subtle point about evaluating the cost functions $C_{\mathsf{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ and $C_{\mathsf{LHST}}(U, V_{\vec{k}}(\vec{\alpha}))$ is that the complex conjugate $V_{\vec{k}}(\vec{\alpha})^*$ must be executed on the quantum computer, not $V_{\vec{k}}(\vec{\alpha})$ itself. The complex conjugate of a unitary corresponding to a gate sequence can be obtained by taking the complex conjugate of each unitary in the gate sequence. However, if each gate in the sequence comes from a gate alphabet $\mathcal{A}$, it is possible that the complex conjugate of a gate in the sequence is not contained in the alphabet; for example, if $\mathcal{A} = \{R_x(\pi/2), R_z(\theta)\}$, then the complex conjugate of $R_x(\pi/2)$, which is $R_x(-\pi/2)$, is not contained in $\mathcal{A}$. But the unitary $R_z(\pi)R_x(\pi/2)R_z(\pi)$ is equal (up to a global phase) to $R_x(-\pi/2)$. There are thus two ways to proceed when performing the compilation procedure: during the optimization over the continuous parameters, directly run the gate sequence corresponding to $V_{\vec{k}}(\vec{\alpha})$, expressing it in terms of the native gate alphabet of the quantum computer, then at the end establish the complex conjugate of the optimal unitary as the unitary to which U has been compiled. This would involve translating the complex conjugate of each gate in the optimal sequence into the native gate alphabet of the quantum computer. An alternative is to first take the complex conjugate $V_{\vec{k}}(\vec{\alpha})^*$ by translating the complex conjugate of each gate in the sequence into the native gate alphabet, then execute the resulting sequence on the quantum computer. In each case, we allow for a small-scale classical compiler that can perform the simple translation of the complex conjugate of a gate sequence into the native gate alphabet of the quantum computer. Note that this small-scale classical compiler

does not come with exponential overhead because it is only compiling one- and two-qubit gates.

Also, observe that if a gate alphabet is not closed under complex conjugation, then the depth of a gate sequence from that alphabet can increase by taking its complex conjugate. This is true for the example given above, in which the complex conjugate $R_x(-\pi/2)$ of $R_x(\pi/2)$ has a depth of three under the alphabet $\mathcal{A} = \{R_x(\pi/2), R_z(\theta)\}$, while the original gate has a depth of only one. However, in general, note that the final depth increases by at most a constant factor relative to the original depth.

3.11 Faithfulness of LHST cost function

Proposition 1: For all unitaries U and V, it holds that $C_{\text{LHST}}(U, V) = 0$ if and only if $U = V$ (up to a global phase).

Proof: First, we note that since $0 \leq C^{(j)}_{\text{LHST}}(U, V) \leq 1$ for all $j \in \{1, 2, \ldots, n\}$, we get that $C_{\text{LHST}}(U, V) = 0$ if and only if $C^{(j)}_{\text{LHST}}(U, V) = 0$, i.e., $F_e^{(j)} = 1$, for all $j \in \{1, 2, \ldots, n\}$. Next, since $F_e^{(j)}$ is by definition the entanglement fidelity of the channel $\mathcal{E}_j$, we have that $F_e^{(j)} = 1$ if and only if $\mathcal{E}_j$ is the identity channel $\mathcal{I}$. Finally, the condition $U = V$ is equivalent to $W \coloneqq UV^\dagger = I$. Therefore, it suffices to prove that $W = I$ if and only if $\mathcal{E}_j$ is the identity channel for all $j \in \{1, 2, \ldots, n\}$. The implication $W = I \Rightarrow \mathcal{E}_j = \mathcal{I}$ for all $j \in \{1, 2, \ldots, n\}$ is immediate. We now prove the converse.

Let $j = 1$, and suppose that W has the following operator Schmidt decomposition under the bipartite cut $A_1|A_2 \cdots A_n$:

$$W = \sum_{i=1}^{r} \sqrt{\sigma_i} X_i^{A_1} \otimes Y_i^{A_2 \cdots A_n}, \tag{3.40}$$

where $\{X_i\}_{i=1}^r$ and $\{Y_i\}_{i=1}^r$ are orthonormal sets of operators, $\sigma_i > 0$ are the Schmidt coefficients of W, and r is the Schmidt rank of W. Since W is unitary, we have

$$W^\dagger W = \sum_{i,i'=1}^{r} \sqrt{\sigma_i \sigma_{i'}} X_i^\dagger X_{i'} \otimes Y_i^\dagger Y_{i'} = I_{A_1 \cdots A_n}, \tag{3.41}$$

which implies that

$$\text{Tr}_{A_2 \cdots A_n}(W^\dagger W) = \sum_{i=1}^{r} \sigma_i X_i^\dagger X_i = 2^{n-1} I_{A_1}. \tag{3.42}$$

Plugging in the Schmidt decomposition of W into the definition of $\mathcal{E}_1$ in (3.24), we get

$$\mathcal{E}_1(\rho) = \sum_{i=1}^{r} \frac{1}{2^{n-1}} \sigma_i X_i \rho X_i^\dagger. \tag{3.43}$$

The operators $K_i := \sqrt{\frac{\sigma_i}{2^{n-1}}} X_i$ can therefore be regarded as Kraus operators for $\mathcal{E}_1$. Indeed, they satisfy the following condition for trace preservation:

$$\sum_{i=1}^{r} K_i^\dagger K_i = \frac{1}{2^{n-1}} \sum_{i=1}^{r} \sigma_i X_i^\dagger X_i \tag{3.44}$$

$$= \frac{1}{2^{n-1}} \mathrm{Tr}_{A_2 \cdots A_n}(W^\dagger W) \tag{3.45}$$

$$= I_{A_1}, \tag{3.46}$$

where to obtain the second equality we used (3.42).

Now, we assume that $\mathcal{E}_1$ is the identity channel, meaning that $\mathcal{E}_1(\rho) = \sum_{i=1}^{r} \frac{\sigma_i}{2^{n-1}} X_i \rho X_i^\dagger = \rho$ for all states ρ. By the non-uniqueness of Kraus representations of quantum channels, there exists an isometry V relating the Kraus operators $\{K_i\}_{i=1}^r$ to another set $\{N_j\}_{j=1}^s$ of Kraus operators according to $K_i = \sum_{j=1}^{s} V_{i,j} N_j$. Since one Kraus representation of the identity channel is the one consisting of only the identity operator I, we let the set $\{N_j\}_{j=1}^s$ consist of only the identity operator. The isometry V is then a $r \times 1$ matrix, so that $V_{i,1} = \alpha_i \in \mathbb{C}$ for all $i \in \{1, 2, \ldots, r\}$. This implies that $K_i = \sqrt{\frac{\sigma_i}{2^{n-1}}} X_i = \alpha_i I_{A_1}$ for all $i \in \{1, 2, \ldots, r\}$. Therefore,

$$W_{A_1 \cdots A_n} = \sum_{i=1}^{r} \sqrt{\sigma_i} X_i^{A_1} \otimes Y_i^{A_2 \cdots A_n} \tag{3.47}$$

$$= \sum_{i=1}^{r} \sqrt{\sigma_i} \left(\sqrt{\frac{2^{n-1}}{\sigma_i}} \alpha_i I_{A_1} \right) \otimes Y_i^{A_2 \cdots A_n} \tag{3.48}$$

$$= I_{A_1} \otimes \sqrt{2^{n-1}} \sum_{i=1}^{r} \alpha_i Y_i^{A_2 \cdots A_n} \tag{3.49}$$

$$=: I_{A_1} \otimes W'_{A_2 \cdots A_n}, \tag{3.50}$$

where in the last line we have defined the unitary $W'_{A_2 \cdots A_n} = \sqrt{2^{n-1}} \sum_{i=1}^{r} \alpha_i Y_i^{A_2 \cdots A_n}$.

Now, given the assumption that $\mathcal{E}_1 = \mathcal{I}$, so that W has the form in (3.50), we get that

$$\mathcal{E}_2(\rho) = \mathrm{Tr}_{A_3 \cdots A_n} \left(W' \left(\rho \otimes \frac{I_{A_3 \cdots A_n}}{2^{n-2}} \right) (W')^\dagger \right). \tag{3.51}$$

Therefore, applying the procedure above for $j = 2$ by taking the bipartite cut in the operator Schmidt decomposition of W' to be $A_2|A_3 \cdots A_n$, we get that if $\mathcal{E}_2$ is the identity channel, then $W = I_{A_1} \otimes I_{A_2} \otimes W''$ for some unitary W'' acting on $A_3 \cdots A_n$. Continuing in this manner for all j up to $j = n$, assuming in each case that $\mathcal{E}_j$ is the identity channel, we ultimately obtain $W = I_{A_1} \otimes I_{A_2} \otimes \cdots \otimes I_{A_n}$, which implies that $U = V$, as required. □

3.12 Relation between C_{LHST} and C_{HST}

Proposition 2: Let U and V be $2^n \times 2^n$ unitaries. Then,

$$C_{\text{LHST}}(U, V) \leq C_{\text{HST}}(U, V) \leq nC_{\text{LHST}}(U, V)\,.$$

Proof: First we rewrite the global cost function:

$$\begin{aligned} C_{\text{HST}}(U, V) &= 1 - \frac{1}{d^2}\left|\text{Tr}[V^\dagger U]\right|^2 \\ &= 1 - \text{Tr}[|\Phi^+\rangle\langle\Phi^+|_{AB} \\ &\quad \times (W \otimes I_B)|\Phi^+\rangle\langle\Phi^+|_{AB}(W^\dagger \otimes I_B)], \end{aligned} \tag{3.52}$$

where $W = UV^\dagger$. Also, for the local cost function, we have

$$C_{\text{LHST}}(U, V) \coloneqq \frac{1}{n}\sum_{j=1}^{n} C^{(j)}_{\text{LHST}}(U, V), \tag{3.53}$$

where

$$\begin{aligned} &C^{(j)}_{\text{LHST}}(U, V) \\ &= 1 - \text{Tr}[\Pi_j(W \otimes I_B)|\Phi^+\rangle\langle\Phi^+|_{AB}(W^\dagger \otimes I)] \end{aligned} \tag{3.54}$$

and we have defined

$$\Pi_j \coloneqq I_{A_1B_1} \otimes \cdots \otimes |\Phi^+\rangle\langle\Phi^+|_{A_jB_j} \otimes \cdots \otimes I_{A_nB_n}, \tag{3.55}$$

which are projectors that all mutually commute. Let

$$\rho \coloneqq (W \otimes I_B)|\Phi^+\rangle\langle\Phi^+|_{AB}(W^\dagger \otimes I_B). \tag{3.56}$$

Then, we can write $C_{\text{HST}}(U, V)$ as

$$C_{\text{HST}}(U, V) = 1 - \text{Tr}[\Pi_n \cdots \Pi_1 \rho], \tag{3.57}$$

and we can write $C^{(j)}_{\mathrm{LHST}}(U,V)$ as

$$C^{(j)}_{\mathrm{LHST}}(U,V) = 1 - \mathrm{Tr}[\Pi_j \rho] \tag{3.58}$$

for all $1 \leq j \leq n$. If we associate the events E_j with the projectors Π_j, so that $\Pr[E_j] = \mathrm{Tr}[\Pi_j \rho]$, then, $\mathrm{Tr}[\Pi_n \cdots \Pi_1 \rho] = \Pr\left[\bigcap_{i=1}^{n} E_i\right]$.

To prove (3.28), namely $C_{\mathrm{LHST}}(U,V) \leq C_{\mathrm{HST}}(U,V)$, we recall a basic inequality in probability theory. For any set $\{A_1, A_2, \ldots, A_n\}$ of events, it holds that

$$\Pr\left[\bigcup_{i=1}^{n} A_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} \Pr[A_i]. \tag{3.59}$$

Let us take $A_i = \overline{E_i}$ in (3.59). Then,

$$\Pr\left[\bigcup_{i=1}^{n} \overline{E_i}\right] \geq \frac{1}{n}\sum_{i=1}^{n} \Pr[\overline{E_i}] \tag{3.60}$$

$$\Rightarrow 1 - \Pr\left[\bigcap_{i=1}^{n} E_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} (1 - \Pr[E_i]). \tag{3.61}$$

By definition of the events E_i, the last equality is precisely $C_{\mathrm{HST}}(U,V) \geq C_{\mathrm{LHST}}(U,V)$, as required.

To prove (3.29), we make use of the union bound:

$$\Pr\left[\bigcup_{i=1}^{n} \overline{E_i}\right] \leq \sum_{i=1}^{n} \Pr[\overline{E_i}] \tag{3.62}$$

$$\Rightarrow 1 - \Pr\left[\bigcap_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} (1 - \Pr[E_i]) \tag{3.63}$$

$$= n C_{\mathrm{LHST}}(U,V). \tag{3.64}$$

Given that the left-hand side of the above inequality is precisely $C_{\mathrm{HST}}(U,V)$, we have that $C_{\mathrm{HST}}(U,V) \leq n C_{\mathrm{LHST}}(U,V)$, as required. □

3.13 Proofs of complexity theorems

Theorem 6: Let U and V be poly(n)-sized quantum circuits specified by $2^n \times 2^n$ unitary matrices, and let $\epsilon = O(1/\mathrm{poly}(n))$. Then, the problem of approximating $C_{\mathsf{HST}}(U,V)$ up to ϵ-precision is DQC1-hard.

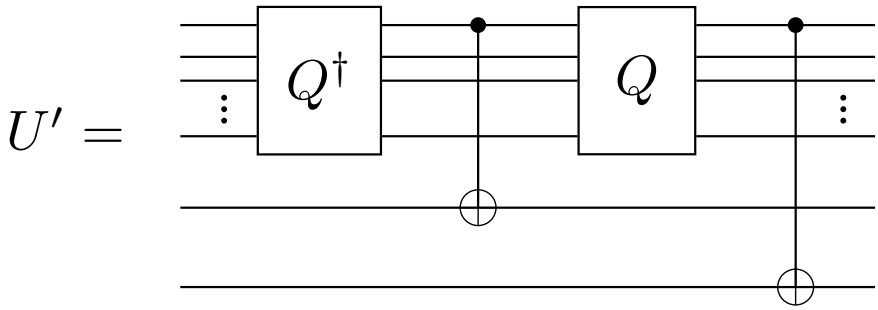


Figure 3.11: The trace of the unitary U' defined by the circuit above is equal to the trace of the non-unitary operator $(|0\rangle\langle 0| \otimes I)Q(|0\rangle\langle 0| \otimes I)Q^\dagger$ up to a factor of 4 [4].

Proof: We show that the problem of approximating the cost $C_{\mathsf{HST}}(U, V)$ is hard for DQC1. In other words, we have to show that any problem in DQC1 reduces to an instance of approximating $C_{\mathsf{HST}}(U, V)$ for some $\epsilon = O(1/\text{poly}(n))$. Recall that, given as input a poly(n)-sized unitary Q on n-qubits, any problem in DQC1 requires us to estimate the acceptance probability p_{acc} when measuring the outcome "0" on input $\rho = |0\rangle\langle 0| \otimes I/2^n$, i.e.

$$p_{\text{acc}} = \text{Tr}[(|0\rangle\langle 0| \otimes I)Q\rho Q^\dagger]. \tag{3.65}$$

Note that, since the above equation describes a probability via the positive semi-definite operator $|0\rangle\langle 0| \otimes I$, the trace will result in a non-negative real number. Let us re-write Eq. (3.65) as follows:

$$p_{\text{acc}} = \frac{1}{2^n}\left|\text{Tr}[(|0\rangle\langle 0| \otimes I)Q(|0\rangle\langle 0| \otimes I)Q^\dagger]\right|. \tag{3.66}$$

When letting U' as in Fig. 3.11, we can also write

$$\text{Tr}[(|0\rangle\langle 0| \otimes I)Q(|0\rangle\langle 0| \otimes I)Q^\dagger] = \text{Tr}[U']/4, \tag{3.67}$$

hence the problem is equivalent to approximating the absolute value of the trace of a unitary U'. In fact, given our choice of U' and when taking V to be the identity, the problem reduces to an instance of approximating the cost $C_{\mathsf{HST}}(U', I)$ up to some precision $\epsilon = O(1/\text{poly}(n))$ via a simple reduction. Therefore, we have shown that the problem of approximating $C_{\mathsf{HST}}(U, V)$ up to ϵ-precision is DQC1-hard. □

Theorem 7: Let U and V be poly(n)-sized quantum circuits specified by $2^n \times 2^n$ unitary matrices, and let $\epsilon = O(1/\text{poly}(n))$. Then, the problem of approximating $C_{\mathsf{LHST}}(U, V)$ up to ϵ-precision is DQC1-hard.

Proof: We show that any problem in DQC1 reduces to an instance of approximating $C_{\mathsf{LHST}}(U, V)$ via a reduction. We are given as input a poly(n)-sized unitary Q on n-qubits, and the task is to estimate the acceptance probability of outputting "0". Our proof strategy is to show that one can efficiently extract $\text{Tr}(U')$, the trace of an n-qubit unitary U', from two distinct evaluations of C_{LHST} and elementary post-processing. This implies that computing C_{LHST} is hard for DQC1, since all problems in DQC1 can be seen as estimating the real part of $\text{Tr}(U')$ via Eq. (3.66) and Eq. (3.67).

The two cost function evaluations that we consider are $C_{\mathsf{LHST}}(U_1, I)$ and $C_{\mathsf{LHST}}(U_2, I)$, where

$$U_1 = U' \tag{3.68}$$

$$U_2 = C_{U'} \,. \tag{3.69}$$

Here, $C_{U'}$ denotes controlled-U' operation.

First, consider U_2 and let the $j = n+1$ qubit correspond to the control qubit for the $C_{U'}$ controlled unitary. Then one can show that

$$C^{(j)}_{\mathsf{LHST}}(U_2, I) = \frac{1}{2} C^{(j)}_{\mathsf{LHST}}(U_1, I) \quad \forall j \in \{1, ..., n\} \,, \tag{3.70}$$

$$C^{(n+1)}_{\mathsf{LHST}}(U_2, I) = \frac{1}{2} - \frac{1}{2^{n+1}} \text{Re}(\text{Tr}(U')) \,. \tag{3.71}$$

This gives

$$C_{\mathsf{LHST}}(U_2, I) = \frac{1}{2(n+1)} \left(1 + n C_{\mathsf{LHST}}(U_1, I) - \frac{\text{Re}(\text{Tr}(U'))}{2^n} \right) . \tag{3.72}$$

For notational simplicity, let $B(U) := 1 - C_{\mathsf{LHST}}(U, I)$. Then, we can rewrite Eqs. (3.72) as

$$B(U_2) = \frac{1}{2} \left(1 + \frac{2^{-n}}{n+1} \text{Re}(\text{Tr}(U')) + \frac{n}{n+1} B(U_1) \right) . \tag{3.73}$$

Hence, we have that

$$\mathrm{Re}(\mathrm{Tr}(U')) = 2^n \left((n+1)(2B(U_2)-1) - nB(U_1)\right) . \tag{3.74}$$

By choosing U' according to Fig. 3.11, one can see from Eq. (3.66) and Eq. (3.67) that the problem is equivalent to ϵ-approximating our local cost function for some $\epsilon = O(1/\mathrm{poly}(n))$. Hence, any DQC1 problem can be efficiently solved for by computing a simple linear combination of two instances of C_{LHST}. Therefore, we have shown that the problem of approximating the cost $C_{\mathsf{LHST}}(U, V)$ is hard for DQC1. □

3.14 Gradient-free optimization method

We now outline our approach to gradient-free optimization of over the continuous gate parameters $\vec{\alpha}$ in the trainable unitary $V_{\vec{k}}(\vec{\alpha})$. This approach was used to obtain the results in Sec. 3.6. Given that this is an implementation for small problem size, we employ the cost function $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$. However, we note that one can replace C_{HST} with our general cost function C_q for larger problem sizes.

Algorithm 1: Gradient-free Continuous Optimization for QAQC via the HST

Input: Unitary U to be compiled; trainable unitary $V_{\vec{k}}(\vec{\alpha})$ of a given structure; error tolerance $\varepsilon' \in (0, 1)$; maximum number of starting points N; maximum number of iterations N_{iter} for `gp_minimize`; sample precision $\delta > 0$.

Output: Parameters $\vec{\alpha}_{\mathrm{opt}}$ such that at best $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}_{\mathrm{opt}})) \leq \varepsilon'$.

Init: $\vec{\alpha}_{\mathrm{opt}} \leftarrow 0$; `cost` $\leftarrow 1$

1 **repeat**
2 choose an initial parameter $\vec{\alpha}^{(0)}$ at random;
3 run `gp_minimize` with $\vec{\alpha}^{(0)}$ and N_{iter} as input and $\vec{\alpha}_{\mathrm{min}}$ as output; whenever the cost is called upon for some $\vec{\alpha}$, run the HST on $V_{\vec{k}}(\vec{\alpha})^*$ and U approximately $1/\delta^2$ times to estimate the cost $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$;
4 **if** `cost` $\geq C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}_{\mathrm{min}}))$ **then**
5 `cost` $\leftarrow C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}_{\mathrm{min}}))$; $\vec{\alpha}_{\mathrm{opt}} \leftarrow \vec{\alpha}_{\mathrm{min}}$
6 **until** `cost` $\leq \varepsilon'$, at most N times.
7 **return** $\vec{\alpha}_{\mathrm{opt}}$, `cost`

Recall that we compute the cost function $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ using the Hilbert-Schmidt Test (HST), as described in Sec. 3.4.1 and illustrated in Fig. 3.4(a). For a given set of gate structure parameters

$\vec{k}$, the calculation of the cost on a quantum computer (as well as on a simulator) is affected by the fact that, due to finite sampling, the HST allows us to obtain only an estimate of the magnitude of the Hilbert-Schmidt inner product. Noise within the quantum computer itself also affects the calculation of the cost. Therefore, in order to perform gradient-free optimization over the continuous gate parameters $\vec{\alpha}$, we make use of stochastic optimization techniques that are designed to optimize noisy functions. Specifically, we make use of the `gp_minimize` routine in the scikit-optimize Python library [105], which is a gradient-free optimization routine that performs Bayesian optimization using Gaussian processes [106, 107]. See Algorithm 1 for a general overview of the optimization procedure. Note that with this algorithm, we obtain an ε-approximate compilation of U, with

$$\varepsilon = \left(\frac{d}{d+1}\right)\varepsilon' . \tag{3.75}$$

In the small-scale quantum computer implementations of Fig. 3.5(c) and Fig. 3.6, we use 50 objective function evaluations in `gp_minimize` per iteration. Note that evaluating the objective function involves running the quantum circuit many times in order to sample from the output distribution of the circuit.

For large problem sizes, as described in Sec. 3.3.4, we propose using the cost function $C_q = qC_{\mathrm{HST}} + (1-q)C_{\mathrm{LHST}}$. The gradient-free continuous parameter optimization algorithm for C_q is similar to the one for C_{HST} in Algorithm 1, except that in addition to running the HST we run the LHST for every qubit $j \in \{1, 2, \ldots, n\}$ in order to compute the local cost C_{LHST}. In this case, the algorithm provides an ε-approximate compilation of U, with

$$\varepsilon = \left(\frac{n}{1-q+nq}\right)\left(\frac{d}{d+1}\right)\varepsilon' . \tag{3.76}$$

We emphasize that our approach to gradient-free optimization avoids the exponential overhead of evaluating the cost function classically, yet at the same time makes use of fast and efficient classical heuristics for optimization. In fact, using the HST, Algorithm 1 requires only $O(1/\delta^2)$ calls to the quantum computer in order to evaluate the cost, where $\delta = 1/\sqrt{n_{\mathrm{shots}}}$ is the sample precision, which is related to the number of samples n_{shots} taken from the device.

3.14.1 Alternative method for gradient-free optimization

Here we propose an alternative algorithm for gradient-free optimization that, on average, significantly reduces the number of times the objective function is evaluated. As a result, it is more suitable for cloud computing under a queue submission system (e.g., IBM's Quantum Experience). This algorithm performs a "multi-scale bisection" of the parameter space based on simulated annealing. We implement this method in Sec. 3.6.1.1 specifically for the hardware of IBM because the queue submission system can require a significant amount of time to make many calls to the quantum computer.

This alternative approach to performing gradient-free continuous parameter optimization is outlined in Algorithm 2. We start with four angles spread uniformly in the interval $[0, 2\pi)$—namely $0, \pi/2, \pi$, and $3\pi/2$. This significantly reduces the size of the search space and allows us to get close to, or find exactly, an optimal gate sequence. Once the optimal structure is reached from this step, we then bisect the angles for each gate $R_z(\alpha)$ by evaluating the cost with a new circuit containing $R_z(\alpha \pm \pi/2^{t+1})$, where $t = 1, 2, \ldots, t_{\max}$ is determined by the iteration in the procedure. Although we do not explore all angles in the interval, the runtime is logarithmically faster than a continuous search due to the bisection procedure. An additional advantage of this approach is that many gates have angles that are simple fractions of π, e.g., $T = R_z(\pi/4)$ and $H = R_z(\pi/2)R_x(\pi/2)R_z(\pi/2)$. In a noiseless environment, the two steps above are sufficient. On actual devices, we implement a third step of stochastic optimization by evaluating the cost for the new circuit with each gate $R_z(\alpha)$ replaced by $R_z(\alpha \pm \Delta(t))$ for some small value $\Delta(t) \ll 1$ decreasing monotonically with the iteration t. This allows us to compile for a given device by accounting for noise and gate errors. This can be thought of as a "fine-grained" angular optimization in contrast to the previous "coarse-grained" angular optimization.

Algorithm 2: Gradient-free Optimization using Bisection for QAQC

Input: Unitary U to be compiled; trainable unitary $V_{\vec{k}}(\vec{\alpha})$ of a given structure and gate alphabet $\mathcal{A}$; error tolerance $\varepsilon' \in (0, 1)$; maximum number of iterations N; maximum number of bisections $t_{\max}$ of the unit circle; sample precision $\delta > 0$.

Output: Parameters $\vec{\alpha}_{\text{opt}}$ such that at best $C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}_{\text{opt}})) \leq \varepsilon'$.

Init: Restrict all gates in $\mathcal{A}$ with continuous parameters to discrete angles in the set $\Omega_0 = \{0, \pi/2, \pi, 3\pi/2\}$; $\alpha_{\text{opt}} \leftarrow 0$; $\texttt{cost} \leftarrow 1$

for $t = 1, 2, \ldots, t_{\max}$ **do**
  repeat
    anneal over all possible bisected angles in the set $\Omega_t := \{\alpha \pm \pi/2^{t+1} \mid \text{for } \alpha \in \Omega_0\} \cup \Omega_{t-1}$;
    whenever the cost is called upon for some $\alpha \in \Omega_t$, run the HST on $V_{\vec{k}}(\vec{\alpha})^*$ and U approximately $1/\delta^2$ times to estimate the cost $C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$;
    if $\texttt{cost} \geq C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ **then**
      $\texttt{cost} \leftarrow C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$;
  until $\texttt{cost} \leq \varepsilon'$ at most N times.
  repeat
    minimize the cost over all small continuous increments $\Delta(t) \ll 1$ within the set of bisected angles Ω_t; whenever the cost is called upon for some $\alpha + \Delta(t)$, with $\alpha \in \Omega_t$, run the HST on $V_{\vec{k}}(\alpha + \Delta(t))^*$ and U approximately $1/\delta^2$ times to estimate the cost $C_{\text{HST}}(U, V_{\vec{k}}(\alpha + \Delta(t)))$;
    if $\texttt{cost} \geq C_{\text{HST}}(U, V_{\vec{k}}(\alpha + \Delta(t)))$ **then**
      $\texttt{cost} \leftarrow C_{\text{HST}}(U, V_{\vec{k}}(\alpha + \Delta(t)))$; $\vec{\alpha}_{\text{opt}} \leftarrow \alpha + \Delta(t)$
  until $\texttt{cost} \leq \varepsilon'$ at most N times.
return $\vec{\alpha}_{\text{opt}}, \texttt{cost}$

3.15 Gradient-based optimization method

We now describe a gradient-based approach to performing the optimization over the continuous parameters in the trainable gate sequence $V_{\vec{k}}(\vec{\alpha})$. In Sec. 3.15.1, we define a new cost function for this purpose, and we introduce a quantum circuit to calculate this cost function on a quantum computer. In Sec. 3.15.2, we present the results of implementing this method on a quantum simulator. In Sec. 3.15.3, we briefly describe how the original cost functions C_{HST} and C_{LHST} can also be optimized using a gradient-based method.

While recent work on gradient descent continuous optimization has shown vast quantum speedups over classical variants [108, 109, 110], the majority of proposals still appear to be

out of reach for implementations on NISQ devices, mainly due to their use of certain algorithmic techniques, such as quantum random-access memory, the quantum Fourier transform, and the Grover search algorithm, which have high resource requirements. Instead, we focus on continuous optimization procedures that are feasible on current quantum computers and leave improvements to our algorithms as an open problem.

The gradient with respect to $\vec{\alpha}$ of the gate sequence $V_{\vec{k}}(\vec{\alpha})$ given by

$$V_{\vec{k}}(\vec{\alpha}) = G_{k_L}(\alpha_L)G_{k_{L-1}}(\alpha_{L-1})\cdots G_{k_1}(\alpha_1), \tag{3.77}$$

is defined by

$$\nabla_{\vec{\alpha}} V_{\vec{k}}(\vec{\alpha}) = \left(\frac{\partial V_{\vec{k}}(\vec{\alpha})}{\partial \alpha_1}, \ldots, \frac{\partial V_{\vec{k}}(\vec{\alpha})}{\partial \alpha_L}\right), \tag{3.78}$$

where the (i, j) matrix element of the ℓ-th component is

$$\left(\frac{\partial V_{\vec{k}}(\vec{\alpha})}{\partial \alpha_\ell}\right)_{i,j} = \frac{\partial V_{\vec{k}}(\vec{\alpha})_{i,j}}{\partial \alpha_\ell}. \tag{3.79}$$

For example, consider the rotation gate $R_z(\alpha) = e^{-i\alpha\sigma_z/2}$, which is parametrized by the angle α. Then, the derivative with respect to α can be written as

$$\frac{\partial}{\partial \alpha} R_z(\alpha) = -\frac{i}{2}\sigma_z R_z(\alpha)\ , \tag{3.80}$$

which follows from the Taylor series expansion of the exponent.

Now, evaluating the gradient on a quantum computer is possible due to the fact that for the gate alphabets we consider in this paper, only the single-qubit gates are parameterized, and these gates are simply rotation gates. In fact, any unitary gate can be decomposed into circuits in which only the single-qubit rotation gates are present. This is illustrated in Fig. 3.12. Furthermore, the circuits in Fig. 3.12(a) and Fig. 3.12(b) are universal for one- and two-qubit gates, respectively (see [6], which also contains universal circuits for n-qubit gates). This means that our gradient-based approach can be applied to any n-qubit unitary without explicitly searching over gate structures, though the compilations obtained in this manner will generally have sub-optimal depth.

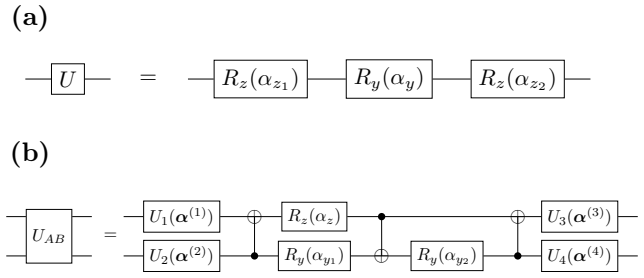


Figure 3.12: (a) Any single-qubit gate U can be decomposed into three elementary rotations (up to a global phase). Given appropriate parameters $\vec{\alpha} = (\alpha_{z_1}, \alpha_y, \alpha_{z_2})$, U can be written as $V(\vec{\alpha}) = e^{-i\alpha_{z_2}\sigma_z/2} e^{-i\alpha_y\sigma_y/2} e^{-i\alpha_{z_1}\sigma_z/2}$. (b) Any two-qubit gate U_{AB} can be decomposed into three CNOT gates as well as 15 elementary single-qubit gates, where each unitary $U_j(\vec{\alpha}^{(j)})$ can be written as in (a). This decomposition is known to be optimal [5], i.e., it uses the least number of continuous parameters and CNOT gates. General universal quantum circuits for n-qubit gates are discussed in [6].

3.15.1 The Power of Two Qubits

Consider the following cost function based on the normalized Hilbert-Schmidt distance between the unitaries U and V:

$$\begin{aligned} C_{\text{POTQ}}(U, V) &\coloneqq \frac{1}{2d}\|U - V\|_{\text{HS}}^2 \\ &= 1 - \frac{1}{d}\text{Re}\left[\text{Tr}(V^\dagger U)\right], \end{aligned} \tag{3.81}$$

where POTQ stands for "Power of Two Qubits" and refers to the circuit used to evaluate it, which we present below. Note that $C_{\text{POTQ}}(U, V)$ is zero if and only if $U = V$. Contrary to the cost function $C_{\text{HST}}(U, V)$, which is defined using the magnitude of the inner product $\langle V, U\rangle$, this cost function is defined using the real part of the inner product. Consequently, it does not vanish if U and V differ only by a global phase. Indeed, if $V = e^{i\varphi}U$, then $C_{\text{POTQ}}(U, V) = 1 - \cos(\varphi)$.

Before discussing the circuit used to evaluate the cost function $C_{\text{POTQ}}(U, V)$, let us review the Power of One Qubit (POOQ) [7], shown in Fig. 3.13(a), which is a circuit for computing the trace of a d-dimensional unitary U. This circuit acts on a d-dimensional system A, initially in the maximally mixed state, I/d, and on a single-qubit ancilla Q initially in the $|0\rangle$ state. After applying a Hadamard gate to Q and a controlled-U gate to QA (with Q the control system), the reduced density matrix ρ_Q has its off-diagonal elements proportional to $\text{Tr}(U)$. Hence, one can measure Q in the X and Y bases, respectively, to read off the real and imaginary parts of $\text{Tr}(U)$.

We now introduce a circuit for computing the real and imaginary parts of $\langle V, U\rangle$ that generalizes the POOQ and is called the Power of Two Qubits (POTQ), depicted in Fig. 3.13(b). As the name suggests, the POTQ employs two single-qubit ancillas, Q and Q', each initially in the $|0\rangle$ state. In addition, two d-dimensional systems, A and B, are initially prepared in the Bell state $|\Phi^+\rangle$ defined in Eq. (3.14). (Although not shown in Fig. 3.13(b), this Bell state is prepared with a depth-two circuit, as shown in Fig. 3.4.)

The first step in the POTQ is to prepare the two-qubit maximally entangled state $\frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)$ between Q and Q', using the Hadamard and CNOT gates as shown in Fig. 3.13(b). The second step is to apply a controlled-U gate between Q and A (with Q the control system). In parallel with this gate, the anticontrolled-V^T gate is applied to $Q'B$, with Q' the control system, where anticontrolled means that the roles of the $|0\rangle$ and $|1\rangle$ states on the control system are reversed in comparison to a controlled gate. This results in the state:

$$\begin{aligned}
&\frac{1}{\sqrt{2}}(|0\rangle_Q|0\rangle_{Q'}(I_A \otimes V^T)|\Phi^+\rangle \\
&\quad + |1\rangle_Q|1\rangle_{Q'}(U \otimes I_B)|\Phi^+\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\rangle_Q|0\rangle_{Q'}(V \otimes I_B)|\Phi^+\rangle \\
&\quad + |1\rangle_Q|1\rangle_{Q'}(U \otimes I_B)|\Phi^+\rangle),
\end{aligned} \tag{3.82}$$

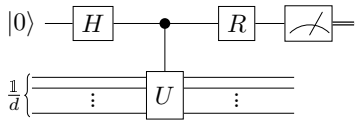
where to obtain the equality we used the ricochet property in Eq. (3.18). As in the HST, note that V itself is not implemented. In this case, its transpose is implemented.

Finally, a CNOT gate is applied to QQ', with Q the control system. This results in the reduced state on Q being

$$\begin{aligned}
\rho_Q = \frac{1}{2}\Big(&|0\rangle\langle 0| + \mathrm{Tr}(V^\dagger U)|0\rangle\langle 1| \\
&+\mathrm{Tr}(U^\dagger V)|1\rangle\langle 0| + |1\rangle\langle 1|\Big).
\end{aligned} \tag{3.83}$$

By inspection of ρ_Q, one can see that measuring Q in the X and Y bases, respectively, gives the real and imaginary parts of $\mathrm{Tr}(V^\dagger U)$.

(a) Power of One Qubit



(b) Power of Two Qubits

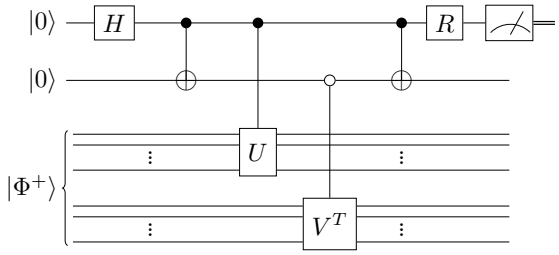


Figure 3.13: **(a)** The Power of One Qubit (POOQ) [7]. This can be used to compute the trace of a unitary U acting on a d-dimensional space. The R gate represents either H, in which case the circuit computes $\mathrm{Re}[\mathrm{Tr}(U)]$, or the S gate followed by H, in which case the circuit computes $\mathrm{Im}[\mathrm{Tr}(U)]$. **(b)** The Power of Two Qubits (POTQ). This is a generalization of the POOQ, as can be seen by setting $V = I$. The POTQ can be used to compute the Hilbert-Schmidt inner product $\mathrm{Tr}(V^\dagger U)$ between two unitaries U and V acting on a d-dimensional space. As with the POOQ, $R = H$ leads to $\mathrm{Re}[\mathrm{Tr}(V^\dagger U)]$, while $R = HS$ leads to $\mathrm{Im}[\mathrm{Tr}(V^\dagger U)]$.

Interestingly, if we set V to the identity in the POTQ, then since the CNOT gate commutes with the controlled-U gate and the reduced state of $|\Phi^+\rangle$ is the maximally mixed state I/d, we recover the POOQ. The POTQ is therefore a generalization of the POOQ.

Note that while the POOQ can also be used to determine $\mathrm{Tr}(V^\dagger U)$, the POTQ has the advantage that the controlled gates for U and V can be executed in parallel, while in the POOQ they would have to be executed in series. This makes the POTQ better suited for NISQ devices, where short depth is crucial. Consider the depth of the POTQ. Denoting the controlled-U and the anticontrolled-V^T as C_U and $\overline{C}_{V^T}$ respectively, the overall depth is

$$D(\mathrm{POTQ}) = 4 + \max\{D(C_U), D(\overline{C}_{V^T})\} \tag{3.84}$$

Note the similarity here to Eq. (3.19). The overall depth is essentially determined by whichever controlled gate has the largest depth.

3.15.2 Gradient-based optimization via the POTQ

The gradient with respect to $\vec{\alpha}$ of $C_{\mathrm{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}))$ can be computed using the POTQ. This is due to the fact that

$$\frac{\partial}{\partial \alpha_\ell} \mathrm{Re}\left[\mathrm{Tr}(V_{\vec{k}}(\vec{\alpha})^\dagger U)\right] = \frac{1}{2}\,\mathrm{Re}\left[\mathrm{Tr}\left(\widetilde{V}^{(\ell)}_{\vec{k}}(\vec{\alpha})^\dagger U\right)\right], \tag{3.85}$$

where

$$\begin{aligned}\widetilde{V}^{(\ell)}_{\vec{k}}(\vec{\alpha}) \coloneqq\ & G_{k_L}(\alpha_L)\cdots G_{k_{\ell+1}}(\alpha_{\ell+1})(-i\sigma_{k_\ell}) \\ & \times G_{k_\ell}(\alpha_\ell) G_{k_{\ell-1}}(\alpha_{\ell-1})\cdots G_{k_1}(\alpha_1)\end{aligned} \tag{3.86}$$

is the original gate sequence $V_{\vec{k}}(\vec{\alpha})$ except with an additional Pauli gate σ_{k_ℓ} corresponding to the variable with respect to which the derivative is taken. (Note that for the gate alphabets that we consider in this paper, only the single-qubit gates are parameterized, and these gates are simply rotation gates. The derivative of any one-qubit rotation gate is analogous to the expression in (3.80) for the derivative of the rotation gate $R_z(\alpha)$.) This means that to compute the gradient of $C_{\mathrm{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}))$, we simply add the appropriate local Pauli gate to the original gate sequence and run the POTQ on this new gate sequence.

Our gradient-based optimization procedure is outlined in Algorithm 3. Given an arbitrary unitary U as input, Algorithm 3 compiles U to a unitary $V_{\vec{k}}(\vec{\alpha}_{\mathrm{opt}})$ of a given structure $\vec{k}$ that minimizes the cost C_{POTQ}. The gradient is evaluated with the POTQ circuit as a subroutine within a classical gradient-descent algorithm. The overall query complexity in the number of calls to the cost evaluation routine of Algorithm 3 is $O(NTL/\delta^2)$, where $\delta = 1/\sqrt{n_{\mathrm{shots}}}$ is the sample precision, N is the maximum number of repetitions over random initial parameters $\vec{\alpha}^0$, L is the dimension of the continuous parameter space of $\vec{\alpha}$, and T is the number of gradient descent iterations for a suitable learning rate $\eta > 0$. In order to improve convergence, it may also be useful to supply the quantum subroutines for computing the cost function and the gradient to a more advanced minimization routine, for example as found in the Python library SciPy [61]. We present below the results on compiling both single-qubit and two-qubit gates on a simulator.

When performing Algorithm 3, we rely on the ability to perform the controlled-U gate. The unitary U may be unknown, e.g., as in Fig. 3.1(b). In general, to perform a controlled operation

Algorithm 3: Gradient-based Continuous Optimization for QAQC via the POTQ

Input: Unitary U to be compiled; a trainable unitary $V_{\vec{k}}(\vec{\alpha})$ of a given structure, where $\vec{\alpha}$ is a continuous circuit parameter of dimension L; maximum number of iterations N; error tolerance $\varepsilon' \in (0, 1)$; learning rate $\eta > 0$; sample precision $\delta > 0$.

Output: Parameters $\vec{\alpha}_{\text{opt}}$ such that at best $C_{\text{POTQ}}(U, V_{\vec{k}}(\vec{\alpha_{\text{opt}}})) \leq \varepsilon'$.

Init: $\vec{\alpha}_{\text{opt}} \leftarrow 0$; $\texttt{cost} \leftarrow 1$

repeat
- choose initial parameters $\vec{\alpha}^{(0)}$ at random
- **for** $\tau = 1, 2, \ldots, T$ **do**
  - **for** $i = 1, 2, \ldots, L$ **do**
    - run the POTQ on $\partial_{\alpha_i} V_{\vec{k}}(\vec{\alpha}^{(\tau-1)})^T$ and U approximately $1/\delta^2$ times to estimate $\text{Re}\left(\text{Tr}\left[\partial_{\alpha_i} V_{\vec{k}}(\vec{\alpha}^{(\tau-1)})^\dagger U\right]\right)$
  - **update** $\vec{\alpha}^{(\tau)} \leftarrow \vec{\alpha}^{(\tau-1)} - \eta \nabla_{\vec{\alpha}} C_{\text{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}^{(\tau-1)}))$
- run the POTQ on $V_{\vec{k}}(\vec{\alpha}^{(\tau)})^T$ and U approximately $1/\delta^2$ times to estimate the cost $C_{\text{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}^{(\tau)}))$
- **if** $\texttt{cost} \geq C_{\text{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}^{(\tau)}))$ **then**
  - $\texttt{cost} \leftarrow C_{\text{POTQ}}(U, V_{\vec{k}}(\vec{\alpha}^{(\tau)}))$; $\vec{\alpha}_{\text{opt}} \leftarrow \vec{\alpha}^{(\tau)}$

until $\texttt{cost} \leq \varepsilon'$, at most N times

return $\vec{\alpha}_{\text{opt}}, \texttt{cost}$

with respect to a target unitary U, one can use a method for "remote control" [111]. This method employs a local U gate and controlled-SWAP operations in order to realize the controlled-U gate. In practice, since any controlled unitary gate can be decomposed into native gates, the ability to compile controlled-SWAP, the Toffoli gate, and the set of controlled rotations is sufficient. In order to perform such a translation, we allow the user to have access to a small-scale classical compiler. This does not incur exponential overhead since the gates to be translated are one- and two-qubit gates (or their controlled versions). While this may cause the depth of our compiled unitary to increase, it will only be by a constant factor.

We note that decoherence, gate infidelity, and readout errors on NISQ computers are all more pronounced when attempting to execute controlled unitaries. This means that there is significant performance loss for controlled unitaries, as required in the POTQ. Consequently, we did not implement our gradient-based optimization method on current quantum devices, but we speculate that improvements to quantum hardware will enable this application.

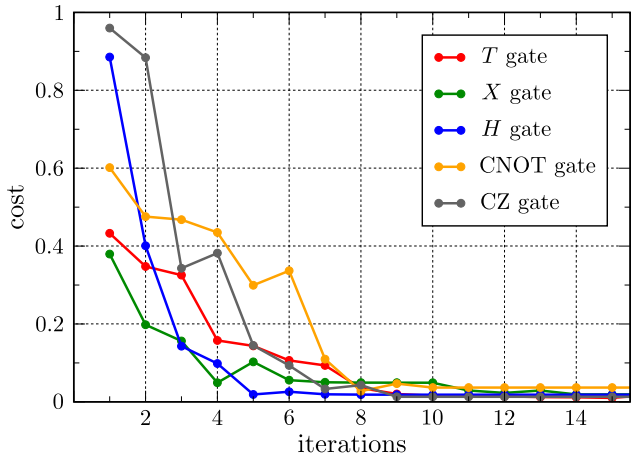


Figure 3.14: Compiling one- and two-qubit gates on a simulator with the gate alphabet in (3.35) using the gradient-based optimization technique described in Algorithm 3, with $n_{\text{shots}} = 10,000$. Shown is the cost as a function of the number of gradient calls of the continuous parameter optimization using the `minimize` routine in the SciPy-optimize Python library. The gate structure for the single-qubit gates is fixed to the one shown in Fig. 3.12(a), while the gate structure for the two-qubit gates is fixed to the one shown in Fig. 3.12(b).

3.15.2.1 Implementation on a quantum simulator

We use IBM's simulator [3] to compile a selection of single-qubit and two-qubit gates by performing the gradient-based optimization procedure in Algorithm 3. In order to improve convergence, we additionally supply the gradient, as well as the cost function, to the `minimize` routine in the SciPy-optimize Python library [61]. For the single-qubit gates, we assume a fixed structure for the trainable gate sequence according to the decomposition in Fig. 3.12(a), while for the two-qubit gates we assume a fixed structure for the trainable gate sequence according to the decomposition in Fig. 3.12(b). We compile the T gate, X gate, Hadamard (H) gate, as well as the CNOT and CZ gates, all with $n_{\text{shots}} = 10,000$. The results are shown in Fig. 3.14. We note that increasing n_{shots} to higher orders of magnitude significantly reduces the sampling error and results in more stable convergence at the cost of an increase in runtime.

3.15.3 Gradient-based optimization via the HST and LHST

We now show that it is possible to perform gradient-based optimization of the original cost function C_{HST} and its local variant C_{LHST}. This allows us to perform gradient-based optimization of the

general cost function $C_q = qC_{\text{HST}} + (1-q)C_{\text{LHST}}$. The algorithm for gradient-based optimization of C_{HST} and C_{LHST} is presented in Algorithm 4.

The gradient with respect to $\vec{\alpha}$ of both $C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ and $C_{\text{LHST}}(U, V_{\vec{k}}(\vec{\alpha}))$ can be computed using the HST and the LHST, respectively. Specifically, for a gate sequence of the form in (3.77), in which the only parameterized gates are the single-qubit rotation gates, we have that

$$\begin{aligned}\frac{\partial}{\partial \alpha_\ell} C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha})) = \frac{1}{2} C_{\text{HST}}(U, \widehat{V}^{(\ell)}_{\vec{k},+}(\vec{\alpha})) \\ - \frac{1}{2} C_{\text{HST}}(U, \widehat{V}^{(\ell)}_{\vec{k},-}(\vec{\alpha})),\end{aligned} \tag{3.87}$$

and

$$\begin{aligned}\frac{\partial}{\partial \alpha_\ell} C^{(j)}_{\text{LHST}}(U, V_{\vec{k}}(\vec{\alpha})) = \frac{1}{2} C^{(j)}_{\text{LHST}}(U, \widehat{V}^{(\ell)}_{\vec{k},+}(\vec{\alpha})) \\ - \frac{1}{2} C^{(j)}_{\text{LHST}}(U, \widehat{V}^{(\ell)}_{\vec{k},-}(\vec{\alpha}))\end{aligned} \tag{3.88}$$

for all $j \in \{1, 2, \ldots, n\}$. Here,

$$\begin{aligned}\widehat{V}^{(\ell)}_{\vec{k},\pm}(\vec{\alpha}) \coloneqq G_{k_L}(\alpha_L) \cdots G_{k_{\ell+1}}(\alpha_{\ell+1}) G_{k_\ell}\left(\pm\frac{\pi}{2}\right) \\ \times G_{k_\ell}(\alpha_\ell) G_{k_{\ell-1}}(\alpha_{\ell-1}) \cdots G_{k_1}(\alpha_1)\end{aligned} \tag{3.89}$$

is the original gate sequence $V_{\vec{k}}(\vec{\alpha})$ with an additional rotation gate $G_{k_\ell}\left(\pm\frac{\pi}{2}\right)$ corresponding to the variable with respect to which the derivative is taken. In other words, to compute the gradient of the cost function $C_{\text{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$, we run the HST in Fig. 3.4(a) twice, once with the gate sequence $\widehat{V}^{(\ell)}_{\vec{k},+}(\vec{\alpha})$ and once with the gate sequence $\widehat{V}^{(\ell)}_{\vec{k},-}(\vec{\alpha})$. Similarly, to compute the gradient of the functions $C^{(j)}_{\text{LHST}}(U, V_{\vec{k}}(\vec{\alpha}))$, we run the LHST in Fig. 3.4(b) twice, once with the gate sequence $\widehat{V}^{(\ell)}_{\vec{k},+}(\vec{\alpha})$ and once with the gate sequence $\widehat{V}^{(\ell)}_{\vec{k},-}(\vec{\alpha})$.

The expressions for the gradient in (3.87) and (3.88) can be verified by recalling that only the one-qubit gates need to be parameterized and that they can always be assumed to have the form $e^{-i\alpha\sigma/2}$ for some Pauli operator σ, where α is the continuous parameter specifying the gate. Then,

for the gate sequence $V_{\vec{k}}(\vec{\alpha})$ in (3.77), we get

$$\begin{aligned}\frac{\partial V_{\vec{k}}(\vec{\alpha})}{\partial \alpha_\ell} &= G_{k_L}(\alpha_L)\cdots G_{k_{\ell+1}}(\alpha_{\ell+1})\frac{\partial G_{k_\ell}(\alpha_\ell)}{\partial \alpha_\ell} \\ &\quad \times G_{k_{\ell-1}}(\alpha_{\ell-1})\cdots G_{k_1}(\alpha_1) \qquad (3.90)\\ &= -\frac{i}{2}G_{k_\ell}(\alpha_\ell)\cdots G_{k_{\ell+1}}(\alpha_{\ell+1})\sigma_{k_\ell}G_{k_\ell}(\alpha_\ell) \\ &\quad \times G_{k_{\ell-1}}(\alpha_{\ell-1})\cdots G_{k_1}(\alpha_1) \qquad (3.91)\end{aligned}$$

Then, we use the identity

$$\begin{aligned}i[\sigma_{k_\ell},\rho] = G_{k_\ell}\left(-\frac{\pi}{2}\right)\rho G_{k_\ell}\left(-\frac{\pi}{2}\right)^\dagger \\ - G_{k_\ell}\left(\frac{\pi}{2}\right)\rho G_{k_\ell}\left(\frac{\pi}{2}\right)^\dagger,\end{aligned} \qquad (3.92)$$

which holds for any state ρ. We also observe that both the functions $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$ and $C^{(j)}_{\mathrm{LHST}}(U, V_{\vec{k}}(\vec{\alpha}))$ are of the form

$$F(\vec{\alpha}) = \mathrm{Tr}[H(U\otimes V_{\vec{k}}(\vec{\alpha})^*)\rho(U^\dagger\otimes V_{\vec{k}}(\vec{\alpha})^T)], \qquad (3.93)$$

where $\rho = |\Phi^+\rangle\langle\Phi^+|_{A_1\cdots A_n}$ for both functions, $H = |\Phi^+\rangle\langle\Phi^+|_{A_1\cdots A_n}$ for $C_{\mathrm{HST}}(U, V_{\vec{k}}(\vec{\alpha}))$, and $H = |\Phi^+\rangle\langle\Phi^+|_{A_jB_j}\otimes I_{\bar{A}_j\bar{B}_j}$ for $C^{(j)}_{\mathrm{LHST}}(U, V_{\vec{k}}(\vec{\alpha}))$. Finally, using

$$\begin{aligned}\frac{\partial F(\vec{\alpha})}{\partial \alpha_\ell} &= \mathrm{Tr}\left[H\left(U\otimes\left(\frac{V_{\vec{k}}(\vec{\alpha})}{\partial\alpha_\ell}\right)^*\right)\rho(U^\dagger\otimes V_{\vec{k}}(\vec{\alpha})^T)\right] \\ &+ \mathrm{Tr}\left[H(U\otimes V_{\vec{k}}(\vec{\alpha})^*)\rho\left(U^\dagger\otimes\left(\frac{\partial V_{\vec{k}}(\vec{\alpha})}{\partial\alpha_\ell}\right)^T\right)\right],\end{aligned} \qquad (3.94)$$

substituting (3.91) into this expression, and using (3.92) to simplify, we obtain (3.87) and (3.88).

The quantum algorithms we developed in the first part of this thesis have several advantages for near-term quantum computers, but in order to scale them to problem sizes that are challenging for classical methods, several problems must be solved. Such problems include optimization strategies and techniques for dealing with errors. In this second part of the thesis, we zoom in on the problem of dealing with errors. As we have described in Chapter 1, the usual long-term solution is quantum error correction and fault tolerance [112, 113, 114], but this requires significant overhead beyond

Algorithm 4: Gradient-based Continuous Optimization for QAQC via the (L)HST

Input: Unitary U to be compiled; a trainable unitary $V_{\vec{k}}(\vec{\alpha})$ of a given structure, where $\vec{\alpha}$ is a continuous circuit parameter of dimension L; maximum number of iterations N; gradient tolerance $\varepsilon' \in (0, 1)$; sample precision $\delta > 0$; cost function $C \in \{C_{\text{HST}}, C_{\text{LHST}}\}$.

Output: Parameters $\vec{\alpha}_{\text{opt}}$ such that at best $\|\nabla_{\vec{\alpha}} C(U, V_{\vec{k}}(\vec{\alpha_{\text{opt}}}))\|^2 \leq \varepsilon'$.

Init: $\vec{\alpha}_{\text{opt}} \leftarrow 0$; $\texttt{cost} \leftarrow 0$; $\texttt{grad} \leftarrow \infty$; $\tau \leftarrow 0$; $\texttt{gradCount} \leftarrow 0$; $\eta \leftarrow 1$

choose initial parameters $\vec{\alpha}^{(0)}$ at random
$\texttt{cost} \leftarrow C(U, V_{\vec{k}}(\vec{\alpha}^{(0)}))$
while $\texttt{count} < N$ *and* $\texttt{gradCount} < 4$ **do**
  $\tau \leftarrow \tau + 1$
  for $i = 1, 2, \ldots, L$ **do**
    Calculate $\frac{\partial C}{\partial \alpha_i}$ using either (3.87) or (3.88), taking approximately $\frac{1}{\delta^2}$ samples for each circuit.
  $\texttt{grad} \leftarrow \|\nabla_{\vec{\alpha}} C(U, V_{\vec{k}}(\vec{\alpha}^{(\tau-1)}))\|^2$
  if $\texttt{grad} \leq \varepsilon'$ **then**
    $\texttt{gradCount} \leftarrow \texttt{gradCount} + 1$
  $\vec{\alpha}_1^{(\tau-1)} \leftarrow \vec{\alpha}^{(\tau-1)} - \eta \nabla_{\vec{\alpha}} C(U, V_{\vec{k}}(\vec{\alpha}^{(\tau-1)}))$
  $\vec{\alpha}_2^{(\tau-1)} \leftarrow \vec{\alpha}_1^{(\tau-1)} - \eta \nabla_{\vec{\alpha}} C(U, V_{\vec{k}}(\vec{\alpha}^{(\tau-1)}))$
  if $\texttt{cost} - C(U, V_{\vec{k}}(\vec{\alpha}_2^{(\tau-1)})) \geq \eta \cdot \texttt{grad}$ **then**
    $\eta \leftarrow 2\eta$
    $\alpha^{(\tau)} \leftarrow \alpha_2^{(\tau-1)}$
  else if $\texttt{cost} - C(U, V_{\vec{k}}(\vec{\alpha}_1^{(\tau-1)})) < \frac{\eta}{2} \cdot \texttt{grad}$ **then**
    $\eta \leftarrow \frac{\eta}{2}$
    $\alpha^{(\tau)} \leftarrow \alpha_1^{(\tau-1)}$
  else
    $\alpha^{(\tau)} \leftarrow \alpha_1^{(\tau-1)}$
  $\texttt{cost} \leftarrow C(U, V_{\vec{k}}(\vec{\alpha}^{(\tau)}))$
  $\vec{\alpha}_{\text{opt}} \leftarrow \vec{\alpha}^{(\tau)}$
return $\vec{\alpha}_{\text{opt}}, \texttt{cost}$

current experimental capabilities. It is therefore interesting and important to develop methods for dealing with errors with less overhead. The general name used to refer to this is *quantum error mitigation* (QEM) [115].

A central task in near-term quantum algorithms (and many other areas of quantum information processing) is to estimate the expectation value of an observable O with respect to a pure state $\rho = |\psi\rangle\langle\psi|$, i.e. $\langle O\rangle = \langle\psi|O|\psi\rangle = \mathrm{Tr}(\rho O)$. If the state is prepared by a quantum device, it can be noisy, and we instead evaluate $\langle O\rangle_{\mathrm{noisy}} = \mathrm{Tr}(\rho_{\mathcal{E}} O)$ where $\rho_{\mathcal{E}} = \mathcal{E}(\rho)$ denotes the noisy state that is corrupted from the target state ρ by an unknown noisy quantum channel $\mathcal{E}$. Given the corrupted state $\rho_{\mathcal{E}}$, the goal of quantum error mitigation is to estimate a quantity $\langle O\rangle_{\mathrm{QEM}}$ that is closer to the target value $\langle O\rangle$ compared to the noisy result $\langle O\rangle_{\mathrm{noisy}}$. In other words, we seek to compute $\langle O\rangle_{\mathrm{QEM}}$ such that

$$\left|\langle O\rangle_{\mathrm{QEM}} - \langle O\rangle\right| < \left|\langle O\rangle_{\mathrm{noisy}} - \langle O\rangle\right|. \tag{3.95}$$

A relatively large number of QEM techniques have been proposed in recent literature including zero-noise extrapolation [116, 117], probabilistic error cancellation [116, 118], randomized compiling [119], Pauli-frame randomization [120], dynamical decoupling [121, 122, 123, 124], quantum optimal control [125, 126], subspace expansion [127], virtual distillation [128, 129] and others [130, 131, 132, 133, 134]. Some authors have defined common frameworks which encapsulate one or more of these techniques [135, 136]. At its core, any QEM technique uses additional quantum resources (qubits, gates, and/or samples) in a clever way to approximate what would happen in an ideal device.

CHAPTER 4

ADVANCES IN ZERO-NOISE EXTRAPOLATION

4.1 Digital and adaptive ZNE

4.1.1 Introduction

Zero-noise extrapolation (ZNE) was introduced concurrently in [116] and [117]. In ZNE, a quantum program is altered to run at different effective levels of processor noise. The result of the computation is then extrapolated to an estimated value at a noiseless level. More formally, one can parameterize the noise-level of a quantum system with a dimensionless scale factor λ. For $\lambda = 0$ the noise is removed, while for $\lambda = 1$ the true noise-level of the physical hardware is matched. For example, λ could be a multiplicative factor that scales the dissipative terms of a master equation [116]. More generally, λ could represent a re-scaling of any physical quantity which introduces some noise in the quantum computation: the calibration uncertainty of variational parameters, the temperature of the quantum processor, etc.

For a given quantum program, we can measure an arbitrary expectation value $E(\lambda)$. By construction, $E(1)$ represents the expectation value evaluated with the natural noise of the hardware, whereas $E(0)$ denotes the noiseless observable which, despite being not directly measurable, we would like to estimate.

To implement ZNE, one needs a direct or indirect way to scale the quantum computation's noise level to values of λ larger than one. With such a method, ZNE can be implemented in two main steps:

1. **Noise-scaling:** Measure $E(\lambda)$ at m different values of $\lambda \geq 1$.

2. **Extrapolation:** Infer $E(0)$ from the m expectation values measured in previous step.

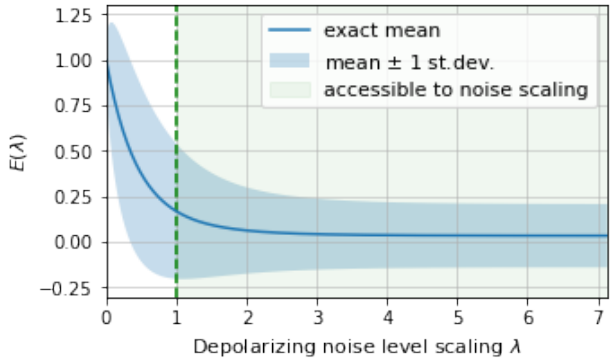


Figure 4.1: An example of the change of an expectation value, $E(\lambda)$, with the underlying scaling λ of the depolarizing noise level. Here the simulated base noise value is 5% (marked by the green dashed vertical line). ZNE increases that noise and back extrapolates to the $\lambda = 0$ expectation value. In this example, an accurate extrapolation should be non-linear and take advantage of a known asymptotic behavior.

Figure 4.1 shows an example *noise curve* given by scaling depolarizing noise for a randomized benchmarking circuit.

In this work, we introduce improvements to both noise-scaling and extrapolation methods for quantum error mitigation. In Section 4.1.2.1 we introduce unitary folding, a framework for digital noise scaling of generic gate noise. We then move to the extrapolation step of ZNE, which we characterize as an inference problem. We study non-adaptive (Section 4.1.3) extrapolation methods and introduce adaptive (Section 4.1.4) extrapolation to improve performance and reduce resource overhead for ZNE.

4.1.2 Noise scaling methods

In [116] and [137] a time-scaling approach implements the scaling of effective noise on the back-end quantum processor. Control pulses for each gate are re-calibrated to execute the same unitary evolution but applied over a longer amount of time. This effectively scales up the noise. While successfully used to suppress errors in single and two-qubit quantum programs on a superconducting quantum processor [137], time-scaling has some disadvantages:

- It requires programmer access to low-level physical-control parameters. This level of access

is not available on all quantum hardware and breaks the gate model abstraction.

- Control pulses must be re-calibrated for each time duration and error-scaling. This calibration can be resource intensive.

Instead, we study alternative approaches that require only a gate-level access to the system. Rather than increasing the time duration of each gate, we increase the total number of gates or, similarly, the circuit depth. This procedure is similar to what is usually done by a *q*uantum compiler but with the opposite goal: instead of optimizing a circuit to reduce its depth or its gate count, we are interested in "de-optimizing" to increase the effect of noise and decoherence. We use the term *digital* to describe noise-scaling techniques that manipulate just the quantum program at the instruction set layer. Their advantage is that they can be used with the gate model access that is common to most quantum assembly languages [138, 139, 140]. Low level access to pulse shaping and detailed physical knowledge of quantum processor physics is no longer required. Our digital framework incorporates and generalizes some recent related work [141, 142].

4.1.2.1 Unitary folding

We describe two methods–circuit folding and gate folding–for scaling the effective noise of a quantum computation based on *u*nitary folding, i.e., replacing a unitary circuit (or gate) U by:

$$U \rightarrow U(U^\dagger U)^n, \tag{4.1}$$

where n is a positive integer. In an ideal circuit, since $U^\dagger U$ is equal to the identity, this folding operation has no logical effect. However, on a real quantum computer, we expect that the noise increases since the number of physical operations scales by a factor of $1 + 2n$. This effect is clearly visible in the quantum computing experiment reported in Figure 4.5.

A similar trick was used in Ref. [141, 142], where noise was artificially increased by inserting pairs of CNOT gates into quantum circuits. In our framework, U can represent the full input circuit or, alternately, some local gates which are inserted with different strategies.

4.1.2.2 Circuit folding

Assume that the circuit is composed of d unitary layers:

$$U = L_d ... L_2 L_1, \tag{4.2}$$

where d represents the $d$epth of the circuit and each block L_j can either represent a single layer of operations or just a single gate.

In circuit folding, the substitution rule in Eq. (4.1) is applied globally, i.e., to the entire circuit. This scales the effective depth by odd integers. In order to have a more fine-grained resolution of the scaling factor, we can also allow for a final folding applied to a subset of the circuit corresponding to its last s layers. The general $c$ircuit folding replacement rule is therefore:

$$U \rightarrow U(U^\dagger U)^n L_d^\dagger L_{d-1}^\dagger \dots L_s^\dagger L_s \dots L_{d-1} L_d. \tag{4.3}$$

The total number of layers of the new circuit is $d(2n + 1) + 2s$. This means that we can stretch the depth of a circuit up to a scale resolution of $2/d$, i.e., we can apply the scaling $d \rightarrow \lambda d$, where:

$$\lambda = 1 + \frac{2k}{d}, \quad k = 1, 2, 3, \dots. \tag{4.4}$$

Conversely, for every real λ, one can apply the following procedure:

1. Determine the closest integer k to the real quantity $d(\lambda - 1)/2$.

2. Perform an integer division of k by d. The quotient corresponds to n and the reminder to s.

3. Apply n integer foldings and a final partial folding as described in Eq. (4.3).

From a physical point of view, the circuit folding method corresponds to repeatedly driving the Hamiltonian of the qubits forwards and backwards in time, such that the ideal unitary part of the dynamics is not changed while the non-unitary effect of the noise is amplified.

Table 4.1: Different methods for implementing gate (or layer) folding

Method	**Subset of indices to fold**
From left	$S = \{1, 2, \ldots, s\}$
From right	$S = \{d, d-1, \ldots, d-s+1\}$
At random	$S = s$ different indices randomly sampled without replacement from $\{1, 2, \ldots, d\}$.

4.1.2.3 Gate (or layer) folding

Instead of globally folding a quantum circuit, appending the folds at the end, one could fold a subset of individual gates (or layers) in place. Let us consider the circuit decomposition of Eq. (4.2) where we can assume that each unitary operator L_j represents just a single gate applied to one or two qubits of the system or, alternatively, each L_j could be a layer of several gates.

If we apply the replacement rule given in Eq. (4.1) to each gate (or layer) L_j of the circuit, it is clear that the initial number of gates (layers) d is scaled by an odd integer $1 + 2n$. Similarly to the case of circuit folding, we can add a final partial folding operation to get a scaling factor which is more fine grained. In order to achieve such "partial" folding, let us define an arbitrary subset S of the full set of indices $\{1, 2, \ldots d\}$, such that its number of elements is a given integer $s = |S|$. In this setting, we can define the following *g*ate (layer) folding rule:

$$\forall j \in \{1, 2, \ldots d\}, \quad L_j \rightarrow \begin{cases} L_j (L_j^\dagger L_j)^n & \mathrm{if}\, j \notin S, \\ \\ L_j (L_j^\dagger L_j)^{n+1} & \mathrm{if}\, j \in S. \end{cases} \tag{4.5}$$

Depending on how we chose the elements of the subset S, different noise channels will be added at different positions along the circuit and so we can have different results. The optimal choice may depend on the particular circuit and noise model. We focus on three different ways of selecting the subset of gates (layers) to be folded: *f*rom left, *f*rom right and *a*t random. Depending on the method, the prescription for selecting the subset S of indices is reported in Table 4.1.

It is easy to check that the number of gates (or layers), obtained after the application of the gate folding rule given in Eq. (4.5) is $d(2n + 1) + 2s$. This is exactly the same number obtained after the application of the global circuit-folding rule given in Eq. (4.3). As a consequence, the number

of gates (layers) is still stretched by a factor λ, i.e., $d \to \lambda d$, where λ can take the specific values reported in Eq. (4.4). Moreover, if we are given an arbitrary λ and we want to determine the values of n and s, we can simply apply the same procedure that was given in the case of circuit-folding.

While preparing this manuscript we became aware of [141] whose technique is similar to our gate folding (at random). The main difference is that [141] focuses mainly on CNOT gates and uses random sampling with replacement, in our case any gate (or layer) can be folded and the sampling is performed without replacement. The rationale of this choice is to sample in a more uniform way the input circuit, and to converge smoothly to the odd integer values of $\lambda = 1 + 2n$ where all the input gates are folded exactly n times.

4.1.2.4 Advantages and limitations of unitary folding

The main advantage of the unitary folding approach is that is is digital, i.e., noise is scaled using a high level of abstraction from the physical hardware. Moreover, it can be applied without knowing the details of the underlying noise-model. It is natural to ask: how justified is this approach physically? Does unitary folding actually correspond to an effective scaling of the physical noise of the hardware?

For example, unitary folding may fail to amplify systematic and coherent errors since applying the inverse of a gate will usually *u*ndo such errors instead of increasing them. It is also clear that unitary folding is not appropriate to scale state preparation and measurement (SPAM) noise, since this noise is independent of the circuit depth. Instead, we expect that unitary folding can be used for scaling incoherent noise models which are associated both to the application of individual gates and/or to the time-length of the overall computation. The more we increase the depth of the circuit, the more such kinds of noise are usually amplified. In this work this intuition is confirmed by numerical and experimental examples in which unitary folding is successfully used for implementing ZNE (see Figures 4.2, 4.3, 4.4 and 4.5).

The effect of unitary folding can be analytically derived when the noise-model for each gate L_j

is a global depolarizing channel with a gate-dependent parameter $p_j \in [0, 1]$, acting as:

$$\rho \xrightarrow{\text{noisy gate}} p_j L_j \rho L_j^\dagger + (1 - p_j)\mathbb{I}/D, \tag{4.6}$$

where D is the dimension of the Hilbert space associated to all the qubits of the circuit. Since the depolarizing channel commutes with unitary operations, we can postpone the noise channels of all the gates until the end of the full circuit U, resulting into a single final depolarizing channel:

$$\rho \xrightarrow{\text{noisy circuit}} pU\rho U^\dagger + (1 - p)\mathbb{I}/D, \tag{4.7}$$

where $p = \Pi_j p_j$ is the product of all the gate-dependent noise parameters p_j. This simple commutation property does not hold for local depolarizing noise, unless we are dealing with singe-qubit circuits.

Consider what happens if we apply unitary folding with a scale factor $\lambda = 1 + 2n$ (odd positive integer). For both the circuit folding and the gate folding methods, defined in Eq. (4.3) and (4.5) respectively, the final result is exactly equivalent to an exponential scaling of all the depolarizing parameters of each gate $p_j \to p_j^\lambda$ or, equivalently, to the global operation:

$$\rho \xrightarrow{\text{noise + unitary folding}} p^\lambda U\rho U^\dagger + (1 - p^\lambda)\mathbb{I}/D. \tag{4.8}$$

This implies that unitary folding is equivalent to an exponential parameterization of the noise level p, and so any expectation value is also scaled according to an exponential ansatz:

$$E(\lambda) = a + bp^\lambda, \tag{4.9}$$

which we can fit and extrapolate according to the methods discussed in the Sections 4.1.3 and 4.1.4.

Equations (4.8) and (4.9) are valid only for depolarizing noise and for odd scaling factors λ. For gate-independent depolarizing noise, the global parameter p is a function of the total number of gates only. This means that all the folding methods (circuit, from left, from right and at random) become equivalent, and induce the exponential scalings of Eqs. (4.8) and (4.9)) for all values of λ.

4.1.2.5 Numerical results

We executed density matrix simulations using unitary folding for zero-noise extrapolation. Broadly these results show that unitary folding is effective in a variety of situations. Furthermore, we benchmark on both random circuits and a variational algorithm at 6 and more qubits. This extends previous work that focuses on the single and two qubit cases [116, 117, 137, 118]. Figure 4.2 shows a simulated two qubit randomized benchmarking experiment under 1% depolarizing noise with and without error-mitigation. Noise was scaled using circuit folding as described in Section 4.1.2.2.

Figure 4.3 shows the distribution of noise reduction by ZNE with circuit folding on randomly generated six qubit circuits. Let E_m be the mitigated expectation value of a circuit after zero-noise extrapolation. Then $R_m = |E_m - E(0)|$ is the absolute value of the error in the mitigated expectation and $R_u = |E(1) - E(0)|$ is the absolute value of the error of the unmitigated circuit. The improvement from ZNE is quantified as R_u/R_m.

Table 4.2 (see Section 4.1.3) provides a comparison different combinations of folding and extrapolation techniques on a set of randomized benchmarking circuits.

Figure 4.4 shows the performance of unitary folding ZNE on a variational algorithm. Using exact density matrix simulation we study the percentage closer to optimal achieved by the quantum approximation optimization algorithm [143] on random instances of MAXCUT.

4.1.3 Non-adaptive extrapolation methods: Zero noise extrapolation as statistical inference

In Section 4.1.2, we discussed several methods to scale noise. In this section we study, from an estimation theory perspective, the second component of ZNE: extrapolating the measured data to the zero-nose limit.

We assume that the output of the quantum computation is a single expectation value $E(\lambda)$, where λ is the noise scale factor. This expectation could be the result of a single quantum circuit or some combinations of quantum circuits with classical post-processing. The expectation value $E(\lambda)$ is a real number which, in principle, can only be estimated in the limit of infinite measurement samples. In a real situation with N samples, only a statistical estimation of the expectation value is

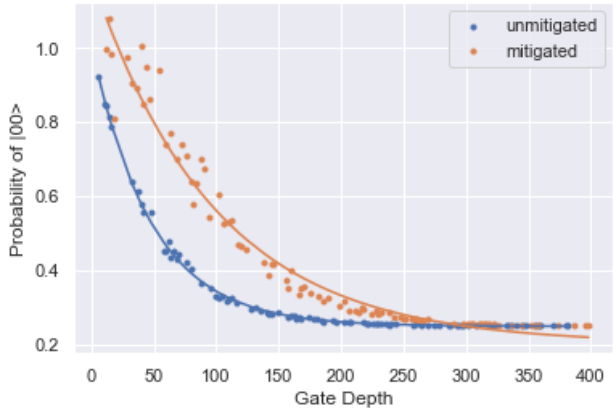


Figure 4.2: Comparison of two qubit randomized benchmarking with & without error mitigation. Data is taken by density matrix simulation with a 1% depolarizing noise model. The unmitigated simulation results in a randomized benchmarking decay of 97.9%. Mitigation is applied using circuit folding and an order-2 polynomial extrapolation at $\lambda = 1, 1.5, 2.0$. With mitigation the randomized benchmarking decay improves to 99.0%. Since we do not impose any constraint on the domain of the extrapolated results, some of the mitigated expectation values are slightly beyond the physical upper limit of 1. This is an expected effect of the noise introduced by the extrapolation fit. If necessary, one could enforce the result to be physical by using a more advanced Bayesian estimator.

Algorithm 5: Generic non-adaptive extrapolation

Data: A set of increasing noise scale factors $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$, with $\lambda_j \geq 1$ and fixed number of samples N for each λ_j.

Result: A mitigated expectation value

$\boldsymbol{y} \longleftarrow \emptyset$;

begin

  for $\lambda_j \in \boldsymbol{\lambda}$ **do**

    $y_j \longleftarrow ComputeExpectation(\lambda_j, N)$;

    $Append\,(\boldsymbol{y}, y_j)$;

  `/* Abitrary best fit algorithm (e.g., least squares) */`

  $\boldsymbol{\Gamma}^* \longleftarrow BestFit(E_{\text{model}}(\lambda; \boldsymbol{\Gamma}), (\boldsymbol{\lambda}, \boldsymbol{y}))$;

  return $E_{model}(0; \boldsymbol{\Gamma}^*)$;

actually possible:

$$\hat{E}(\lambda) = E(\lambda) + \hat{\delta}, \tag{4.10}$$

where $\hat{\delta}$ is a random variable with zero mean and variance $\sigma^2 = \mathbb{E}(\hat{\delta}^2) = \sigma_0^2/N$, with σ_0^2 corresponding to the single-shot variance. In other words, we can sample a real prediction y from

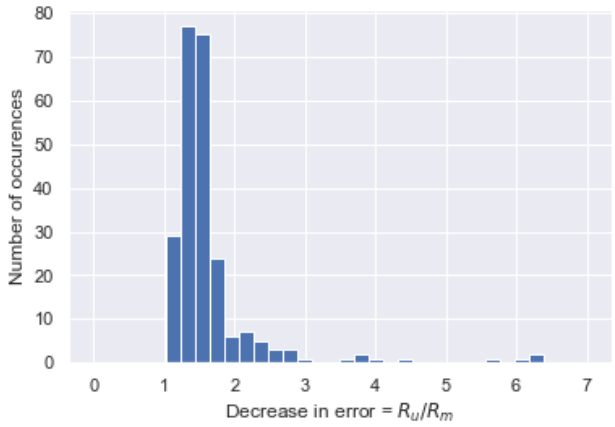


Figure 4.3: A comparison of improvements from ZNE (using quadratic extrapolation with folding from left) averaged across all output bitstrings from 250 random six-qubit circuits. Results are from exact density matrix simulations with a base of 1% depolarizing noise. The horizontal axis shows a ratio of L_2 distances from the noiseless probability distribution and the vertical axis shows the frequency of obtaining this result. ZNE improves on the noisy result by factors of 1-7X. The average mitigated error is 0.075 ± 0.035, while the unmitigated errors average 0.114 ± 0.050. Each circuit has 40 moments with single-qubit gates sampled randomly from $\{H, X, Y, Z, S, T\}$ and two-qubit gates sampled randomly from $\{\text{iSWAP}, \text{CZ}\}$ with arbitrary connectivity.

the probability distribution:

$$P(\hat{E}(\lambda) = y) = \mathcal{N}(E(\lambda) - y, \sigma^2), \tag{4.11}$$

where $\mathcal{N}(\mu, \sigma^2)$ is a generic distribution (typically Gaussian), with mean μ and variance $\sigma^2 = \sigma_0^2/N$.

Given a set of m scaling parameters $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$, with $\lambda_j \geq 1$, and the corresponding results

$$\mathbf{y} = \{y_1, y_2, \ldots\ y_m\}, \tag{4.12}$$

the ZNE problem is to build a good estimator $\hat{E}(0)$ for $E(\lambda = 0)$, such that its bias

$$\text{B}ias(\hat{E}(0)) = \mathbb{E}(\hat{E}(0) - E(0)), \tag{4.13}$$

and its variance

$$\text{V}ar(\hat{E}(0)) = \mathbb{E}(\hat{E}(0)^2) - \mathbb{E}(\hat{E}(0))^2, \tag{4.14}$$

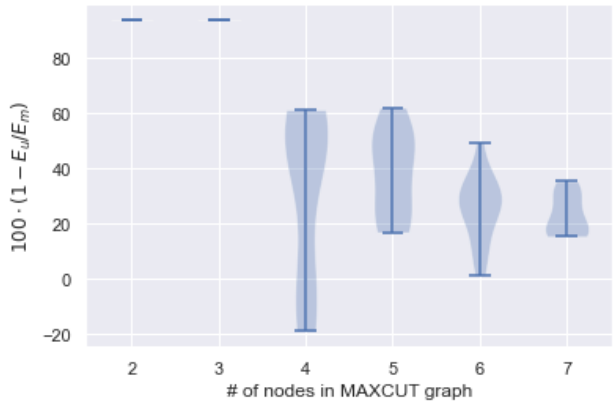


Figure 4.4: Percent closer to optimal on random MAXCUT executions. 14 Erdos-Renyi random graphs were generated at each number n. Each random graph has n nodes and n edges. QAOA was then run (with $p = 2$ QAOA steps) and optimized using Nelder-Mead with and without error mitigation. Results are from exact density matrix simulations with a base of 2% depolarizing noise. For the mitigated case, we used zero noise extrapolation with global unitary folding for scaling and linear extrapolation at noise scalings of 1, 1.5 and 2. The y axis shows the percent closer to the optimal solution that was gained by ZNE. Here E_u is the absolute error in the unmitigated expectation and E_m is the absolute error in the mitigated expectation. The violin plot shows the distribution of percentage improvements over the 14 sampled instances. Variance is zero for 2 and 3 nodes graphs as there is only a single valid graph with n nodes and edges for $n = 2, 3$.

are both reasonably small. More precisely, a typical figure of merit for the quality the estimator is its mean squared error with respect to the true unknown parameter:

$$\mathrm{MSE}(\hat{E}(0)) = \mathbb{E}(\hat{E}(0) - E(0))^2 \tag{4.15}$$

$$= \mathrm{Var}(\hat{E}(0)) + \mathrm{Bias}(\hat{E}(0))^2. \tag{4.16}$$

If the expectation value $E(\lambda)$ can be an arbitrary function of λ without any regularity assumption, then zero-noise extrapolation is impossible. Indeed its value at $\lambda = 0$ would be arbitrary and unrelated to its values at $\lambda \geq 1$. However from physical considerations, it is reasonable to have a model for $E(\lambda)$, e.g., we can assume a linear, a polynomial or an exponential dependence with respect to λ. For example, for a depolarizing noise model, one can use the exponential ansatz given in Eq. (4.9).

If we chose a generic model $E_{\mathrm{model}}(\lambda; \mathbf{\Gamma})$ for the quantum expectation value, where $\mathbf{\Gamma}$ represents the model parameters, then the zero-noise-extrapolation problem reduces to a regression problem. Algorithm 5 is the general form for a non-adaptive ZNE. Alternatively, the scale factors λ_j and the

associated numbers of samples N_j can be chosen in an adaptive way, depending on the results of intermediate steps. This adaptive extrapolation method is studied in more details in Section 4.1.4.

We focus on two main non-adaptive models, the polynomial ansatz and the poly-exponential ansatz. These two general models, give rise to a large variety of specific extrapolation algorithms. Some well known methods, such as Richardson's extrapolation, are particular cases. Some other methods have, to our knowledge, not been applied before for quantum error mitigation.

4.1.3.1 Polynomial extrapolation

The polynomial extrapolation method is based on the following polynomial model of degree d:

$$E^{(d)}_{poly}(\lambda) = c_0 + c_1\lambda + \dots c_d\lambda^d, \tag{4.17}$$

where $c_0, c_1, \dots c_d$ are $d+1$ unknown real parameters. This essentially corresponds to a Taylor series approximation and is physically justified in the weak noise regime.

In general, the problem is well defined only if the number of data points m is at least equal to the number of free parameters $d+1$. As opposed to Richardson's extrapolation [116], a useful feature of this method is that we can keep the extrapolation order d small but still use a large number of data points m. This avoids an over-fitting effect: if we increase the order d by too much, then the model is forced to follow the random statistical fluctuations of our data at the price of a large generalization error for the zero-noise extrapolation. In terms of the inference error given in Eq. (4.15), if we increase d by too much, then the bias is reduced but the variance can grow so much that the total mean squared error is actually increased.

4.1.3.2 Linear extrapolation

Linear extrapolation is perhaps the simplest method and is a particular case of polynomial extrapolation. It corresponds to the model:

$$E_{linear}(\lambda) = E^{(d=1)}_{poly}(\lambda) = c_0 + c_1\lambda. \tag{4.18}$$

In this case a simple analytic solution exists, corresponding to the ordinary least squared estimator of the intercept parameter:

$$\hat{E}_{linear}(0) = \bar{y} - \frac{S_{\lambda y}}{S_{\lambda\lambda}}\bar{x}, \tag{4.19}$$

where

$$\bar{\lambda} = \frac{1}{m}\sum_j \lambda_j, \qquad \bar{y} = \frac{1}{m}\sum_j y_j,$$
$$S_{\lambda y} = \sum_j (\lambda_j - \bar{\lambda})(y_j - \bar{y}), \qquad S_{\lambda\lambda} = \sum_j (\lambda_j - \bar{\lambda})^2. \tag{4.20}$$

With respect to the zero noise value of the model $E_{linear}(0)$, the estimator is unbiased. If the statistical uncertainty σ^2 for each y_j is the same, the variance for $\hat{E}_{linear}(0)$ is:

$$\mathbb{V}ar[\hat{E}_{linear}(0)] = \sigma^2 \left[\frac{1}{m} + \frac{\bar{\lambda}^2}{S_{\lambda\lambda}}\right]. \tag{4.21}$$

4.1.3.3 Richardson extrapolation

Richardson's extrapolation is also a particular case of polynomial extrapolation where $d = m - 1$, i.e., the order is maximized given the number of data points:

$$E_{Rich}(\lambda) = E^{(d=m-1)}_{poly}(\lambda) = c_0 + c_1\lambda + \ldots c_{m-1}\lambda^{m-1}. \tag{4.22}$$

This is the only case in which the fitted polynomial perfectly interpolates the m data points such that, in the ideal limit of an infinite number of samples $N \to \infty$, the error with respect to the true expectation value is by construction $O(m)$. Using the interpolating *L*agrange polynomial, the estimator can be explicitly expressed as:

$$\hat{E}_{Rich}(0) = \hat{c}_0 = \sum_{k=1}^{m} y_k \prod_{i\neq k} \frac{\lambda_i}{\lambda_i - \lambda_k}, \tag{4.23}$$

where we assumed that all the elements of $\boldsymbol{\lambda}$ are different.

The error of the estimator is $O(m)$ only in the asymptotic limit $N \to \infty$. In other words $O(m)$ corresponds to the bias term in Eq. (4.15). In a real scenario, N is finite, and the variance term in

Eq. (4.15) grows exponentially as we increase m. This fact can be easily shown in the simplified case in which the noise scale factors are equally spaced, i.e., $\lambda_k = k\,\lambda_1$ where $k = 1, 2, \ldots m$. Substituting this assumption into Eq. (4.23) we get:

$$\hat{E}_{\mathrm{R}ich}(0) = \sum_{k=1}^{m} y_k \prod_{i \neq k} \frac{i}{i-k} = \sum_{k=1}^{m} y_k \,(-1)^{k-1} \binom{m}{k}. \tag{4.24}$$

If we assume that each expectation value is sampled with the same statistical variance σ^2 as described in Eq. (4.11), since $\hat{E}_{\mathrm{R}ich}(0)$ is a linear combination of the measured expectation values $\{y_k\}$, its variance is given by:

$$\begin{aligned} \mathrm{V}ar(\hat{E}_{\mathrm{R}ich}(0)) &= \sigma^2 \sum_{k=1}^{m} \binom{m}{k}^2 \\ &= \sigma^2 \left[\binom{2m}{m} - 1 \right] \xrightarrow{m \longrightarrow \infty} \sigma^2 \, \frac{2^{2m}}{\sqrt{\pi m}}, \end{aligned} \tag{4.25}$$

where we used the Vandermonde's identity and, in the last step, the Stirling approximation.

The practical implication of Eq. (4.25) is that the zero-nose limit predicted by the Richardson's estimator is characterized by a statistical uncertainty which scales exponentially with the number of data points.

4.1.3.4 Poly-Exponential extrapolation

The poly-exponential ansatz of degree d is:

$$E^{(d)}_{\mathrm{poly}exp}(\lambda) = a \pm e^{z(\lambda)}, \; z(\lambda) := z_0 + z_1\lambda + \ldots z_d\lambda^d. \tag{4.26}$$

where $a, z_0, z_1, \ldots z_d$ are $d+2$ parameters. From physical considerations, it is reasonable to assume that $E(\lambda)$ converges to a finite asymptotic value i.e.:

$$E(\lambda) \xrightarrow{\lambda \to \infty} a \quad \Longleftrightarrow \quad z(\lambda) \xrightarrow{\lambda \to \infty} -\infty. \tag{4.27}$$

There are two important scenarios: (i) where a is unknown and so a non-linear fit should be performed and (ii) where a is deduced from asymptotic physical considerations. For example, if we know that in the limit of $\lambda \to \infty$ the state of the system is completely mixed or thermal, it is

possible to fix the value of a such that the poly-exponential ansatz (4.26) is left with only $d+1$ unknown parameters: $z_0, z_1, \ldots z_d$. If the asymptotic limit a is known, we can apply the following procedure:

1. Evaluate $\{y'_k\} = \{\log(|y_k - a| + \epsilon)\}$, representing the measurement results in a convenient logarithmic space with coordinates (y'_k, λ_k), with a small regularizing constant $\epsilon > 0$.

2. The model of Eq. (4.26) in the logarithmic space (y'_k, λ_k) reduces to the polynomial $z(\lambda)$.

3. Estimate the zero-noise limit in the logarithmic space $\hat{z}(0) = \hat{z}_0$ with a standard polynomial extrapolation. If necessary different weights can be used for different scale factors, taking into account the non-linear propagation of statistical errors.

4. Convert back to the original space, obtaining the final estimator $\hat{E}(0) = a \pm e^{\hat{z}(0)}$.

This allows us to map a non-linear regression problem into a polynomial fit that is linear with respect to the parameters and therefore much more stable. However, many reasonable alternative approaches exist like maximum likelihood optimization. Alternatively a Bayesian approach could be used, especially if we have prior information about the parameters of the model.

4.1.3.5 Exponential extrapolation

Exponential extrapolation is a particular case of the more general poly-exponential method. It corresponds to the model:

$$E_{exp}(\lambda) = E^{(d=1)}_{polyexp}(\lambda) = a \pm e^{z_0 + z_1 \lambda} = a + b e^{-c\lambda}, \tag{4.28}$$

where the set of real coefficients a, b, c is a way of parametrizing the same ansatz, alternative but equivalent to a, z_0, z_1. This model was discussed in [118] and is generalized by our extrapolation framework. In particular, increasing the order d, for example to $d = 2$, and using the poly-exponential model (4.26) we can capture small deviations from the ideal exponential assumption, possibly obtaining a more accurate zero-noise extrapolation.

Scaling	Extrapolation	Error % (dep.)	Error % (amp. damp.)
none	unmitigated	29.9 ± 5.1	16.7 ± 4.0
circuit	linear ($d = 1$)	14.6 ± 4.6	5.40 ± 2.3
circuit	quadratic ($d = 2$)	6.35 ± 3.6	3.53 ± 3.4
circuit	Richardson ($d = 3$)	17.6 ± 11	*17.9 ± 16*
circuit	exponential ($a = 0.25$)	2.73 ± 1.9	2.06 ± 1.6
circuit	adapt. exp. ($a = 0.25$)	$\mathbf{1.27 \pm 1.1}$	2.69 ± 2.8
at random	linear ($d = 1$)	15.6 ± 5.3	5.20 ± 2.4
at random	quadratic ($d = 2$)	5.54 ± 4.4	8.00 ± 8.1
at random	Richardson ($d = 3$)	*30.0 ± 24*	*24.0 ± 18*
at random	exponential ($a = 0.25$)	2.84 ± 1.8	$\mathbf{0.95 \pm 1.0}$
at random	adapt. exp. ($a = 0.25$)	1.77 ± 1.4	2.18 ± 1.2
from left	linear ($d = 1$)	14.4 ± 4.5	5.16 ± 2.3
from left	quadratic ($d = 2$)	6.73 ± 3.7	3.88 ± 3.7
from left	Richardson ($d = 3$)	18.4 ± 12	16.1 ± 13
from left	exponential ($a = 0.25$)	3.17 ± 2.1	2.19 ± 2.0
from left	adapt. exp. ($a = 0.25$)	1.43 ± 1.1	3.08 ± 3.6

Table 4.2: Average of 20 different two-qubit randomized benchmarking circuits with mean depth 27. The percent mean absolute error from the exact value of 1 is reported for a depolarizing noise with $p = 1\%$ and an amplitude damping channel with $\gamma = 0.01$. For all non-adaptive methods we used $\lambda = \{1, 1.5, 2, 2.5\}$. Adaptive extrapolation was iterated up to 4 scale factors. All the results reported in this table are obtained with exact density matrix simulations. The best result for each noise model is highlighted with a bold font, while errors larger than the unmitigated one are *italicized*.

4.1.3.6 Benchmark comparisons of ZNE methods

Benchmarks comparing the performance of ZNE methods are given in Table 4.2. In all cases, besides for Richardson extrapolation, ZNE improves on the unmitigated noise value, however the performance varies significantly. Furthermore, one scaling or extrapolation method does not strictly dominate others.

Different extrapolation methods are compared on IBMQ's London superconducting quantum processor in Fig. 4.5. Here random gate folding scales the noise of 50 different two-qubit randomized benchmarking circuits. The ideal expectation value for all circuits is 1. The order 2 polynomial fit, and the exponential fit outperform Richardson extrapolation. In fact, Fig. 4.5 shows the expectation value for Richardson extrapolation when only the first 3 data points are considered. Instability in

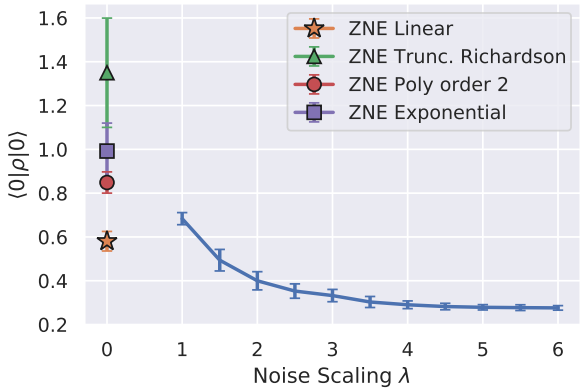


Figure 4.5: Comparison of extrapolation methods averaged over 50 two-qubit randomized benchmarking circuits executed on IBMQ's "London" five-qubit chip. The circuits had, on average, 97 single qubit gates and 17 two-qubit gates. The true zero-noise value is $\langle 0|\rho|0\rangle = 1$ and different markers show extrapolated values from different fitting techniques.

the Richardson extrapolation for more points, as described in Section 4.1.3.3, causes nonphysical results when applied to all the measured data. This is an example in which vanilla Richardson extrapolation is not sufficient to provide stable results.

4.1.4 Adaptive zero noise extrapolation

In Section 4.1.3, we considered only non-adaptive extrapolation methods. However, in order to reduce the computational overhead, we can choose the scale factors and the number of samples in an adaptive way as described in Algorithm 6.

Differently from the non-adaptive case, in this adaptive procedure (Alg. 6) the measured scale factors $\boldsymbol{\lambda}$ are not monotonically increasing. Indeed in the adaptive step, λ_{next} can take any value (above or equal to 1). In particular, λ_{next} could also be equal to a previous scale factor λ_j, for some j. In this case, the additional measurement samples N_{next} will improve the statistical estimation of $E(\lambda_j)$.

Now, we present an example of adaptive extrapolation which is based on the exponential ansatz $E_{\text{exp}}(\lambda) = a + be^{-c\lambda}$ that we have already introduced in Eq. (4.28). We also assume that the asymptotic value a is known. This implies that at least two scale factors should be measured to fit

Algorithm 6: Generic adaptive extrapolation

Data: An initial set of m noise scale factors $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$, with $\lambda_j \geq 1$, m sample numbers $\boldsymbol{N} = (N_1, N_2, \ldots N_m)$ and a maximum number of total samples $N_{\max}$.

Result: A mitigated expectation value

begin

  /* Initialization */

  $\boldsymbol{y} \longleftarrow \emptyset$;

  for $\lambda_j \in \boldsymbol{\lambda}$ **do**

    $y_j \longleftarrow ComputeExpectation(\lambda_j, N_j)$;

    $Append\,(\boldsymbol{y}, y_j)$;

  /* Adaptive loop */

  $N_{\text{used}} \longleftarrow 0$;

  while $N_{used} < N_{max}$ **do**

    $\boldsymbol{\Gamma}^* \longleftarrow BestFit(E_{model}(\lambda; \boldsymbol{\Gamma}), (\boldsymbol{\lambda}, \boldsymbol{y}))$;

    $\lambda_{\text{next}} \longleftarrow NewScale(\boldsymbol{\Gamma}^*, \boldsymbol{\lambda}, \boldsymbol{y})$;

    $N_{\text{next}} \longleftarrow NewNumSamples(\boldsymbol{\Gamma}^*, \boldsymbol{\lambda}, \boldsymbol{y})$;

    $y_{\text{next}} \longleftarrow ComputeExpectation(\lambda_{\text{next}}, N_{\text{next}})$;

    $Append\,(\boldsymbol{\lambda}, \lambda_{\text{next}})$;

    $Append\,(\boldsymbol{y}, y_{\text{next}})$;

    $N_{\text{used}} \longleftarrow N_{\text{used}} + N_{\text{next}}$;

  return $E_{model}(0; \boldsymbol{\Gamma}^*)$;

the parameters b and c. We first consider this particular case and then we generalize the method to an a arbitrary number of scale factors, which will be chosen in an adaptive way.

4.1.4.1 Exponential extrapolation with two scale factors

We assume only two scale factors λ_1 and λ_2 (typically, λ_1 is 1). As discussed in Section 4.1.3, we can estimate the corresponding expectation values, $E(\lambda_1)$ and $E(\lambda_2)$, with a statistical uncertainty of $\sigma_1^2 = \sigma_0^2/N_1$ and $\sigma_2^2 = \sigma_0^2/N_2$, respectively. Here, we are implicitly assuming that the single shot variance σ_0^2 is independent of λ, such that the estimation precision is only determined by number of samples N_1 and N_2. The measurement process will produce two results y_1 and y_2, whose statistical distribution is given by Eq. (4.11).

Since the parameter a is known, we can use the points (λ_1, y_1) and (λ_2, y_2) to estimate b and c of Eq. (4.28). The two estimators $\hat{b}$ and $\hat{c}$ can be determined by the unique ansatz interpolating

the two points, whose parameters are:

$$\hat{c} = \frac{1}{\lambda_2 - \lambda_1} \log \frac{y_1 - a}{y_2 - a}, \tag{4.29}$$

$$\hat{b} = (y_1 - a)^{\frac{\lambda_2}{\lambda_2 - \lambda_1}} (y_2 - a)^{-\frac{\lambda_1}{\lambda_2 - \lambda_1}} . \tag{4.30}$$

The corresponding estimator for the zero-noise limit is $\hat{E}_{\text{exp}}(0) = a + \hat{b}$ where, since a is known, the error is only due to the statistical noise of $\hat{b}$.

This estimator depends on the empirical variables y_1, y_2, with statistical variances $\sigma_1^2 = \sigma_0^2/N_1$ and $\sigma_2^2 = \sigma_0^2/N_2$ respectively. Such measurement errors will propagate to the estimator $\hat{b}$. To leading order in σ_1^2 and σ_2^2, we have:

$$\text{MSE}(\hat{b}) = \left(\frac{\partial \hat{b}}{\partial y_1}\right)^2 \sigma_1^2 + \left(\frac{\partial \hat{b}}{\partial y_2}\right)^2 \sigma_2^2 . \tag{4.31}$$

The explicit evaluation of Eq. (4.31), yields:

$$\text{MSE}(\hat{b}) = \frac{\sigma_0^2}{(\lambda_2 - \lambda_1)^2} \left[\frac{\lambda_2^2\, e^{2c\lambda_1}}{N_1} + \frac{\lambda_1^2\, e^{2c\lambda_2}}{N_2}\right] . \tag{4.32}$$

The previous equation shows that the error depends on the choice of the scale factors λ_1 and λ_2 but also on the associated measurement samples N_1 and N_2.

Error minimization Let us first assume that we have at disposal only a total budget $N_{\text{max}} = N_1 + N_2$ of circuit evaluations and that λ_1 and λ_2 are fixed. Minimizing Eq. (4.32), with respect to N_1 and N_2, we get:

$$\begin{aligned} N_1 &= N_{\text{max}} \frac{\lambda_1}{\lambda_1 + \lambda_2\, e^{-c(\lambda_2 - \lambda_1)}} \\ N_2 &= N_{\text{max}} \frac{\lambda_2\, e^{-c(\lambda_2 - \lambda_1)}}{\lambda_1 + \lambda_2\, e^{-c(\lambda_2 - \lambda_1)}} \end{aligned} \tag{4.33}$$

and the corresponding error becomes:

$$\text{MSE}(\hat{b}) = \sigma_0^2 \left[\frac{\lambda_2\, e^{c\lambda_1} + \lambda_1\, e^{c\lambda_2}}{\lambda_2 - \lambda_1}\right]^2 . \tag{4.34}$$

Algorithm 7: Adaptive exponential extrapolation

Data: An exponential model $E_{\exp}(\lambda) = a + be^{-c\lambda}$ with a known/estimated a. A maximum number of total samples $N_{\max}$, a fixed number of samples per iteration N_{batch} and a minimum scale factor λ_1 (typically equal to 1).

Result: A mitigated expectation value

begin
- $c \longleftarrow 1$; /* Initial guess */
- $\alpha \longleftarrow 1.27846$; /* Alpha in Eq. (4.36) */
- $\mathbf{d}ata \longleftarrow \emptyset$;
- $N_{\text{used}} \longleftarrow 0$;
- **while** $N_{used} < N_{max}$ **do**
  - $\lambda_2 \longleftarrow \lambda_1 + \alpha/c$;
  - $N_1 \longleftarrow N_{\text{batch}} \times \frac{c\,\lambda_1/\alpha}{c\,\lambda_1+\alpha-1}$;
  - $N_2 \longleftarrow N_{\text{batch}} \times \frac{(1+c\,\lambda_1/\alpha)(\alpha-1)}{c\,\lambda_1+\alpha-1}$;
  - $N_{\text{used}} \longleftarrow N_{\text{used}} + N_1 + N_2$;
  - $y_1 \longleftarrow ComputeExpectation(\lambda_1, N_1)$;
  - $y_2 \longleftarrow ComputeExpectation(\lambda_2, N_2)$;
  - $Append\,(\mathbf{d}ata, (\lambda_1, y_1))$;
  - $Append\,(\mathbf{d}ata, (\lambda_2, y_2))$;
  - /* New estimate of c */
  - $c \longleftarrow BestFit(E_{exp}(\lambda; a, b, c), \mathbf{d}ata)$;
- **return** $E_{exp}(0; a, b, c)$;

This error can be further minimized with respect to the choice of the scale factors. Since λ_1 is usually fixed to 1, we optimize over λ_2, leading to the condition:

$$e^{c(\lambda_2-\lambda_1)}\,(c(\lambda_2-\lambda_1)-1)-1=0. \tag{4.35}$$

We can solve the previous equation numerically, obtaining:

$$c(\lambda_2-\lambda_1)=\alpha, \tag{4.36}$$

where $\alpha \simeq 1.27846$ is a numerical constant. For a fixed λ_1, the previous condition determines the optimal choice of the scale factor λ_2 which minimizes the zero-nose extrapolation error. From a practical point of view, Eqs. (4.33) and (4.36) can only be used if we have some prior knowledge about c. This motivates the following adaptive algorithm.

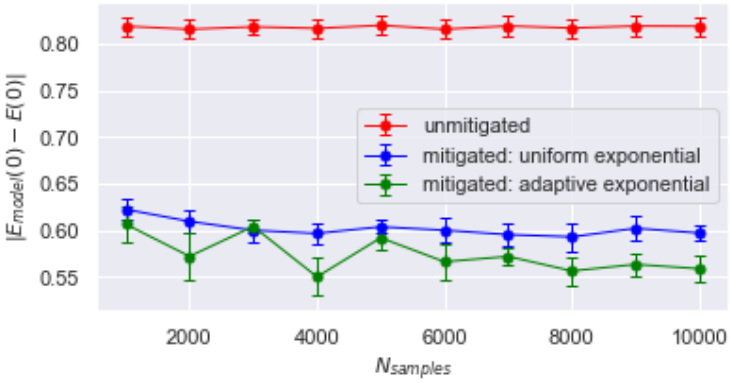


Figure 4.6: Comparison of adaptive and non-adaptive exponential zero noise extrapolation, given a fixed budget of samples. The adaptive method generally produces a more accurate extrapolation with less samples. On the other hand, in this example, the advantage of adaptivity is not particularly large. Likely, this is due to the fact that the scale factors used for the non-adaptive method are already quite good and not far from their optimal values. Data was generated by exact density matrix simulation of 5-qubit randomized benchmarking circuits of depth 10 under 5% depolarizing noise and measured in the computational basis. Noise was scaled directly by access to the back-end simulator rather than with a folding method.

4.1.4.2 An adaptive exponential extrapolation algorithm

Algorithm 7 is an adaptive exponential algorithm based on the exponential ansatz $E_{\exp}(\lambda) = a + be^{-c\lambda}$, where a is a known constant. Figure 4.6 shows a comparison of adaptive exponential extrapolation with non-adaptive exponential extrapolation. At almost all sample levels, adaptive extrapolation outperforms the non adaptive approach.

4.1.5 Conclusion

We make zero-noise extrapolation digital, developing the unitary folding framework to run error mitigation with instruction set level access. We then demonstrate improved performance through a set of non-adaptive and adaptive extrapolation methods. We emphasize that zero-noise extrapolation is in general an inference problem with many avenues for further optimization.

While ZNE has previously been benchmarked on randomized benchmarking circuits or VQE, we give benchmarks of ZNE on MAXCUT problems solved with QAOA. This allows us to smoothly benchmark the performance of ZNE on larger variational quantum circuits then have been consid-

ered previously.

We also consider specialization of zero-noise extrapolation to different noise models, using calibration noise as an example. With more sophisticated multi-parameter noise models (such as a combination of calibration noise and amplitude dampening), it is likely that multi-dimensional noise extrapolation [144] will be of interest.

This work is a first step towards viewing zero-noise extrapolation as an inference problem and has opportunities for extension. Priors or constraints from observable, noise or circuit structure could be included. Data could be gathered from similar executions over time so that inference includes a historical database of previous computations.

4.2 Reducing the impact of time-correlated noise on ZNE

4.2.1 Introduction

Zero-noise extrapolation (ZNE) techniques have been primarily investigated under the assumption that the errors to be mitigated are uncorrelated in time. On the other hand, time-correlated noise (in particular $1/f^{\alpha}$ noise) has been widely observed in physical systems including superconducting devices [145, 146, 147, 148, 149], quantum dots [150, 151], and spin qubits [152]. To estimate the noise present in these real physical systems, one can use quantum noise spectroscopy (QNS) [153, 154, 155] wherein the outcomes of a set of distinct control pulses or circuits are analyzed. Key to this approach is that while these different probe sequences may in fact represent identical circuits under ideal conditions, they interact with any noise present in different ways. This can be understood through the filter function formalism [156, 157] which describes the "frequency response" of a given probe sequence. Broadly speaking, the impacts of noise (in terms of fidelity) are approximately proportional to the integral of the power spectrum of the noise with the filter function of the control. In what follows, we will show how this intuition can also be applied to different ZNE schemes in the presence of temporally correlated dephasing noise.

The recently developed [158] and experimentally validated [159] Schroödinger wave autoregressive moving average (SchWARMA) technique provides a natural mechanism for the exploration

of so-called *digital* ZNE techniques [142, 160, 161] that operate at the gate level in a quantum circuit. Building on techniques from classical time-series modeling in statistics and signal processing, SchWARMA was conceived as a highly flexible mechanism for simulating a wide-range of spatiotemporally correlated errors in quantum circuits.

In the following, we first review the SchWARMA modeling and simulation formalism and its relationship to the filter function formalism. Next, we provide a concise overview of ZNE and discuss different methods for scaling noise. Next, we show how these different schemes are impacted by time-correlated dephasing noise despite the fact that they behave equivalently for uncorrelated noise. We then interpret these noise scaling schemes using the language of filter functions and show that these results are well described by the intuition provided by the filter functions. Our findings indicate that, for time-correlated noise, the noise scaling method known as *g*lobal unitary folding [160, 162] produces more accurate noise-scaled expectation values and ZNE results.

4.2.2 Background

4.2.2.1 Time-correlated noise: The SchWARMA model

Consider a single-qubit Hamiltonian

$$H(t) = H_z(t) + H_c(t) \tag{4.37}$$

consisting of a semiclassical dephasing noise component $H_z(t)$ along with a deterministic idealized control component $H_c(t)$ corresponding, for example, to the external driving induced by laser pulses. If we further define $H_z(t) = \eta(t)\sigma^z$ with $\eta(t)$ a wide-sense stationary Gaussian stochastic process, we can say that this noise process is not time-correlated if $\mathbb{E}[\eta(t)\eta(t')] = \mathbb{E}[\eta(|t - t'|)\eta(0)] = 0$ for all $t \neq t'$, where $\mathbb{E}(\cdot)$ represents the average over many statistical realizations. σ^i, $i = x, y, z$ are the Pauli matrices. Equivalently, we can say that the noise process is time-correlated if the power spectrum

$$S_\eta(\omega) = \int_0^\infty dt\, \mathbb{E}[\eta(t)\eta(0)]e^{-i\omega t} \tag{4.38}$$

is not constant as a function of ω (i.e., not a "white" process).

In the SchWARMA modeling approach [158], the impact of the continuous time Hamiltonian in (4.37) is modeled in a quantum circuit formalism by inserting correlated Z-error operators after each "gate" determined by the control H_c. This is accomplished by generating a time-correlated sequence of rotation angles y_k defined from independent Gaussian inputs x_k using an auto-regressive moving average(ARMA) model [163, 164],

$$y_k = \underbrace{\sum_{i=1}^{p} a_i y_{k-i}}_{AR} + \underbrace{\sum_{j=0}^{q} b_j x_{k-j}}_{MA}, \tag{4.39}$$

where the set $\{a_i\}$ defines the autoregressive portion of the model, and $\{b_j\}$ the moving average portion with p and $q+1$ elements of each set respectively. The time correlations are defined via the resulting power spectrum

$$S_y(\omega) = \frac{\left|\sum_{k=0}^{q} b_k \exp(-ik\omega)\right|^2}{\left|1 + \sum_{k=1}^{p} a_k \exp(-ik\omega)\right|^2}, \tag{4.40}$$

and ARMA models can approximate *any* discrete-time power spectrum to arbitrary accuracy [165]. For the scope of this work we focus on the four paradigmatic noise spectra shown in Fig. 4.7, namely: white noise, low-pass noise, $1/f$ noise and $1/f^2$ noise.

Dividing the circuit trajectory defined by $H_c(t)$ into consecutive gates G_k, the SchWARMA approach models the impact of correlated noise $H_z(t)$ by adding in a random $Z(\theta_k) = \exp(iy_k\sigma^z)$ after each gate, which can then be Monte Carlo averaged to produce an expectation value. This model can be extended to multi-qubit Hamiltonians

$$H(t) = \sum_{j=1}^{n} \eta_j(t)\sigma_j^z + H_c(t), \tag{4.41}$$

by generating independent, yet identically defined, SchWARMA-generated errors on each qubit. In principle, these could of course be heterogeneous and correlated between qubits.

4.2.2.2 Zero-noise extrapolation with colored noise

Zero-noise extrapolation (ZNE) is a heuristic error mitigation technique which relies on the ability to increase the noise in a quantum circuit [166, 167, 168]. Like other error mitigation techniques, the target is to estimate an expectation value

$$E(\lambda) := \mathrm{Tr}[\rho(\lambda)O] \tag{4.42}$$

at zero noise. The noise scale factor λ dictates how much the base noise level $\lambda = 1$ is scaled in the quantum circuit which prepares the system density matrix ρ, and O is a problem-dependent observable. The key insight of ZNE is to (i) evaluate $E(\lambda)$ at several noise scale factors $\lambda \geq 1$, then (ii) fit a statistical model to the collected data and infer the zero-noise value $E(\lambda \to 0)$. We refer to these two steps as noise scaling and inference, respectively.

Compared to other error mitigation techniques, zero-noise extrapolation requires very few additional quantum resources. Correspondingly, it has received some attention in recent literature, e.g. it was implemented in Refs. [169, 142, 160, 162, 170, 171] and in [172] on twenty six superconducting qubits to produce results competitive with classical approximation techniques. References [142, 160, 161] formally introduced digital noise scaling, in which noise is scaled at a gate-level without pulse-level control.

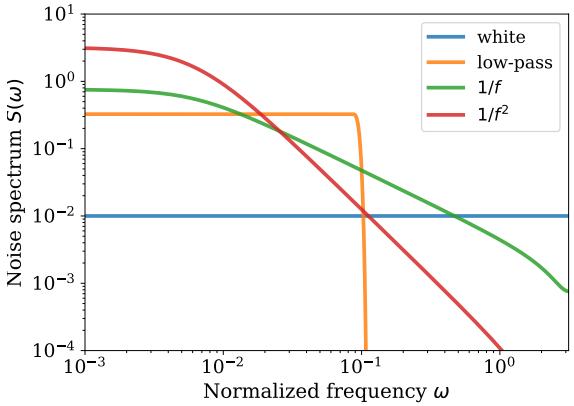


Figure 4.7: Noise power spectrum of four different dephasing SchWARMA noise models corresponding to white noise, low-pass noise, $1/f$ noise and $1/f^2$ noise. These noise models are used in Sec. 4.2.3 to test the effect of time-correlated noise on zero-noise extrapolation.

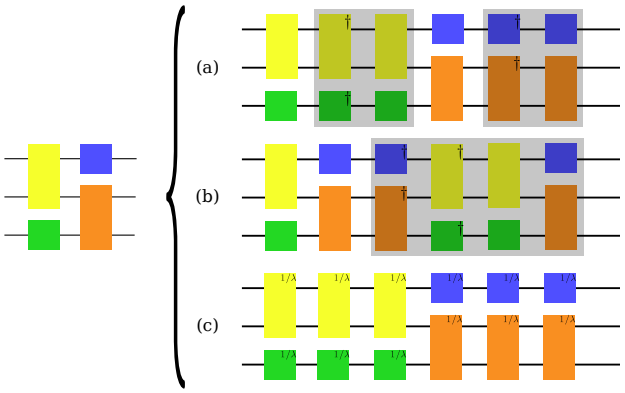


Figure 4.8: A sample three-qubit circuit with four gates under the action of three digital noise scaling methods we consider in this work. (a) Local folding, in which each gate G gets mapped to $G \mapsto G\left(G^\dagger G\right)^n$ for scale factor $\lambda = 2n - 1$. (b) Global folding, in which the entire circuit C gets mapped to $C \mapsto C\left(C^\dagger C\right)^n$. In (a) and (b), grey shading shows the "virtual gates" which logically compile to identity. (c) Gate Trotterization, in which $G \mapsto \left(G^{1/\lambda}\right)^\lambda$ for each gate G.

While ZNE is straightforward to implement and requires relatively few additional quantum resources, it is nonetheless a heuristic method. The quality of the solution depends critically on both the inference and noise-scaling method. In this work, we fix the inference method by assuming a particular noise model and focus on the effects of the noise-scaling method.

4.2.2.3 Noise scaling methods

Ideal noise scaling In a purely theoretical setting, the ideal way of scaling the noise would be to multiply the Hamiltonian H_z in Eq. (4.37) by a constant $\sqrt{\lambda}$:

$$H'(t) = \sqrt{\lambda} H_z(t) + H_c(t). \tag{4.43}$$

Equivalently, the scale factor can be absorbed into a redefinition of the stochastic noise amplitude: $\eta'(t) = \sqrt{\lambda}\eta(t)$. From Eq. (4.40), it is evident that the noise power spectrum gets scaled by λ,

$$S_{\eta'}(\omega) = S_{\sqrt{\lambda}\eta}(\omega) = \lambda\, S_\eta(\omega). \tag{4.44}$$

If one could directly control the noise, this would be the ideal way of scaling its power and, therefore, the ideal way of applying zero-noise extrapolation. In a typical experimental scenario,

of course, one cannot directly control the noise of a quantum device. For this reason, several indirect noise scaling techniques of have been proposed and applied in recent literature. We define several of these in the following subsections (see Fig. 4.8 for an overview) in order to analyze their performance in the presence of time-correlated noise in Sec. 4.2.3.

Pulse stretching The intent of pulse stretching is to scale the impacts of the noise on the system by "stretching" the underlying control Hamiltonian, replacing (4.37) with

$$H(t) = H_z(t) + \frac{1}{\lambda} H_c(t/\lambda)\,, \tag{4.45}$$

for some dimensionless time-scaling factor λ. In principle, this scales the impacts of the noise by increasing the overall time duration of the circuit. More precisely, if we define $t' = t/\lambda$, the density operator $\rho(t')$ of the system evolves with respect to the effective Hamiltonian:

$$H'(t') = \lambda\, H_z(\lambda t') + H_c(t')\,. \tag{4.46}$$

The corresponding noise power spectrum is:

$$\begin{aligned} S_{\eta'}(\omega) &= \lambda^2 \int_0^\infty dt'\, \mathbb{E}[\eta(\lambda t')\eta(0)] e^{-i\omega t'} \\ &= \lambda \int_0^\infty dt\, \mathbb{E}[\eta(t)\eta(0)] e^{-i\omega t/\lambda} = \lambda\, S_\eta(\omega/\lambda). \end{aligned} \tag{4.47}$$

From the equation above, it is evident that for a white (constant) spectrum, pulse stretching can be used to effectively scale the noise power by λ as in the ideal case defined in Eq. (4.44). In fact, the equivalence between the ideal noise scaling and the pulse-stretching technique was already shown in Ref. [166], under the hypothesis of a quantum state ρ evolving according to a master equation with a time-independent noise operator acting as $\mathcal{L}(\rho)$ (more details about the consistency between our findings with the results of Ref. [166] are given in Section 4.2.6). On the other hand, Eq. (4.47) shows that, for a colored spectrum, pulse-stretching does not exactly reproduce the ideal noise scaling defined in (4.44). Indeed, on the r.h.s. of Eq. (4.47) we observe that the original spectrum is also stretched with respect to the frequency variable ω. This fact is a manifestation of the intuitive idea that slowing down the dynamics the system corresponds to effectively speeding up the time

scale of the environment. Such frequency stretching, while irrelevant in the white noise limit, becomes relevant for time-correlated noise.

In the SchWARMA formalism, there is not a mechanism for stretching pulses *per se* as it operates at the gate level in a circuit (without pulse-level control on $H_c(t)$). However, as discussed in the supplement to [158], it is possible to manipulate and stretch the spectrum of a SchWARMA model. So, for the task of numerically simulating pulse stretching, instead of implementing equation Eq. (4.46) one can simply implement Eq. (4.47) by directly transforming the spectrum of the SchWARMA model.

Local unitary folding A possible way of effectively increasing the noise of a circuit is to insert after each noisy CNOT gate, the product of two additional CNOT gates [142, 161]. In this way the ideal unitary is not changed, but the real dynamics is more noisy. More generally, in Sec. 4.1 we introduced several digital noise scaling methods that are based on the unitary folding replacement rule

$$G \to G(G^\dagger G)^n, \quad n = 0, 1, 2, \ldots, \tag{4.48}$$

where G is a unitary operation associated to an individual gate. If noise is absent, the replacement rule leaves the operation unchanged since $G^\dagger G$ is equal to the identity. On the contrary, if some base noise is associated to G, the unitary folding operation approximately scales the noise by an odd integer factor $\lambda = 1 + 2n$.

More precisely, by applying the unitary folding replacement to all the gates of an input circuit

$$U = G_d G_{d-1} \ldots G_1 \tag{4.49}$$

which is composed of d gates G_j, we obtain new circuit U' of depth $d' = (1 + 2n)d$ given by

$$U' = G_d (G_d^\dagger G_d)^n G_{d-1} (G_{d-1}^\dagger G_{d-1})^n \ldots G_1 (G_1^\dagger G_1)^n. \tag{4.50}$$

The depth of the new circuit U' is scaled by $\lambda = d'/d = 1+2n$ and, similarly, any type of noise which depends on the total number of gates will be effectively scaled by the same constant λ. In Sec. 4.1,

partial folding methods were proposed to obtain arbitrary real values of λ, but for simplicity in this work we only consider odd-integer scale factors. We refer to (4.50) as local unitary folding.

Global unitary folding Instead of locally folding all the gates, we can apply Eq. (4.48) to the entire circuit. In this way, the circuit U defined in Eq. (4.49) is simply mapped to

$$U' = U(U^\dagger U)^n. \tag{4.51}$$

Also in this case the total number of gates of the new circuit U' is multiplied by $\lambda = d'/d = 1 + 2n$ corresponding to an effective scaling of the noise.

Gate Trotterization In this work we also introduce another local noise-scaling method, acting at the level of individual gates, that we call *g*ate Trotterization since it can be considered as a discretization of the continuous pulse-stretching technique. According to the gate Trotterization technique, each gate of the circuit is replaced as follows:

$$G \to \left(G^{1/\lambda}\right)^\lambda, \quad \lambda = 0, 1, 2, \ldots. \tag{4.52}$$

For example, a Pauli X rotation gate $R_X(\theta)$ is replaced by λ applications of $R_X(\theta/\lambda)$. Eq. (4.52) is similar to the local version of the unitary folding rule (4.48) and, indeed, both methods replace a single gate with the product of λ gates. Compared to Eq. (4.48), the Trotter-like decomposition used in Eq. (4.52) is more uniform since equal elementary gates are used. On the other hand, a possible drawback of the gate Trotterization method is that $G^{1/\lambda}$ may be compiled by the hardware in different ways depending on λ and, therefore, the circuit depth may not get scaled as expected.

4.2.3 Results

In the previous section, we defined several noise-scaling methods that can be used in zero-noise extrapolation. In this section, we study how these different methods affect the performance of ZNE in the presence of time-correlated noise. For all the simulations presented in this section we used the following Python libraries: Mezze [158] for modeling SchWARMA noise, Mezze's

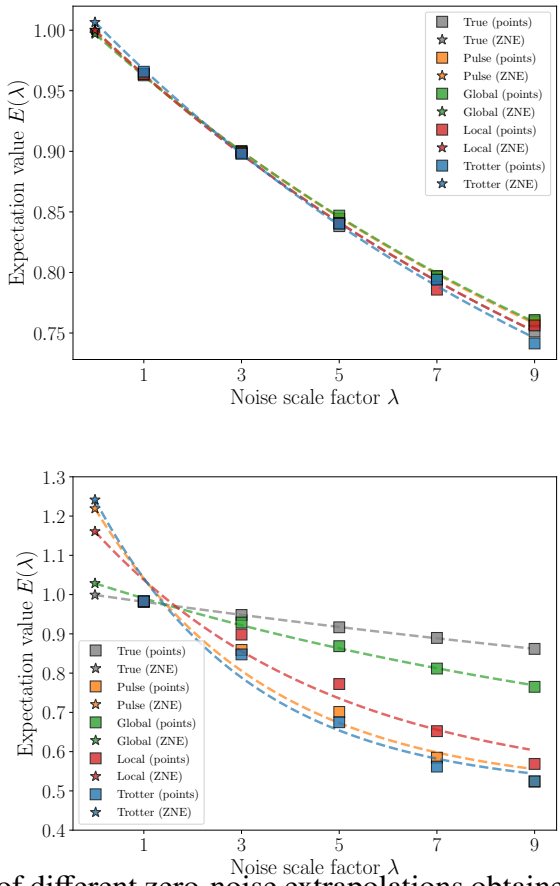


Figure 4.9: Comparison of different zero-noise extrapolations obtained with different noise scaling methods. We consider a single-qubit randomized benchmarking circuit affected by dephasing noise of fixed integrated power. The two subfigures correspond to different noise spectra: (top) white noise, (bottom) $1/f$ pink noise. Both spectra are shown in Fig. 4.7. The expectation value $E(\lambda) = tr(O\rho(\lambda))$ is associated to the observable $O = |0\rangle\langle 0|$ measured with respect to the noise-scaled quantum state $\rho(\lambda)$. The colored squares represent the noise-scaled expectation values; the dotted lines represent the associated exponential fitting curves; the colored *s*tars represent the corresponding zero-noise extrapolations. The figure shows that the zero-noise limit obtained with global unitary folding (green star) is relatively close to the ideal result (gray star) even in the presence of strong time correlations in the noise.

TensorFlow Quantum [173] interface for simulating quantum circuits and Mitiq [162] for applying unitary folding and zero-noise extrapolation.

4.2.3.1 Zero-noise extrapolation with colored noise

In this section we numerically simulate a simple ZNE experiment with different noise scaling methods and with different noise spectra. The results are reported Fig. 4.9 and demonstrate the

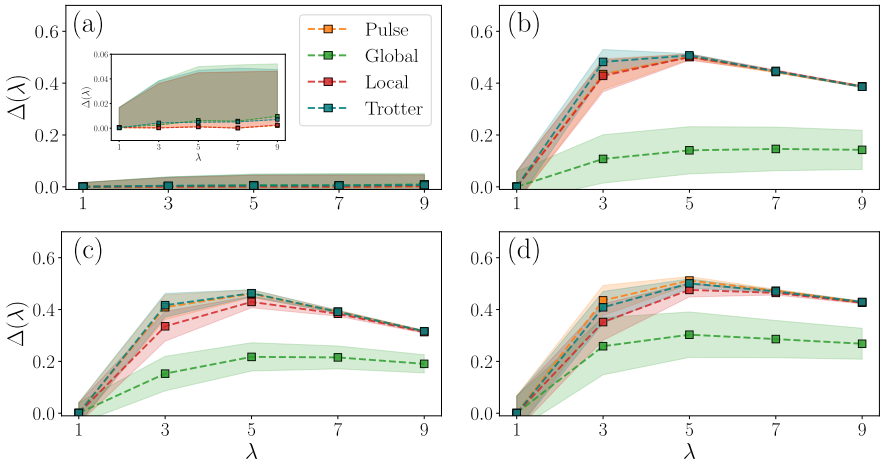


Figure 4.10: Average relative errors in noise scaling two-qubit randomized benchmarking circuits with (a) white noise, (b) lowpass noise, (c) $1/f$ noise, and (d) $1/f^2$ noise. Panel (a) shows no significant difference in scaling methods under white noise (no time correlations). (Inset shows zoomed vertical scale.) Panels (b)-(d) show that global scaling is the lowest-error digital scaling method. The two-qubit randomized benchmarking circuits used here have, on average, 27 single-qubit gates and five two-qubit gates. For each circuit execution, 3000 samples were taken to estimate the probability of the ground state as the observable. Points show the average results over fifty such circuits and error bars show one standard deviation.

detrimental effect of time-correlated noise on ZNE. In Fig. 4.9(a) the noise spectrum is white and all noise scaling methods produce nearly identical expectation values. Correspondingly, the zero-noise limits (marked with stars in the plot) are nearly identical. On the other hand, in Fig. 4.9(b), the noise is colored (a $1/f$ "pink" spectrum) and different noise-scaling methods produce different expectation values. Correspondingly, the zero-noise limits (marked with starts in the plot) are also different. This is the main qualitative result that this work aims to highlight: compared to white noise, time-correlated noise can be much harder to mitigate via zero-noise extrapolation.

In the rest of this section, we study this aspect in a more quantitative way. In particular we study the performances of different noise-scaling methods for different types of noise spectra and different types of circuits.

4.2.3.2 Comparing noise scaling methods

Observing Fig. 4.9(b) we notice that, at least for the particular circuit considered in the example, some noise sclaing methods perform better than others in the presence of time-correlated noise.

In particular the extrapolation based on the global folding technique produces a relatively good approximation of the ideal result even in the presence of time-correlated noise.

To better investigate this phenomenon, we consider the *relative noise-scaling error*

$$\Delta(\lambda) := \left| \frac{E(\lambda) - E^*(\lambda)}{E^*(\lambda)} \right|, \tag{4.53}$$

as a figure of merit. Here, $E(\lambda)$ is the expectation value of interest evaluated with some particular noise scaling method and scale factor λ, and $E^*(\lambda)$ is the expectation value simulated with a noise spectrum ideally scaled according to Eq. (4.44). In Fig. 4.10 we plot the relative error defined in Eq. (4.53) for each noise-scaling method, after averaging the results over multiple instances of two-qubit randomized-benchmarking circuits. Here the expectation value of the observable $O = |00\rangle\langle 00|$ is considered. The results of Fig. 4.10 are consistent with those of Fig. 4.9 discussed in the previous subsection. In fact, even after averaging over multiple random circuits, we observe that in the presence of white noise all noise scaling methods are practically equivalent to each other and are characterized by a small relative noise-scaling error. For all colored noise spectra instead, global folding is optimal when compared to other noise scaling methods.

We repeat the same experiments using mirror circuits [174] and QAOA-like circuits instead of RB circuits. The former provides another type of randomized circuit structure used for benchmarking, and the latter provides a structured circuit. Fig. 4.11 shows the results using two-qubit mirror circuits. These circuits have 26 single-qubit gates and eight two-qubit gates on average. As with the randomized benchmarking circuits, 3000 samples were taken when executing each circuit to estimate the probability of sampling the correct bitstring. As shown in Fig. 4.11, the conclusion that global unitary folding most closely matches true noise scaling holds on average for mirror circuits as well. These results were averaged over fifty random mirror circuits.

Fig. 4.12 shows the same experiment using QAOA circuits. These $n = 2$ qubit circuits have $p = 2$ QAOA rounds using the standard mixer Hamiltonian $H_M = \sum_{i=1}^{n} X_i$ and driver Hamiltonian $H_C = \sum_{ij} Z_i Z_j$. Denoting this circuit as U, we append $U^\dagger$ such that the final noiseless state is $|00\rangle$ independent of the randomly chosen angles β and γ. A total of fifty circuits with random angles were simulated for the final results, again using 3000 samples to estimate the ground state

probability for each circuit execution. The results in Fig. 4.12 have the highest variance of the three circuit types, but on average we still see that global unitary folding is closest to true noise scaling out of all scaling methods considered.

The conclusions of this subsection suggest that, even for different types of circuits, the effect of time-correlated noise on noise scaling methods is qualitatively similar. This intuition is consistent with the theoretical discussion presented in the next section, in which the performances of noise scaling methods are linked to their effective frequency modulation effects.

We emphasize that the comparison considered in this work is focused on one particular figure of merit: the robustness of a noise scaling method with respect to time-correlated noise. Our results suggest that global folding outperforms the other methods considered with respect to this specific figure of merit. In a real-world scenario, the optimal noise-scaling method should be determined according to a more general cost-benefit analysis, e.g. taking into account the sampling cost, coherence time, and other hardware limitations. For instance, it may not be possible to use global noise scaling if the circuit length is comparable to the coherence time of the computer; in such circumstances, pulse stretching can amplify errors via small scale factors [172], although potentially inaccurately in the presence of time-correlated noise as we have shown in this section.

4.2.4 Discussion and physical interpretation

4.2.4.1 Frequency response of a circuit

The impacts of time-correlated dephasing noise can be interpreted using the filter function formalism [156, 157]. In a single qubit scenario, we can associate with a given deterministic $H_c(t)$ (or gate sequence G_k) a frequency response $F_z(\omega)$ that relates the expected reduction in fidelity due to the noise as a decay $\exp(-\chi)$ where χ is defined via the "overlap integral,"

$$\chi = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} S_\eta(\omega) F_z(\omega) F_z(\omega)^\dagger \,. \tag{4.54}$$

The overlap integral can be used to derive approximations to noise-averaged observables, via

$$E[\mathrm{Tr}[\rho\, O]] \approx A + B \exp(-\chi) \,. \tag{4.55}$$

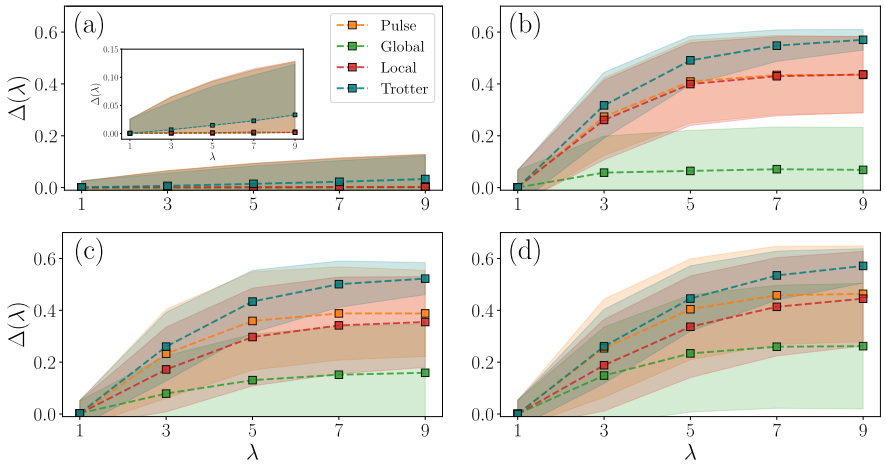


Figure 4.11: Relative errors in noise scaling two-qubit mirror circuits with (a) white noise, (b) lowpass noise, (c) $1/f$ noise, and (d) $1/f^2$ noise. Panel (a) shows no significant difference in scaling methods under white noise (no time correlations). (Inset shows zoomed vertical scale.) Panels (b), (c) and (d) show global scaling is optimal with time-correlated noise. The two-qubit mirror benchmarking circuits used here have, on average, 26 single-qubit gates and eight two-qubit gates. For each circuit execution, 3000 samples were taken to estimate the probability of the correct bitstring (defined by the particular mirror circuit instance) as the observable. Points show the average results over fifty such circuits and error bars show one standard deviation.

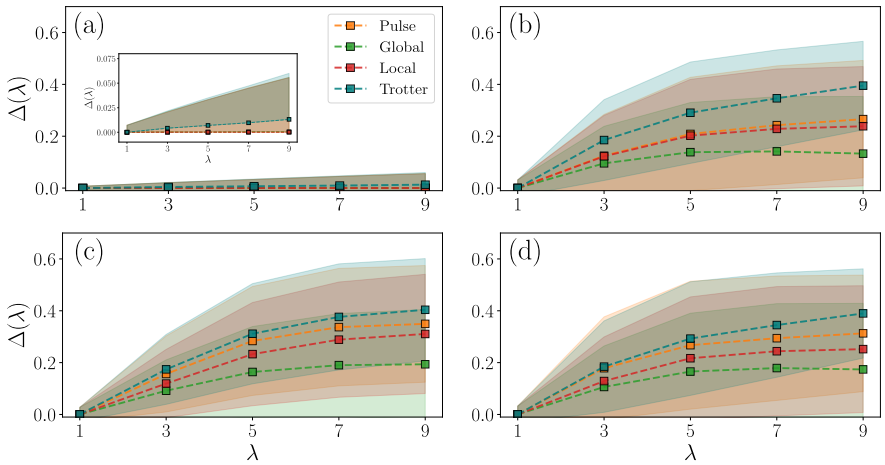


Figure 4.12: Relative errors in noise scaling two-qubit $p = 2$ QAOA circuits with (a) white noise, (b) lowpass noise, (c) $1/f$ noise, and (d) $1/f^2$ noise. Panel (a) shows no significant difference in scaling methods under white noise (no time correlations). (Inset shows zoomed vertical scale.) Panels (b), (c) and (d) show global scaling is optimal with time-correlated noise. The two-qubit $p = 2$ QAOA circuits used here have eight single-qubit gates and four two-qubit gates. For each circuit execution, 3000 samples were taken to estimate the probability of the ground state as the observable. (Note that the QAOA circuit U is echoed such that the total circuit is $UU^\dagger = I$ without noise.) Points show the average results over fifty such circuits and error bars show one standard deviation.

For multiqubit circuits, the overlap integral becomes a more complicated expression involving the second cumulant $C_O^{(2)}(T)$,

$$\frac{C_O^{(2)}(T)}{2} = \sum_{\alpha,\beta,\alpha',\beta'} \mathrm{Re} \int_0^\infty \frac{d\omega}{2\pi} S_{\alpha,\alpha'}(\omega) \mathcal{F}_{\alpha\beta,\alpha'\beta'}(\omega, T) \mathcal{A}_{\beta\beta'}, \tag{4.56}$$

where the overlaps between the noise power spectrum $S_{\alpha,\alpha'}$ and filter functions $\mathcal{F}_{\alpha\beta,\alpha'\beta'}$ scale operators $\mathcal{A}_{\beta\beta'}$.This expression captures potential cross correlations in noise, but here $S_{\alpha,\alpha'} = 0$ when $\alpha \neq \alpha'$ and α is not a σ_z operator on given qubit. Furthermore, for the examples below we compute the filter functions using instantaneous gates as specified by a circuit, but these expressions can hold for piecewise constant controls to accommodate pulse shaping. In the context of noise scaling experiments, Eq. (4.56) provides a mechanism for understanding how the different noise scaling techniques impact the resulting scaled expectations and thus the interpolation process.

4.2.4.2 Spectral analysis of noise scaling methods

Using the filter function prediction from Eq. (4.55) we have that direct noise scaling produces states $\rho_{dir}(\lambda)$ with expectation

$$E\left[\mathrm{Tr}[\rho_{dir}(\lambda)\, O]\right] \approx A + B \exp(-\lambda \chi_1)\,, \tag{4.57}$$

where χ_1 is the overlap integral of the base circuit. Similarly, following Eq. (4.46), we have that pulse stretching produces the expectation

$$E\left[\mathrm{Tr}[\rho_{pul}(\lambda)\, O]\right] \approx A + B \exp\left(-\lambda \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} S_\eta(\omega/\lambda) F_z^\dagger(\omega) F_z(\omega)\right), \tag{4.58}$$

with similar expressions for Eq. (4.56), which is clearly not equal to Eq. (4.57) in general. Equivalently, stretching the pulse amounts to "compressing" a filter function response by a factor of λ, which shifts the filter function to lower frequencies, and thus the overlap with low-frequency noise will likely increase by a factor greater than λ. An example of the impact of pulse stretching on a sample filter function is shown in Fig. 4.13a. Gate Trotterization is similar in spirit to pulse

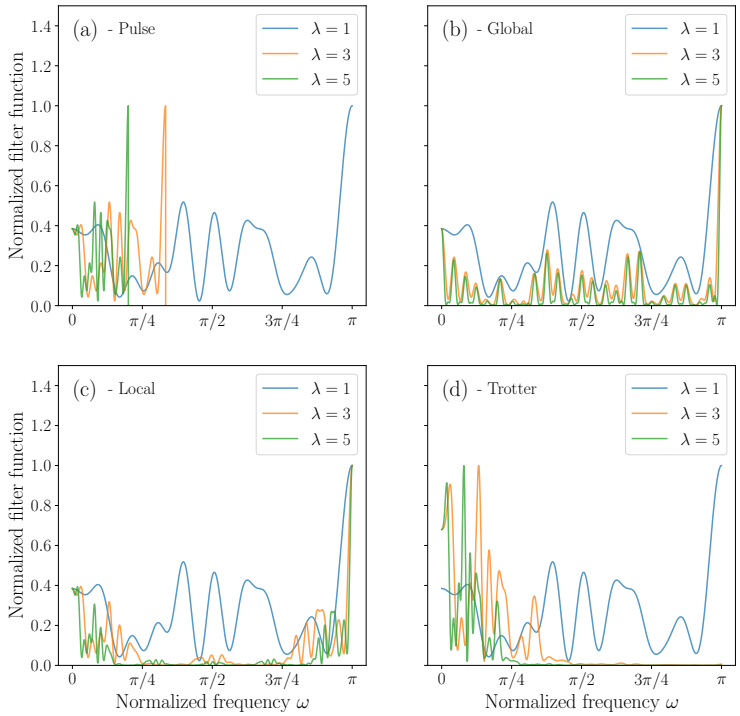


Figure 4.13: Largest magnitude filter function of a two-qubit randomized benchmarking circuit of Clifford depth 2 (actual depth 24) for different scale factors λ. All filter functions are normalized by their maximum values (otherwise the integral of the filter function scales by λ). Different subplots correspond to different noise scaling methods. All noise scaling methods change the frequency response of the circuit, however, global folding tends to preserve the qualitative shape of response function and, for this reason, it gives better performances for zero-noise extrapolation with colored noise.

stretching, but performed "digitally." However, repeating a gate-pulse λ times with amplitude $1/\lambda$ is in general different from stretching a gate's pulse (except in the case of rectangular pulses). Fig. 4.13d shows a similar qualitative impact of gate Trotterization on the filter function as pulse stretching, in that the filter function is compressed to the low frequencies. However, unlike pulse stretching, it is distorted and not a "perfect" compression.

Like pulse stretching and gate Trotterization, local folding also increases the proportion of the filter function that overlaps with low frequency noise, see Fig. 4.13c. However unlike pulse stretching and gate Trotterization, local folding also appears to generate response at high frequency.

Qualitatively, local folding "pulls" the filter function to the extreme frequencies from the middle of the spectrum. With these general trends, we would again expect that the overlap integrals produced would not be particularly close to direct noise scaling.

Of the noise scaling methods studied, it appears that global folding preserves the most structure from the unscaled filter function. The circuit responses shown in Fig. 4.13b shows that scaling preserves the qualitative shape of the base circuit's filter function. Qualitatively, it looks like the impact of global folding serves to "resolve" a coarse frequency response of the base circuit. Thus, scaling in this case preserves some structure and produces overlap integrals that are somewhat close to direct noise scaling.

These observations in the different noise scaling strategies explain the trends in Figs. 4.9 and 4.10. As global folding produces scaled filter functions that best preserve the general balance across different frequency ranges, the overlap integrals of the globally folded circuits are the closest to the ideal scaling produced by direct noise scaling. The remaining three scaling approaches all produce some level of concentration at low frequencies, and thus tend to have much greater overlap with the low-frequency noise here. As the pulse stretching and gate Trotterization approaches are very similar in spirit, they produce similar extrapolations. Furthermore, unlike local folding, these two approaches have all their concentration at low frequency, thus producing the most overlap leading to the worst extrapolation error. Local folding, which includes some high frequency content (based on the proportion of the original circuit's frequency response above $\pi/2$), produces overlaps that lie between the global folding and the stretching/Trotterization approaches.

We note that the trends observed above and the intuition behind them is a direct consequence of the correlated noise classes considered, all of which are fundamentally low frequency. Thus, pulse stretching, gate Trotterization, and local folding produce larger overlaps with the low-frequency noise and drastically bias the noise extrapolation process. In contrast, if the noise was band limited (say between $\pi/4$ and $3\pi/4$ in normalized frequency) we would expect that global folding would continue to track direct noise scaling the best. However, analysis of the other three techniques would be challenging as the overlap integral with these would essentially vanish as the scaling

increased. Without knowing the true expectation and the underlying noise spectra, it would be unclear if the leveling out of the scaled expectation values would be due to the overlap integral approaching infinity (i.e., too much noise) or vanishing (i.e., decoupling from the noise). Similarly, if the noise were purely high frequency, we would expect the pulse stretching and gate Trotterization approaches to be insensitive, local folding method to be more sensitive, and global folding between them. Finally, extremely narrow band noise could potentially lie in a "valley" in the scaled response (obviously this is circuit dependent), and thus overlap integrals would vanish for all the noise scaling approaches considered here.

4.2.5 Conclusion

In this work, we have demonstrated the effect of time-correlated noise on zero-noise extrapolation. Using the SchWARMA technique to model time-correlated dephasing noise, we presented the results of several numerical experiments showing that global unitary folding produces the lowest error relative to direct noise scaling. We analyzed our observed results and provided a physical interpretation in terms of the spectral analysis of the considered noise scaling methods.

A takeaway from our work is to use global noise scaling in zero-noise extrapolation, if possible, whenever noise may be time-correlated. An obvious important consideration is which quantum computer architectures may have time-correlated noise, a question we do not consider in this paper and leave to future work. We note that global folding is not the only possible noise scaling method suitable for time-correlated noise: other methods could be defined and analyzed, e.g. folding the first half and second half of the gates in a unitary separately. Our work provides the theoretical and practical tools to analyze the performance of such methods under a wide variety of noise models.

Data availability Software for reproducing all numerical results is available at https://github.com/mezze-team/mezze.

4.2.6 Consistency between different theories of pulse-stretching

Our work is based on a semi-classical theory of time-correlated noise, according to which, the pulse-stretching technique induces two effective changes on the noise spectrum: (i) it scales the noise level by a constant λ, (ii) it also stretches the noise spectrum on the frequency axis by the same constant. Both effects are formally summarized in Eq. (4.47) derived in the main text.

In Ref. [166], a different formalism, based on a master equation with a time-independent noise operator, was used to study the pulse-stretching technique. More precisely, a system evolving according to the following master equation was considered:

$$\frac{\partial}{\partial t}\rho(t) = -[K(t), \rho(t)] + \mathcal{L}(\rho(t)), \tag{4.59}$$

where $K(t)$ is the system Hamiltonian and $\mathcal{L}$ is a time-independent noise super-operator. As shown in Ref. [166], the effect of pulse stretching (i.e., $K(t) \longrightarrow 1/\lambda K(t/\lambda)$) is equivalent to an effective master equation:

$$\frac{\partial}{\partial t'}\rho(t') = -[K(t'), \rho(t')] + \lambda\mathcal{L}(\rho(t')), \tag{4.60}$$

where $t' = \lambda t$. In practice pulse-stretching induces a multiplicative scaling of the noise operator $\mathcal{L} \longrightarrow \lambda\mathcal{L}$.

The master equation Eq. (4.59) is typically used to model Markovian noise (no time-correlations). In this case, the Hilbert space of the environment can be traced out such that ρ represents the reduced state of the system evolving according to the master equation Eq. (4.59). In this white-noise regime, also our semi-classical theory of pulse-stretching predicts a simple multiplicative scaling of noise power and this is indeed consistent with Eq. (4.60).

What happens for a non-Markovian environment with a colored noise spectrum? In this case, our semi-classical theory suggests that pulse-stretching induces, in addition to a multiplicative scaling, also a scaling of the frequency axis of the noise spectrum (see Eq. (4.47)). This may seem to contradict the simple multiplicative scaling of the noise $\mathcal{L} \longrightarrow \lambda\mathcal{L}$ derived in Ref. [166] and reported in Eq. (4.60). However, as explained below, both theoretical derivations are actually consistent with each other.

In principle, the master equation (4.59) can be used to model a non-Markovian bath by representing with ρ the global quantum state (system + bath) instead of the reduced state of the system. In this global picture, a non-Markovian bath can be modeled by a time-independent noise operator $\mathcal{L}(\rho)$ that includes an interaction Hamiltonian term H_{SB} and the bare Hamiltonian H_{B} acting on the bath only (see Supplemental Material of Ref. [166])

$$\mathcal{L}(\rho(t)) = -i[H_{SB} + H_{\mathrm{B}}, \rho(t)], \tag{4.61}$$

which we can split as the sum of two terms $\mathcal{L} = \mathcal{L}_{SB} + \mathcal{L}_{\mathrm{B}}$, where $\mathcal{L}_{SB}(\rho) = -i[H_{SB}, \rho]$ and $\mathcal{L}_{\mathrm{B}}(\rho) = -i[H_{\mathrm{B}}, \rho]$. In this case, the simple multiplicative scaling $\mathcal{L} \longrightarrow \lambda\mathcal{L}$ induced by the pulse-stretching technique according to Eq. (4.60) has actually two physically different effects: (i) $\mathcal{L}_{SB} \longrightarrow \lambda\mathcal{L}_{SB}$ corresponding to a scaling of the noise power and (ii) $\mathcal{L}_{B} \longrightarrow \lambda\mathcal{L}_{B}$ corresponding to an effective scaling of the all the characteristic frequencies of the bath and, therefore, to a frequency stretching of the noise spectrum. These two effects are consistent with the semi-classical theory of pulse-stretching presented in this work and, in particular, with Eq. (4.47).

4.3 Increasing the effective quantum volume of quantum computers

4.3.1 Introduction

Quantum volume [8] is a single-number metric which, loosely speaking, reports the number of usable qubits on a quantum computer[1]. While improvements to the underlying hardware are a direct means of increasing quantum volume, the metric is "full-stack" and can be increased by an improvement to any component, e.g. software for compilation to produce an equivalent quantum circuit with fewer elementary operations [175].

Given an m qubit quantum circuit C, the *heavy set* is $\mathcal{H}_C := \{z \in \{0,1\}^m : p(z) > p_{\mathrm{median}}\}$ where $p(z) := |\langle z|C|0\rangle|^2$ is the probability of sampling bitstring z and p_{median} is the median probability over all bitstrings. A *heavy bitstring* is one in the heavy set. Quantum volume is

[1] Some authors define quantum volume as the effective Hilbert space dimension. Here we report the logarithm of this number which corresponds to the number of qubits.

determined by counting the number of heavy bitstrings n_h measured over n_c random circuits, each sampled n_s times. If the experiment is run with m qubit circuits of depth $d = m$, $n_c \geq 100$, and

$$\hat{h}_d := n_h / n_c n_s > 2/3 + 2\sigma \tag{4.62}$$

where σ is the standard deviation of the estimate, then the volume is at least m. The actual volume is the largest m such that these conditions are true. The particular structure of these random circuits, which we refer to as *quantum volume circuits*, is defined in [8].

4.3.2 Method

Given a quantum volume circuit C, we define the projector on the heavy subspace

$$\Pi_{h,C} := \sum_{z \in \mathcal{H}_C} |z\rangle\langle z| \tag{4.63}$$

so that the expected number of heavy bitstrings for this circuit is $n_{h,C} := n_s \langle 0|C^\dagger \Pi_{h,C} C|0\rangle$. We use zero-noise extrapolation (ZNE) [176, 177] with $\Pi_{h,C}$ as the observable for each quantum volume circuit C to estimate the noise-free value of $n_h := \sum_C n_{h,C}$. This amounts to evaluating $\langle \Pi_{h,C}^{(\lambda)} \rangle$ at several noise-scale factors $\lambda \geq 1$ then using these results to estimate $\langle \Pi_{h,C}^{(0)} \rangle$, i.e., the zero-noise limit of the heavy output probability. In practice, this means compiling the circuit C to a set of circuits $\{C_{\lambda_i}\}_{i=1}^k$. For fairness with the unmitigated experiment, we use n_s/k samples for each C_{λ_i} so that the total number of samples drawn is equal in the mitigated and unmitigated experiments. The main difference to previous work improving quantum volume by compiling [175] is that we compile a single circuit to a set of circuits, following the pattern of many error mitigation methods (e.g. [177]), in contrast with compilation that does rewrites on a single circuit following algebraic rules or optimized routing.

After executing each C_{λ_i} to obtain scaled heavy output counts $n_{h,C}^{(i)}$, we use Richardson extrapolation [177, 178] to estimate the zero-noise result via

$$n_{h,C}^{(0)} = \sum_{i=1}^{k} \eta_i n_{h,C}^{(i)} \tag{4.64}$$

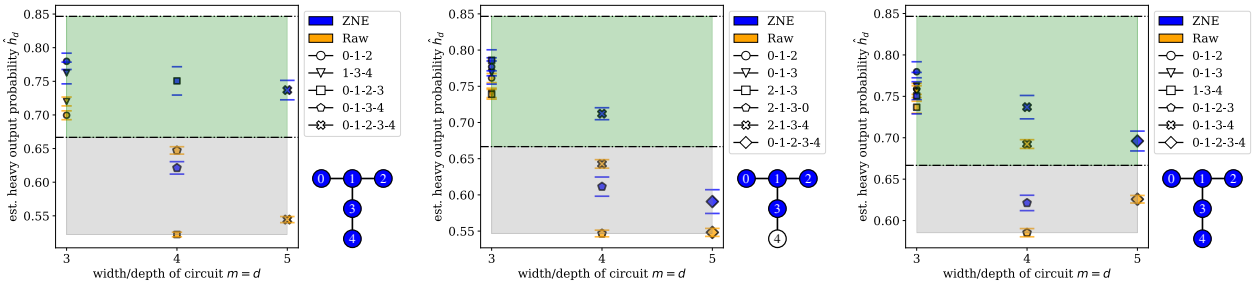


Figure 4.14: Results of unmitigated and mitigated quantum volume experiments on three five-qubit quantum computers (left-to-right: Belem, Lima, and Quito) using $n_c = 500$ circuits and $n_s = 10^4$ total samples. Each marker shows the estimated heavy output probability $\hat{h}_d$ on a different qubit configuration defined in the legend and error bars show 2σ intervals evaluated by bootstrapping. The connectivity of each device is shown below each legend. Dashed black lines show the $2/3$ threshold and noiseless asymptote $(1+\ln 2)/2$ [8]. For the mitigated experiments, $\lambda_i \in \{1, 3, 5, 7, 9\}$ and $n_s = 10^4/5$. Local unitary folding of two-qubit gates is used to compile the circuits (i.e., scale noise) and Richardson's method of extrapolation is used to infer the zero-noise result. The qubit subsets which achieved the largest quantum volume in the mitigated experiments are colored blue in each device diagram. As can be seen, on Belem error mitigation increases the effective quantum volume from three to five, on Lima error mitigation increases the effective quantum volume from three to four, and on Quito error mitigation increases the effective quantum volume from four to five.

where coefficients are given by

$$\eta_i := \prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i}. \tag{4.65}$$

In practice, we use $\lambda_i \in \{1, 3, 5, 7, 9\}$ and scale circuits by locally folding two-qubit (CNOT) gates [178]. In other words, the scaled circuit for $\lambda_i = t$ has each CNOT replaced by t CNOTs.

4.3.3 Results

Using this strategy, we perform unmitigated and mitigated quantum volume experiments on the Belem, Lima, and Quito devices available through IBM [1] (see Section 4.3.6 for device specifications). The results, shown in Fig. 4.14, demonstrate that we are able to increase the effective quantum volume from three to five on Belem, three to four on Lima, and from four to five on Quito. Note that ZNE increases the estimated heavy output probability $\hat{h}_d$ on all qubit subsets even though the $2/3$ threshold is not always crossed. We therefore expect that ZNE can increase effective quantum volume independent of the size of the device, so long as cross-talk and other

errors do not scale with the device size. The largest ZNE experiment to date was performed on 26 qubit circuits with 1080 two-qubit gates [179] and, in the context of our work, provides evidence that error mitigation can continue to increase the effective volume of larger quantum devices, e.g. those listed in Section 4.3.7.

Because we estimate the noiseless result by taking a linear combination of noisy results, the way we compute σ in (4.62) changes relative to [8]. For any technique, such as Richardson extrapolation, that evaluates an error-mitigated expectation value as a linear combination of noisy expectation values (4.64), one can show (see Section 4.3.8) that

$$\sigma^2 = \frac{1}{n_c^2} \sum_C \sigma_C^2, \qquad \sigma_C^2 = \sum_{i=1}^{k} |\eta_i|^2 (\sigma_C^{(i)})^2, \tag{4.66}$$

where $(\sigma_C^{(i)})^2$ is the variance of each noise-scaled expectation value, while σ_C^2 is the variance of the error-mitigated expectation value associated to the quantum circuit C. The previous expressions correspond to a theoretical estimate of the error, but in practice we can estimate error bars by repeating the experiment multiple times or by bootstrapping. The 2σ error bars in Fig. 4.14 are calculated by bootstrapping with 500 resamples. See Section 4.3.8 for more details.

4.3.4 Discussion

There is a subtle point in interpreting our results in the general context of quantum computer performance. Our error mitigation procedure improves the expectation value of the heavy output projector but does not produce more heavy bitstrings — in fact, our procedure likely produces fewer heavy bitstrings because we distribute samples across circuits at amplified noise levels. However, as we have shown, we are able to use this information to estimate the expected number of heavy bitstrings in a statistically significant way. To carefully distinguish between the two cases, we refer to our results as increasing the *effective* quantum volume.

The restriction to evaluating expectation values but not directly sampling bitstrings raises interesting questions about physicality and the role of a quantum computer in a computational procedure. If an algorithm only requires expectation values and we apply the error mitigation

procedure used in this work, is it the case that we effectively have access to a quantum computer with a larger quantum volume? These questions are linked to the physical interpretation of error mitigation. One way to interpret ZNE is that we evaluate expectation values with respect to the "extrapolated density matrix"

$$\rho_0 = \sum_i \eta_i \rho_{\lambda_i} \tag{4.67}$$

where ρ_{λ_i} are the noise-scaled physical states and η_i are the real coefficients in (4.64). Clearly we did not physically prepare ρ_0 in our experiment, but should we restrict the use of a quantum computer to only preparing a single physical state from which we can sample bitstrings? Or do we allow ourselves to "virtually" prepare non-physical but mathematically well-defined states from which we can compute expectation values more accurately? We note that similar questions have been asked in the context of virtual distillation techniques [180, 129] which have been proposed to artificially purify a quantum state or to reduce its effective temperature [181].

4.3.5 Conclusion

In this work we have experimentally demonstrated that error mitigation improves the effective quantum volume of several quantum computers. We use the term *effective* quantum volume to emphasize that our procedure is appropriate for algorithms computing expectation values and not for algorithms requiring individual bitstrings. The error mitigation technique is not tailored to the structure of quantum volume circuits or to the architecture of the quantum computers we used. Indeed, we did not run any additional calibration experiments or use any calibration information to obtain our results. Similar software-level techniques have been used in previous quantum volume experiments, e.g. (approximate) compilation [8, 175] and dynamical decoupling [175]. The novelty of our proposal is that, by relaxing the strong requirement of directly sampling heavy bitstrings to the weaker requirement of estimating the expectation value of the heavy output projector, more general error mitigation techniques can be applied to improve the effective quantum volume of a device. We expect this approach to improve the effective quantum volume of additional quantum

	Lima	Belem	Quito
# Qubits	5	5	5
ϵ_{1Q}	4.446×10^{-4}	2.808×10^{-4}	2.980×10^{-4}
ϵ_{CNOT}	1.131×10^{-2}	1.098×10^{-2}	8.292×10^{-3}
ϵ_{M}	3.790×10^{-2}	2.868×10^{-2}	2.546×10^{-2}

Table 4.3: Device specifications and error rates for the quantum computers we used in our experiments. Device connectivities are shown in Fig. 4.14. Parameters ϵ_{1Q}, ϵ_{CX}, ϵ_{M} denote, respectively, averages (over all qubits) of single-qubit $\sqrt{X}$ gate errors, two-qubit CNOT gate errors, and readout errors $(p(0|1) + p(1|0))/2$ accessed from [1].

computers such as those in Section 4.3.7. Our open source error mitigation software [182] can be used on many quantum computers to repeat the experiments we performed here.

In the context of error mitigation, our work provides additional benchmarks to the relatively few experimental results in literature [183, 182, 179, 184, 185, 186, 187]. We encourage the use of quantum volume as a benchmark for error mitigation techniques due to its relatively widespread adoption and clear operational meaning. Normalizing by additional resources used (gates, shots, qubits, etc.) in error-mitigated quantum volume experiments provides a way to directly compare different techniques and drive progress in this area. As most error mitigation techniques act on expectation values, they can be used for effective quantum volume experiments as we have done in this work.

Code and data availability The code we used to run experiments as well as the data we collected are available at https://github.com/unitaryfund/mitiq-qv.

4.3.6 Device specifications

In Table 4.3 we provide more information about the quantum computers we used in our experiments. Note that the quantum volume of Belem is listed as four at [1] but we are unable to reproduce this result in our unmitigated experiments, presumably due to device degradation over time.

Quantum computer	log QV	Reference
Rigetti Aspen-4	3	[188]
Lima	3 (4)	[1] (this work)
Belem	3 (5)	[1] (this work)
Jakarta	4	[1]
Bogota	4	[1]
Quito	4 (5)	[1] (this work)
Manila	5	[1]
Nairobi	5	[1]
Lagos	5	[1]
Perth	5	[1]
Guadalupe	5	[1]
Toronto	5	[1]
Brooklyn	5	[1]
Trapped-ion QCCD	6	[189]
Hanoi	6	[1]
Auckland	6	[1]
Cairo	6	[1]
Washington	6	[1]
Mumbai	7	[1]
Kolkata	7	[1]
Honeywell System Model H1	10	[190]

Table 4.4: Measured quantum volumes (in increasing order). Values in parentheses show effective quantum volumes measured in this work.

4.3.7 Table of quantum volumes

As discussed in the main text, error mitigation consistently increased the estimated heavy output probability in all of our experiments. To increment the effective quantum volume of a device, this increase must cross the $2/3$ threshold with statistical significance. While there is no guarantee that this will happen, we expect there to be several cases of already-measured quantum volumes where this will be true. For this reason, as well as general context, we include a list of quantum computer volumes in Table 4.4.

4.3.8 Statistical uncertainty of error-mitigated volume

4.3.8.1 Theoretical estimation of error bars

For a large number of error mitigation techniques, including Richardson extrapolation, an error-mitigated expectation value E_C associated with an ideal circuit C is evaluated as linear combination of different noisy expectation values:

$$E_C = \sum_j \eta_j \tilde{E}_j. \tag{4.68}$$

Because of shot noise, each noisy expectation value $\tilde{E}_j$ can only be measured up to a statistical variance $\sigma_j^2 = \mathbb{E}(\tilde{E}_j^2) - [\mathbb{E}(\tilde{E}_j)]^2$, where $\mathbb{E}$ represents the statistical average over n_j measurement shots.

Since different noisy expectation values are statistically uncorrelated, the variance σ_C^2 of the error-mitigated result E_C is:

$$\sigma_C^2 = \mathbb{E}(E_C^2) - [\mathbb{E}(E_C)]^2 = \sum_j |\eta_j|^2 \sigma_j^2. \tag{4.69}$$

If we assume that each noisy expectation value is obtained by sampling a binomial distribution $\mathcal{B}(p_j, n_j)$ with probability $p_j = \tilde{E}_j$ and normalizing the result over n_j measurement shots, we have $\sigma_j^2 = E_j(1 - E_j)/n_j$. The variance σ_C^2 of the error-mitigated result is therefore:

$$\sigma_C^2 = \sum_{j=1}^{k} |\eta_j|^2 \tilde{E}_j(1 - \tilde{E}_j)/n_j. \tag{4.70}$$

The previous expression is valid for a generic expectation value. In the specific case of a quantum volume experiment, we can identify with E_C the heavy-output probability associated with a specific random circuit C. Averaging E_C over multiple n_c noisy circuits C of depth d, produces the estimated heavy output probability visualized in Fig. 4.14:

$$h_d = \frac{1}{n_c} \sum E_C. \tag{4.71}$$

This is again a sum of independent random variables and so its variance is given by:

$$\sigma^2 = \frac{1}{n_c^2} \sum_C \sigma_C^2. \tag{4.72}$$

4.3.8.2 Bootstrapping empirical error bars

The previous way of estimating error bars is based on theoretical assumptions and, even though it provides useful analytical expressions, it may underestimate unknown sources of errors such as systematic errors.

A brute-force way of estimating error bars is to repeat the estimation of the quantity of interest (in our case h_d) with N independent experiments and to evaluate the empirical variance of the results. This method can be expensive with respect to classical and quantum computational resources, and the results can be sensitive to how the independent samples are grouped. So while we can split independent samples into five groups of $n_c = 100$ circuits to estimate the standard deviation this way, a more feasible alternative is instead given by bootstrapping. This is a statistical inference technique in which, instead of performing N new experiments, one *resamples* the raw results of a single experiment N times in order to estimate properties of the underlying statistical distribution.

In our specific quantum volume experiment, the heavy-output probability h_d is estimated as an average over n_C random circuits C as shown in equation (4.71). Let us define the set

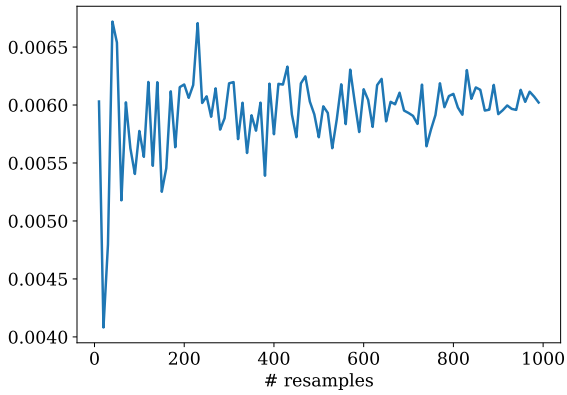


Figure 4.15: The value of σ for different resampling numbers in bootstrapping.

$S = \{E_{C_1}, E_{C_2}, ...E_{C_{n_C}}\}$ containing all the estimated heavy-output probabilities associated with different random circuits. We can now resample N sets of data $S_1, S_2, ...S_N$, each one containing n_c values that are randomly sampled from S with replacements. For each resampled set S_j we evaluate the associated bootstrapped mean $\mu_j = \langle E_C \rangle_{S_j}$.

The empirical standard deviation of all the bootstrap means $\{\mu_1, \mu_2, ...\mu_N\}$ provides an estimate of the statistical uncertainty:

$$\sigma = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\mu_j - \bar{\mu})^2}. \tag{4.73}$$

where $\bar{\mu} = \sum_{j=1}^{N} \mu_j / N$. This is the method that we used to evaluate error bars in Fig. 4.14 (with $N = 500$).

One may ask how large should N be, in order to obtain a reasonable estimate of the error. In Fig. 4.15 we show the dependence of σ for an arbitrary error-mitigated point of Fig. 4.14 (the results are qualitatively similar for all points). Fig. 4.15 provides empirical evidence that, for $N > 400$, the bootstrapped estimate converges around a stable result.

CHAPTER 5

LOGICAL SHADOW TOMOGRAPHY

5.1 Background

5.1.1 Subspace expansion

Let $\mathcal{S} = \langle S_1, ..., S_r \rangle$ be a stabilizer code with generators $S_1, ..., S_r$. Recall from Chapter 1.4 that a stabilizer code (group) $\mathcal{S}$ is a subgroup of $\mathcal{P}_n$, the Pauli group on n-qubits, such that $-I \notin \mathcal{S}$ and $\mathcal{S}$ is abelian. These properties ensure the codespace defined by $\mathcal{S}$ is non-trivial. The *codespace* $V_{\mathcal{S}}$ of $\mathcal{S}$ is the span of the codewords of $\mathcal{S}$, where a *codeword* is a plus one eigenstate of all stabilizer elements. That is,

$$V_{\mathcal{S}} := \text{span } \{|c_1\rangle, ..., |c_k\rangle\} \tag{5.1}$$

where each codeword $|c_i\rangle$ for all $i = 1, ..., k$ satisfies $S|c_i\rangle = |c_i\rangle$ for all $S \in \mathcal{S}$. Note that the dimension of the codespace is $\dim(V_{\mathcal{S}}) = k = 2^{n-r}$.

Given a generator $S_i \in \{S_1, ..., S_r\}$, define the projector

$$P_i := \frac{1}{2}(I + S_i). \tag{5.2}$$

Note that $P_i^2 = \frac{1}{4}(I + 2S_i + S_i^2) = \frac{1}{4}(2I + 2S_i) = P_i$ so that P_i is indeed a projector. Now, define the *complete projector* as follows.

Definition 8. Given a stabilizer $\mathcal{S} = \langle S_1, ..., S_r \rangle$, the complete projector is defined as

$$P := \prod_{i=1}^{r} P_i = \prod_{i=1}^{r} \frac{1}{2}(I + S_i). \tag{5.3}$$

Theorem 9: The complete projector can be written as a convex combination over all stabilizer elements

$$P = \frac{1}{2^r} \sum_{S \in \mathcal{S}} S. \tag{5.4}$$

Proof. This follows from expanding (5.3) and using closure. Note that there are 2^r elements in $\mathcal{S}$. □

For quantum subspace expansion [127], we want to compute the expectation value $\langle\Gamma\rangle \equiv \mathrm{Tr}[\rho\Gamma]$ for an observable

$$\Gamma := \sum_{i=1}^{m} \gamma_i \Gamma_i. \tag{5.5}$$

We start in the codespace with a state $|\psi\rangle\langle\psi|$ and evolve unitarily to $\rho := U|\psi\rangle\langle\psi|U^\dagger$. Suppose some error E_i occurs which takes us out of our codespace, i.e.

$$|\psi\rangle \mapsto E_i|\psi\rangle. \tag{5.6}$$

If E_i is correctable by our stabilizer code, then there exists a projector P_i such that $P_i E_i|\psi\rangle = |\psi\rangle$ (ignoring any renormalization). Hence for any correctable errors the complete projector (5.3) will map the errored state $|\psi\rangle_{\text{error}} \in V_{\mathcal{S}}^{\perp}$ back into the codespace $V_{\mathcal{S}}$. The idea of quantum subspace expansion is thus to measure

$$\langle\Gamma\rangle_{\text{corrected}} := \frac{1}{c}\mathrm{Tr}[P\rho P^\dagger\Gamma] \tag{5.7}$$

where c is a normalization factor. Using (5.3) and (5.5), we can expand this as

$$\langle\Gamma\rangle_{\text{corrected}} = \frac{1}{2^{2r}c}\sum_{i=1}^{m}\sum_{j=1}^{2^r}\sum_{k=1}^{2^r}\gamma_i \mathrm{Tr}[S_j\rho S_k^\dagger\Gamma_i]. \tag{5.8}$$

Here each Γ_i is a logical operator of the stabilizer code $\mathcal{S}$. For subspace codes, all logical operators commute with stabilizers. Using cyclicity of the trace along with this property, we can write

$$\langle\Gamma\rangle_{\text{corrected}} = \frac{1}{2^{2r}c}\sum_{i=1}^{m}\sum_{j=1}^{2^r}\sum_{k=1}^{2^r}\gamma_i \mathrm{Tr}[\rho S_k^\dagger S_j\Gamma_i]. \tag{5.9}$$

Again by closure, the product $S_k^\dagger S_j$ is another stabilizer element. This allows us to eliminate one summation and arrive at the following theorem.

Theorem 10: If $[\Gamma_i, S_j] = 0$ for all logical operators Γ_i and stabilizers S_j (which is true for subspace codes), then

$$\langle\Gamma\rangle_{\text{corrected}} = \frac{1}{2^r c}\sum_{i=1}^{m}\sum_{j=1}^{2^r}\gamma_i \mathrm{Tr}[\rho S_j\Gamma_i]. \tag{5.10}$$

We remark that for subsystem codes, not all logical operators commute with stabilizers, so (5.10) is invalid for subsystem codes. Here, there is an additional "residue" term containing $[\Gamma_i, S_j] \neq 0$. The full expression is

$$\langle \Gamma \rangle_{\text{corrected}} = \frac{1}{2^r c} \sum_{i=1}^{m} \sum_{j=1}^{2^r} \gamma_i \text{Tr}[\rho S_j \Gamma_i] - \frac{1}{2^{2r} c} \sum_{i=1}^{m} \sum_{j=1}^{2^r} \sum_{k=1}^{2^r} \gamma_i \text{Tr}[\rho S_k^\dagger [\Gamma_i, S_j]] \tag{5.11}$$

5.1.2 Virtual distillation

Given a noisy state $\rho_{\mathcal{E}} = \mathcal{E}(\rho)$, write its spectral decomposition

$$\rho_{\mathcal{E}} = p_0 |\psi_0\rangle\langle\psi_0| + \sum_k p_k |\psi_k\rangle\langle\psi_k| \tag{5.12}$$

where $p_0 > p_1 \geq \cdots \geq 0$, and $\langle\psi_i|\psi_j\rangle = \delta_{ij}$ (the Kronecker delta). Assume that the ideal (noiseless) state is $\rho = |\psi_0\rangle\langle\psi_0|$. (In an idealized noise model, we can think of the noise as adding "orthogonal errors", e.g. if our ideal state is $|0\rangle\langle 0|$ then under bitflip noise with probability $p < 1/2$ we obtain the noisy state $\rho_{\mathcal{E}} = (1-p)|0\rangle\langle 0| + p|1\rangle\langle 1|$. See [128] for more justification of this assumption.) Given an observable O, the idea of virtual distillation [128, 129] is to evaluate the quantity

$$\langle O \rangle_{\text{VD}} := \frac{\text{Tr}[\rho_{\mathcal{E}}^m O]}{\text{Tr}[\rho_{\mathcal{E}}^m]} \tag{5.13}$$

for positive integer m. Using the spectral decomposition (5.12), one can show that

$$\langle O \rangle_{\text{VD}} = \langle O \rangle \left[\frac{1 + \sum_k (p_k/p_0)^m \langle\psi_k|O|\psi_k\rangle / \langle O \rangle}{1 + \sum_k (p_k/p_0)^m} \right] \tag{5.14}$$

where $\langle O \rangle := \langle\psi_0|O|\psi_0\rangle$ is the true (noiseless) expectation value by assumption. It's easy to see that $\lim_{m\to\infty} \langle O \rangle_{\text{VD}}(m) = \langle O \rangle$. In fact, to leading order

$$\langle O \rangle_{\text{VD}} = \langle O \rangle \left[1 + \mathcal{O}\left((p_1/p_0)^m \right) \right], \tag{5.15}$$

showing that errors are suppressed exponentially in the power m.

The remaining question is how to evaluate (5.13). Authors of [128, 129] propose physically preparing m copies of the state then using the "SWAP trick" $\text{Tr}[S_m \rho^{\otimes m} O_1] = \text{Tr}[\rho^m O]$. Here, S_m is

the SWAP or shift operator which cyclically permutes m copies of a state $S_m|1\rangle\otimes|2\rangle\otimes|3\rangle\otimes\cdots|m\rangle = |2\rangle\otimes|3\rangle\otimes\cdots|m\rangle\otimes|1\rangle$ and O_1 is the observable O acting on the first copy of the state $\rho^{\otimes m}$. The denominator of (5.13) is computed in the same way but by omitting O_1. Even for small m, this requires many additional qubits and operations to prepare the states, and the SWAP operation is very hard to implement experimentally. Recognizing these challenges, Ref. [191] proposes using active reset to reduce the number of ancilla qubits needed for this approach. Our logical shadow tomography procedure only requires sampling from a single copy of the state. We now describe this method and how it encapsulates both subspace expansion and virtual distillation with significantly fewer resources.

5.2 Introduction

In this chapter, we are interested in error mitigation in the region between noisy-intermediate scale quantum (NISQ) and fault-tolerant quantum computation. We present a conceptually simple and practical QEM technique. In our setup, we use quantum error correction code to distribute logical qubits information with multiple physical qubits. Without active quantum error correction which requires extra ancilla qubits and parity check measurements, the only step which happens on a quantum computer is repeatedly sampling from an encoded state to perform shadow tomography [192, 193]. After this, one can use classical post-processing to project out errors as in subspace expansion [127] and calculate powers of the density matrix as in virtual distillation [128]. In addition to enabling both of these methods, our technique requires significantly fewer resources. Specifically, we show that (i) the quantum gate overhead is independent of the number of logical qubits and we do not require multiple copies of the physical systems, (ii) the sample complexity for estimating error mitigated Pauli observables only scales with the number of logical qubits instead of the total number of physical qubits, and (iii) there exists an efficient classical algorithm that can post-process data with polynomial time. We show the new method is practical for relatively large systems with numerical experiments.

5.3 Logical shadow tomography

5.3.1 Motivation

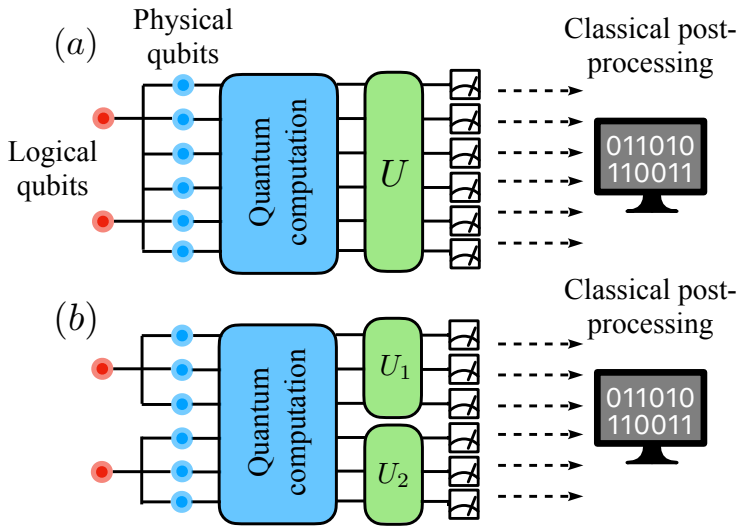


Figure 5.1: Graphic illustration of logical shadow tomography. (a) Red dots are logical qubits, and blue dots are physical qubits. Logical information is distributed to physical qubits by error correction code, then followed by noisy quantum computation on physical qubits. To get estimation of error mitigated observables, we perform classical shadow tomography on the noisy physical state. Particularly, we can apply random Clifford gates denoted as green blocks from some unitary ensemble $\mathcal{U}$, and take computational basis measurements. (b) A special case using $[[n, 1]]$ code for each logical qubit. In shadow tomography, we apply random unitary from tensor product of Clifford groups $\mathcal{C}\ell(2^n)^{\otimes k}$. Additional gate depth will not scale with number of logical qubits k, and sample complexity for estimating error mitigated logical Pauli observables is the same as using global Clifford group $\mathcal{C}\ell(2^{nk})$.

In this work, we are interested in error mitigation in the region between noisy intermediate-scale quantum (NISQ) and fault-tolerant quantum computation, where the number of qubits is more than a few but cannot fulfill the requirement of fault-tolerance. We ask the question whether we can use those resources cleverly to achieve a more reliable quantum computation. QEM methods related to this idea are subspace expansion [127] and the virtual distillation [128]. The key contribution of this work is to propose the application of classical shadow tomography [192, 193] to reduce the quantum and classical resources needed to perform these QEM schemes, and provide rigorous analysis on its error mitigation capability.

The subspace expansion approach starts by encoding k logical qubits with N physical qubits via an error correction code $[[N, k]]$. Consider using a stabilizer code defined by a stabilizer group $\mathcal{S} = \langle S_1, S_2, \cdots, S_{N-k} \rangle$, the code subspace is specified by the projection operator

$$\Pi = \prod_{j=1}^{N-k} \frac{I + S_j}{2} = \frac{1}{2^{N-k}} \sum_{M \in \mathcal{S}} M, \tag{5.16}$$

where M denote group elements in $\mathcal{S}$ as products of the stabilizers. Errors may occur to the physical state during the quantum information processing, which generally takes the physical state away from the code subspace. Suppose the goal is to estimate the logical observable O only, then even without active error correction, a simple projection of the the corrupted physical state $\rho_{\mathcal{E}}$ back to the code subspace can already mitigate the error for the logical observable [127]

$$\langle O \rangle_{\text{QEM}} = \frac{\text{Tr}(\Pi \rho_{\mathcal{E}} \Pi^\dagger O)}{\text{Tr}(\Pi \rho_{\mathcal{E}} \Pi^\dagger)} = \frac{1}{2^{n-k} c} \sum_{M \in \mathcal{S}} \text{Tr}(\rho_{\mathcal{E}} M O), \tag{5.17}$$

where $c = \text{Tr}(\Pi \rho_{\mathcal{E}} \Pi^\dagger)$. This amounts to measuring $\langle MO \rangle$ on the corrupted physical state for all elements M in the stabilizer group $\mathcal{S}$ (or for a majority of M sampled from $\mathcal{S}$). This approach can quickly become exponentially expensive when $N - k$ becomes large.

Another approach for QEM goes under the name of virtual distillation. Assuming the target state is a pure state as the leading eigen state of $\rho_{\mathcal{E}}$. The sub-leading eigen states of $\rho_{\mathcal{E}}$ (as errors orthogonal to the target state) can be suppressed by powering the density matrix $\rho_{\mathcal{E}}^m$, and we estimate $\langle O \rangle_{\text{QEM}} = \text{Tr}(\rho_{\mathcal{E}}^m O)/\text{Tr}(\rho_{\mathcal{E}}^m)$. Or more generally, a polynomial function $f(\rho_{\mathcal{E}}) = c_0 I + c_1 \rho_{\mathcal{E}} + c_2 \rho_{\mathcal{E}}^2 + \cdots + c_m \rho_{\mathcal{E}}^m$ of $\rho_{\mathcal{E}}$ can be considered, with an optimal choice of the coefficients $c_0, \cdots, c_m$ to best mitigate the error, such that

$$\langle O \rangle_{\text{QEM}} = \frac{\text{Tr}(f(\rho_{\mathcal{E}}) O)}{\text{Tr}(f(\rho_{\mathcal{E}}))}. \tag{5.18}$$

However, powering a density matrix on quantum devices typically involves making multiple copies of the quantum system, which can be challenging and expensive in quantum resources.

The key observation of this work is that both Eq. (5.17) and Eq. (5.18) (or their combination) can be efficiently evaluated in the classical post-processing phase after performing the classical

shadow tomography [193] on the noisy physical state $\rho_{\mathcal{E}}$. The classical shadow tomography uses randomized measurements to extract information from an unknown quantum state, and predicts physical properties about the state efficiently by post-processing collected measurement outcomes on a classical computer. When the measurement basis are chosen to be Pauli or Clifford basis, the post-processing can be made efficient. The code subspace projection and the powering of density matrix can all be implemented efficiently in the post-processing phase by classical computation, given the Clifford nature of the classical shadows. In this way, we can implement the existing error mitigation schemes with significantly reduced quantum and classical resources.

5.3.2 Procedure

Let us now introduce our technique which we will refer to as logical shadow tomography (LST). LST consists of the following steps[1] (see Fig. 5.1 for a graphical overview):

1. Given a k-qubit logical state, encode it into a N-qubit physical state by a $[[N, k]]$ stabilizer code.

2. Perform the quantum information processing on the physical state, the resulting physical state is denoted as $\rho_{\mathcal{E}}$. Due to the error accumulated in the processing, $\rho_{\mathcal{E}} = \mathcal{E}(\rho)$ may be corrupted from the ideal result $\rho = |\psi\rangle\langle\psi|$ by some noisy quantum channel $\mathcal{E}$. The goal is the mitigate the error for predicting logical observables based on the noisy physical state $\rho_{\mathcal{E}}$.

3. Perform shadow tomography on the noisy state.

    3.1. Apply a randomly sampled unitary U from a unitary ensemble $\mathcal{U}$ to the physical qubits. Measure all physical qubits in the computational basis to obtain a bit-string $b \in \{0, 1\}^n$. Store U, b.

    3.2. Repeat step 3.1 for N times to obtain a data ensemble $\{(U_s, b_s)\}_{s=1}^{N}$ (s labels the samples in the ensemble).

[1] Steps 2 and 3 can be aptly summarized by "perform shadow tomography [193] on the logical state," whence *logical shadow tomography*. We explain these steps in detail for completeness.

4. Post-process the data on a classical computer.

   4.1. Construct the classical shadow ensemble

$$\Sigma(\rho_{\mathcal{E}}) = \{\hat{\rho}_s = \mathcal{M}^{-1}(U_s^\dagger|b_s\rangle\langle b_s|U_s)\}_{s=1}^N, \tag{5.19}$$

   where $\mathcal{M}^{-1}$ is the classical shadow reconstruction map that depends on the unitary ensemble $\mathcal{U}$.

   4.2. Given any logical observable O (i.e. $[\Pi, O] = 0$) estimate the error-mitigated expectation value by

$$\langle O\rangle_{\text{LST}} = \frac{\text{Tr}(\Pi f(\rho_{\mathcal{E}})\Pi^\dagger O)}{\text{Tr}(\Pi f(\rho_{\mathcal{E}})\Pi^\dagger)}, \tag{5.20}$$

   where Π is the code subspace projection operator defined in Eq. (5.16), and $f(x) = \sum_{p=1}^m c_p x^p$ can be a generic polynomial function up to the power m. The proposed QEM estimator $\langle O\rangle_{\text{LST}}$ in Eq. (5.20) combines the subspace expansion Eq. (5.17) and the virtual distillation Eq. (5.18) approaches. In particular, the numerator $\text{Tr}(\Pi f(\rho_{\mathcal{E}})\Pi^\dagger O)$ is evaluated by

$$\sum_{p=1}^m c_p \mathop{\mathbb{E}}_{\{\hat{\rho}_s\}\in\Sigma(\rho_{\mathcal{E}})^{\times p}} \text{Tr}\Big(\Pi\Big(\prod_{s=1}^p \hat{\rho}_s\Big)\Pi^\dagger O\Big), \tag{5.21}$$

   and the denominator $\text{Tr}(\Pi f(\rho_{\mathcal{E}})\Pi^\dagger)$ is evaluated independently in a similar manner (by replacing O with I).

The map $\mathcal{M}^{-1}$ depends on the unitary ensemble $\mathcal{U}$ for which there are several proposals, e.g. Pauli ensembles, random Clifford circuits, and chaotic dynamics [193, 194, 195, 196]. In this work, we find the sample complexity for predicting logical Pauli observable is the same between using a full Clifford ensemble $C\ell(2^N)$ as shown in Fig. 5.1(a) and a tensor product of Clifford ensemble $C\ell(2^{N/k})^{\otimes k}$ as shown in Fig. 5.1(b). In the following, we will focus on the scheme where each qubit is encoded with a $[[n, 1]]$ error correction code and apply random unitaries from Clifford group $C\ell(2^n)$ at each logical qubit sector. And the total number of physical qubits is $N = nk$.

5.3.3 Analysis

In the previous section, we have outlined the procedure of logical shadow tomography (LST). Here, we will analyze its performance from three perspectives: 1) error mitigation capacity, 2) quantum resources, and 3) classical resources. Particularly, in *error mitigation capacity* subsection, we show how error is suppressed with the code distance d of the error correction code and powers of density matrix. In *quantum resources* subsection, we show the gate overhead is similar to the original proposal of subspace expansion, except for a shallow depth Clifford circuit whose depth does not depend on the number of logical qubits. Compared to virtual distillation, our method only requires one copy of the physical system. In addition, we also show the sample complexity has an exponential reduction compared to the direct implementation of subspace expansion for estimating logical Pauli observables. In *classical resources* subsection, we outline the general classical algorithm for post-processing the data. Particularly, we show there exists fast algorithm for LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ and its algorithm time complexity is $O(N^3)$. This allows our method scale to large system size.

5.3.3.1 Error mitigation capability

Code space projection. Let $\mathcal{S}$ be the stabilizer code used in LST. For any correctable error E, there exists a stabilizer generator S such that $SE = -ES$ and so

$$\Pi E|\psi\rangle \propto (I+S)E|\psi\rangle = E(|\psi\rangle - |\psi\rangle) = 0. \tag{5.22}$$

Analogously, if no error has occurred then $|\psi\rangle$ is a codeword and

$$\Pi|\psi\rangle = |\psi\rangle. \tag{5.23}$$

Thus the projector Π discards results in which correctable errors have occurred [127]. The set of correctable errors is determined by the chosen code. Assume a simple noise model where each qubit is subjected to depolarizing noise with rate p. If a single error happened on one qubit. Then it can be projected out given the fact that single Pauli operator is not the stabilizer group. Those errors are non-logical errors.

The code space projection fails when more local error happens and they form a logical operator. The probability of having this failure is p^d, where d is the code distance of the error correction code. Mathematically, we can write down

$$\rho_{\mathcal{E}} = (1-p)^N \rho_0 + p\rho_1 + p^2\rho_2 + \dots, \tag{5.24}$$

where $\rho_0 = |\psi_0\rangle\langle\psi_0|$ is the ideal quantum state, and N is the system size. ρ_1 is the quantum state subjected to error happened to one qubit, i.e.

$$\rho_1 = (XI\cdots I)\rho_0(XI\cdots I) + (IX\cdots I)\rho_0(IX\cdots I) + \dots \tag{5.25}$$

Similarly, ρ_2 is the quantum state subjected to two error happened, i.e.

$$\begin{aligned}\rho_2 =& (XIY\cdots I)\rho_0(XIY\cdots I) \\ &+ (IXX\cdots I)\rho_0(IXX\cdots I) + \dots\end{aligned} \tag{5.26}$$

Therefore, for any logical observable O, we have

$$\begin{aligned}&\frac{\mathrm{Tr}(\Pi\rho_{\mathcal{E}}\Pi O)}{\mathrm{Tr}(\Pi\rho_{\mathcal{E}}\Pi)} \\ &= \langle\psi_0|O|\psi_0\rangle\left[1 + O\left(p^d\frac{\mathrm{Tr}(\rho_d O)}{\langle\psi_0|O|\psi_0\rangle}\right)\right].\end{aligned} \tag{5.27}$$

More details can be found in Sec. 5.7.

Virtual distillation. For completeness, we will review the virtual distillation theory here. With the spectral decomposition of the noisy density matrix, one can write

$$\rho_{\mathcal{E}} = p_0|\psi_0\rangle\langle\psi_0| + p_1|\psi_1\rangle\langle\psi_1| + \cdots + p_n|\psi_n\rangle\langle\psi_n|, \tag{5.28}$$

where $p_0 > \cdots > p_n$ and $\langle\psi_i|\psi_j\rangle = \delta_{ij}$. For shallow circuit, it is reasonable to assume $|\psi_0\rangle \equiv |\psi\rangle$ is the noiseless state. In this case, for any observable O and positive integer m we have [128, 135]

$$\begin{aligned}\frac{\mathrm{Tr}(\rho_{\mathcal{E}}^m O)}{\mathrm{Tr}(\rho_{\mathcal{E}}^m)} &= \frac{p_0^m\langle\psi_0|O|\psi_0\rangle + \sum_i p_i^m\langle\psi_i|O|\psi_i\rangle}{p_0^m + \sum_i p_i^m} \\ &= \langle\psi|O|\psi\rangle\left[1 + O\left(\left(\frac{p_1}{p_0}\right)^m\frac{\langle\psi_1|O|\psi_1\rangle}{\langle\psi|O|\psi\rangle}\right)\right].\end{aligned} \tag{5.29}$$

Thus, computing the expectation value with the mth power of the state suppresses errors exponentially in m, a phenomenon which can also be interpreted as artificially cooling the system [197]. Here we allow for an arbitrary function f acting on the noisy state via its Taylor expansion $f(x) = \sum_{p=1}^{m} c_p x^p$. Including sums of powers up to m (instead of just the highest power m) was shown to improve results of numerical experiments in [135].

LST (Combined approach). LST has the error mitigation capability of code space projection and virtual distillation. When we combine code space projection with virtual distillation, we expect both code distance d of the error correction code and mth power of the density matrix will suppress the error. Especially, when $m = 2$ in $\mathrm{Tr}(\Pi\rho_\epsilon^m \Pi O)$, i.e. one projects the squared density matrix to code space, the order of error suppression is improved from $O(p^d)$ to $O(p^{2d})$. In general, higher order of the power m will lead to stronger error mitigation effect. (See Sec. 5.7 for more details.)

5.3.3.2 Quantum resources

Gate overhead. LST requires additional qubits and gates to encode the logical state, the exact number of which depends on the chosen code. Restricting to stabilizer codes, the logical state preparation only requires implementing Clifford gates, which are presumably easier to implement on NISQ devices compared to universal quantum computation gate set. This encoding overhead is the same as the subspace expansion method. To perform the classical shadow tomography, an element from a unitary ensemble $\mathcal{U}$ is appended to the quantum circuit before measuring all qubits in the computational basis. If $\mathcal{U}$ is chosen to be a global Clifford ensemble, the added circuit depth is $O(N)$ with local random unitary gates [198]. The gate overhead associated with this can be significant. If $\mathcal{U}$ is chosen to be a tensor-product Pauli ensemble, the added circuit depth is $O(1)$ [193]. The gate overhead is more affordable. However, the Pauli measurement will increase the sample complexity exponentially for non-local observables. Facing this dilemma between Clifford v.s. Pauli measurement, a recent work [195] by part of the authors has developed an efficient classical shadow tomography approach for *finite-depth local* Clifford circuits, which can smoothly interpolate between the global Clifford and the local Pauli limit. Using shallow (finite-

depth) Clifford circuits for shadow tomography, the gate overhead can be significantly reduced with only a mild increase of sample complexity. [195] This approach can be combined with our error mitigation technique seamlessly to achieve an optimal balance between gate overhead and sample complexity. For this work, we will use a tensor-product Clifford group $C\ell(2^n)^{\otimes k}$ as shown in Fig. 5.1 (b). The added circuit depth is $O(n)$, where n is the physical qubits in $[[n, 1]]$ code for one logical qubit. It is important to notice that the depth of additional circuit is independent of the number of logical qubit k.

Sample complexity. We now consider number of measurements needed for classical post-processing in LST. If one uses a $[[n, 1]]$ code for each logical qubit and a tensor-product Clifford group $C\ell(2^n)^{\otimes k}$, where k is number of logical qubits, we have the following theorem which dictates the sample complexity of LST.

Theorem 11: Let ρ be a k logical qubits quantum state, where each logical qubit is encoded with a $[[n, 1]]$ code, and Π be the associated projection operator to the code subspace. Then one needs $O(\log(M)4^k/\epsilon\delta^2)$ samples to produce an estimation $\tilde{O}_i$ of $\mathrm{Tr}(\Pi\rho\Pi O_i)$ with $\{i = 1, \ldots, M\}$ of logical Pauli observables O_i such that

$$\Pr\left(\left|\tilde{O}_i - \mathrm{Tr}(\Pi\rho\Pi O_i)\right| \geq \epsilon\right) \leq \delta. \tag{5.30}$$

The result (5.30) applies to both the numerator and denominator of the LST estimate Eq. (5.20) separately. We emphasize that the number of samples does not depend on the total number of physical qubits nk, but only depends on the number of logical qubits k, and scales as $O(4^k)$. Compared to direct implementation of subspace expansion whose sample complexity is $O(2^{nk})$, our method dramatically reduces the sample complexity.

In Ref. [127], the authors also mentioned stochastic sampling of the stabilizer group elements to reduce the sample complexity at the price of losing the error mitigation capacity. Our approach bypasses the need to sample the all elements in the stabilizer group, as we can implement the projection operator directly and efficiently by data post-processing. Our advantage will be even

more apparent if one uses larger n error correction code $[[n, 1]]$, which provides larger code distance and better error mitigation capacity.

5.3.3.3 Classical resources

In this section, we show that there exists efficient algorithm for classical post-processing. Especially, for LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ (no virtual distillation), the classical post-processing can be performed with polynomial classical memory and time.

After sampling, we have the classical shadow Eq. (5.19) consisting of N stabilizer states $U^{\dagger}|b\rangle\langle b|U$, the observable O, and the projector Π. We need to estimate the numerator and denominator of Eq. (5.20), both of which can be written

$$\mathrm{Tr}[f(\rho_{\mathcal{E}})\Gamma] \tag{5.31}$$

where $\Gamma = \Pi O$ for the numerator and $\Gamma = \Pi$ for the denominator. Let m be the highest degree of the Taylor expansion of f. As in [193], we can evaluate the expectation of this term via

$$\mathbb{E}\mathrm{Tr}[\hat{\rho}_{i_1} \cdots \hat{\rho}_{i_m}\Gamma], \tag{5.32}$$

where $\{\hat{\rho}_{i_1} \ldots \hat{\rho}_{i_m}$ are independent samples. A single classical shadow $\hat{\rho}_i$ requires $O(N^2)$ classical memory to store (where $N = nk$ is the total number of qubits), so the argument inside the trace in (5.32) requires $O(N^2 m)$ classical memory [12]. In Sec. 5.6.1, we showed that the evaluation of (5.32) boils down to evaluating the general form

$$\mathrm{Tr}\left(\prod_{j=1}^{l}(a_j I + b_j M_j)\right), \tag{5.33}$$

where M_j are Pauli operators. This can be solved by finding the null space $\mathcal{N}_A$ of a binary matrix representation of Pauli operators $\{M_j\}$. By simple counting argument (see Sec. 5.6.1 for more details), we see the evaluation of Eq. (5.32) has time complexity upper bounded by $O\left(mN^2(mN + N - k + 1) + m|\mathcal{N}_A|\right)$, when $m \geq 2$, where k is number of logical qubits, $N = nk$ is the total number of physical qubits, m is power of density matrix, and $|\mathcal{N}_A|$ is the volume of

binary null space $\mathcal{N}_A$ determined by a set of classical shadows $\{\hat{\rho}_{i_1}, \ldots, \hat{\rho}_{i_m}\}$. When m is large, the null space $\mathcal{N}_A$ can be very large. In practice, the evaluation of (5.32) can be slow given the time complexity is proportional to $O(m|\mathcal{N}_A|)$ for $m \geq 2$.

We also find an improved classical algorithm for $m = 1$ with time complexity $O(N^3)$. When $m = 1$, we would like to evaluate $\mathrm{Tr}(\Pi\rho\Pi O)$, where ρ is proportional to a stabilizer state, which can be represented using stabilizer tableau. The intuition behind the efficient algorithm is that a stabilizer state can be efficiently projected by another stabilizer group projector by updating its stabilizer tableau. We leave the detail of the algorithm to Sec. 5.6.2.

In conclusion, at least for the $m = 1$ (no virtual distillation) case, the post-processing complexity is polynomial $O(N^3)$ in the total qubit number N.

5.4 Numerical results

In Sec. 5.3.3, we discussed the error mitigation capabilities, quantum resources, and classical resources of LST. One should notice that the discussion of sample complexity is mainly focused on estimation of $\mathrm{Tr}(\Pi\rho\Pi^\dagger O)$. While in practice, we would like to estimate the ratio of $\mathrm{Tr}(\Pi\rho\Pi^\dagger O)/\mathrm{Tr}(\Pi\rho\Pi^\dagger)$. The sample complexity of the ratio does not have a closed form in general. This problem is not unique to our approach. It also exists for the subspace expansion [127] and virtual distillation [128].

Nevertheless, in the following, we will use numerical experiments to investigate the sample complexity. We demonstrate the performance of LST through numerical simulation of large systems. In particular, we find LST outperforms the direct implementation of the subspace expansion and the sample complexity scaling is very close to our theoretical prediction in small noise region.

5.4.1 Pseudo-threshold with the $[[5, 1, 3]]$ code

We first consider a simple example with one logical qubit encoded in five physical qubits with the $[[5, 1, 3]]$ stabilizer code. Each physical qubit is subjected to depolarizing noise with error rate p. The same model is considered in the subspace expansion literature [127], which shows

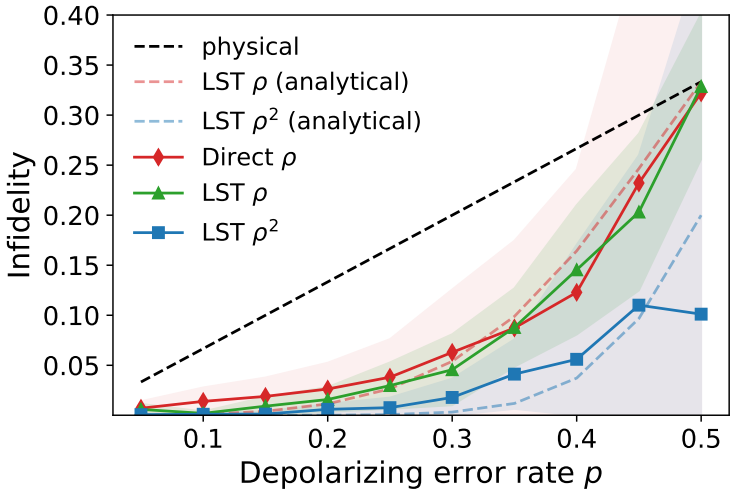


Figure 5.2: LST with the $[[5, 1, 3]]$ code. Here, $|\psi\rangle$ is taken to be the logical $|\bar{0}\rangle$ and we estimate infidelity $1 - F$ with samples. The dashed black line shows the physical infidelity, i.e., the noisy expectation value of single qubit without any encoding. The green and blue dashed line are analytical performance of logical shadow tomography with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ and $f(\rho_{\mathcal{E}}^2) = \rho_{\mathcal{E}}^2$ respectively. The red dots and red shaded area indicates the mean value and standard deviation of error mitigation with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ by direct implementation of subspace expansion with 3000 measurements. The green line and green shaded area indicate the mean value and standard deviation with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ and 3000 measurements by LST. And the performance of LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}^2$ is indicated by blue line and blue shaded area.

pseudo-threshold $p = 0.5$. Here, we want to compare the practical performance of logical shadow tomography and direct measurement by subspace expansion.

We evaluate Eq. (5.20) with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ and $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}^2$. The results are shown in Fig. 5.2. Here, the dashed black line shows the infidelity without any error mitigation (the "physical" curve), and the dashed colored lines show the infidelity using LST. We see that the LST estimates have lower infidelity than the physical curve, showing that errors are indeed being mitigated. LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}^2$ outperforms LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$ showing that the combination of codespace projection and virtually distillation outperforms only projecting into the codespace. This phenomenon agrees with the behavior of the error mitigation capability p^{md} with $m = 1, 2$ argued in 5.3.3.1 (See Sec. 5.7 for proof details). In addition to expected performance, we also care about sample efficiency, since one major contribution of our work is showing the exponential reduction in sample complexity with LST.

To show this practical advantage, we collect 3000 measurements results and use the data to estimate error mitigated value. The colored lines/points shows the mean values of the estimation and the colored regions shows the standard deviation to the mean value. For $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}$, LST is the same as the subspace expansion. If we implement the direct subspace expansion by measuring every Pauli observables, each Pauli observable is measured around 100 times. The red points/line and red shaded region shows the result of direct implementation of subspace expansion. Given each Pauli observable is not measured many times, small error in the denominator can cause large error of the ratio. We see the standard deviation is very huge. The mean estimation and standard deviation of LST is shown as green points/line and green shaded region. In contrast to direct implementation of subspace expansion, LST has a much smaller fluctuation with the same amount of measurements. This shows the practical advantage of our method. In addition, the blue points/line and blue shaded region shows the result of LST with $f(\rho_{\mathcal{E}}) = \rho_{\mathcal{E}}^2$. With the same amount of data, we see it will suppress the noise even more.

5.4.2 Convergence vs. code size

As we have pointed out in Sec. 5.3.3.2, the sample complexity for estimating $\mathrm{Tr}(\Pi\rho\Pi^\dagger O)$ for logical Pauli observables O only scales with number of logical qubits k as $O(4^k)$ but does not scale with the number of encoding qubits n of the $[[n, 1]]$ code for each logical qubit. In practice, we will estimate error mitigated values as a ratio, i.e. $\mathrm{Tr}(\Pi\rho\Pi^\dagger O)/\mathrm{Tr}(\Pi\rho\Pi^\dagger)$. Since there is no closed form for the statistical fluctuation of the above ratio. We will investigate the sample complexity of it via numerical simulation. Interestingly, we find the sample complexity of the ratio agrees well with our theoretical analysis in small noise region.

We now consider $[[n, 1]]$ codes for one logical qubit and vary the number of physical qubits ranging from $n = 10$ to $n = 60$. For quantum noise, each physical qubit is subjected to 1% depolarizing noise in all the following experiments. The LST estimated fidelity (using $f(\rho_\epsilon) = \rho_\epsilon$ for all code sizes) vs. number of samples is shown in Fig. 5.3 (a). Using a relatively small number of samples (at most 10^5), all LST values converge to the noiseless fidelity. Note that the direct

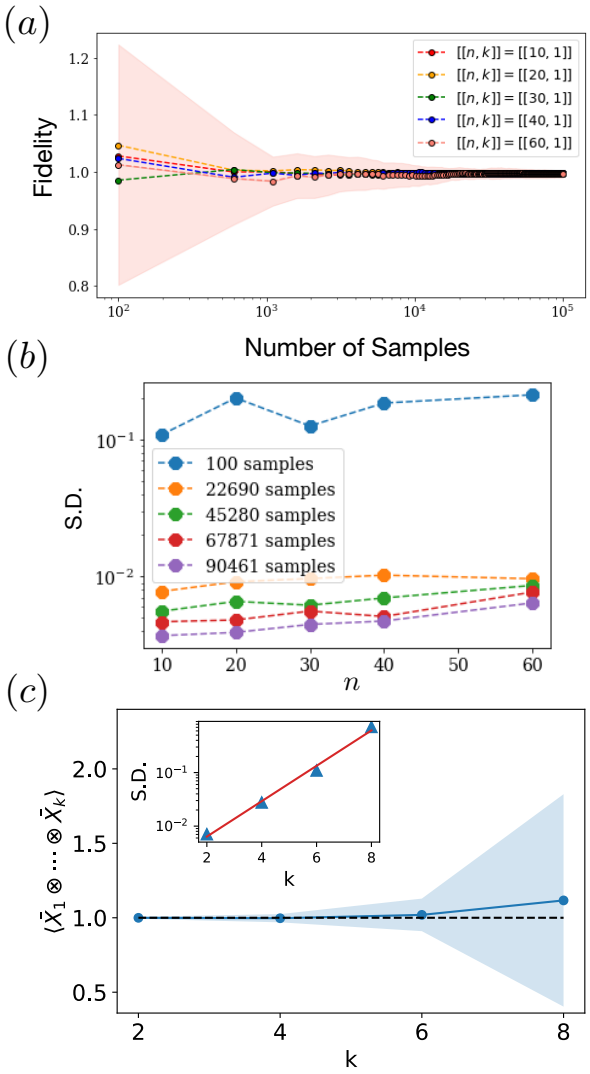


Figure 5.3: Scaling study of LST with $f(\rho_\epsilon) = \rho_\epsilon$. In all figures, each physical qubit is subjected to 1% depolarizing noise. (a) LST estimated fidelity vs. number of samples from 10^2 - 10^5 with various $[[n, 1]]$ code sizes. The noiseless fidelity value of 1.0 is shown with the dashed black line. For all code sizes up to $n = 60$ physical qubits, the LST estimate converges to the true noiseless value. Codes used are the minimum distance constructions from [9].(b) Standard deviation vs. number of physical qubits n. The standard deviation of estimation doesn't scale with number of encoding physical qubits. (c) Mean value and standard deviation scaling vs. number of logical qubits k. Each logical qubit is encoded with $[[5, 1, 3]]$ code, and the state is prepared as logical GHZ state $|\bar{0}\ldots\bar{0}\rangle + |\bar{1}\ldots\bar{1}\rangle$. We see standard deviation scales exponentially with number of logical qubits k as predicted.

implementation of subspace expansion with the full projection Π as used here would require at least 2^{59} samples, a number infeasible to implement in any practical experiment. In addition, we estimate the standard deviation of each predicted point by LST using the bootstrap method, and the result is shown in Fig. 5.3 (b). We see the standard deviation does not show strong dependence of number of encoding qubits n, and it indicates the sample complexity does not increase much if one increase n.

In addition, we also study the sample complexity scaling with number of logical qubits k. Particularly, we use $[[5, 1, 3]]$ code to encode each logical qubit and prepare a multi-logical qubits GHZ state, $(|\bar{0}\ldots\bar{0}\rangle + |\bar{1}\ldots\bar{1}\rangle)/\sqrt{2}$. In Fig. 5.3 (c), the blue dots/line shows the estimated mean value for logical operator $\bar{X}_1 \otimes \cdots \otimes \bar{X}_k$, and the blue shaded area indicates the standard deviation. Especially, in the inset, we see the standard deviation increases exponentially with number of logical qubits k. This result also agrees well with our sample complexity analysis, even though the analysis focuses on estimating $\mathrm{Tr}(\Pi\rho\Pi^\dagger O)$.

Through the large scale numerical simulation, we find LST indeed outperforms previous methods in terms of sample complexity. Interestingly, we also find the sample complexity agrees well with the theoretical analysis in small noise region. This indicates the sample complexity of LST does not scale much with the number of physical encoding qubits n with $[[n, 1]]$ error correction code and only scales with number of logical qubits k.

5.5 Discussion

We have presented a procedure for estimating error-mitigated observables on noisy quantum computers. Our procedure is flexible enough to be performed on virtually any quantum computer: the only additional quantum resources needed are qubits and Clifford gates for encoding the logical state. After sampling from the logical state, a classical computer processes the obtained classical shadow to return the error-mitigated expectation value. In the analysis, we show if the error correction code has code distance d and $f(\rho_\mathcal{E}) = \rho_\mathcal{E}$ for LST, then error will be suppressed to $O(p^d)$, assuming independent depolarizing noise with rate p on each physical qubit. And we also show

higher power of density matrix will further improve the performance. For sample complexity, we show it scales only with number of logical qubits k but not the number of encoding physical qubits n. This result is also supported with large scale numerics. In addition, we provide efficient classical algorithms to post-process the classical shadow data and remark that this post-processing can be easily parallelized for practical efficiency in real-world experiments.

With respect to error mitigation, our procedure provides an experimentally simple procedure to carry out proposed error mitigation techniques [127, 128] at scale. In particular, we have demonstrated subspace expansion with up to $n = 60$ physical qubits encoding a single logical qubit, i.e., a stabilizer group with 2^{59} elements, an experiment which would be practically infeasible with the direct or stochastic sampling schemes of [127, 135]. We have also implemented virtual distillation [128] without expensive swap operations to compute powers of the density matrix. Rather, our procedure uses the same quantum circuit to evaluate the error-mitigated expectation with any function $f(\rho_{\mathcal{E}})$ of the noisy state $\rho_{\mathcal{E}}$; the only difference is in classical post-processing (and number of shadows needed). Note that virtual distillation without subspace expansion can be implemented in our protocol by using a trivial code (i.e., not encoding a logical state). Further, beyond making both of these techniques significantly more practical to implement, our procedure enables them to be composed with one another, and we have shown numerically the composition of both techniques results in further reduction of errors. Additional error mitigation techniques which act on the noisy state, e.g., those in [135], may also be implementable with our framework.

In our analysis, we assumed the Clifford circuit in the classical shadow part is noise-free. If there are noise in the Clifford circuit part, it can be mitigated if the noise is independent of the Clifford gates, as in [199, 200, 201], where similar idea was used for randomized benchmarking [202, 203].

Shadow tomography since proposed in [193] has found a number of applications in quantum information processing, including the recently proposed process tomography [204], and avoiding barren plateau in variational quantum circuits [205]. This work constitutes an application in the error mitigation realm. We are optimistic our procedure will be effective on current and near-term quantum computers for a variety of experiments on relatively large systems.

5.6 Stabilizer algorithms

5.6.1 Evaluating the trace in Eq. (5.21)

In this Section, we explain an efficient approach to evaluate the trace in Eq. (5.21). For random Clifford ensemble, the reconstruction map reads $\hat{\rho} = \mathcal{M}^{-1}(\hat{\sigma}) = (2^n + 1)\hat{\sigma} - I$, such that

$$\mathbb{E}_{\hat{\rho}} \mathrm{Tr}\Big(\Pi\Big(\prod_{s=1}^{m} \hat{\rho}_s\Big)\Pi^\dagger O\Big) = \sum_{q=0}^{m} \binom{m}{q} (2^n+1)^q (-1)^{m-q} \mathbb{E}_{\hat{\sigma}} \mathrm{Tr}\Big(\Pi\Big(\prod_{s=1}^{q} \hat{\sigma}_s\Big)\Pi^\dagger O\Big). \tag{5.34}$$

Notice that the projection operator $\Pi = \prod_{j=1}^{n-k}(I + S_j)/2$ and the snapshot state $\hat{\sigma} = U^\dagger|b\rangle\langle b|U = \prod_{i=1}^{n}(I + b_i U^\dagger Z_i U)/2$ both take the from of stabilizer states. So the problem boils down to evaluating the trace of the following general form

$$\mathrm{Tr}\Big(\prod_{j=1}^{l}(a_j I + b_j M_j)\Big), \tag{5.35}$$

where M_j are Pauli operators and a_j, b_j are real coefficients. As we expand the product, the only terms that survive the trace are those terms with the Pauli operators multiplied to the identity operator. To find these combination of Pauli operators, we can first encode every Pauli operator M_j as a binary vector following

$$M_j = \mathrm{i}^{\sum_{i=1}^{n} \xi_{ij}\zeta_{ij}} \prod_{i=1}^{n} X_i^{\xi_{ij}} \prod_{i=1}^{n} Z_i^{\zeta_{ij}} \rightarrow \begin{pmatrix} \vdots \\ \xi_{ij} \\ \zeta_{ij} \\ \vdots \end{pmatrix}, \tag{5.36}$$

where $\xi_{ij}, \zeta_{ij} \in \{0, 1\}$ are binary variables. Arranging all the binary vector representations of M_j as column vectors, together they form a $2n \times l$ matrix, denoted as A. Each combination of Pauli operators M_j that multiply to identity corresponds to a binary null vector solution x of the binary matrix A, as $Ax = 0$ (modulo 2). The null vectors form the null space of A, denoted as $\mathcal{N}_A$. The null space of binary matrix A can be found using Gaussian elimination method, and its time complexity is $O(nl \times \min(2n, l))$. For $x \in \mathcal{N}_A$,

$$\prod_{j=1}^{l}(M_j)^{x_j} = z(x)I, \tag{5.37}$$

which defines the phase factor $z(x)$ given x. Then the trace in Eq. (5.35) is given by

$$\mathrm{Tr}\left(\prod_{j=1}^{l}(a_j I + b_j M_j)\right) = 2^n \sum_{x\in\mathcal{N}_A} z(x) \prod_{j=1}^{l} a_j^{1-x_j} b_j^{x_j}. \tag{5.38}$$

Therefore the time complexity of evaluating the general trace Eq. (5.38) is $O(nl\times\min(2n,l)+|\mathcal{N}_A|)$, where $|\mathcal{N}_A|$ is the volume of the null space $\mathcal{N}_A$ that is determined by the set of Pauli operators $\{M_i\}$. Applying this result for Eq. (5.34), we get the time complexity for evaluating $\mathrm{Tr}\left(\Pi\left(\prod_{s=1}^{m}\hat{\rho}_s\right)\Pi^\dagger O\right)$ is upper bounded by $O(mnl\times\min(2n,l)+m|\mathcal{N}_A|)$, with $l = mn+n-k+1$. For large m, the volume of null space $|\mathcal{N}_A|$ can be troublesome, and scale exponentially with m. But luckily for $m = 1$, there exists more efficient polynomial time algorithm, which is illustrated in Sec. 5.6.2.

5.6.2 Efficient projection of a stabilizer state

As shown in Eq. (5.36), any Pauli string operator can be represented as a one-hot binary vector x and z, with $x_i, z_i = 0, 1$ for $i = 1,\ldots,N$, where N is the total number of qubits,

$$\sigma_{(x,z)} = i^{x\cdot z}\prod_{i=1}^{N} X_i^{x_i} Z_i^{y_i}, \tag{5.39}$$

where X_i, Z_i are Pauli operators, and x_i, z_i are binary values. The multiplication of two Pauli operators can be represented as

$$\sigma_{(x,z)}\sigma_{(x',z')} = i^{p(x,z;x',z')}\sigma_{(x+x',z+z')\%2}, \tag{5.40}$$

where the phase factor is

$$p(x,z;x',z') = \sum_{i=1}^{N}\left(z_i x_i' - x_i z_i' + 2(z_i+z_i')\left\lfloor\frac{x_i+x_i'}{2}\right\rfloor + 2(x_i+x_i')\left\lfloor\frac{z_i+z_i'}{2}\right\rfloor\right) \mod 4. \tag{5.41}$$

Any two Pauli strings either commute or anti-commute,

$$\sigma_{(x,z)}\sigma_{(x',z')} = (-)^{c(x,z;x',z')}\sigma_{(x',z')}\sigma_{(x,z)}, \tag{5.42}$$

where the anticommutation indicator c has a simpler form

$$c(x,z;x',z') = \frac{p(x,z;x',z') - p(x',z';x,z)}{2} = \sum_{i=1}^{N}\left(z_i x_i' - x_i z_i'\right) \mod 2. \tag{5.43}$$

Therefore, the complexity of calculating anticommutation indicator is $O(N)$. The binary vectors x and z can be interweaved into a $2N$-component vector $g = (x_0, z_0, x_1, z_1, \cdots)$, which forms the binary representation of a Pauli operator σ_g.

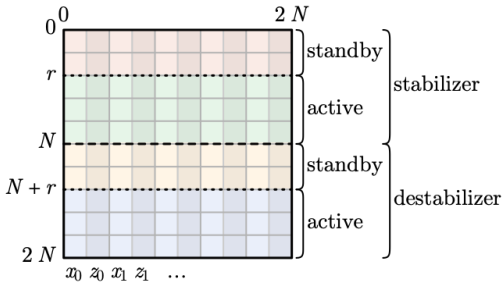


Figure 5.4: Data structure of a stabilizer state. Each Pauli string is represented as a binary vector. First N rows store the stabilizers of the state, and second N rows store the destabilizers of the state.

In Fig. 5.4, each row is a binary representation of a Pauli string. For a Hilbert space with dimension $\mathbb{C}^{2^N}$, we can find at most N stabilizers $\{S_i, i = 1 \ldots N\}$ and N destabilizers $\{D_i, i = 1 \ldots N\}$, with $[S_i, S_j] = 0$, $[D_i, D_j] = 0$ and $\{S_i, D_j\} = \delta_{i,j}$. Fig. 5.4 is called the stabilizer tableau of stabilizer state. We use r in Fig. 5.4 indicates the $\log -2$ rank of the density matrix. For pure stabilizer state, $r = 0$. And if $r > 0$, then stabilizer tableau represents a mixed state and only partial Hilbert space is stabilized.

A stabilizer projector $\Pi = \prod_{k=1}^{l} \frac{I_k}{2}$ can also be represented as a l-row tableau (only stabilizer). Now we are going to present an efficient algorithm to calculate $\mathrm{Tr}(\Pi\rho\Pi)$, and update the stabilizer tableau of a full rank stabilizer state ρ according to projection Π.

Outline of the algorithm: First, we set trace = 1. Then we scan over every observable G_k in the generator of the operator. For each G_k, we continue to scan over all operators in the stabilizer tableau. If the observable G_k anticommute with:

1. At least one active stabilizer (the first of them being S_p) $\rightarrow$ G_k is an *error* operator that take the state out of the code subspace $\rightarrow$ stabilizer tableau need to be updated according to $I \pm G_k$. And trace will be multiplied by 1/2.

2. Otherwise, G_k is in the stabilizer group generated by $\{S_k\}$. If the phase factor is compatible,

then stabilizer state is eigenstate of $(I \pm G_k)/2$ with eigenvalue 1, i.e. $(I \pm G_k)|\psi\rangle/2 = |\psi\rangle$; if the phase factor is incompatible, it means $(I \pm G_k)|\psi\rangle/2 = 0$.

We see this algorithm has a double loop of $O(N)$ items, and each anticommutation check takes $O(N)$ time. Therefore, the time complexity of this algorithm is $O(N^3)$, where N is the total number of qubits in the system.

5.7 Error mitigation capability

In Eq. (5.24), we show the density matrix subjected to depolarizing noise is naturally the spectral decomposition form. Here, we provide detailed proof of it. Let $|\psi_0\rangle$ be the ideal quantum state encoded with $[[n, k, d]]$ error correction code. And let $\{|\bar{i}\rangle, i = 1, \ldots, 2^k\}$ be the orthonormal basis for the logical space. In general, $|\psi_0\rangle = \sum_{i=1}^{2^k} c_i|\bar{i}\rangle$. We assume the simple depolarizing error for each physical qubit. Then

$$\rho_\epsilon = (1-p)^N|\psi_0\rangle\langle\psi_0| + p\rho_1 + p^2\rho_2 + \cdots, \tag{5.44}$$

where ρ_i is the density matrix with i local error happened, and N is the system size. For example,

$$\begin{aligned}\rho_1 =&(XII\cdots I)|\psi_0\rangle\langle\psi_0|(XII\cdots I) + (YII\cdots I)|\psi_0\rangle\langle\psi_0|(YII\cdots I)\\ &+ (ZII\cdots I)|\psi_0\rangle\langle\psi_0|(ZII\cdots I) + (IXI\cdots I)|\psi_0\rangle\langle\psi_0|(IXI\cdots I)\\ &+\cdots+ (II\cdots IZ)|\psi_0\rangle\langle\psi_0|(II\cdots IZ),\end{aligned} \tag{5.45}$$

and

$$\begin{aligned}\rho_2 =&(XXI\cdots I)|\psi_0\rangle\langle\psi_0|(XXI\cdots I) + (XYI\cdots I)|\psi_0\rangle\langle\psi_0|(XYI\cdots I)\\ &+\cdots+ (I\cdots ZZ)|\psi_0\rangle\langle\psi_0|(I\cdots ZZ).\end{aligned} \tag{5.46}$$

We define the support of a Pauli string as number of Pauli operators that is not the identity operator. Let P_l be a Pauli string operator with non-trivial support l. For example, the support of Pauli string $XIZYI$ is three. Then any term in ρ_l can be written as $P_l|\psi_0\rangle\langle\psi_0|P_l$ with some P_l. It is easy to check $\Pi P_l|\psi_0\rangle\langle\psi_0|P_l\Pi = 0$ for any $l < d$. By definition of the code distance d, any P_l with $l < d$ is not in the stabilizer group. Therefore, it must anti-commute with some stabilizer generator S,

such that $\{S, P_l\} = 0$. We write $\Pi = \Pi^{'}(I + S)$, where $\Pi^{'}$ includes all other stabilizer generators. Then

$$\Pi P_l|\psi_0\rangle = \Pi^{'}(I + S)P_l|\psi_0\rangle = \Pi^{'}P_l(I - S)|\psi_0\rangle = \Pi^{'}P_l(|\psi_0\rangle - |\psi_0\rangle) = 0, \tag{5.47}$$

and

$$\Pi P_l|\psi_0\rangle\langle\psi_0|P_l\Pi = 0 \ \ (l < d). \tag{5.48}$$

Therefore, we conclude $\Pi\rho_l\Pi = 0$ for any $l < d$, and

$$\begin{aligned} &\frac{\mathrm{Tr}(\Pi\rho_{\mathcal{E}}\Pi O)}{\mathrm{Tr}(\Pi\rho_{\mathcal{E}}\Pi)} \\ &= \langle\psi_0|O|\psi_0\rangle \left[1 + O\left(p^d \frac{\mathrm{Tr}(\rho_d O)}{\langle\psi_0|O|\psi_0\rangle}\right)\right]. \end{aligned} \tag{5.49}$$

Now we want to prove the leading order correction of $\Pi\rho_\epsilon^2\Pi$ is of order $O(p^{2d})$ with contradiction. Suppose the leading order correction is of order $O(p^s)$ with $s < 2d$, then there exist P_l and P_r with $l + r = s < 2d$ such that

$$\Pi(P_l|\psi_0\rangle\langle\psi_0|P_l)(P_r|\psi_0\rangle\langle\psi_0|P_r)\Pi = \langle\psi_0|P_lP_r|\psi_0\rangle(\Pi P_l|\psi_0\rangle)(\langle\psi_0|P_r\Pi) \neq 0. \tag{5.50}$$

This requires $\Pi P_l|\psi_0\rangle \neq 0$ and $\Pi P_r|\psi_0\rangle \neq 0$. From Eq. (5.48), we know this requires $l \geq d$ and $r \geq d$, and it contradicts with $l + r < 2d$. Therefore, we conclude the leading order correction of $\mathrm{Tr}(\Pi\rho_\epsilon^2\Pi O)$ is of order $O(p^{2d})$.

For higher power of $\Pi\rho_\epsilon^m\Pi$, one may expect the leading order correction will be $O(p^{md})$. Depending on the particular logical state $|\psi_0\rangle$ and error correction code, the performance may or may not reach $O(p^{md})$. This is because there can exist shortcuts that make the leading order correction larger than $O(p^{md})$. In practice, we do witness the performance will be improved with larger m.

5.8 Mean and variance of a ratio of two random variables

Consider random variables P and Q and let $G = g(P, Q) = P/Q$. In general, there is no closed form expression for $\mathbb{E}[G(P, Q)]$, and $\mathrm{Var}[G(P, Q)]$. Here, we find approximations for the mean and variance using Taylor expansions of $g(P, Q)$.

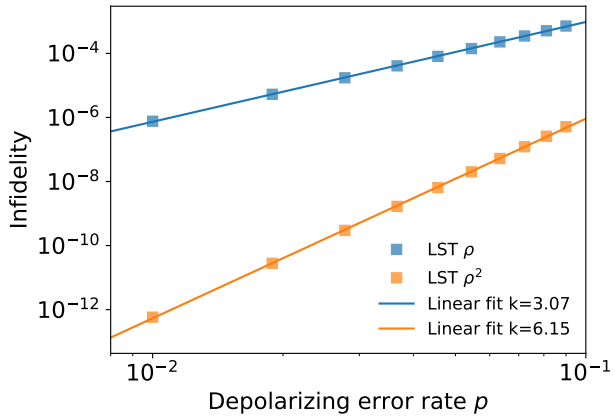


Figure 5.5: Infidelity in small error rate region. Theoretically we have shown the leading order correction to infidelity will be $O(p^{md})$ with $m = 1, 2$. Here, we use $[[5, 1, 3]]$ code with LST as a demonstration. We prepare random logical states and calculate the infidelity. We see the numerical results give linear order correction $O(p^{3.07})$ and $O(p^{6,15})$, which is very close to theoretical prediction $O(p^3)$ and $O(p^6)$.

The approximation for the mean value is

$$\begin{aligned}\mathbb{E}[g(P,Q)] &= \mathbb{E}[g(\mu_P,\mu_Q) + g'_P(\mu_P,\mu_Q)(P-\mu_P) + g'_Q(\mu_P,\mu_Q)(Q-\mu_Q) + R] \\ &\approx \mathbb{E}[g(\mu_P,\mu_Q)] + g'_P(\mu_P,\mu_Q)\mathbb{E}[(P-\mu_P)] + g'_Q(\mu_P,\mu_Q)\mathbb{E}[(Q-\mu_Q)] \\ &= g(\mu_P,\mu_Q),\end{aligned} \tag{5.51}$$

where R is the higher order reminders of the Taylor expansion. For keeping the Taylor expansion to the first order, we ignore higher order remainders.

For the variance, we have

$$\begin{aligned}\mathrm{Var}[g(P,Q)] &= \mathbb{E}\left\{[g(P,Q) - \mathbb{E}(g(P,Q))]^2\right\} \\ &\approx \mathbb{E}\left\{[g(P,Q) - g(\mu_P,\mu_Q)]^2\right\} \\ &\approx \mathbb{E}\left\{[g'_P(\mu_P,\mu_Q)(P-\mu_P) + g'_Q(\mu_P,\mu_Q)(Q-\mu_Q)]^2\right\} \\ &= g'^2_P(\mu_P,\mu_Q)\,\mathrm{Var}(P) + g'^2_Q(\mu_P,\mu_Q)\,\mathrm{Var}(Q) + 2g'_P(\mu_P,\mu_Q)g'_Q(\mu_P,\mu_Q)\,\mathrm{Cov}(P,Q).\end{aligned} \tag{5.52}$$

For our case, $g(P,Q) = P/Q$, therefore $g'_P(\mu_P,\mu_Q) = 1/\mu_Q$, $g'_Q(\mu_P,\mu_Q) = -\mu_P/\mu_Q^2$, and

$$\mathrm{Var}(P/Q) \approx (\frac{\mu_P}{\mu_Q})^2 \left[\frac{\mathrm{Var}(P)}{\mu_P^2} + \frac{\mathrm{Var}(Q)}{\mu_Q^2} - 2\frac{\mathrm{Cov}(P,Q)}{\mu_P\mu_Q}\right] \tag{5.53}$$

5.9 Proof of sample complexities

Suppose we want to predict a linear property of the underlying quantum state,

$$o = \mathrm{Tr}(\rho O). \tag{5.54}$$

We simply replace the unknown quantum state ρ with the classical shadows $\hat{\rho} = \mathcal{M}^{-1}(\hat{\sigma})$. This yields a stochastic number $\hat{o} = \mathrm{Tr}(\hat{\rho}O)$, and it will converge to correct answer with sufficient amount of classical shadows,

$$\mathbb{E}\hat{o} = \mathrm{Tr}(\rho O). \tag{5.55}$$

In practice, the expectation $\mathbb{E}\hat{o}_i$ is replaced by a sample mean estimator, $o_{\mathrm{avg}} = \frac{1}{M}\sum_{i=1}^{M}\hat{o}_i = \frac{1}{M}\sum_{i=1}^{M}\mathrm{Tr}(O\hat{\rho}_i)$. Based on Chebyshev's inequality, the probability of the estimation o_{avg} to deviate from its expectation value o is bounded by its variance $\mathrm{Var}(o_{\mathrm{avg}})$ as $\Pr(|o_{\mathrm{avg}} - o| \geq \delta) \leq \mathrm{Var}(o_{\mathrm{avg}})/\delta^2$. To control the deviation within a desired statistial accuracy ϵ, we require $\mathrm{Var}(o_{\mathrm{avg}})/\delta^2 = \mathrm{Var}(\hat{o})/(M\delta^2) \leq \epsilon$, where M is the number of classical shadows. In other words, the number of experiments needed to achieve the statistical error ϵ is given by

$$M \geq \mathrm{Var}(\hat{o})/\epsilon\delta^2. \tag{5.56}$$

Therefore, the sample complexity is directly related to the variance of single-shot estimation $\mathrm{Var}(\hat{o})$. We can further bound the variance by

$$\begin{aligned}
\mathrm{Var}(\hat{o}) &= \mathbb{E}[\hat{o}^2] - \mathbb{E}[\hat{o}]^2 \leq \mathbb{E}[\hat{o}^2] \\
&= \mathbb{E}_{U\sim\mathcal{U}} \sum_{b\in\{0,1\}^n} \langle b|U\sigma U^\dagger|b\rangle\langle b|U\mathcal{M}^{-1}(O)U^\dagger|b\rangle^2 \\
&\leq ||O||^2_{\mathrm{shadow}},
\end{aligned} \tag{5.57}$$

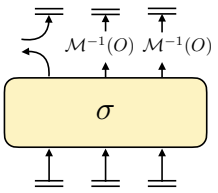
where the shadow norm of an observable is defined as

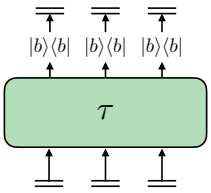
$$\begin{aligned}
||O||_{\mathrm{shadow}} &= \max_{\sigma:\mathrm{state}} \left(\mathbb{E}_{U\sim\mathcal{U}} \sum_{b\in\{0,1\}^n} \langle b|U\sigma U^\dagger|b\rangle\langle b|U\mathcal{M}^{-1}(O)U^\dagger|b\rangle^2 \right)^{1/2} \\
&= \max_{\sigma:\mathrm{state}} \left(\mathbb{E}_{U\sim\mathcal{U}} \sum_{b\in\{0,1\}^n} \mathrm{Tr}(\sigma U^\dagger|b\rangle\langle b|U\langle b|U\mathcal{M}^{-1}(O)U^\dagger|b\rangle^2) \right)^{1/2} \\
&= \max_{\sigma:\mathrm{state}} \left(\mathrm{Tr}\sigma V_{\mathcal{U}}[O] \right)^{1/2},
\end{aligned} \tag{5.58}$$

where we define a new operator $V_{\mathcal{U}}[O] = \mathbb{E}_{U\sim\mathcal{U}} \sum_{b\in\{0,1\}^n} U^\dagger|b\rangle\langle b|U\langle b|U\mathcal{M}^{-1}(O)U^\dagger|b\rangle^2$ that depends both on the unitary ensemble $\mathcal{U}$ and observable O. If the unitary ensemble $\mathcal{U}$ satisfies unitary 3-design, it can be simplified as

$$\begin{aligned} V_{\mathcal{U}}[O] &= \mathbb{E}_{U\sim\mathcal{U}} \sum_{b\in\{0,1\}^n} U^\dagger|b\rangle\langle b|U\langle b|U\mathcal{M}^{-1}(O)U^\dagger|b\rangle^2 \\ &= \sum_{b\in\{0,1\}^n} \sum_{\sigma,\tau\in S_3} \mathrm{Wg}[\sigma\tau^{-1}g_0]A[\sigma]B[\tau], \end{aligned} \tag{5.59}$$

where σ, τ are permutations from permutation group S_3, $\mathrm{Wg}[g]$ is the Weingarten function of the permutation group element g, $g_0 = (2,3)$ is a fixed permutation to match the tensor network connection, and $A[\sigma]$, $B[\tau]$ are defined as:

$A[\sigma] =$ 

, $B[\tau] =$ 

(5.60)

In the following, we will mainly focus on the analysis of $V_{\mathcal{U}}[O]$ operator and $||O||^2_{\text{shadow}}$. In the main text, we focus on the scheme of encoding each logical qubit with $[n,1]$ stabilizer code, and doing quantum computation with total physical qubits $N = n \times l$, where l is the number of logical qubits. For the classical shadow tomography part, we will use random unitaries sampled from $C\ell(2^n)^{\otimes l}$. One reason of choosing this factorized random unitary group is global clifford group $C\ell(2^{nl})$ is harder to implement in experiments. And the difficulty of implementing this factorized scheme does not depend on number of logical qubits. In practice, it is possible to encode each logical qubit with a small error correction code, such as $[5,1]$ code, and implement random circuits from $C\ell(2^n)$. If the random unitary ensemble is $C\ell(2^n)^{\otimes l}$, then it is easy to show the reconstruction map is

$$\mathcal{M}^{-1}[\sigma] = \otimes_{i=1}^{l} \left[(2^n+1)\sigma_i - \mathrm{Tr}(A_i)I\right], \tag{5.61}$$

where σ_i is the reduced classical shadow on part i. The logical Pauli observables will be factorized on each logical sectors, i.e. $O = O_1 \otimes O_2 \otimes \cdots \otimes O_l$. And since the random untaries are sampled

from ensemble $C\ell(2^n)^{\otimes l}$, they also have the tensor product structure, i.e. $U = U_1 \otimes U_2 \otimes \cdots \otimes U_l$. By combining those two properties, we can show

$$V_{\mathcal{U}}[O] = \otimes_{i=1}^{l} V_{\mathcal{U}_i}[O_i]. \tag{5.62}$$

Therefore, we only need to focus on the property of $V_{\mathcal{U}_i}[O_i]$ for each logical sector. In the following, we will use $d = 2^n$ for the Hilbert space dimension for one logical sector.

The calculation for $V_{\mathcal{U}_i}[P_i I_i]$: For projection operator P_i, $\mathcal{M}^{-1}(P_i I_i) = (d+1)P_i - 2I_i$. And Eq. (5.59) can be evaluated

$$V_{C\ell(d)}[P_i I_i] = \frac{2d-2}{d+2}(P_i + I_i). \tag{5.63}$$

The calculation for $V_{\mathcal{U}_i}[P_i O_i]$: For non-trivial Pauli string O_i, $\mathcal{M}^{-1}(P_i O_i) = (d+1)P_i O_i$. And Eq. (5.59) can be evaluated

$$V_{C\ell(d)}[P_i O_i] = \frac{2d+2}{d+2}(P_i + I_i). \tag{5.64}$$

As we can see, $V_{C\ell(d)}[P_i I_i] \lesssim V_{C\ell(d)}[P_i O_i] = \frac{2d+2}{d+2}(P_i + I_i)$. This result indicates the sample complexity for predicting logical Pauli operators $O = \otimes_{i=1}^{l} O_i$ after projection by $P = \otimes_{i=1}^{l} P_i$ does not depend on the locality of the logical Pauli operators,

$$||PO||^2_{\text{shadow}} \lessapprox \max_{\sigma:\text{state}} \text{Tr}\left(\sigma \left(\frac{2d+2}{d+2}\right)^l \otimes_{i=1}^{l} (P_i + I_i)\right) \lesssim 4^l. \tag{5.65}$$

This result is different from the sample complexity from local Clifford group or tensored Clifford group $C\ell(d)^{\otimes l}$, where the sample complexity will depends on the locality of Pauli string O. This difference is mainly introduced by the logical subspace projection P. Even the Pauli string O is trivial in some region, the subspace projection P_i will still introduce fluctuation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] IBM Quantum, (2022), https://quantum-computing.ibm.com/.

[2] R. S. Smith, M. J. Curtis, and W. J. Zeng, *A practical quantum instruction set architecture,* arXiv:1608.03355 (2016).

[3] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, *Open Quantum Assembly Language,* arXiv:1707.03429 (2017).

[4] P. W. Shor and S. P. Jordan, *Estimating jones polynomials is a complete problem for one clean qubit,* Quantum Information & Computation **8**, 681 (2008).

[5] F. Vatan and C. Williams, *Optimal quantum circuits for general two-qubit gates,* Physical Review A **69**, 032315 (2004).

[6] P. B. M. Sousa and R. V. Ramos, *Universal quantum circuit for n-qubit quantum gate: A programmable quantum gate,* Quantum Information and Computation **7**, 228 (2007).

[7] E. Knill and R. Laflamme, *Power of one bit of quantum information,* Physical Review Letters **81**, 5672 (1998).

[8] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, *Validating quantum computers using randomized model circuits,* Physical Review A (2018), 10.1103/PhysRevA.100.032328.

[9] M. Grassl, Bounds on the minimum distance of linear codes and quantum codes, (2021).

[10] P. W. Shor, in *Proceedings 35th annual symposium on foundations of computer science* (Ieee, 1994) pp. 124–134.

[11] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[12] S. Aaronson and D. Gottesman, *Improved simulation of stabilizer circuits,* Physical Review A **70** (2004), 10.1103/PhysRevA.70.052328, arXiv: quant-ph/0406196.

[13] D. Gottesman, *The heisenberg representation of quantum computers,* arXiv:quant-ph/9807006 (1998), arXiv: quant-ph/9807006.

[14] J. Preskill, *Quantum computing in the NISQ era and beyond,* Quantum **2**, 79 (2018).

[15] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a*

quantum computer, SIAM review **41**, 303 (1999).

[16] S. Lloyd, M. Mohseni, and P. Rebentrost, *Quantum principal component analysis,* Nature Physics **10**, 631 (2014), 1307.0401 .

[17] A. W. Harrow, A. Hassidim, and S. Lloyd, *Quantum algorithm for linear systems of equations,* Physical review letters **103**, 150502 (2009).

[18] P. Rebentrost, A. Steffens, I. Marvian, and S. Lloyd, *Quantum singular-value decomposition of nonsparse low-rank matrices,* Physical review A **97**, 012327 (2018).

[19] S. K. Leyton and T. J. Osborne, *A quantum algorithm to solve nonlinear differential equations,* arXiv:0812.4423 (2008).

[20] D. W. Berry, *High-order quantum algorithm for solving linear differential equations,* Journal of Physics A: Mathematical and Theoretical **47**, 105301 (2014).

[21] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm,* arXiv:1411.4028 (2014).

[22] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, *A variational eigenvalue solver on a photonic quantum processor,* Nature communications **5**, 4213 (2014).

[23] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,* Nature **549**, 242 (2017).

[24] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, *Simulating hamiltonian dynamics with a truncated taylor series,* Physical review letters **114**, 090502 (2015).

[25] J. Preskill, *Quantum computing and the entanglement frontier,* arXiv:1203.5813 (2012).

[26] A. W. Harrow and A. Montanaro, *Quantum computational supremacy,* Nature **549**, 203 (2017).

[27] S. Bravyi, G. Smith, and J. A. Smolin, *Trading classical and quantum computational resources,* Physical Review X **6**, 021043 (2016).

[28] O. Higgott, D. Wang, and S. Brierley, *Variational Quantum Computation of Excited States,* arXiv:1805.08138 (2018), arXiv:1805.08138 [quant-ph] .

[29] S. Endo, T. Jones, S. McArdle, X. Yuan, and S. Benjamin, *Variational quantum algorithms for discovering Hamiltonian spectra,* arXiv:1806.05707 (2018), arXiv:1806.05707 [quant-ph] .

[30] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, *QVECTOR: an algorithm for device-tailored quantum error correction,* arXiv:1711.02249 (2017).

[31] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum autoencoders for efficient compression of quantum data,* Quantum Science and Technology **2**, 045001 (2017).

[32] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, and M. H. Amin, *Quantum variational autoencoder,* Quantum Science and Technology **4**, 014001 (2018).

[33] Y. Li and S. C. Benjamin, *Efficient variational quantum simulator incorporating active error minimization,* Physical Review X **7**, 021050 (2017).

[34] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, *et al.*, *Self-verifying variational quantum simulation of the lattice schwinger model,* arXiv:1810.03421 (2018).

[35] H. Li and F. D. M. Haldane, *Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum hall effect states,* Physical Review Letters **101**, 010504 (2008).

[36] V. Giovannetti, S. Lloyd, and L. Maccone, *Quantum random access memory,* Physical review letters **100**, 160501 (2008).

[37] K. Pearson, *On lines and planes of closest fit to systems of points in space,* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559 (1901).

[38] L. N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, 1997).

[39] This can be seen by noting that computing eigenvalues is equivalent to computing roots of a polynomial equation (namely the characteristic polynomial of the matrix) and that no closed-form solution exists for the roots of general polynomials of degree greater than or equal to five [38].

[40] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Barren plateaus in quantum neural network training landscapes,* Nature Communications **9**, 4812 (2018).

[41] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti, *An initialization strategy for addressing barren plateaus in parametrized quantum circuits,* arXiv:1903.05076 (2019), 1903.05076 [quant-ph] .

[42] T. Baumgratz, M. Cramer, and M. B. Plenio, *Quantifying coherence,* Physical review letters **113**, 140401 (2014).

[43] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, *Quantum fingerprinting,* Physical Review Letters **87**, 167902 (2001).

[44] D. Gottesman and I. Chuang, *Quantum digital signatures,* quant-ph/0105032 (2001).

[45] VQSD source code, `https://github.com/rmlarose/vqsd` (2019).

[46] R. S. Smith, M. J. Curtis, and W. J. Zeng, *A Practical Quantum Instruction Set Architecture,* arXiv:1608.03355 (2016).

[47] M. B. Hastings, *An area law for one-dimensional quantum systems,* Journal of Statistical Mechanics: Theory and Experiment **2007**, 08024 (2007), arXiv:0705.2024 [quant-ph] .

[48] B. Bauer and C. Nayak, *Area laws in a many-body localized state and its implications for topological order,* Journal of Statistical Mechanics: Theory and Experiment **2013**, 09005 (2013), arXiv:1306.5753 [cond-mat.dis-nn] .

[49] T. Grover, *Certain General Constraints on the Many-Body Localization Transition,* arXiv e-prints (2014), arXiv:1405.1471 [cond-mat.dis-nn] .

[50] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, *Learning the quantum algorithm for state overlap,* New Journal of Physics **20**, 113022 (2018).

[51] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles, *Quantum assisted quantum compiling,* arXiv:1807.00800 (2018).

[52] T. Jones and S. C. Benjamin, *Quantum compilation and circuit optimisation via energy dissipation,* arXiv:1811.03147 (2018).

[53] E. Tang, *Quantum-inspired classical algorithms for principal component analysis and supervised clustering,* arXiv:1811.00414 (2018).

[54] H. Li and F. D. M. Haldane, *Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum hall effect states,* Phys. Rev. Lett. **101**, 010504 (2008).

[55] J. C. Garcia-Escartin and P. Chamorro-Posada, *Swap test and Hong-Ou-Mandel effect are equivalent,* Physical Review A **87**, 052330 (2013).

[56] G. Smith, J. A. Smolin, X. Yuan, Q. Zhao, D. Girolami, and X. Ma, *Quantifying coherence and entanglement via simple measurements,* arXiv:1707.09928 (2017).

[57] L. M. Rios and N. V. Sahinidis, *Derivative-free optimization: a review of algorithms and comparison of software implementations,* Journal of Global Optimization **56**, 1247 (2013).

[58] G. G. Guerreschi and M. Smelyanskiy, *Practical optimization for hybrid quantum-classical algorithms,* arXiv:1701.01450 (2017).

[59] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, *The theory of variational hybrid quantum-classical algorithms,* New Journal of Physics **18**, 023023 (2016).

[60] M. J. D. Powell, in *Numerical Analysis*, Lecture Notes in Mathematics, edited by G. A. Watson (Springer Berlin Heidelberg, 1978) pp. 144–157.

[61] Scipy optimization and root finding, (2018).

[62] M. J. D. Powell, *Direct search algorithms for optimization calculations,* Acta Numerica **7**, 287 (1998).

[63] M. J. D. Powell (2009).

[64] F. Gao and L. Han, *Implementing the nelder-mead simplex algorithm with adaptive parameters,* Computational Optimization and Applications **51**, 259 (2012).

[65] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer Series in Operations Research and Financial Engineering (Springer-Verlag, 2006).

[66] C. Cartis, J. Fiala, B. Marteau, and L. Roberts, *Improving the Flexibility and Robustness of Model-Based Derivative-Free Optimization Solvers,* arXiv:1804.00154 (2018).

[67] D. Venturelli, M. Do, E. Rieffel, and J. Frank, *Compiling quantum circuits to realistic hardware architectures using temporal planners,* Quantum Science and Technology **3**, 025004 (2018).

[68] K. E. Booth, M. Do, J. C. Beck, E. Rieffel, D. Venturelli, and J. Frank, *Comparing and integrating constraint programming and temporal planning for quantum circuit compilation,* arXiv:1803.06775 (2018).

[69] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, *Quantum circuit simplification and level compaction,* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**, 436 (2008).

[70] A. G. Fowler, *Constructing arbitrary Steane code single logical qubit fault-tolerant gates,* Quantum Information and Computation **11**, 867 (2011).

[71] J. Booth Jr, *Quantum compiler optimizations,* arXiv:1206.3348 (2012).

[72] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, *Automated optimization of large quantum circuits with continuous parameters,* npj Quantum Information **4**, 23 (2018).

[73] F. T. Chong, D. Franklin, and M. Martonosi, *Programming languages and compiler design for realistic quantum hardware,* Nature **549**, 180 (2017).

[74] L. E. Heyfron and E. T. Campbell, *An efficient quantum compiler that reduces T count,* Quantum Science and Technology **4**, 015004 (2018).

[75] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, *A software methodology for compiling quantum programs,* Quantum Science and Technology **3**, 020501 (2018).

[76] A. Oddi and R. Rasconi, in *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (Springer, 2018) pp. 446–461.

[77] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm,* arXiv:1411.4028 (2014).

[78] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor,* Nature Communications **5**, 4213 (2014).

[79] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, *A generative modeling approach for benchmarking and training shallow quantum circuits,* arXiv:1801.07686 (2018).

[80] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, *Quantum circuit learning,* Physical Review A **98**, 032309 (2018).

[81] G. Verdon, J. Pye, and M. Broughton, *A Universal Training Algorithm for Quantum Deep Learning,* arXiv:1806.09729 (2018).

[82] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum autoencoders for efficient compression of quantum data,* Quantum Science and Technology **2**, 045001 (2017).

[83] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *Quantum autoencoders for short depth quantum circuit synthesis,* GitHub article (2018).

[84] B. Dive, A. Pitchford, F. Mintert, and D. Burgarth, *In situ upgrade of quantum simulators to universal computers,* Quantum **2**, 80 (2018).

[85] K. Fujii, H. Kobayashi, T. Morimae, H. Nishimura, S. Tamate, and S. Tani, *Impossibility of Classically Simulating One-Clean-Qubit Model with Multiplicative Error,* Physical Review Letters **120**, 200502 (2018).

[86] B. Rosgen and J. Watrous, in *20th Annual IEEE Conference on Computational Complexity (CCC'05)* (2005) pp. 344–354.

[87] A. Kitaev, *Quantum computations: algorithms and error correction,* Russian Mathematical Surveys **52**, 1191 (1997).

[88] C. M. Dawson and M. A. Nielsen, *The Solovay-Kitaev algorithm,* Quantum Information and Compututation **6**, 81 (2006).

[89] T. T. Pham, R. Van Meter, and C. Horsman, *Optimization of the Solovay-Kitaev algorithm,* Physical Review A **87**, 052332 (2013).

[90] V. Kliuchnikov, D. Maslov, and M. Mosca, *Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits,* Physical Review Letters **110**, 190502 (2013).

[91] V. Kliuchnikov, A. Bocharov, and K. M. Svore, *Asymptotically optimal topological quantum compiling,* Physical Review Letters **112**, 140504 (2014).

[92] Y. Zhiyenbayev, V. M. Akulin, and A. Mandilara, *Quantum compiling with diffusive sets of gates,* Physical Review A **98**, 012325 (2018).

[93] M. Horodecki, P. Horodecki, and R. Horodecki, *General teleportation channel, singlet fraction, and quasidistillation,* Physical Review A **60**, 1888 (1999).

[94] M. A. Nielsen, *A simple formula for the average gate fidelity of a quantum dynamical operation,* Physics Letters A **303**, 249 (2002).

[95] A. Gepp and P. Stocks, *A review of procedures to evolve quantum algorithms,* Genetic Programming and Evolvable Machines **10**, 181 (2009).

[96] M. Suzuki, *Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations,* Physics Letters A **146**, 319 (1990).

[97] IBM Q 5 Tenerife backend specification, `https://github.com/QISKit/qiskit-backend-information/tree/master/backends/tenerife/V1` (2018).

[98] IBM Q 16 Rueschlikon backend specification, (2018).

[99] Rigetti 8Q-Agave specification v.2.0.0.dev0, `http://docs.rigetti.com/en/latest/qpu.html` (2018).

[100] A. G. R. Day, M. Bukov, P. Weinberg, P. Mehta, and D. Sels, *Glassy phase of optimal quantum control,* Physical Review Letters **122**, 020601 (2019).

[101] X. Glorot and Y. Bengio, in *In Proceedings of the International Conference on Artificial Intelligence and Statistics* (2010) pp. 249–256.

[102] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, *A generative modeling approach for benchmarking and training shallow quantum circuits,* arXiv:1801.07686 (2018).

[103] R. LaRose, A. Tikku, É. O'Neel-Judy, L. Cincio, and P. J. Coles, *Variational quantum state diagonalization,* arXiv:1810.10506 (2018).

[104] A. Kandala, K. Temme, A. D. Corcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, *Extending the computational reach of a noisy superconducting quantum processor,* Nature **567**, 491 (2018).

[105] Scikit-optimize, (2018).

[106] J. Močkus, in *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974* (Springer Berlin Heidelberg, Berlin, Heidelberg, 1975) pp. 400–404.

[107] M. A. Osborne, R. Garnett, and S. J. Roberts, in *3rd International Conference on Learning and Intelligent Optimization (LION3) 2009* (2009).

[108] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, *Quantum gradient descent and Newton's method for constrained polynomial optimization,* arXiv:1612.01789 (2016).

[109] I. Kerenidis and A. Prakash, *Quantum gradient descent for linear systems and least squares,* arXiv:1704.04992 (2017).

[110] A. Gilyén, S. Arunachalam, and N. Wiebe, Optimizing quantum optimization algorithms via faster quantum gradient computation, in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (ACM, 2019) pp. 1425–1444.

[111] X.-Q. Zhou, T. C. Ralph, P. Kalasuwan, M. Zhang, A. Peruzzo, B. P. Lanyon, and J. L. O'Brien, *Adding control to arbitrary unknown quantum operations,* Nature Communications **2**, 413 (2011).

[112] B. M. Terhal, Quantum error correction for quantum memories, (2015).

[113] D. Gottesman (AMS eBooks, 2010) pp. 13–58.

[114] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface codes: Towards practical large-scale quantum computation,* Physical Review A (2012), 10.1103/PhysRevA.86.032324.

[115] R. Takagi, S. Endo, S. Minagawa, and M. Gu, *Fundamental limits of quantum error mitigation,* arXiv:2109.04457 [quant-ph] (2021), arXiv: 2109.04457.

[116] K. Temme, S. Bravyi, and J. M. Gambetta, *Error Mitigation for Short-Depth Quantum Circuits,* Physical Review Letters **119**, 180509 (2017).

[117] Y. Li and S. C. Benjamin, *Efficient Variational Quantum Simulator Incorporating Active*

Error Minimization, Physical Review X **7** (2017), 10.1103/physrevx.7.021050.

[118] S. Endo, S. C. Benjamin, and Y. Li, *Practical quantum error mitigation for near-future applications,* Physical Review X **8**, 031027 (2018).

[119] J. J. Wallman and J. Emerson, *Noise tailoring for scalable quantum computation via randomized compiling,* Physical Review A **94**, 052325 (2016).

[120] E. Knill, *Quantum computing with realistically noisy devices,* Nature **434**, 39 (2005).

[121] L. F. Santos and L. Viola, *Dynamical control of qubit coherence: Random versus deterministic schemes,* Physical Review A **72**, 062303 (2005).

[122] L. Viola and E. Knill, *Random decoupling schemes for quantum dynamical control and error suppression,* Physical review letters **94**, 060502 (2005).

[123] B. Pokharel, N. Anand, B. Fortman, and D. A. Lidar, *Demonstration of fidelity improvement using dynamical decoupling with superconducting qubits,* Physical review letters **121**, 220502 (2018).

[124] P. Sekatski, M. Skotiniotis, and W. Dür, *Dynamical decoupling leads to improved scaling in noisy quantum metrology,* New Journal of Physics **18**, 073034 (2016).

[125] H. Ball, M. J. Biercuk, A. Carvalho, R. Chakravorty, J. Chen, L. A. de Castro, S. Gore, D. Hover, M. Hush, P. J. Liebermann, *et al.*, *Software tools for quantum control: Improving quantum computer performance through noise and error suppression,* arXiv preprint arXiv:2001.04060 (2020).

[126] T. J. Green, J. Sastrawan, H. Uys, and M. J. Biercuk, *Arbitrary quantum control of qubits in the presence of universal noise,* New Journal of Physics **15**, 095004 (2013).

[127] J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven, *Decoding quantum errors with subspace expansions,* arXiv:1903.05786 [physics, physics:quant-ph] (2019), arXiv: 1903.05786.

[128] W. J. Huggins, S. McArdle, T. E. O'Brien, J. Lee, N. C. Rubin, S. Boixo, K. B. Whaley, R. Babbush, and J. R. McClean, *Virtual Distillation for Quantum Error Mitigation,* arXiv:2011.07064 [quant-ph] (2021), arXiv: 2011.07064.

[129] B. Koczor, *Exponential error suppression for near-term quantum devices,* Physical Review X **11**, 031057 (2021), arXiv: 2011.05942.

[130] Y. Li and S. C. Benjamin, *Efficient Variational Quantum Simulator Incorporating Active Error Minimization,* Physical Review X **7**, 021050 (2017).

[131] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *Error mitigation with Clifford quantum-circuit data,* arXiv:2005.10189 [quant-ph] (2021), arXiv: 2005.10189.

[132] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, *Error mitigation for universal gates on encoded qubits,* Physical Review Letters **127** (2021), 10.1103/physrevlett.127.200505.

[133] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *Unified approach to data-driven quantum error mitigation,* Phys. Rev. Research **3**, 033098 (2021).

[134] A. T. Arrasmith, P. J. Czarnik, P. J. Coles, and L. Cincio, *Error mitigation with clifford quantum-circuit data,* Quantum **5** (2021), 10.22331/q-2021-11-26-592.

[135] N. Yoshioka, H. Hakoshima, Y. Matsuzaki, Y. Tokunaga, Y. Suzuki, and S. Endo, *Generalized quantum subspace expansion,* arXiv:2107.02611 [quant-ph] (2021), arXiv: 2107.02611.

[136] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *Unified approach to data-driven quantum error mitigation,* arXiv:2011.01157 [quant-ph] (2020), arXiv: 2011.01157.

[137] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, *Error mitigation extends the computational reach of a noisy quantum processor,* Nature **567**, 491 (2019).

[138] R. S. Smith, M. J. Curtis, and W. J. Zeng, *A practical quantum instruction set architecture,* arXiv preprint arXiv:1608.03355 (2016).

[139] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, *Efficient Z gates for quantum computing,* Physical Review A **96** (2017), 10.1103/PhysRevA.96.022330.

[140] X. Fu, L. Riesebos, M. Rol, J. van Straten, J. van Someren, N. Khammassi, I. Ashraf, R. Vermeulen, V. Newsum, K. Loh, *et al.*, in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (IEEE, 2019) pp. 224–237.

[141] A. He, B. Nachman, W. A. de Jong, and C. W. Bauer, *Resource efficient zero noise extrapolation with identity insertions,* arXiv preprint arXiv:2003.04941 (2020).

[142] E. F. Dumitrescu, A. J. McCaskey, G. Hagen, G. R. Jansen, T. D. Morris, T. Papenbrock, R. C. Pooser, D. J. Dean, and P. Lougovski, *Cloud quantum computing of an atomic nucleus,* Physical review letters **120**, 210501 (2018).

[143] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm,* arXiv (2014).

[144] M. Otten and S. K. Gray, *Recovering noise-free quantum observables,* Physical Review A **99**, 012338 (2019).

[145] J. Bylander, S. Gustavsson, F. Yan, F. Yoshihara, K. Harrabi, G. Fitch, D. G. Cory, Y. Nakamura, J.-S. Tsia, and W. D. Oliver, *Noise spectroscopy through dynamical decoupling with a superconducting flux qubit,* Nature Physics **7**, 656 (2019).

[146] F. Yan, S. Gustavsson, J. Bylander, X. Jin, F. Yoshihara, D. G. Cory, Y. Nakamura, T. P. Orlando, and W. D. Oliver, *Rotationg-frame relaxation as a noise spectrum analyser of a superconducting qubit undergoing driven evolution,* Nature Communications **4**, 22337 (2013).

[147] J. Meeson, A. Ya. Tzalenchuk, and T. Lindström, *Evidence for interacting two-level systems from the 1/f noise of a superconducting resonator,* Nature Communications **5**, 4119 (2014).

[148] C. Müller, J. Lisenfeld, A. Shnirman, and P. S., *Non-gaussian noise spectroscopy with a superconducting qubit sensor,* Physical Review B **92**, 035442 (2015).

[149] J. J. Burnett, A. Bengtsson, M. Scigliuzzo, D. Niepce, M. Kudra, P. Delsing, and J. Bylander, *Decoherence benchmarking of superconducting qubits,* npj Quantum Information **5**, 54 (2019).

[150] J. Basset, A. Stockklauser, D.-D.. Jarausch, T. Frey, C. Reichl, W. Wegscheider, A. Wallraff, K. Ensslin, and I. T., *Evaluating charge noise acting on semiconductor quantum dots in the circuit quantum electrodynamics architecture,* Applied Physics Letters **105**, 063105 (2014).

[151] K. W. Chan, W. Huang, C. H. Yang, J. C. C. Hwang, B. Hensen, T. Tanttu, F. E. Hudson, K. M. Itoh, A. Laucht, A. Morello, and A. S. Dzurak, *Assessment of a silicon quantum dot spin qubit environment via noise spectroscopy,* Phys. Rev. Applied **10**, 044017 (2018).

[152] T. Struck, A. Hollmann, F. Schauer, O. Fedorets, A. Schmidbauer, K. Sawano, H. Riemann, N. V. Abrosimov, L. Cywiński, B. D., and L. R. Schreiber, *Low-frequency spin qubit energy splitting noise in highly purified* ^{28}*si/sige,* npj Quantum Information **6**, 40 (2020).

[153] G. A. Álvarez and D. Suter, *Measuring the spectrum of colored noise by dynamical decoupling,* Phys. Rev. Lett. **107**, 230501 (2011).

[154] P. Szańkowski, G. Ramon, J. Krzywda, D. Kwiatkowski, and Ł. Cywiński, *Environmental noise spectroscopy with qubits subjected to dynamical decoupling,* Journal of Physics: Condensed Matter **29**, 333001 (2017).

[155] G. A. Paz-Silva, L. M. Norris, and L. Viola, *Multiqubit spectroscopy of gaussian quantum noise,* Phys. Rev. A **95**, 022121 (2017).

[156] L. Cywiński, R. M. Lutchyn, C. P. Nave, and S. Das Sarma, *How to enhance dephasing time*

in superconducting qubits, Phys. Rev. B **77**, 174509 (2008).

[157] G. A. Paz-Silva and L. Viola, *General transfer-function approach to noise filtering in open-loop quantum control,* Phys. Rev. Lett. **113**, 250501 (2014).

[158] K. Schultz, G. Quiroz, P. Titum, and B. D. Clader, *SchWARMA: A model-based approach for time-correlated noise in quantum circuits,* Phys. Rev. Research **3**, 033229 (2021).

[159] A. Murphy, J. Epstein, G. Quiroz, K. Schultz, L. Tewala, K. McElroy, C. Trout, B. Tien-Street, J. A. Hoffmann, B. Clader, *et al.*, *Universal dephasing noise injection via Schrodinger wave autoregressive moving average models,* arXiv preprint arXiv:2102.03370 (2021).

[160] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, *Digital zero noise extrapolation for quantum error mitigation,* 2020 IEEE International Conference on Quantum Computing and Engineering (QCE) , 306 (2020).

[161] A. He, B. Nachman, W. A. de Jong, and C. W. Bauer, *Zero-noise extrapolation for quantum-gate error mitigation with identity insertions,* Phys. Rev. A **102**, 012426 (2020).

[162] R. LaRose, A. Mari, P. J. Karalekas, N. Shammah, and W. J. Zeng, *Mitiq: A software package for error mitigation on noisy quantum computers,* arXiv preprint arXiv:2009.04417 (2020).

[163] P. Whittle, *Prediction and regulation by linear least-square methods* (English Universities Press, 1963).

[164] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control* (John Wiley & Sons, 2015).

[165] S. H. Holan, R. Lund, G. Davis, *et al.*, *The arma alphabet soup: A tour of arma model variants,* Statistics Surveys **4**, 232 (2010).

[166] K. Temme, S. Bravyi, and J. M. Gambetta, *Error mitigation for short-depth quantum circuits,* Physical Review Letters **119**, 180509 (2017).

[167] Y. Li and S. C. Benjamin, *Efficient variational quantum simulator incorporating active error minimization,* Phys. Rev. X **7**, 021050 (2017).

[168] S. Endo, S. C. Benjamin, and Y. Li, *Practical quantum error mitigation for near-future applications,* Phys. Rev. X **8**, 031027 (2018).

[169] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, *Error mitigation extends the computational reach of a noisy quantum processor,* Nature **567**, 491 (2019).

[170] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *Unified approach to data-driven quantum error mitigation,* Phys. Rev. Research **3**, 033098 (2021).

[171] A. Mari, N. Shammah, and W. J. Zeng, *Extending quantum probabilistic error cancellation by noise scaling,* Phys. Rev. A **104**, 052607 (2021).

[172] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kandala, *Scalable error mitigation for noisy quantum circuits produces competitive expectation values,* arXiv:2108.09197 [cond-mat, physics:quant-ph] (2021), arXiv: 2108.09197.

[173] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. V. Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, Tensorflow quantum: A software framework for quantum machine learning, (2021), arXiv:2003.02989 [quant-ph] .

[174] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, *Measuring the capabilities of quantum computers,* arXiv:2008.11294 [quant-ph] (2020), arXiv: 2008.11294.

[175] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa, A. Kandala, G. A. Keefe, K. Krsulich, W. Landers, E. P. Lewandowski, D. T. McClure, G. Nannicini, A. Narasgond, H. M. Nayfeh, E. Pritchett, M. B. Rothwell, S. Srinivasan, N. Sundaresan, C. Wang, K. X. Wei, C. J. Wood, J.-B. Yau, E. J. Zhang, O. E. Dial, J. M. Chow, and J. M. Gambetta, *Demonstration of quantum volume 64 on a superconducting quantum computing system,* Quantum Science and Technology **6**, 025020 (2021).

[176] Y. Li and S. C. Benjamin, *Efficient variational quantum simulator incorporating active error minimisation,* Physical Review X (2016), 10.1103/PhysRevX.7.021050.

[177] K. Temme, S. Bravyi, and J. M. Gambetta, *Error mitigation for short-depth quantum circuits,* Physical Review Letters (2016), 10.1103/PhysRevLett.119.180509.

[178] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, *Digital zero noise extrapolation for quantum error mitigation,* 2020 IEEE International Conference on Quantum Computing and Engineering (QCE) , 306–316 (2020), arXiv: 2005.10921.

[179] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kandala, *Scalable error mitigation for noisy quantum circuits produces competitive expectation values,* arXiv:2108.09197 [cond-mat, physics:quant-ph] (2021), arXiv: 2108.09197.

[180] W. J. Huggins, S. McArdle, T. E. O'Brien, J. Lee, N. C. Rubin, S. Boixo, K. B. Whaley, R. Babbush, and J. R. McClean, *Virtual distillation for quantum error mitigation,* Physical Review X **11**, 041036 (2021), arXiv: 2011.07064.

[181] J. Cotler, S. Choi, A. Lukin, H. Gharibyan, T. Grover, M. E. Tai, M. Rispoli, R. Schittko, P. M. Preiss, A. M. Kaufman, M. Greiner, H. Pichler, and P. Hayden, *Quantum virtual cooling,* Physical Review X **9**, 031013 (2019), arXiv: 1812.02175.

[182] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. E. Mandouh, M. H. Gordon, Y. Hindy, A. Robertson, P. Thakre, N. Shammah, and W. J. Zeng, *Mitiq: A software package for error mitigation on noisy quantum computers,* arXiv:2009.04417 [quant-ph] (2021), arXiv: 2009.04417.

[183] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, *Error mitigation extends the computational reach of a noisy quantum processor,* Nature **567**, 491–495 (2019).

[184] S. Zhang, Y. Lu, K. Zhang, W. Chen, Y. Li, J.-N. Zhang, and K. Kim, *Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system,* Nature Communications **11**, 587 (2020).

[185] E. v. d. Berg, Z. K. Minev, A. Kandala, and K. Temme, *Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors,* arXiv:2201.09866 [quant-ph] (2022), arXiv: 2201.09866.

[186] D. Bultrini, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *Unifying and benchmarking state-of-the-art quantum error mitigation techniques,* arXiv:2107.13470 [quant-ph] (2021), arXiv: 2107.13470.

[187] E. Huffman, M. G. Vera, and D. Banerjee, *Real-time dynamics of plaquette models using NISQ hardware,* arXiv:2109.15065 [cond-mat, physics:hep-lat, physics:quant-ph] (2021), arXiv: 2109.15065.

[188] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, *A quantum-classical cloud platform optimized for variational hybrid algorithms,* Quantum Science and Technology **5**, 024003 (2020), arXiv: 2001.04449.

[189] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, *Demonstration of the trapped-ion quantum-CCD computer architecture,* Nature **592**, 209–213 (2021), arXiv: 2003.01293.

[190] C. H. Baldwin, K. Mayer, N. C. Brown, C. Ryan-Anderson, and D. Hayes, *Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations,* arXiv:2110.14808 [quant-ph] (2021), arXiv: 2110.14808.

[191] P. Czarnik, A. Arrasmith, L. Cincio, and P. J. Coles, *Qubit-efficient exponential suppression of errors,* arXiv:2102.06056 [quant-ph] (2021), arXiv: 2102.06056.

[192] S. Aaronson, *Shadow Tomography of Quantum States,* arXiv:1711.01053 [quant-ph] (2018), arXiv: 1711.01053.

[193] H.-Y. Huang, R. Kueng, and J. Preskill, *Predicting Many Properties of a Quantum System from Very Few Measurements,* Nature Physics **16**, 1050 (2020), arXiv: 2002.08953.

[194] H.-Y. Hu and Y.-Z. You, *Hamiltonian-driven shadow tomography of quantum states,* Phys. Rev. Research **4**, 013054 (2022).

[195] H.-Y. Hu, S. Choi, and Y.-Z. You, *Classical Shadow Tomography with Locally Scrambled Quantum Dynamics,* arXiv:2107.04817 [cond-mat, physics:quant-ph] (2021), arXiv: 2107.04817.

[196] M. Ohliger, V. Nesme, and J. Eisert, *Efficient and feasible state tomography of quantum many-body systems,* New Journal of Physics **15**, 015024 (2013).

[197] J. Cotler, S. Choi, A. Lukin, H. Gharibyan, T. Grover, M. E. Tai, M. Rispoli, R. Schittko, P. M. Preiss, A. M. Kaufman, M. Greiner, H. Pichler, and P. Hayden, *Quantum Virtual Cooling,* Physical Review X **9**, 031013 (2019), arXiv: 1812.02175.

[198] F. G. S. L. Brandao, A. W. Harrow, and M. Horodecki, *Local random quantum circuits are approximate polynomial-designs,* Communications in Mathematical Physics **346**, 397 (2016), arXiv: 1208.0692.

[199] S. Chen, W. Yu, P. Zeng, and S. T. Flammia, *Robust shadow estimation,* PRX Quantum **2** (2021), 10.1103/prxquantum.2.030348.

[200] D. Enshan Koh and S. Grewal, *Classical Shadows with Noise,* arXiv e-prints , arXiv:2011.11580 (2020), arXiv:2011.11580 [quant-ph] .

[201] K. Bu, D. Enshan Koh, R. J. Garcia, and A. Jaffe, *Classical shadows with Pauli-invariant unitary ensembles,* arXiv e-prints , arXiv:2202.03272 (2022), arXiv:2202.03272 [quant-ph] .

[202] E. Magesan, J. M. Gambetta, and J. Emerson, *Scalable and robust randomized benchmarking of quantum processes,* Phys. Rev. Lett. **106**, 180504 (2011).

[203] J. Claes, E. Rieffel, and Z. Wang, *Character randomized benchmarking for non-multiplicity-free groups with applications to subspace, leakage, and matchgate randomized benchmarking,* PRX Quantum **2**, 010351 (2021).

[204] R. Levy, D. Luo, and B. K. Clark, *Classical Shadows for Quantum Process Tomography on Near-term Quantum Computers,* arXiv:2110.02965 [cond-mat, physics:physics, physics:quant-ph] (2021), arXiv: 2110.02965.

[205] S. H. Sack, R. A. Medina, A. A. Michailidis, R. Kueng, and M. Serbyn, *Avoiding barren plateaus using classical shadows,* arXiv e-prints , arXiv:2201.08194 (2022), arXiv:2201.08194 [quant-ph] .