Efficient and trusted operation of quantum computers and quantum simulators

by

Ryan Matthew Shaffer

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Physics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Hartmut Häffner, Chair
Professor Ehud Altman
Professor K. Birgitta Whaley

Fall 2022

Efficient and trusted operation of quantum computers and quantum simulators

Abstract

Efficient and trusted operation of quantum computers and quantum simulators

by

Ryan Matthew Shaffer

Doctor of Philosophy in Physics

University of California, Berkeley

Professor Hartmut Häffner, Chair

Quantum computers have the potential to revolutionize the computational power accessible to humanity. But because the scale and quality of current devices is limited, maximizing their usefulness requires that researchers make efficient use of the resources that they provide.

Motivated by this, we introduce in this work several techniques for efficient verification and operation of near-term quantum devices. We first develop and demonstrate a stochastic quantum compilation technique (STOQ) which produces approximate compilations of unitaries in terms of any arbitrary native instruction set. We then construct a novel verification protocol for quantum devices called randomized analog verification (RAV), which uses STOQ to generate verification sequences that ideally leave the system near a measurement basis state, which allows for efficient estimation of the success rate.

We first apply RAV to the context of quantum computers with continuously-parameterized native gate sets. We demonstrate both numerically and experimentally that RAV provides an efficiency advantage over cross-entropy benchmarking (XEB) when estimating an error rate experimentally. We then adapt the RAV protocol for verification of analog quantum simulators, and we demonstrate its sensitivity to various types of error both numerically and experimentally.

In addition, we discuss practical techniques for efficient operation of quantum computers. We develop and implement a surrogate-based optimization (SBO) technique for variational quantum algorithms, and we demonstrate that the technique has significant performance advantages over the most commonly-used optimization methods. Then, applying a more experimental lens, we describe work toward realistic simulation of various aspects of a trapped-ion experiment, such as experimental control sequences and calibration runs.

Taken together, the techniques introduced in this work are a collection of incremental steps toward the broader goal of increasing efficiency in near-term quantum computers and quantum simulators.

To my wife Irene,

my source of endless

inspiration, motivation, and support.

Acknowledgments

First and foremost, I would like to express my gratitude to Hartmut Häffner, my research advisor and mentor throughout my PhD journey at Berkeley. His constant encouragement was essential to my perseverance, and the work in this thesis would not have been remotely possible without his scientific expertise and advice. Additionally, I would like to express thanks to Birgitta Whaley, who selected me to be a graduate student instructor for her undergraduate quantum computing course, and who was always willing to spend time reviewing and providing valuable feedback on my research.

I would also like to thank everyone in the Häffner group that I worked with and learned from over the course of my time in Berkeley. This includes those I worked with directly on research and experiments: Eli Megidish, Emiliia Dyrenkova, Hang Ren, Joseph Broz, Shuqi Xu, Sumanta Khan, and Wei-Ting Chen. And it also includes everyone else in the group who joined in frequent meals and coffee breaks, social outings, and several highly competitive seasons of fantasy football: Alberto Alonso, Ben Saarel, Chi Chi An, Clemens Matthiesen, Crystal Noel, Elia Perego, Erik Urban, Isabel Sacksteder, Justin Phillips, Maya Berlin-Udi, Neha Yadav, Neil Glikin, Nicole Greene, Qian Yu, Ryan Tollefsen, and Sara Mouradian. These relationships are among the most fun, memorable, and lasting parts of my PhD experience.

In addition, I would like to acknowledge my collaborators at Sandia National Laboratories. First, I would like to express great appreciation to Mohan Sarovar, who mentored me during my internship at Sandia, introducing me to the area of research that we pursued and guiding me to the most interesting questions. I would also like to thank Lucas Kocia, who contributed his expertise to our project and helped to make it a success. Additionally, I would like to thank the team behind the Quantum Scientific Computing Open User Testbed (QSCOUT) at Sandia, and in particular Christopher Yale, who spent many hours over the course of months working with me to iterate on experimental procedures and provide results for our research.

I also acknowledge the generous funding support I received from two fellowships during my PhD program. First, the National Defense Science and Engineering (NDSEG) fellowship, which funded the first four years of my program. And second, the Quantum Information Science and Engineering Network (QISE-NET) fellowship, which provided funding during my final two years and facilitated my collaboration with Sandia.

Finally, I would like to express the ultimate appreciation to those in my family who have constantly supported me throughout these years of seemingly endless work, much of which occurred amid a world-altering pandemic. To my wife Irene, who has always encouraged me to pursue the challenge. To my daughter Vivian, who is too young to remember these days but will always hear the stories. To my parents Jerry and Lee Ann, who encouraged a love of science from my earliest days. To my in-laws Tim and Twila, who are always willing to support me in my endeavors. And to my grandparents Jerry, JoAnn, Frank, and Delores, who always wondered when their grandson would finally become a doctor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Quantum computers have the potential to revolutionize the computational power accessible to humanity. Many believe that quantum computers will have the ability to significantly outperform their classical counterparts across a range of applications, such as breaking common encryption schemes [143], simulating complex chemical dynamics [84], and finding solutions to difficult optimization problems [64].

But achieving computational advantage with quantum computers will require hardware significantly beyond the current state-of-the-art in both quality and scale. Despite many scientific breakthroughs in recent decades, current quantum computers are still relatively small and unreliable compared to what will be needed to surpass conventional supercomputers on real-world applications. The devices that exist today, which have been labeled as "noisy intermediate-scale quantum" (NISQ) devices [125], are most useful as tools for research and exploration of what will be possible as the devices continue to improve in reliability and size over the coming years. As illustrated in Figure 1.1, these devices can be implemented using a wide range of physical platforms, including chains of atomic ions trapped in electrical potentials, superconducting circuits cooled nearly to absolute zero, and beams of photons controlled via high-precision optics.

Because the scale and quality of current devices is limited, maximizing their usefulness requires that researchers make efficient use of the resources that they provide. First, since the devices are fundamentally error-prone, finding efficient techniques for calibrating and verifying the configuration of the devices will be critical for ensuring that they are operating to the best of their capability. In addition, because the devices are difficult and expensive to build, demand for computational time will likely exceed supply for some time. Therefore, techniques for improving the efficiency of computations will lead directly to increased benefit to the researchers who are exploring potential algorithms and applications.

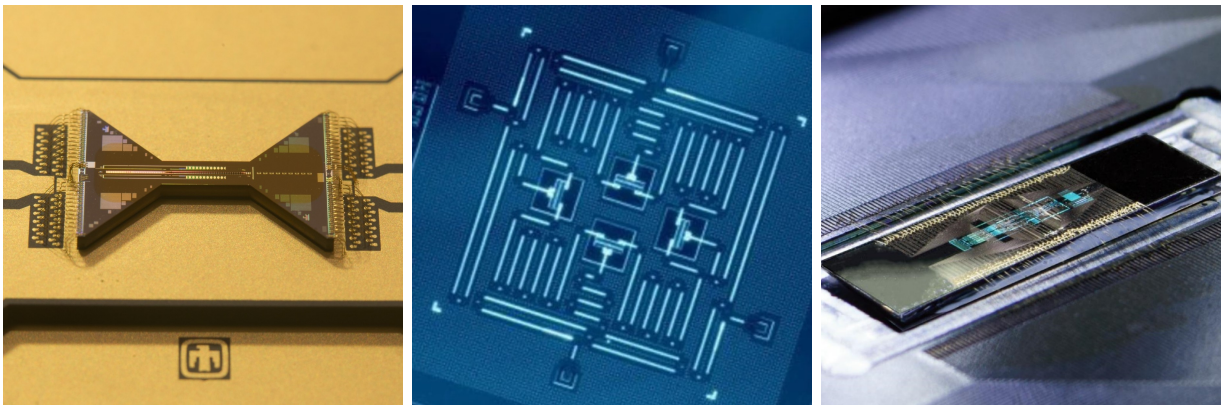


Figure 1.1: Illustrations of various quantum computing platforms. From left to right: an ion trap produced by Sandia National Laboratories [129], a superconducting quantum processor produced by IBM [48], and an integrated silicon photonic quantum processor produced by PsiQuantum [153].

1.2 Structure

With this motivation, then, the subsequent chapters of this work introduce several techniques for efficient verification and operation of near-term quantum devices. Chapter 2 provides an overview of basic quantum computing concepts such as qubits, gates, and algorithms, as well as an introduction to various types of errors that affect the operation of quantum computers. Subsequently, Chapter 3 introduces the problem of quantum compilation and describes several approaches to solving this problem, including a novel stochastic technique which produces approximate compilations.

This stochastic compilation technique is then applied in the context of verification protocols for quantum devices. First, in Chapter 4, an efficient technique is introduced for verification of quantum computers with continuously-parameterized gates, including numerical and experimental demonstrations of its efficiency advantage over existing protocols. Next, in Chapter 5, this technique is adapted for verification of analog quantum simulators, in which a quantum device is programmed to emulate a particular physical system of interest; again, numerical and experimental demonstrations are provided to demonstrate the effectiveness of the procedure.

Efficiency is critical not only for verifying the behavior of quantum computers, but also for operating them. Toward this end, Chapter 6 introduces an efficient optimization technique for a class of near-term quantum algorithms known as variational quantum algorithms, and the superiority of this method over existing optimization techniques is demonstrated on a variety of common problems. In addition to improving the operation of quantum computers, the ability to simulate the physical device itself is useful for developing and testing new experimental protocols. In Appendix A, this experimentally-focused lens is applied, and basic

procedures are developed for simulating various aspects of trapped-ion quantum computing hardware.

The work concludes in Chapter 7 with a summary of the topics covered in this work and a future-looking assessment of the field of quantum computing.

Chapter 2

Overview of quantum computation

2.1 Introduction

This chapter is organized as follows. Section 2.2 introduces qubits and gates, the fundamental building blocks of quantum computation. Next, Section 2.3 describes how to build quantum programs – i.e., quantum circuits and quantum algorithms – from these building blocks. Section 2.4 provides an overview of errors in quantum computation, including classification and characterization of errors. Finally, the chapter is summarized in Section 2.5.

Most topics in this chapter are introduced very briefly. It is assumed that the reader already has some familiarity with the concepts, as well as a solid background in quantum mechanics and linear algebra. For a more thorough introduction to the fundamental concepts of quantum computing, the reader is encouraged to consult one of many outstanding textbooks in the field, such as *Quantum Computation and Quantum Information* by Nielsen and Chuang [119].

2.2 Qubits and operations

Single-qubit states

The fundamental unit of quantum information is known as a *qubit*. A single qubit represents a two-level quantum system whose state can be represented as a vector in a two-dimensional complex vector space known as a Hilbert space. A standard orthonormal basis is chosen for this system, often referred to as the *computational basis*, and the corresponding basis states are conventionally defined as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

According to the principles of quantum mechanics, an arbitrary single qubit state $|\psi\rangle$ can be represented as a linear combination of these basis states,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, in order to satisfy the normalization constraint $\langle\psi|\psi\rangle = 1$.

Because the normalization condition enforces that the the state vector must be of unit length, it is common to represent a pure state of a single qubit as a point on a unit sphere, known as the *Bloch sphere*. A diagram of this sphere is shown in Figure 2.1. This depiction suggests an alternative representation of an arbitrary single qubit state,

$$|\psi\rangle = e^{i\gamma}\left[\cos\tfrac{\theta}{2}|0\rangle + e^{i\varphi}\sin\tfrac{\theta}{2}|1\rangle\right], \tag{2.3}$$

where θ is the polar angle (the angle with respect to the $+z$ axis), φ is the azimuthal angle (the angle of rotation when projected onto the x-y plane), and γ is a global phase. Note that this global phase is unobservable and therefore irrelevant when considering only a single-qubit system, but it may become relevant when considering the relative phases among multiple qubits. Note also that in this representation, the normalization condition $\langle\psi|\psi\rangle = 1$ is enforced automatically.

We note that $\theta = 0$ and $\theta = \pi$ correspond to the basis states $|0\rangle$ and $|1\rangle$, respectively. Since the computational basis vectors align with the $+z$ and $-z$ axes on the Bloch sphere, it is logical to define two additional orthonormal bases. By considering the unit vectors along the $+x$ and $-x$ axes on the Bloch sphere, we can define the *Hadamard basis* as

$$|+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \qquad |-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right). \tag{2.4}$$

Likewise, we can define an orthonormal basis using unit vectors along the $+y$ and $-y$ axes as

$$|+i\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right) \qquad |-i\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - i|1\rangle\right). \tag{2.5}$$

Single-qubit operations

To manipulate the state of a qubit, we must be able to perform operations on that qubit. In general, such operations are represented as a unitary operator acting on the quantum state of the system. A single-qubit operator can be represented as a 2×2 matrix, which is typically written in the computational basis. A commonly-used set of such operators are

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.6}$$

where $\{X, Y, Z\}$ are the usual single-qubit *Pauli operators*, which are often equivalently denoted as $\{\sigma_x, \sigma_y, \sigma_z\}$. We note that each of these operators is self-adjoint, i.e., $I = I^\dagger$,

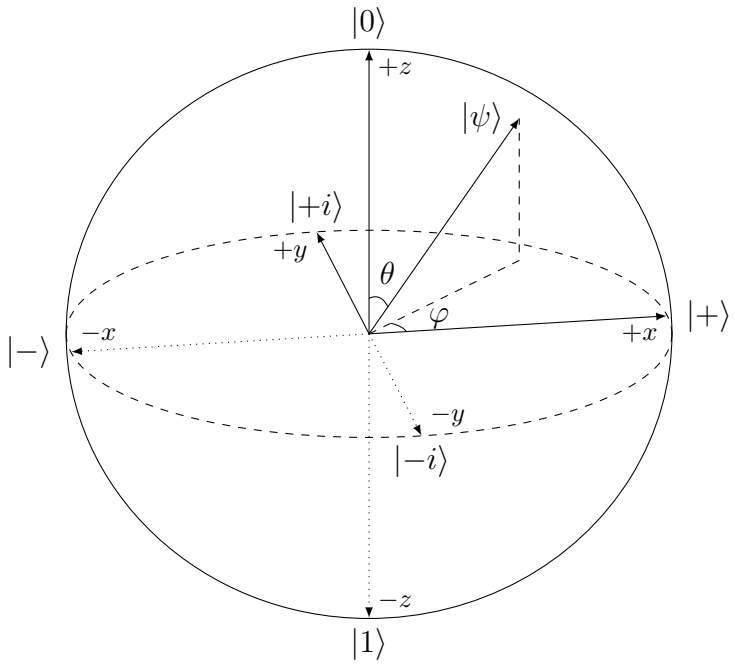


Figure 2.1: Bloch sphere representation of a single-qubit state $|\psi\rangle$, along with the angles θ and φ from the single-qubit state representation given in Equation 2.3. Note that the $\{|0\rangle, |1\rangle\}$ states lie along the $\{+z, -z\}$ axes, the $\{|+\rangle, |-\rangle\}$ states lie along the $\{+x, -x\}$ axes, and the $\{|+i\rangle, |-i\rangle\}$ states lie along the $\{+y, -y\}$ axes.

$X = X^\dagger$, $Y = Y^\dagger$, and $Z = Z^\dagger$, where the dagger denotes the Hermitian conjugate. Given that $U^\dagger U = I$ for any unitary operator U, this implies that each of the Pauli operators squares to the identity operator, i.e., $XX = YY = ZZ = I$.

When considering their action on a single qubit, we often refer to operators as *gates*, since they manipulate the state of the qubit much like gates in classical computing manipulate the states of classical bits. For example, the X gate acts on the computational basis states as

$$X|0\rangle = |1\rangle \qquad X|1\rangle = |0\rangle. \tag{2.7}$$

Because the X gate inverts the state of the qubit, it is sometimes called a NOT gate, since it is analogous to the classical gate of the same name which inverts the state of a classical bit.

Using linearity, then, we can calculate the effect of the X gate on the arbitrary single-qubit state of Equation 2.2 as

$$X\left[\alpha|0\rangle + \beta|1\rangle\right] = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle \tag{2.8}$$

A unitary single-qubit operator corresponds to a *rotation* of the state vector in the Bloch sphere picture. We can represent an arbitrary single-qubit rotation as a linear combination

of Pauli operators. It is conventional to define such a rotation as

$$R(\theta, \hat{n}) = \exp\left(-i\frac{\theta}{2}\hat{n}\cdot\vec{\sigma}\right) = \left(\cos\frac{\theta}{2}\right)I - i\left(\sin\frac{\theta}{2}\right)(n_x X + n_y Y + n_z Z) \tag{2.9}$$

where θ is the angle of rotation, and $\hat{n} = (n_x, n_y, n_z)$ is the unit vector representing the axis of rotation, $\vec{\sigma} = (X, Y, Z)$ is the vector of Pauli matrices. We note that any single-qubit operator can be written in this form (up to some global phase, which we will disregard).

Often it is useful to specialize this to define parameterized operators for rotations around each of the three axes, i.e., by setting $\hat{n} = \hat{x}$, $\hat{n} = \hat{y}$, and $\hat{n} = \hat{z}$. By so doing, we obtain the following:

$$R_X(\theta) = \exp\left(-i\frac{\theta}{2}X\right) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.10}$$

$$R_Y(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.11}$$

$$R_Z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.12}$$

These operators are often realizable in physical quantum computing devices. Note that we have $R_X(\pi) = -iX$, $R_Y(\pi) = iY$, and $R_Z(\pi) = -iZ$. So, up to a global phase, the Pauli operators can be thought of as π-rotations around the corresponding axis of the Bloch sphere.

Some devices have the ability to implement parameterized single-qubit rotations around an arbitrary axis in the x-y plane. We define the corresponding operator as

$$R(\theta, \varphi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{i\varphi}\sin\frac{\theta}{2} \\ ie^{-i\varphi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{2.13}$$

where φ specifies the axis of rotation in the x-y plane, and θ specifies the angle of rotation.

Several additional specific single-qubit gates are widely used in the literature. A few of these are

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad S = R_Z\left(\frac{\pi}{2}\right) = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \qquad T = R_Z\left(\frac{\pi}{4}\right) = \begin{bmatrix} 0 & 1 \\ 0 & e^{i\pi/4} \end{bmatrix}, \tag{2.14}$$

where H is the Hadamard gate, $S = \sqrt{Z}$ is the phase gate (sometimes denoted as P), and $T = \sqrt{S}$ (also known as the $\pi/8$ gate) is a commonly-used gate in fault-tolerant quantum computation schemes.

Multiple-qubit states

Quantum computers must control and manipulate many qubits. We can represent the state of an n-qubit system as a vector in a 2^n-dimensional Hilbert space. It is convenient to continue to use the computational basis, where the basis vectors for the n-qubit system can be represented as the tensor product of all possible combinations of the computational basis vectors for each of the n individual qubits.

As a simple example, consider a two-qubit system. We can write the computational basis states as

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \qquad |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \tag{2.15}$$

where the notation $|00\rangle$ is shorthand for the tensor product $|0\rangle \otimes |0\rangle$, sometimes denoted as $|0\rangle|0\rangle$, and represents the state in which each of the qubits is in the state $|0\rangle$. Note that depending on the context, it may be convenient to use decimal rather than binary notation for the computational basis states, i.e., $|00\rangle \to |0\rangle$, $|01\rangle \to |1\rangle$, $|10\rangle \to |2\rangle$, and $|11\rangle \to |3\rangle$.

An arbitrary n-qubit state $|\psi\rangle$ can be written as a linear combination of the 2^n basis vectors

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle, \tag{2.16}$$

where the $|j\rangle$ vectors correspond to some orthonormal basis (such as the computational basis), and the α_j coefficients are complex numbers such that $\sum_j |\alpha_j|^2 = 1$.

Multiple-qubit operations

To manipulate the state of multiple qubits simultaneously – and in particular, to generate entanglement, which is essential to nearly all quantum algorithms – we must be able to perform operations that act on a multi-qubit space. A multi-qubit operator acting on n qubits can be represented as a $2^n \times 2^n$ unitary matrix, which is typically written in the computational basis.

Most commonly, discussions of multi-qubit gates are restricted to the case where $n = 2$, since this is both the simplest case and the case most frequently implemented in physical devices. The canonical two-qubit gate is the CNOT ("controlled NOT") gate, which is defined in the computational basis as

$$\mathrm{CNOT}_{0,1} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \tag{2.17}$$

where the subscripts indicate the index[1] of the control and target qubits, respectively.

The action of the CNOT gate on the computational basis states is

$$\begin{aligned} \mathrm{CNOT}_{0,1}|00\rangle &= |00\rangle \qquad \mathrm{CNOT}_{0,1}|10\rangle = |11\rangle \\ \mathrm{CNOT}_{0,1}|01\rangle &= |01\rangle \qquad \mathrm{CNOT}_{0,1}|11\rangle = |10\rangle \end{aligned} \tag{2.18}$$

where the state of the target qubit is inverted if and only if the state of the control qubit is $|1\rangle$. The inversion occurs as an X operation on the target qubit; for this reason, the CNOT gate is also known as the CX ("controlled X") gate.

The CNOT gate can be used to entangle two qubits. For example, if the control qubit is in an arbitrary state $\alpha|0\rangle + \beta|1\rangle$ and the target qubit is in state $|0\rangle$, the CNOT gate will act as follows:

$$\mathrm{CNOT}_{0,1}\left(\alpha|0\rangle + \beta|1\rangle\right)|0\rangle = \alpha|00\rangle + \beta|11\rangle. \tag{2.19}$$

Many other two-qubit gates are commonly used. For example, two-qubit gates can be constructed by taking tensor products of single-qubit Pauli operators. We can therefore define $XX = X \otimes X$, $YY = Y \otimes Y$, and $ZZ = Z \otimes Z$ as

$$XX = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad YY = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \qquad ZZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.20}$$

More generally, we can define a parameterized family of two-qubit gates by exponentiating the XX, YY, and ZZ gates:

$$XX(\theta) = \exp\left(-i\frac{\theta}{2}XX\right) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & -i\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ -i\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.21}$$

$$YY(\theta) = \exp\left(-i\frac{\theta}{2}YY\right) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & i\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ i\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.22}$$

$$ZZ(\theta) = \exp\left(-i\frac{\theta}{2}ZZ\right) = \begin{bmatrix} e^{-i\theta/2} & 0 & 0 & 0 \\ 0 & e^{i\theta/2} & 0 & 0 \\ 0 & 0 & e^{i\theta/2} & 0 \\ 0 & 0 & 0 & e^{-i\theta/2} \end{bmatrix}. \tag{2.23}$$

[1]Here and throughout, we use zero-indexed *little endian* ordering to label qubits. That is, when writing out an n-qubit state, the leftmost qubit is qubit 0, and the rightmost qubit is qubit $n-1$.

The parameter θ can be thought of as the angle of rotation around the corresponding two-qubit "axis". The parameterizations are constructed such that setting $\theta = \pi$ recovers the fixed XX, YY, and ZZ gates up to a global phase: $XX(\pi) = -i\ XX$, $YY(\pi) = -i\ YY$, and $ZZ(\pi) = -i\ ZZ$.

Quantum computing devices built on various physical platforms implement particular two-qubit gates natively, and it is advantageous to use these gates directly when possible. For example, the canonical two-qubit gate for a trapped-ion quantum computer is known as the Mølmer-Sørensen gate [112, 145], which in its fixed form is commonly defined as an $XX(\pi/2)$ gate:

$$MS = XX(\pi/2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & -i \\ 0 & 1 & -i & 0 \\ 0 & -i & 1 & 0 \\ -i & 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

This is an entangling gate which takes any computational basis state to a maximally-entangled two-qubit state:

$$\begin{aligned} MS|00\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle - i|11\rangle) \qquad & MS|10\rangle &= \tfrac{1}{\sqrt{2}}(|10\rangle - i|01\rangle) \\ MS|01\rangle &= \tfrac{1}{\sqrt{2}}(|01\rangle - i|10\rangle) \qquad & MS|11\rangle &= \tfrac{1}{\sqrt{2}}(|11\rangle - i|00\rangle). \end{aligned} \tag{2.25}$$

We can also define a more generalized form of the Mølmer-Sørensen interaction by introducing parameters specifying the "angle" and "axis" of rotation in the two-qubit space:

$$MS(\theta, \varphi) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & -ie^{-i2\varphi}\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ -ie^{i2\varphi}\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix}. \tag{2.26}$$

Here θ and φ represent the angle and axis of rotation, respectively. We note that setting $\varphi = 0$ gives $MS(\theta, 0) = XX(\theta)$, and setting $\varphi = \frac{\pi}{2}$ gives $MS(\theta, \frac{\pi}{2}) = YY(\theta)$.

Clifford operations

The *Pauli group* acting on n qubits, which we denote as $\mathbf{P}_n$, is defined as the set of operations which can be produced by taking the tensor product of one of the Pauli basis elements $\{I, X, Y, Z\}$ acting on each qubit. For example, members of the Pauli group $\mathbf{P}_4$ would include $X_0Y_1I_2Y_3$, $Z_0Z_1Z_2I_3$, etc., where the subscript indicates the qubit number. The set of unitary operations U such that $UPU^\dagger \in \mathbf{P}_n$ for any $P \in \mathbf{P}_n$ is known as the *Clifford group* on n qubits [56], which we denote as $\mathbf{C}_n$.

It turns out that $\mathbf{C}_n$ can be fully generated by the set $\{H, S, \text{CNOT}\}$, where the H and S gates can be applied to any of the n qubits, and the CNOT gate can be applied with any

of the n qubits as its control and target qubits. So we can equivalently define the Clifford group as the set of all operations which can be produced from these generators. It follows that the product of any two Clifford operations is also a Clifford operation, and that the inverse of any Clifford operation is also a Clifford operation.

Famously, the Gottesman-Knill theorem [55] states that any quantum computer which performs only Clifford operations (and measures in the computational basis) can be simulated classically in polynomial time. This implies that Clifford operations cannot perform arbitrary quantum computation, which requires exponential time to simulate classically. It turns out that an element of $\mathbf{C}_n$ can be uniquely identified by a $2n \times 2n$ matrix [1], whereas specifying an arbitrary n-qubit unitary operator requires a matrix of size $2^n \times 2^n$. This leads to the intuition that Clifford operations allow the quantum system to explore only an exponentially-small subset of the full n-qubit Hilbert space. In practice, it also means that simulating sequences of Clifford operations is easy classically and can be done efficiently even with thousands of qubits [1].

Universal gate sets

To achieve universal quantum computation, we need a *universal gate set* which can be used to generate any n-qubit unitary to arbitrary precision. This can be done by taking the Clifford generators $\{H, S, \mathrm{CNOT}\}$ and choosing one additional generator, which could be a single-qubit T gate, a two-qubit $\sqrt{\mathrm{CNOT}}$ gate, or a three-qubit Toffoli gate [55]. The T gate is a $\pi/4$ rotation around the z-axis, which is often natively supported by hardware, and therefore it is a convenient choice. Because $T = \sqrt{S}$, it follows that the set $\{H, T, \mathrm{CNOT}\}$ is an example of a universal gate set for quantum computation.

One of the fundamental results of the field of quantum information, known as the Solovay-Kitaev theorem [88], states that a finite number of gates from a universal gate set such as $\{H, T, \mathrm{CNOT}\}$ can be used to implement any desired unitary to arbitrarily small error. Further discussion of the algorithm that achieves this, which is known as the Solovay-Kitaev algorithm [35], can be found in Section 3.2.

2.3 Circuits and algorithms

Quantum circuits and measurement

Sequences of quantum gates are often referred to as *quantum circuits*. A simple circuit diagram is shown in Figure 2.2, where each horizontal line represents a qubit, and the operations are time-ordered from left to right. To read out the state of the qubits after performing the desired operations, a *measurement* gate is used, which represents a projective measurement in the computational basis. After measurement, the corresponding qubit is left in one of the computational basis states (corresponding to the measurement result) and is therefore unentangled from the rest of the system.

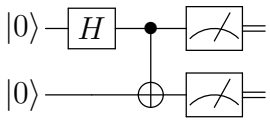


Figure 2.2: Two-qubit circuit diagram for Bell state generation. Each qubit is represented by a horizontal line, and the operations are time-ordered from left to right. Each qubit is initialized in the state $|0\rangle$. A Hadamard gate is applied to the upper qubit, followed by a CNOT gate with the upper qubit as the control qubit and the lower qubit as the target qubit. Finally, each qubit is measured in the computational basis. The double lines after the measurements represent the classical bit of information that is obtained.

The circuit in Figure 2.2 performs Bell state generation, which is a common subroutine in quantum algorithms. The Bell states, which themselves form a basis over the two-qubit space, are typically defined as:

$$\begin{aligned} |\Phi^+\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad |\Psi^+\rangle = \tfrac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\Phi^-\rangle &= \tfrac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \qquad |\Psi^-\rangle = \tfrac{1}{\sqrt{2}}(|01\rangle - |10\rangle). \end{aligned} \tag{2.27}$$

We can write the circuit in operator form as $(\mathrm{CNOT}_{0,1})(H_0 \otimes I_1)$, where the rightmost operator is applied first, and the subscripts indicate the qubit number. Therefore, considering an input state of $|00\rangle$, we observe that a Bell state is indeed generated:

$$(\mathrm{CNOT}_{0,1})(H_0 \otimes I_1)|00\rangle = (\mathrm{CNOT}_{0,1})\tfrac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\Phi^+\rangle. \tag{2.28}$$

By using the remaining computational basis states as inputs, each of the remaining three Bell states can be generated.

This circuit concludes by performing a projective measurement on each of the qubits in the computational basis. Measurement probabilities can be calculated using the Born rule, which states that the probability of measuring outcome j is $P(j) = |\langle j|\psi\rangle|^2$. If the two-qubit system is in state $|\Phi^+\rangle$, then our measurement probabilities for each computational basis state can be calculated as:

$$\begin{aligned} P(00) &= |\langle 00|\Phi^+\rangle|^2 = \tfrac{1}{2} \qquad P(10) = |\langle 10|\Phi^+\rangle|^2 = 0 \\ P(01) &= |\langle 01|\Phi^+\rangle|^2 = 0 \qquad P(11) = |\langle 11|\Phi^+\rangle|^2 = \tfrac{1}{2}. \end{aligned} \tag{2.29}$$

Example: Quantum phase estimation

Quantum circuits can be cleverly constructed to implement *quantum algorithms*. Any classical algorithm can be trivially translated into a quantum circuit and executed on a quantum computer, but most interesting quantum algorithms take advantage of the unique features of

quantum systems (i.e., superposition, entanglement, and interference) to implement procedures which have no direct classical analog. These quantum algorithms can often be shown to have asymptotic speedups over the best known (or, in some cases, best *possible*) classical approaches to solve the same problem.

A broad survey of quantum algorithms is out-of-scope for this work, but the reader is encouraged to consult [77] for a comprehensive review and reference implementations of a wide range of common quantum algorithms. As an example, we provide here a brief overview of an algorithm known as *quantum phase estimation*. We choose to discuss this algorithm because it is relatively concise (it can be expressed in a single circuit diagram) and introduces several key concepts, including energy estimation, phase kickback, probabilistic outcomes, and implementation of arbitrary unitaries.

Suppose we have some unitary operator U which has eigenstate $|u\rangle$ and corresponding eigenvalue $e^{i\varphi}$, where $0 \leq \varphi < 2\pi$. In other words, $U|u\rangle = e^{i\varphi}|u\rangle$, meaning that U applied to $|u\rangle$ changes only the overall phase. The goal of quantum phase estimation is to estimate the phase φ, given many copies of the state $|u\rangle$. In practice, this could be used to estimate the energy of a given eigenstate. For example, suppose we are given a prepared state $|u\rangle$ which is the ground state of some Hamiltonian H with energy E_g, i.e., $H|u\rangle = E_g|u\rangle$. Then, letting $U = e^{-iH\tau}$ be the unitary time-evolution operator under H for time τ (in units such that $\hbar = 1$), we have

$$U|u\rangle = e^{-iH\tau}|u\rangle = e^{-iE_g\tau}|u\rangle. \tag{2.30}$$

So the effect of U is to apply a phase factor $e^{i\varphi}$, where $\varphi = -E_g\tau$. By applying quantum phase estimation to estimate φ, then, we achieve an estimate of the ground state energy

$$E_g = \frac{-\varphi}{\tau}, \tag{2.31}$$

assuming that τ is chosen to be small enough such that $|E_g\tau| \ll \pi$.

The circuit for the quantum phase estimation algorithm is displayed in Figure 2.3. It uses n computational qubits in addition to the provided input state $|u\rangle$, and a layer of Hadamard gates initializes each of the n computational qubits into the $|+\rangle$ state. This is followed by a sequence of controlled unitaries. The key thing to notice is that each of these controlled-U^{2^j} gates (which we abbreviate as cU^{2^j}) provides a *phase kickback*[2] that depends on the value of φ and the number of applications 2^j. To see how this happens, consider the action of the first controlled gate, where $j = 0$ and so we perform U only once:

$$\frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)|u\rangle \xrightarrow{cU^{2^0}} \frac{1}{\sqrt{2}}|0\rangle|u\rangle + e^{i\varphi}|1\rangle|u\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle + e^{i\varphi}|1\rangle\Big)|u\rangle \tag{2.32}$$

The controlled unitary acts only on the part of the state where the control qubit is $|1\rangle$, multiplying it by the eigenvalue $e^{i\varphi}$. Because it leaves the state $|u\rangle$ of the target qubit

[2] A phase kickback is a common trick used in quantum algorithms. It refers to a controlled operation which leaves the state of the target qubit(s) unchanged, but changes the relative phase between the parts of the superposition of the control qubit(s) when expressed in the computational basis.

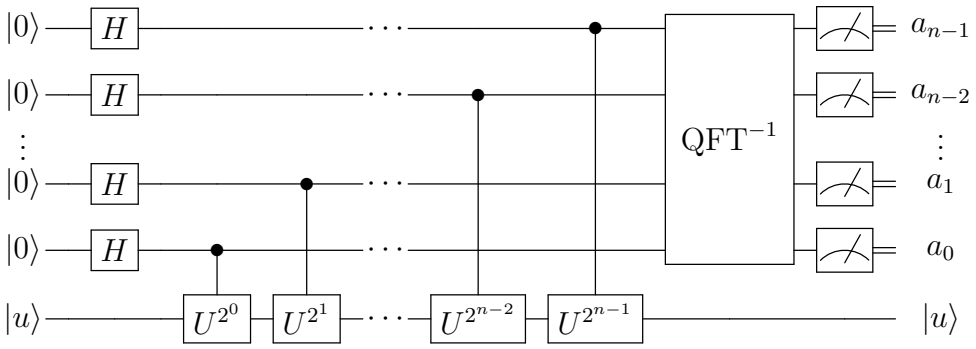


Figure 2.3: Circuit diagram for quantum phase estimation algorithm. The lower qubit (called the "input qubit") is initialized in the state $|u\rangle$, which is an eigenstate of the unitary U. The remaining n qubits (called the "computational qubits") are initialized in the state $|0\rangle$. Hadamard gates are applied to each of the computational qubits, followed by n controlled applications of the unitary U^{2^j}, where each of the computational qubits is used as a control qubit, and the input qubit is used as the target qubit. An inverse QFT operation is then applied to the computational qubits, followed by a computational basis measurement of each computational qubit. The measurements produce the n-bit result $a = a_{n-1}a_{n-2}\cdots a_1a_0$. With high probability, this allows us to calculate the desired phase φ.

unchanged, we see that we have essentially performed a phase kickback, and the phase φ now appears as a relative phase between the superposition terms of the control qubit.

In general, at each controlled gate we perform 2^j controlled repetitions of U, which has the effect of simply applying the phase kickback 2^j times:

$$\frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big)|u\rangle \xrightarrow{cU^{2^j}} \frac{1}{\sqrt{2}}\Big(|0\rangle + e^{i2^j\varphi}|1\rangle\Big)|u\rangle \tag{2.33}$$

After applying the controlled gates, an inverse quantum Fourier transform [29], denoted as QFT^{-1}, is applied to the n computational qubits. The QFT is a quantum analog of the traditional discrete Fourier transform, which is commonly used in classical signal processing to translate between the time domain and the frequency domain. The QFT is defined as the unitary transformation which maps computational basis states as

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle, \tag{2.34}$$

where j and k index the computational basis states, $N = 2^n$ is the dimension of the state space, and $\omega_N^{jk} = e^{2\pi ijk/N}$. In this circuit, the inverse QFT is essentially used to transform the relative phases acquired from the series of controlled operations into amplitudes, which

can then be read out by a measurement in the computational basis. We note that we can write the desired phase φ as

$$\varphi = 2\pi \left(\frac{a}{2^n} + \delta \right) \tag{2.35}$$

for some n-bit integer a, and some residual $0 \leq \delta \leq \frac{1}{2^{n+1}}$. After applying the inverse QFT, the n-bit outcome of the computational basis measurement gives us $a = a_{n-1}a_{n-2}\cdots a_1 a_0$ with high probability. So by repeating this circuit many times and taking the most frequent outcome a, we can extract the best n-bit binary approximation of φ:

$$\varphi \approx 2\pi \frac{a}{2^n}. \tag{2.36}$$

Although we have expressed the quantum phase estimation algorithm in circuit form, it is not necessarily represented in a way that is simple to implement on a quantum computer. In particular, a quantum computer typically implements some fixed set of one-qubit and two-qubit gates, while this algorithm is expressed in terms of a unitary U and controlled unitaries cU^{2^j} that may be operations acting on many qubits, in addition to the n-qubit inverse QFT operation. The problem of translating arbitrary unitaries into a sequence of gates that can be directly implemented by a quantum computer is known as *quantum compilation*, which is the subject of Chapter 3.

Variational quantum algorithms

Although traditional quantum algorithms are extremely powerful, the required qubit count (circuit width) and instruction count (circuit depth) are typically far too large for current hardware to implement reliably. Current quantum computers, which are often referred to as noisy intermediate-scale quantum (NISQ) devices [125], have significant error rates which limit the width and depth at which we can expect to obtain meaningful results.

One potential family of techniques for extracting utility from NISQ devices is known as *variational quantum algorithms* (VQAs). The general approach of a VQA is to combine classical computational power with repeated executions of shallow, parameterized quantum circuits in order to find the parameter values which optimize the expectation value of a particular observable. The optimal parameter values encode the solution to some problem of interest. One textbook example of a VQA is known as the *variational quantum eigensolver* [122], or VQE, which can be used to estimate molecular ground state energies. Another frequently-cited VQA is known as the *quantum approximate optimization algorithm* [44], or QAOA, which can be used to solve combinatorial optimization problems. For additional examples, the reader is encouraged to consult [22] for a comprehensive review of VQAs and their applications.

A general schematic of the VQA approach is shown in Figure 2.4. The quantum circuit implements some parameterized unitary $U(\vec{\boldsymbol{\theta}})$, where $\vec{\boldsymbol{\theta}}$ is a vector of parameters which can be modified for each execution of the circuit. The input state in this case is chosen to be an equal superposition over all basis states $|+\rangle|+\rangle\cdots|+\rangle$, but in general this can be any

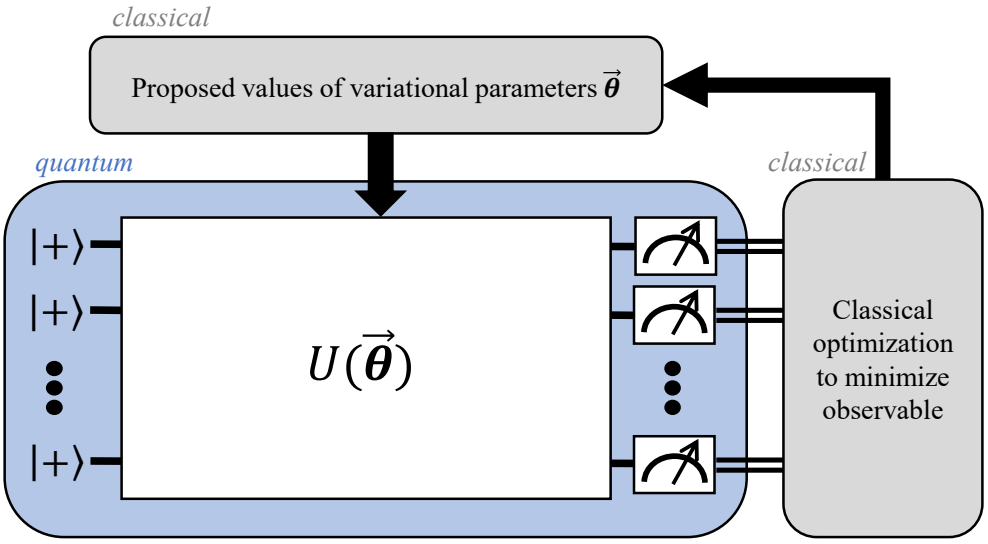


Figure 2.4: Schematic of a typical variational quantum algorithm implementation, including the corresponding classical optimization loop. In this example, each qubit is initialized into the state $|+\rangle$, a parameterized unitary $U(\vec{\boldsymbol{\theta}})$ is applied, and each qubit is then measured in the computational basis in order to calculate some observable of interest. This procedure is repeated as part of a classical optimization loop over the parameters $\vec{\boldsymbol{\theta}}$ in order to find the parameter values which minimize the observable.

desired state. After executing the unitary $U(\vec{\boldsymbol{\theta}})$, the observable of interest is measured. In the figure this is depicted as a computational basis measurement, but in general it may involve measurements in one or more arbitrary bases, where the circuit must be run once for each required basis measurement. In addition, because we are interested in the expectation value of the observable, the observable measurement is repeated many times. Then, the average measured value of the observable for the given parameters $\vec{\boldsymbol{\theta}}$ is treated as the value of the objective function at those parameters and is passed to a classical optimizer, which combines it with results from previous iterations to predict a new set of parameters $\vec{\boldsymbol{\theta}}_{\text{new}}$ which minimize the objective function. This $\vec{\boldsymbol{\theta}}_{\text{new}}$ is then used as the circuit parameterization $\vec{\boldsymbol{\theta}}$ for the next iteration. The process is repeated until the classical optimizer has achieved some desired level of convergence, or until some maximum number of iterations has been reached.

Note that in order for the VQA optimization to successfully converge, the error in the objective function measurement must be small enough that the variations in the underlying objective function landscape can still be detected. This means that the unitary $U(\vec{\boldsymbol{\theta}})$ must be implemented on the quantum device with reasonably high accuracy, and also that enough repetitions (or *shots*) of the observable measurement must be taken such that the statistical noise associated with averaging over finite measurements does not drown out the variations

in the landscape of the true expectation value. Because of these restrictions, the efficiency of VQAs is highly dependent on the choice of the classical optimizer, since standard techniques such as gradient descent often perform very poorly when applied to noisy objective functions. This topic is explored in further detail in Chapter 6.

2.4 Errors

Types of errors

Quantum computers are physical devices which are subject to error from a variety of sources. Classical computers, of course, are also physical devices subject to error, but error rates of modern transistors are so remarkably low that they are essentially negligible. In contrast, current quantum computers experience error rates on the order of 10^{-4} to 10^{-3} for single-qubit gates, and on the order of 10^{-3} to 10^{-2} for two-qubit gates. Error rates at these levels significantly limit the depth of circuits that we can hope to implement reliably.

To understand these errors in more detail, we begin by formalizing notation. Given a quantum system initially in state $|\psi_i\rangle$, the final state after applying some unitary operator U should be $|\psi_f\rangle = U|\psi_i\rangle$. Considering that an arbitrary operation on a quantum state may be non-unitary, we generalize to use density matrix formalism by letting $\rho_i = |\psi_i\rangle\langle\psi_i|$ and $\rho_f = |\psi_f\rangle\langle\psi_f|$, such that $\rho_f = U\rho_i U^\dagger$ is the ideal final state after U is applied.

Two common types of error (among others) are coherent errors and depolarizing errors. *Coherent errors* occur when an incorrect unitary operator is applied to the system. That is, instead of applying unitary U, we instead apply some unitary

$$V = \mathcal{E}U, \tag{2.37}$$

where $\mathcal{E}$ represents the unitary error. For example, if U is a rotation around some axis, then $\mathcal{E}$ may represent an error in the rotation angle (i.e., over- or under-rotation), an error in the rotation axis, or both. If the system experiences this type of coherent error, the resulting state is then

$$V\rho_i V^\dagger = \mathcal{E}U\rho_i U^\dagger \mathcal{E}^\dagger = \mathcal{E}\rho_f \mathcal{E}^\dagger. \tag{2.38}$$

We note that if $\mathcal{E}$ can be determined precisely, then its effect could be reversed simply by applying its inverse to the system.

In contrast, *depolarizing errors* occur when the system, in addition to the intended unitary evolution, undergoes non-unitary evolution which reduces the purity of the state. Depolarization refers to a process in which a state is mapped through a channel Δ_λ onto a linear combination of itself and the maximally mixed state as

$$\Delta_\lambda(\rho) = (1-\lambda)\rho + \frac{\lambda}{N} I, \tag{2.39}$$

where ρ is the system state prior to the channel, $\lambda \in [0, 1]$ is a parameter specifying the fraction to which the state is depolarized, and $N = 2^n$ is the dimension of the system. If

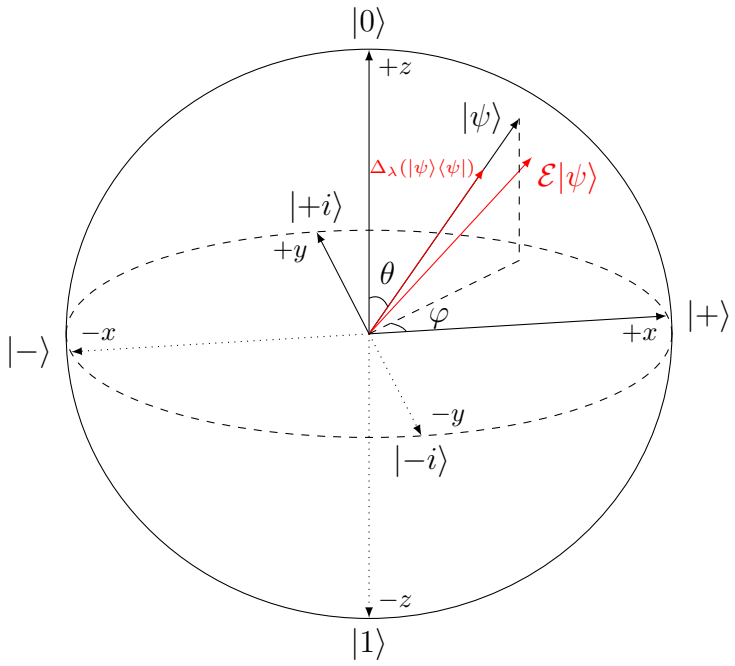


Figure 2.5: Bloch sphere illustration of the effect of coherent and depolarizing errors on a single-qubit state $|\psi\rangle$. The pure state $\mathcal{E}|\psi\rangle$ is on the surface of the sphere and represents the effect of a coherent error $\mathcal{E}$ on the state $|\psi\rangle$. The mixed state $\Delta_\lambda(|\psi\rangle\langle\psi|)$ is *not* on the surface of the sphere and represents the effect of a depolarizing error channel Δ_λ on the state $|\psi\rangle$.

a system experiences purely depolarizing error while executing a particular unitary U, the resulting state is then

$$\Delta_\lambda(U\rho_i U^\dagger) = (1-\lambda)U\rho_i U^\dagger + \frac{\lambda}{N}I = (1-\lambda)\rho_f + \frac{\lambda}{N}I. \tag{2.40}$$

We note that since depolarizing error is non-unitary, it cannot be reversed.

In the case of a single-qubit system, coherent errors and depolarizing errors can be helpfully visualized using the Bloch sphere picture, as shown in Figure 2.5. In particular, a coherent error is a unitary operation which keeps the state on the surface of the Bloch sphere, i.e., takes a pure state $|\psi\rangle$ to another pure state $\mathcal{E}|\psi\rangle$. Conversely, a depolarizing error is a non-unitary operation which pushes the state toward the center of the Bloch sphere, i.e., takes the pure state $|\psi\rangle$ to a mixed state $\Delta_\lambda(|\psi\rangle\langle\psi|)$.

Characterizing errors

In order to characterize errors in quantum information processing, a family of tomographic techniques have been developed, including *quantum state tomography* [156, 30], *quantum pro-*

cess tomography [110], and *gate set tomography* [12, 117]. Tomographic techniques produce a complete characterization of a quantum operation, which provides a detailed mathematical description of the errors present in the system. However, tomography is extremely resource-intensive, and although techniques exist to improve its scalability somewhat [134, 95], its cost still typically scales exponentially with qubit count.

Because of the high cost of tomography, a simpler approach to characterizing error is desirable. One common way to measure the success of a quantum operation is to measure the *fidelity* of the resulting state [82], which is defined as the distance between the ideal state ρ and the actual state ρ' as follows:

$$F(\rho, \rho') = \left(\mathrm{Tr} \sqrt{\sqrt{\rho}\, \rho' \sqrt{\rho}} \right)^2 . \tag{2.41}$$

We note that $F(\rho, \rho')$ is symmetric in the ordering of the input arguments, such that $F(\rho, \rho') = F(\rho', \rho)$. In addition, if $\rho = |\psi\rangle\langle\psi|$ and $\rho' = |\psi'\rangle\langle\psi'|$ are pure states, then the fidelity can be expressed as a function of the pure state vectors:

$$F(|\psi\rangle, |\psi'\rangle) = |\langle\psi|\psi'\rangle|^2 . \tag{2.42}$$

From this definition it is clear that the fidelity is a measure of overlap between $|\psi\rangle$ and $|\psi'\rangle$, such that $F(|\psi\rangle, |\psi'\rangle) = 1$ if the states are identical, and $F(|\psi\rangle, |\psi'\rangle) = 0$ if the states are orthogonal.

As an example, consider the fidelity of a state which has been depolarized according to Equation 2.39, such that $\rho' = (1 - \lambda)\rho + \frac{\lambda}{N} I$. Substituting this into Equation 2.41, we can obtain the following formula for the fidelity of a depolarized state:

$$F_\lambda(\rho, \rho') = 1 - \lambda \frac{N - 1}{N}. \tag{2.43}$$

So the fidelity of a fully-depolarized state (i.e., $\lambda = 1$) is therefore $F_\lambda(\rho, \rho') = \frac{1}{N}$.

Output state fidelity is a simple metric that can be used to quantify the success of a quantum operation, and methods to efficiently approximate this state fidelity are therefore very useful in characterizing the overall reliability of a quantum device. Such methods are often referred to as *benchmarking* techniques, since they can provide a device-independent measure of reliability. The textbook example of a benchmarking technique is known as *randomized benchmarking* (RB) [40, 91], which is based on randomly-generated sequences of Clifford operations and can be used to estimate an average error rate per gate. We discuss RB and other benchmarking protocols in more detail in Chapter 4, including techniques for verifying the behavior of both fixed gate sets and continuously-parameterized gate sets. In addition, Chapter 5 discusses the application of benchmarking-like techniques to verify the correct operation of analog quantum simulators.

Quantum projection noise

Even in the absence of any physical errors, the probabilistic nature of quantum measurement results in some fundamental uncertainty when estimating an expectation value with a finite

number of shots. The error resulting from this uncertainty, which is often referred to as *quantum projection noise* [76], can be understood by modeling a quantum measurement as a sample from an underlying multinomial probability distribution.

To do so, we let $P(x) = |\langle x|\psi\rangle|^2$ represent the probability of obtaining result x when measuring the n-qubit state $|\psi\rangle$ in the computational basis, where $x \in \{0, 1, \ldots, N-1\}$ and $N = 2^n$. We then define a random variable $\mathbf{P_x} \sim \text{Multinomial}(K, N, P(x))$ representing the number of times outcome $\mathbf{x}$ is observed when taking K independent measurements of the state $|\psi\rangle$. This means that the sample probability of observing outcome $\mathbf{x}$ is $\frac{1}{K}\mathbf{P_x}$. Using the properties of variance and the multinomial distribution, then, the variance of the sample probability $\frac{1}{K}\mathbf{P_x}$ is

$$\text{Var}\left[\frac{1}{K}\mathbf{P_x}\right] = \frac{1}{K^2}\text{Var}\left[\mathbf{P_x}\right] = \frac{1}{K^2}\left(KP(x)\big[1 - P(x)\big]\right) = \frac{1}{K}P(x)\left[1 - P(x)\right]. \tag{2.44}$$

Accordingly, the standard deviation σ of the sample probability of observing outcome x is the square root of the variance, which gives

$$\sigma = \frac{1}{\sqrt{K}}\sqrt{P(x)}\sqrt{1 - P(x)}. \tag{2.45}$$

We refer to σ as the "projection noise" associated with estimating $P(x)$ using K shots.

We first note that the projection noise scales as the inverse square root of the number of shots K. Thus, reducing σ by a factor of two requires increasing K by a factor of four.

We also observe that the projection noise for estimating $P(x)$ is minimized when $P(x) = 0$ or $P(x) = 1$, and it is maximized when $P(x) = \frac{1}{2}$. Intuitively, this means that the impact of projection noise when estimating an output probability distribution is minimized when the state $|\psi\rangle$ being measured is close to a basis state, and in fact vanishes if $|\psi\rangle$ is exactly a basis state.

Quantum projection noise is sometimes neglected in discussions of noise in quantum computers, since it can always be reduced to an arbitrarily small value by simply increasing the number of shots. In reality, however, the number of shots cannot be increased arbitrarily. For one, there are often practical limits; it may not be feasible to wait days or weeks for a result, and so the number of shots is bounded by the overall execution time of an algorithm. But even more fundamentally, there is a limitation due to the fact that the behavior of the physical quantum device is changing with time. Increasing the number of shots can provide a more precise estimate of the expectation value only if the same state is being generated each time. But if the underlying system parameters are changing significantly while calculating an expectation value, then the result can no longer be made arbitrarily precise, but instead has some inherent uncertainty due to the fluctuations in the system.

Because of this, techniques that minimize the effect of quantum projection noise, and therefore minimize the number of shots required to obtain a result with the desired precision, can have significant advantages. We discuss examples of this in Chapter 4, where we describe a verification technique which gains an efficiency advantage in part by minimizing quantum

projection noise, and in Chapter 6, where we describe an optimization technique for VQAs which uses statistical methods to smooth out quantum projection noise when estimating the underlying objective function.

2.5 Summary

In this chapter, we have provided an overview of the fundamental concepts of quantum computation, including qubits, gates, circuits, and algorithms. We have also introduced variational quantum algorithms (VQAs) and motivated their potential usefulness. In addition, we have given a very brief introduction to the study and characterization of errors in quantum computation. In the following chapters, we will build on these foundations to develop protocols for verification of both gate-based quantum computers and analog quantum simulators, as well as an efficient classical optimization technique for VQAs. But first, we develop a general technique for stochastic compilation of quantum circuits, which is the subject of the next chapter.

Chapter 3

Compilation of quantum circuits

Portions of this chapter were first released as part of an arXiv preprint [140] and are reproduced here with permission of the authors.

3.1 Introduction

A critical prerequisite to executing any algorithm on a physical quantum computer is the process commonly known as quantum compilation. Traditional compilation, both in the classical and quantum realms, is most often a deterministic process, using rules and heuristics to efficiently synthesize a desired program from the native assembly instructions (in classical compilation) or native physical gates (in quantum compilation). In some cases, adding stochasticity to the compilation process has been shown to produce advantages in the resulting program. In classical compilation, a technique known as stochastic superoptimization [131] has been shown in certain cases to produce significantly shorter programs than the best-in-class compilers and optimizers. In quantum compilation, techniques such as randomized compiling [155] have been demonstrated to improve noise resilience by depolarizing errors that occur during program execution. In this chapter, we provide an overview of existing quantum compilation techniques, and we introduce and analyze a novel stochastic procedure for quantum compilation. We will use this procedure as part of the verification techniques introduced in Chapter 4 and Chapter 5.

This chapter is organized as follows. Section 3.2 provides a brief introduction to general methods for quantum compilation which can be applied to all types of quantum circuits. In Section 3.3, we provide an overview of various product formula approaches to quantum compilation, which are approximate compilation techniques that work especially well for Hamiltonian simulation problems. Next, Section 3.4 introduces STOQ, a novel stochastic approximate quantum unitary compilation scheme. We provide examples of using STOQ to compile approximate gate sequences for time-evolution unitaries in Section 3.5 and for randomly-generated unitaries in Section 3.6. In Section 3.7, we discuss advantages and limitations of STOQ, and we discuss avenues for future work and improvements to the

technique. Finally, we summarize the chapter in Section 3.8.

3.2 General methods for quantum compilation

Definitions

The primary task of quantum compilation is the conversion of a *target unitary* into a sequence of quantum gates that are native to the physical device being used [8, 31, 66, 80]. The target unitary is the 2^n-dimensional unitary operator U implementing some desired effect on the n-qubit system.

The set of gates used for the compilation may be fixed or parameterized. *Fixed gates*, such as Cliffords, are discrete operations that can be represented as a fixed unitary matrix. In contrast, *parameterized gates*, such as rotations, are continuous operations that can be represented as a unitary matrix with one or more continuously-variable parameters.

The allowed set of gates for the compilation may then consist of some combination of fixed and parameterized gates. For an n-qubit system, the *instruction set* (often called *native gate set*) is a set of fixed gates and/or parameterized gates that represent the fundamental set of operations that can be physically applied to the system.

Solovay-Kitaev algorithm

One of the earliest quantum compilation techniques, the Solovay-Kitaev algorithm [88, 89], is an approximate technique which compiles gate sequences that differ from the target unitary by an amount that can be made as small as desired. We provide here a rough sketch of this technique; a full description of the implementation and performance of this algorithm is available in [35]. Although more efficient algorithms exist for certain specialized compilation problems, such as compilation with a restricted instruction set or with a restricted set of target unitaries, the Solovay-Kitaev algorithm remains the most efficient known technique for the general case.

The Solovay-Kitaev algorithm first relies on a preprocessing stage which generates all possible n-qubit sequences up to sufficient length such that any n-qubit unitary U can be approximated by one of the sequences to within a basic approximation error ϵ_0. We denote the product of this basic approximation sequence as U_0. The error in the approximation can be expressed as another unitary operator Δ such that

$$U = \Delta U_0. \tag{3.1}$$

The approximation error Δ can then be decomposed into a balanced group commutator as

$$\Delta = VWV^\dagger W^\dagger. \tag{3.2}$$

We then find basic approximation sequences for V and W using the technique described above. This procedure can be repeated recursively as many times as desired to make the

approximation error as small as desired. It can be shown [35] that an arbitrary unitary can be approximated to any accuracy within $\epsilon > 0$ with runtime and resulting sequence length which are both polylogarithmic in $1/\epsilon$.

Although this technique is asymptotically efficient, in practice the preprocessing step to build the basic approximation sequences is computationally feasible only for very small systems and quickly becomes intractable beyond $n = 2$. As a consequence, Solovay-Kitaev is considered a useful technique for compilation of one-qubit and two-qubit unitaries, but is unlikely to be useful at larger n.

Exact compilation techniques

Because unitary operators belong to a continuous space, quantum compilation in general results in gate sequences which are only approximately equivalent to the target unitary, as with the Solovay-Kitaev algorithm. However, if the set of gates is restricted to a discrete set, and the set of allowable target unitaries is limited to those which can be generated by finite sequences of those gates, then exact compilation is possible. This is often the case, for example, when attempting to find a more efficient compilation of a program which is already specified in terms of a sequence of gates.

Many such compilation techniques involve problems such as qubit mapping and gate ordering, with the goal of minimizing the gate count and optimizing circuit performance on a device with a particular native gate set and given noise characteristics [25]. These procedures are often heuristic-based or rule-based, although methods based on artificial intelligence algorithms have also been explored [94]. Although these techniques are often labeled as compilation techniques, it is more precise to consider these as methods to *transform* (or *transpile*) an existing quantum circuit into another equivalent circuit. This is distinct from the problem of *synthesis*, which is the task of compiling a quantum circuit to implement a unitary for which no circuit is initially provided.

3.3 Product formula approaches for approximate quantum compilation

Suzuki-Trotter decomposition

In the field of quantum compilation, special attention has been paid to compilation of unitaries which result from the time evolution of physically-realizable Hamiltonians. The compiled sequences in these cases can be executed to perform what is known as *Hamiltonian simulation*, or more broadly, *quantum simulation*. Such approaches are of special interest in fields such as quantum chemistry, where it is desirable to use a quantum computer to simulate the dynamics of physical systems. Common approaches to this problem include product formula techniques such as the Suzuki-Trotter decomposition [148] and qubitization

[100], which deterministically compile the time-evolution unitary for a given Hamiltonian into a sequence of quantum gates.

A given Hamiltonian is often expressed as a sum of individual terms,

$$H = \sum_{k=1}^{m} H_k, \tag{3.3}$$

where each H_k is a local interaction that can often be directly implemented on a physical device. The target unitary of a quantum simulation, then, is the unitary $U(t)$ representing the time evolution of this Hamiltonian for some time t,

$$U(t) = e^{-iHt} = e^{-i\left(\sum_{k=1}^{m} H_k\right)t}, \tag{3.4}$$

where we work in units such that $\hbar = 1$. If each H_k is physically realizable, then implementing a unitary of type $e^{-iH_k t}$ is straightforward, since it simply requires applying the H_k interaction for time t. But it is not always possible to implement all H_k interactions simultaneously. For example, some of the H_k terms may be physically implemented as two-qubit interactions between neighboring qubits, and on many platforms it is not feasible to run multiple such interactions simultaneously.

To account for this limitation, we consider the most restrictive case where we can implement only one H_k term at a time. If all of the H_k terms commute, then implementing the full unitary $U(t)$ is trivial, since we can use the relation $e^{-i\left(\sum_{k=1}^{m} H_k\right)t} = \prod_{k=1}^{m} e^{-iH_k t}$ and simply apply each of the H_k interactions for time t (in any order). However, the H_k terms in general do not commute, and so a different approach is needed.

The goal of product formula approaches is to decompose the exponential of the sum of non-commuting operators into a product of exponentials of the individual operators. Product formula techniques for compilation therefore rely on the Trotter product formula, which states that the exponential of the sum of operators A and B can be written as

$$e^{A+B} = \lim_{n\to\infty} \left(e^{A/n} e^{B/n}\right)^n, \tag{3.5}$$

which holds even when A and B do not commute. When applied to our target unitary $U(t)$, this formula becomes

$$U(t) = e^{-iHt} = e^{-i\left(\sum_{k=1}^{m} H_k\right)t} = \lim_{n\to\infty} \left(\prod_{k=1}^{m} e^{-iH_k t/n}\right)^n. \tag{3.6}$$

This is a product of exponentials of individual H_k terms, as desired, but the limit $n \to \infty$ means that implementing it requires an infinitely-long sequence of infinitely short (duration t/n) interactions, which is impractical. Although the infinite limit would be required for an exact compilation, it fortunately turns out that the infinite limit is not necessary to achieve

a good approximation. In fact, if we choose to split the time evolution into a finite number of steps r of duration t/r, it can be shown [69] that

$$U(t) = e^{-iHt} = e^{-i\left(\sum_{k=1}^{m} H_k\right)t} = \left(\prod_{k=1}^{m} e^{-iH_k t/r}\right)^r + O\left(\frac{m^2 t^2}{r}\right), \tag{3.7}$$

where the final term represents the approximation error incurred by reducing the infinite product to a finite one. Thus, in order to limit the approximation error to some finite $\epsilon > 0$, the number of steps r must be chosen such that $r \sim m^2 t^2/\epsilon$. The formula in Equation 3.7 is often called a *first-order Suzuki-Trotter decomposition* or *first-order Trotterization* of the time evolution operator $U(t)$. Higher-order decompositions also exist which produce more accurate approximations [69], but the first-order approximation can often be made good enough in practice by choosing a sufficiently large r.

Stochastic product formula approaches

Basic product formula techniques as laid out in the previous section produce deterministic compilations, but approaches involving stochasticity have been shown to be advantageous in some cases. Adding randomization to the Suzuki-Trotter decomposition [23], in which the application order of the $e^{-iH_k t/r}$ terms is randomized for each of the r steps, creates approximate compilations that are better both theoretically and empirically.

A stochastic compilation protocol known as QDRIFT [20], where gate probabilities are weighted according to the strength of each term in the Hamiltonian rather than using a product formula directly, has been shown to produce much more efficient compilations in many cases. QDRIFT begins by decomposing each of the H_k terms in the Hamiltonian into a coefficient h_k and an operator $\hat{H}_k$ such that

$$H_k = h_k \hat{H}_k, \tag{3.8}$$

where $\hat{H}_k$ is normalized such that its largest singular value is 1. We define the sum of the coefficients as $\lambda = \sum_k h_k$. The compilation is then generated by constructing a probability distribution over these terms such that

$$P(k) = \frac{h_k}{\lambda} \tag{3.9}$$

is the probability of drawing term $\hat{H}_k$. A sequence of N terms $\{\hat{H}^{(1)}, \hat{H}^{(2)}, \ldots, \hat{H}^{(N)}\}$ is drawn from this distribution, and each term is added to the compilation as a time evolution for duration $\lambda t/N$. The full QDRIFT compilation of the time evolution unitary $U(t)$ can then be expressed as

$$U(t) = e^{-iHt} = \prod_{j=1}^{N} e^{-i\hat{H}^{(j)}\lambda t/N} + O\left(\frac{\lambda^2 t^2}{N}\right), \tag{3.10}$$

where the final term represents an upper bound on the approximation error of the compilation. We observe that in this procedure, the "stronger" terms in the Hamiltonian are more likely to be drawn at each step (and the "weaker" terms less likely), but the "strength" of each resulting step is equivalent due to the normalization of the $\hat{H}_k$ terms.

An interpolation of the randomized Suzuki-Trotter and QDRIFT methods [121] has also been proposed. This technique samples from stochastically-sparsified versions of the full Hamiltonian (rather than sampling from the individual Hamiltonian terms as in QDRIFT), and then a randomized Suzuki-Trotter decomposition is applied to each of the sampled Hamiltonians.

We note that when applied to problems involving sparse Hamiltonians, the efficiency of product formula compilation techniques (measured both by compilation runtime and resulting sequence length) is generally independent of system size, and rather depends primarily on the number and relative strengths of terms present in the Hamiltonian.

3.4 Stochastic search for approximate quantum compilation

Motivation

We now introduce STOQ, a stochastic protocol for approximate quantum unitary compilation. We note that this protocol generalizes a similar technique used for variational quantum compilation algorithms [87, 142].

The motivation for STOQ is as follows. For the most general problem of unitary compilation – that is, decomposition of an arbitrary unitary into a sequence of instructions from an arbitrary instruction set – efficient methods are not known for systems larger than those that can be handled by the Solovay-Kitaev algorithm (typically up to two qubits; see Section 3.2). The Solovay-Kitaev algorithm produces compilations which are ϵ-approximate, where ϵ is the error in compilation and can provably be made as small as desired. However, if an application does not require a guarantee on the closeness of the approximation (as is the case in the randomized analog verification protocol described in Chapter 4 and Chapter 5), then we can relax the requirement that ϵ can be made arbitrarily small. This opens the door for a protocol such as STOQ, which can produce compilations for systems of larger size (up to around ten qubits), but with significantly higher error than compilations that would be obtained from a technique such as Solovay-Kitaev.

Definitions

We begin with the concepts of *target unitary*, *fixed gates*, and *parameterized gates* as defined in Section 3.2, and we will slightly generalize the definition of *instruction set*. For some applications, it may be desirable that gates occur in the sequence in specific patterns, which we refer to as *layers*. In this case, we can define the instruction set in terms of these layers

of fixed and/or parameterized native gates, such that the resulting compilation will be a sequence of layers. Each layer consists of a fixed number of each type of native gate, where each gate is assigned a randomly-chosen parameter value (within some allowed parameter range) and the order of the gates within the layer is also randomly chosen. In the following description of the STOQ algorithm, we refer to the components of an instruction set as *instructions*, where each instruction can be either a single gate or a layer, depending on the definition of the instruction set for the given problem.

Given a target unitary U and an instruction set G, then, the goal of an *approximate compilation* protocol is to find a sequence of instructions $\{G_1, \ldots, G_M\}$ such that the product $G_M G_{M-1} \cdots G_1$ is as close as possible to U for some reasonable choice of distance metric. Note that this definition does not require any particular closeness of approximation, but it does require that the quality of approximation can be measured. That is, given some appropriate distance metric which defines a distance d between the sequence product $G_M G_{M-1} \cdots G_1$ and the target unitary U, an approximate compilation procedure treats d as the value of a cost function to be minimized.

Procedure

The STOQ protocol for approximate compilation proceeds according to the pseudocode displayed in Figure 3.1. The procedure is also depicted as a flowchart in Figure 3.2. Intuitively, the STOQ algorithm can be thought of as a randomized exploration of the full space of possible n-qubit unitary operators (or the subspace that can be generated by the instruction set G, if G is not universal), using a technique known as Markov chain Monte Carlo (MCMC) search [68]. The algorithm is always initialized with an empty sequence, meaning that it always starts from the identity operator in the search space. At each iteration, a random step is proposed, in which an item is either added to or removed from the sequence. If this step brings the product of the sequence closer to the target unitary as determined by the cost function, it is accepted; otherwise, it is either accepted or rejected with some probability, where the probability of accepting such "bad" steps decreases with each iteration. The algorithm continues until some maximum number of iterations is reached, at which point the final cost can be evaluated and the sequence either kept or discarded, depending on the accuracy requirements of the given problem.

One critical component of the algorithm is the choice of an appropriate and efficiently-computable cost function. Naturally, the cost function should be a distance measure between the the target unitary U and the unitary V which is the product of the currently-compiled sequence. One commonly-used and operationally-relevant choice, used also in variational quantum compilation approaches [87, 142], is the *Hilbert-Schmidt distance*

$$D_{\mathrm{HS}}(U, V) = \left|\mathrm{Tr}(V^\dagger U)\right|, \tag{3.11}$$

which is related to the fidelity of a process [118] and can be shown in certain cases to be

```
function StochasticCompilation(params U, G, num_iterations):
  sequence := []
  beta := 0
  cost := Cost(U, Product(sequence))
  for i in 1 to num_iterations:
    beta := IncreaseBeta(beta)
    new_sequence := RandomChange(sequence, G)
    new_cost := Cost(U, Product(new_sequence))
    if Accept(cost, new_cost, beta):
      sequence := new_sequence
      cost := new_cost
  return sequence
```

Figure 3.1: Pseudocode for STOQ stochastic compilation algorithm. The inputs to the algorithm are the target unitary `U`, the parameterized instruction set `G`, and the number of iterations to perform `num_iterations`.

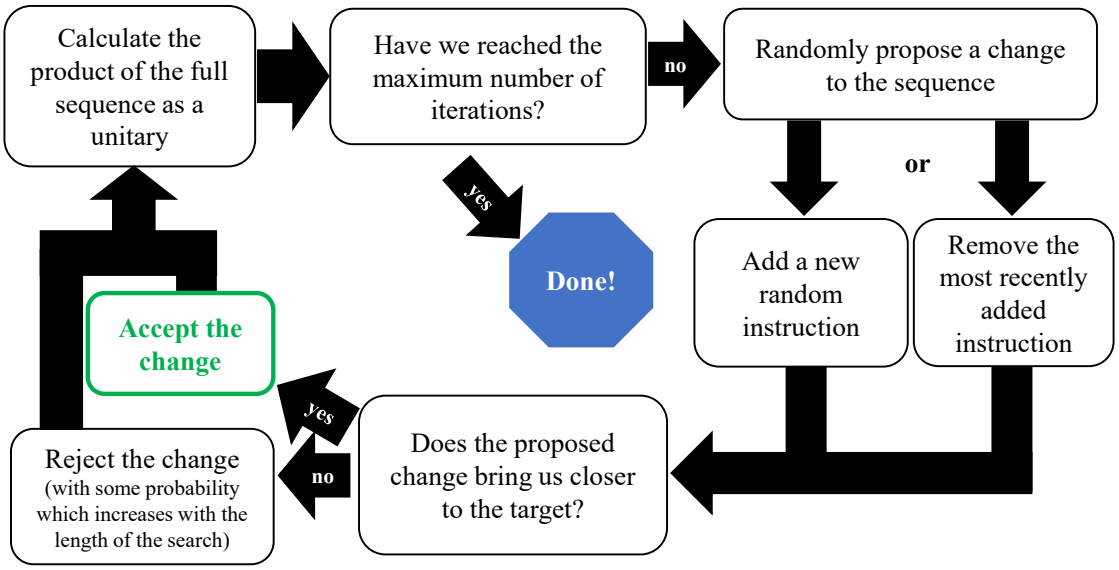


Figure 3.2: Flowchart for STOQ stochastic compilation algorithm. Corresponds to pseudocode in Figure 3.1.

closely related to the trace distance [28]. For these reasons, we use the cost function

$$\texttt{Cost}(U,V) = 1 - \frac{1}{2^n} D_{\mathrm{HS}}(U,V), \tag{3.12}$$

noting that $\texttt{Cost}(U,V)$ ranges from 0 to 1 and vanishes if and only if U and V are equivalent up to a global phase.

Implementation details

Here we provide a few additional implementation details for the pseudocode representation of STOQ provided in Figure 3.1.

The compiled sequence is stored in the `sequence` variable, which is initially empty. The `RandomChange` function returns a modified sequence on each iteration, either by adding a randomly-drawn instruction to the sequence from the parameterized instruction set `G` with randomly-generated parameter values, or by removing an instruction from the sequence. The `Product` function calculates the unitary that represents the product of all of the operations in the sequence, and the `Cost` function is implemented as described in Equation 3.12.

The variable `beta` is used as an annealing parameter for the compilation process. The function `IncreaseBeta` returns a slightly increased value of `beta` on each iteration. Defining the annealing parameter as $\beta = \texttt{beta}$ and the cost difference of such a proposed change as $\Delta = \texttt{new\_cost} - \texttt{cost}$, the `Accept` function calculates the probability of accepting a proposed change as

$$P_{\mathrm{accept}} = \begin{cases} e^{-\beta\Delta} & \Delta > 0 \\ 1 & \Delta \leq 0. \end{cases} \tag{3.13}$$

The probability of accepting "bad" proposed changes where the cost increases (i.e., where $\Delta > 0$) approaches zero as β increases.

A reference implementation of the STOQ compilation protocol, which was used to generate the results in this chapter, is freely available [136].

3.5 Stochastic compilation of time-evolution unitaries

To demonstrate one possible (although not necessarily useful) application of STOQ, we choose an Ising-type Hamiltonian with nearest-neighbor coupling and transverse field

$$H = \sum_{<i,j>} J_{ij} \sigma_x^{(i)} \sigma_x^{(j)} + \sum_i h_i \sigma_y^{(i)} \tag{3.14}$$

where the coefficients J_{ij} and h_i are energies with particular values for each system size, as shown in Table 3.1.

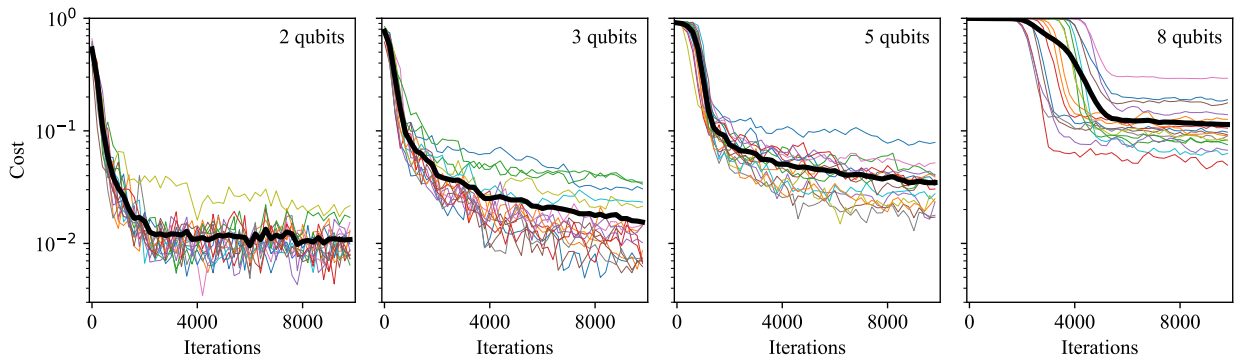


Figure 3.3: Compilation via STOQ for two-qubit, three-qubit, five-qubit, and eight-qubit versions of the time-evolution unitary from Equation 3.15. Each of the 16 thin curves shows the value of the cost function from Equation 3.12 during a single compilation using 10,000 iterations. The thick curve is the average of all runs.

n	J_{12}	J_{23}	J_{34}	J_{45}	J_{56}	J_{67}	J_{78}	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
2	1.27							1.54	1.19						
3	1.81	1.27						1.54	1.19	0.53					
5	1.20	1.40	1.60	1.80				1.60	1.30	1.00	0.70	0.40			
8	1.20	1.30	1.40	1.50	1.60	1.70	1.80	1.40	1.10	0.80	1.00	1.20	1.50	1.70	1.30

Table 3.1: Coefficients used for application of STOQ to the n-qubit Ising model Hamiltonian in Equation 3.14. Values are energies in kHz where $\hbar = 1$.

We then define the time-evolution unitary as $U_t(\tau) = e^{-iH\tau}$, where we choose units such that $\hbar = 1$, and we concretely choose $\tau = 0.5$ ms, such that

$$U = U_t(0.5 \text{ ms}) = e^{-iH(0.5 \text{ ms})} \tag{3.15}$$

is the target unitary for compilation.

To apply STOQ, we need also to choose a parameterized instruction set G from which to approximately compile a sequence. In a physical device, it is often the case that the dynamics are implemented such that each term in H can be individually controlled. To define G for such a device, we express the Hamiltonian as $H = \sum_k H_k$, where each H_k is one of the $\sigma_x\sigma_x$ or σ_y terms from Equation 3.14, and choose

$$G = \bigcup_k \left\{ e^{-iH_k t} \right\} \quad -\epsilon\tau \leq t \leq \epsilon\tau \tag{3.16}$$

where the allowed range for t is chosen such that the duration of each instruction is relatively short in comparison to the timescale of the dynamics of H. (In this demonstration we use $\epsilon = 0.2$.) Negative times correspond to reversing the sign of the coefficient of a given term. (We note that for a general Hamiltonian, it is unlikely that each H_k term is part of the native gate set of the device. In this case, G should instead represent the interactions which are implemented natively on the device.)

We then apply STOQ to compile many sequences that approximately implement U, using two-qubit, three-qubit, five-qubit, and eight-qubit versions of the corresponding Hamiltonian. Figure 3.3 reports the cost for 16 such compilations as a function of the number of iterations. (Each run of 10,000 iterations for the five-qubit system takes around 15 minutes to complete on a typical desktop computer.) We observe that the stochastic search process rapidly reduces the cost at first before noticeably leveling off. For the two-qubit and three-qubit systems, this cost approaches a limit near 10^{-2} after 10,000 iterations. For the larger systems, the final average cost is higher, although even for the eight-qubit system, the final cost reaches a value below 10^{-1} for some compilations.

To compare STOQ to existing compilation techniques, we also compile sequences to approximately implement U using the randomized Suzuki-Trotter decomposition [23] and the QDRIFT stochastic compilation protocol [20]. STOQ is designed to create more randomness in the resulting path taken through state space. To compare these paths quantitatively, we choose to compare the various methods to an ideal version where H is directly implemented for time τ. We define the *ideal path* as the path taken by this ideal time evolution, and we define the *compiled path* as the path taken by the compiled sequence, which we represent as a sequence of instructions $\{G_1, \ldots, G_M\}$. We then calculate the path distance d_m from the ideal path to step m of the compiled path, where $1 \leq m \leq M$, as

$$d_m = \min_{t \in [0,\tau]} D_{\text{HS}} \left(e^{-iHt},\ G_m G_{m-1} \cdots G_1 \right), \tag{3.17}$$

where D_{HS} is the Hilbert-Schmidt distance defined in Equation 3.11. Thus d_m is the shortest distance from step m of the compiled path to any point in the ideal path.

Results for each compilation technique are shown in Figure 3.4, and statistics for the five-qubit example are displayed in Table 3.2. We observe that the STOQ compilations result in a significantly greater path distance from the ideal evolution than the other approaches, and that the total running time of the compiled sequence resulting from the various compilations is within a factor of two.

However, the final cost of the STOQ compilations is typically at least an order of magnitude larger than the compilations created using the randomized Suzuki-Trotter and QDRIFT techniques, both of which can reach arbitrarily low costs by increasing the number of steps. This implies that STOQ would not be a useful tool for applications that require high-fidelity compilations.

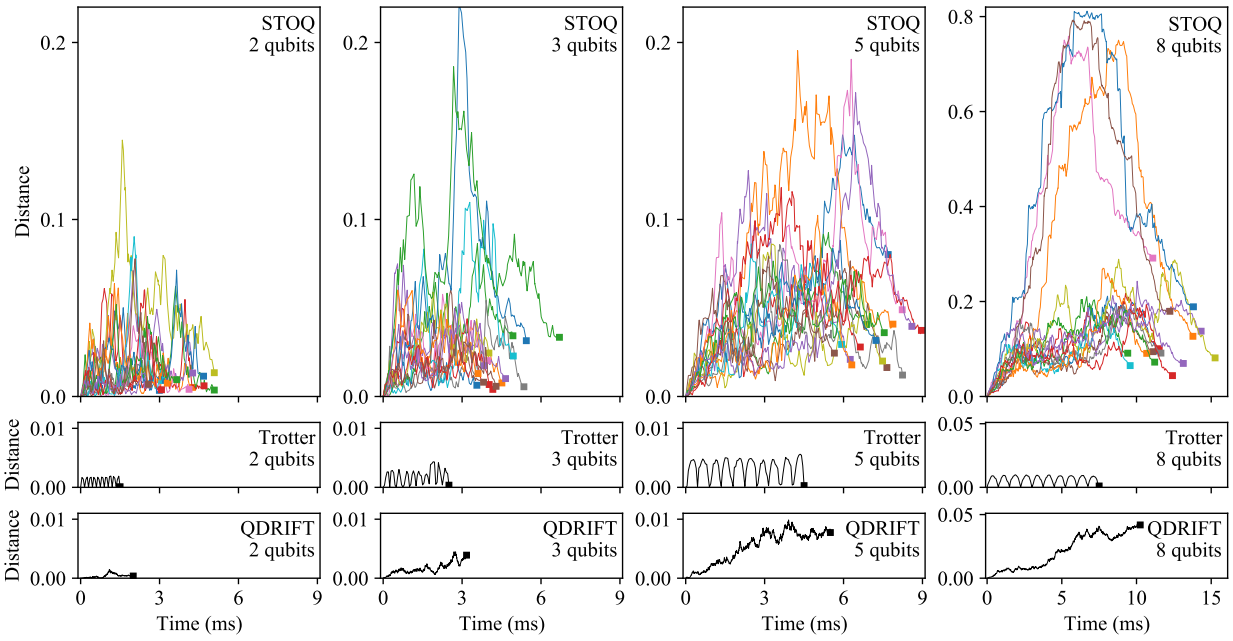


Figure 3.4: Distance from ideal path to compiled path, as defined in Equation 3.17, for the time-evolution unitary from Equation 3.15, illustrating "how different" a particular compilation is from the path that would be followed by the ideal time evolution of the full Hamiltonian. The distance from the horizontal axis quantifies the deviation in the system state from the ideal simulation when executing the specified compilation. The ideal simulation would be a flat curve along the horizontal axis from $\tau = 0.0$ ms to $\tau = 0.5$ ms. Results are shown for 2-qubit, 3-qubit, 5-qubit, and 8-qubit implementations of the Ising model Hamiltonian from Equation 3.14. Each curve represents the execution of one compiled sequence. Filled squares are used to plot the overall running time of the compiled sequence and final cost of each compilation. Top row depicts the execution of 16 independent STOQ compilations, each using 10,000 iterations. Each curve corresponds to a curve of the same color in Figure 3.3. Middle row depicts the execution of a typical randomized Suzuki-Trotter compilation using 10 steps. Bottom row depicts the execution of a typical QDRIFT compilation using 1,000 repetitions. The horizontal axis of each plot represents the execution time of the compiled circuits.

	Ideal	Trotter	QDRIFT	STOQ
(a) Execution time (ms)	0.50	4.50	5.50	7.32
(b) Average distance	—	0.0032	0.0053	0.0469
(c) Maximum distance	—	0.0056	0.0099	0.1133
(d) Final cost	—	0.0003	0.0077	0.0328

Table 3.2: Statistics resulting from various compilations of the five-qubit time-evolution unitary from Equation 3.15, where the ideal evolution occurs for $\tau = 0.5$ ms. For each of the compilation techniques, means are listed for each of the following quantities: (a) total execution time of the compiled sequence, (b) average distance $\sum_{m=1}^{M} d_m$, (c) maximum distance $\max_{m\in[1,M]} d_m$, and (d) final cost d_M. Corresponds to five-qubit plots in Figure 3.4.

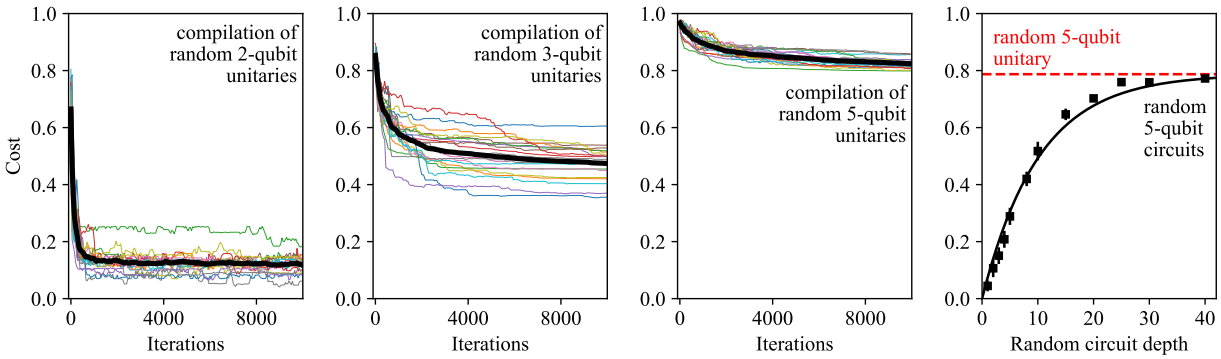


Figure 3.5: Compilation via STOQ of randomly-generated unitaries. The left three plots show the cost during the STOQ compilation process for randomly-generated 2-qubit, 3-qubit, and 5-qubit target unitaries. Each of the 20 thin curves shows the value of the cost function from Equation 3.12 during a single compilation using 10,000 iterations. The thick curve is the average of all runs. The rightmost plot shows the final cost of the STOQ compilation for target unitaries generated by creating random 5-qubit circuits of varying average circuit depth. Circuit depth is calculated as the total number of instructions divided by the number of qubits. Each point is the average of 20 compilations using 100,000 iterations. Error bars indicate standard error of the mean. The solid line is an exponential decay fit with one free parameter. The dashed line represents the average final cost of compiling a randomly-generated 5-qubit unitary. Note that one would expect a similar scaling with circuit depth for general quantum circuits, regardless of whether they are generated randomly.

3.6 Stochastic compilation of random unitaries

In addition to being used for sparse or highly structured unitaries such as those generated from Hamiltonian time-evolution, the STOQ protocol can also be used to compile sequences that approximately implement purely random unitaries in terms of an arbitrary instruction set, without having any prior knowledge of the structure of the unitary. Such an application of STOQ is not necessarily useful in general, but is included here for illustrative purposes.

Figure 3.5 shows typical results of repeatedly using the STOQ protocol to compile sequences for random two-qubit, three-qubit, and five-qubit unitaries, generated according to [111], using a simple universal instruction set $G = \{R(\theta, \varphi), XX(\theta)\}$. $R(\theta, \varphi)$ is a parameterized single-qubit rotation

$$R(\theta, \varphi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{i\varphi}\sin\frac{\theta}{2} \\ ie^{-i\varphi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{3.18}$$

as defined in Equation 2.13, where we require $0 \leq \theta < 2\pi$ and $0 \leq \varphi < 2\pi$. $XX(\theta)$ is the parameterized two-qubit entangling gate

$$XX(\theta) = \exp\left(-i\frac{\theta}{2}XX\right) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & -i\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ -i\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix} \tag{3.19}$$

as defined in Equation 2.21, where we require $0 \leq \theta < 2\pi$. We note that the instruction set G is a typical native gate set that can be implemented by trapped-ion quantum devices.

We observe that the final costs of compilation of these random unitaries are significantly larger than for compilation of the time-evolution unitaries discussed in Section 3.3. In particular, the final cost is approximately 0.1 for two-qubit random unitaries, 0.5 for three-qubit random unitaries, and 0.8 for five-qubit random unitaries. This indicates that the quality of the approximation for such random unitary compilations scales poorly with system size. This is not surprising, since reaching the vast majority of states in the Hilbert space of a system requires circuits of depth which grows exponentially with the dimension of the Hilbert space [90, 124]. Nonetheless, the compilations generated by this method may be useful in scenarios where high-fidelity approximations are not required.

We also observe that the final cost of such random unitary compilations is relatively stable over a wide range of STOQ parameter values. Two primary parameters that can be adjusted in the STOQ algorithm in Figure 3.1 are the annealing rate $\Delta\beta$, which is used to increment β at each step inside the `IncreaseBeta` function, and the probability p_{append} that the search appends an instruction (as opposed to removing an instruction) at each step, which occurs inside the `RandomChange` function. For compilation of three-qubit random unitaries, and for values $\Delta\beta \in \{0.001, 0.01, 0.1, 0.5\}$ and $p_{\text{append}} \in \{0.2, 0.5, 0.8\}$, we find that the average final cost remains between 0.398 (for $\Delta\beta = 0.5$ and $p_{\text{append}} = 0.2$) and 0.448 (for $\Delta\beta = 0.001$ and

$p_{\text{append}} = 0.5$), where each pair of parameter values is averaged over 32 compilations using 100,000 iterations each.

To provide insight into the low-fidelity approximations of random unitaries produced by STOQ, we consider the case of target unitaries generated by random circuits of varying depth. To do this, we generate random five-qubit circuits of average depth ranging from 1 to 40, where the average depth is calculated as the total number of instructions divided by the number of qubits. The rightmost plot in Figure 3.5 shows the final compilation cost after applying STOQ to generate an approximate compilation of the unitary corresponding to each random circuit. As might be expected, we observe that STOQ generates relatively high-fidelity approximations for shallow circuits, since such unitaries are known to be reachable with a fixed number of instructions. But as the circuit depth increases, the resulting unitaries begin to look more like random unitaries, and the final compilation cost approaches that of the randomly-generated five-qubit unitary discussed previously. Indeed, we expect a similar scaling with circuit depth for quantum circuits in general, regardless of whether they are randomly generated, since the size of the reachable state space grows exponentially with the depth of the circuit.

3.7 Discussion

Features

As demonstrated, STOQ has some features that are distinct from other methods. One notable feature is the capability of generating results with arbitrary gate sets, regardless of whether the gates are fixed or parameterized or whether the gate set is universal. In addition, repeated application of STOQ provides many independent approximate compilations of the same unitary, and each compilation creates a sequence that will cause the system state to traverse a different path in state space. As depicted in Figure 3.4, even stochastic techniques such as randomized Suzuki-Trotter or QDRIFT result in a compiled sequence that will cause the system state to follow very nearly the same path in state space as the deterministic version, whereas sequences generated by STOQ cause the system to traverse unique paths that can differ greatly from the ideal path and from each other.

We note that unitaries generated via time evolution of a Hamiltonian often benefit from the sparsity of the Hamiltonian. In general, an n-qubit Hamiltonian has 4^n coefficients when expressed in the basis of Pauli operators. For the five-qubit version of the Hamiltonian in Equation 3.14, only nine of these 1024 coefficients are non-zero. Sparsity in the Hamiltonian greatly limits the subspace of the full operator space that can be reached by via time evolution, which in turn makes compilation a more feasible task and allows techniques such as Suzuki-Trotter and QDRIFT to be highly efficient. Because the number of possible step proposals during each iteration of the STOQ search process is determined by the number of terms in the Hamiltonian, it is reasonable to infer that STOQ is similarly more effective when the problem structure contains such sparsity. This is further evidenced by the inabil-

ity of the STOQ protocol to efficiently obtain low cost values when compiling sequences for random target unitaries, which are not sparse in general.

Future work

We note that because the STOQ protocol requires calculating the product of the compiled sequence during each iteration, the computational cost of each iteration grows exponentially in the system size n. Therefore, the current formulation of STOQ is unlikely to be useful for system sizes of more than around 10 qubits. In particular, for compilation of time-evolution unitaries, this clearly means that STOQ will be less computationally efficient when compared to compilation methods based on product formulas, which in general have a computational cost that depends only on the number of terms in the Hamiltonian and is independent of the system size.

Because of this, it will likely be useful to pursue improvements to the performance of STOQ such that it can be practically used for systems larger than 10 qubits, which is roughly the limit of feasibility when executing the procedure on a typical workstation. One straightforward approach here is to extend the implementation such that the algorithm can be run on a computing cluster, rather than on a single machine, which should increase the maximum system size by several qubits. Search strategies that are more intelligent than a purely random MCMC search should also be investigated. For example, if certain heuristics can be identified that allow the search to converge more quickly and/or with higher probability, this could significantly reduce the resources required to generate these sequences for larger system sizes.

3.8 Summary

In this chapter, we have provided an introduction to the problem of quantum compilation and the broad classes of techniques that are used for this problem. Exact or ϵ-approximate techniques such as Solovay-Kitaev cannot be used for systems larger than around two qubits, and product-formula techniques such as Trotterization and QDRIFT rely on the structure of a particular Hamiltonian and cannot be used for compilation of arbitrary unitaries. Motivated by these limitations, we have introduced STOQ, a stochastic procedure for approximate compilation of a given unitary into a sequence of instructions from an arbitrary gate set. We have demonstrated the performance of STOQ on several practical problems, including compilation of time-evolution unitaries, compilation of random unitaries, and compilation of unitaries derived from random quantum circuits. We have also discussed the limitations of STOQ, particularly the fact that it is computationally intensive and likely impractical to run for systems with a large number of qubits. In Chapter 4 and Chapter 5, we will continue to build on this foundation by using STOQ as a key ingredient in the development of verification protocols for quantum computers and quantum simulators.

Chapter 4

Verification of quantum computers

Portions of this chapter were first released as part of an arXiv preprint [140] and are reproduced here with permission of the authors.

4.1 Introduction

Verifying the correct operation of quantum computations is an essential step toward building a reliable and scalable quantum information processing device [51]. Most commonly, quantum computations are broken down into fundamental building blocks known as *quantum gates*, which may include gates such as the well-known Hadamard, Pauli, and CNOT operations. Verifying the behavior of a device's physically-realizable gates, known as a *native gate set*, is a primary area of research in this field. The most complete techniques for gate verification belong to a family of techniques known as *tomography*. Such techniques include quantum state tomography [156, 30], quantum process tomography [110], and gate set tomography [12, 117]. Tomographic techniques produce a complete characterization of a quantum operation, which provides a detailed mathematical description of the errors present in the system. However, tomography is extremely resource-intensive, and although techniques exist to improve its scalability somewhat [134, 95], its cost still typically scales exponentially with qubit count.

In contrast, *benchmarking* techniques for verifying quantum gates are typically resource-efficient and in principle can be scaled to much larger qubit counts than tomographic techniques. These techniques notably include randomized benchmarking (RB) [40, 91] and more scalable variants such as cycle benchmarking [43] and direct RB [127], which involve executing randomized circuits which are equivalent to the identity, as well as techniques such as cross-entropy benchmarking (XEB) [13] and matchgate benchmarking [71], which compare the ideal and experimental output probabilities of random quantum circuits. Benchmarking provides an incomplete characterization of a quantum system, typically producing a small number of values which attempt to characterize the average error rate of particular operations performed by the system. But as the name implies, such techniques are particularly useful

Protocol	**Building blocks**	**Sequence generation**	**Measurement efficiency**	**Scalability**	**Supports random gate parameters**
Randomized benchmarking	Random n-qubit Cliffords	Easy	Efficient	Infeasible for $n \gg 2$	No
Cycle benchmarking	Native n-qubit Cliffords, random single-qubit Cliffords	Easy	Efficient	Scalable to large n (up to largest native n-qubit gate)	No
Mirror randomized benchmarking	Random native single-qubit and two-qubit Cliffords	Easy	Efficient	Scalable to large n (no maximum)	No
Cross-entropy benchmarking	Random native gates	Easy	Inefficient	Infeasible for $n \gg 50$	Yes
Randomized analog verification	Random native gates	Expensive	Efficient	Infeasible for $n \gg 10$	Yes

Table 4.1: Overview of various characteristics of several verification protocols for gate-based quantum computers. Schematics of these protocols are provided in Figure 4.1.

when attempting to compare the performance of distinct devices, since they provide metrics which are ostensibly hardware-agnostic. For example, benchmarking techniques may provide an estimate of the average error rate of executing a CNOT gate or of a device's average state preparation and measurement (SPAM) error.

Because many quantum algorithms and especially quantum error correction schemes are expressed in terms of particular fixed sets of gates – most commonly, the Clifford+T family, which is universal for quantum computation – much of the benchmarking literature is focused on verifying the operation of these fixed one-qubit and two-qubit gates, or their device-native equivalents. However, near-term quantum devices are unlikely to implement large-scale quantum error correction [125], and instead will implement circuits directly using the physical native gate set of the device. These native gates are often not limited to the fixed gate set used by error correction schemes, but rather have a parameterization which provides the ability to perform a continuous range of operations. For example, a single-qubit operation may frequently be parameterized as $R(\theta, \varphi)$, where θ is the rotation angle and φ is the axis of rotation. Multi-qubit operations may also be parameterized. The typical multi-qubit gate for trapped-ion devices, based on the Mølmer-Sørensen interaction [145, 112], can be parameterized as $MS(\theta, \varphi)$, where θ can be interpreted as the effective rotation angle in the multi-qubit space, and φ is the effective multi-qubit axis of rotation [106]. In many instances, compiling quantum circuits using continuously-parameterized native gate sets can produce more efficient compilations on physical devices than when limited to fixed gate sets [115].

Systematic and efficient verification techniques for continuously-parameterized quantum gates, therefore, are a key ingredient for near-term quantum computers to reliably take ad-

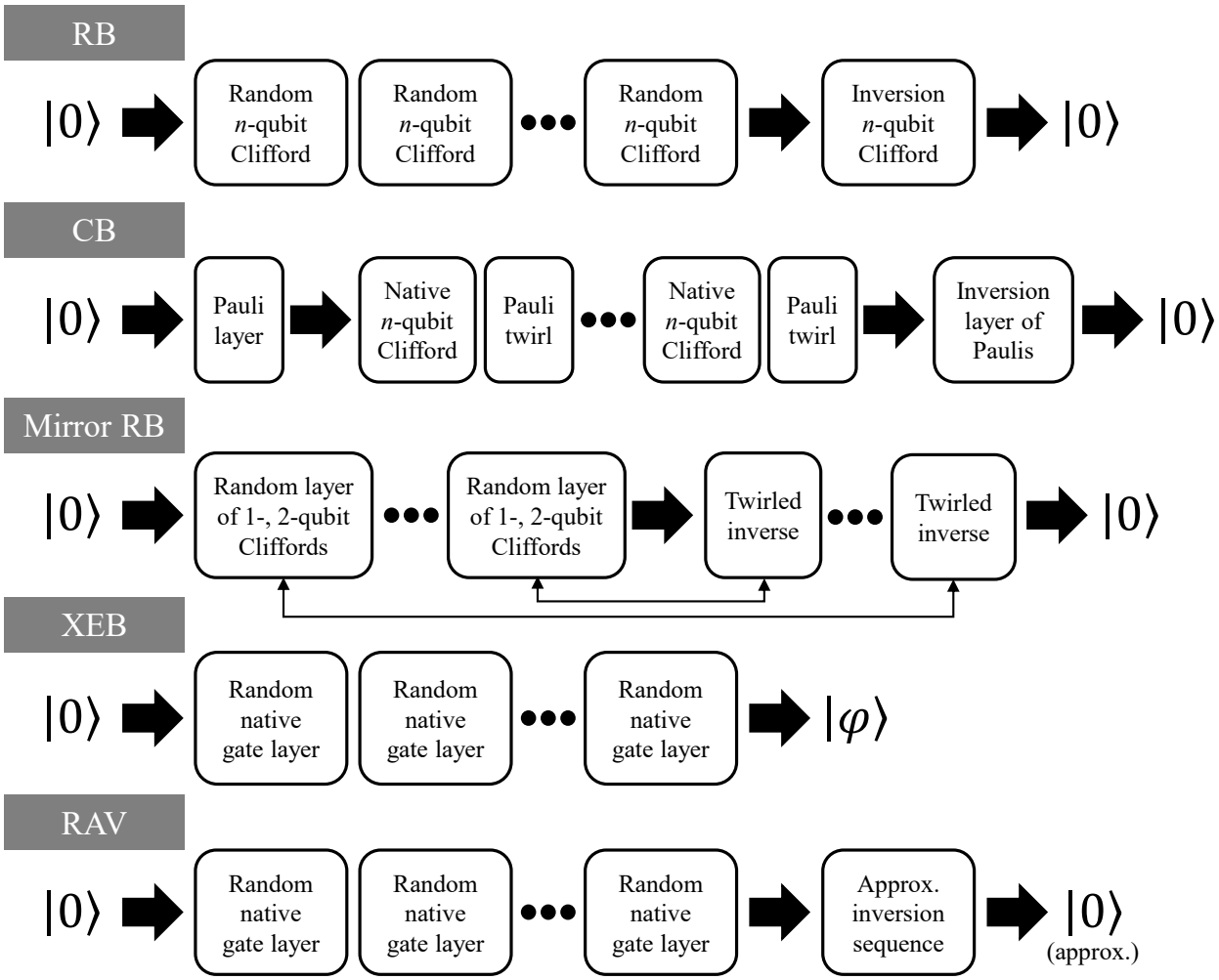


Figure 4.1: Schematics of sequences used in several verification protocols discussed in this chapter: randomized benchmarking (RB), cycle benchmarking (CB), mirror RB, cross-entropy benchmarking (XEB), and randomized analog verification (RAV). The states denoted as $|0\rangle$ can in general be any n-qubit computational basis state. In XEB, the n-qubit state denoted as $|\varphi\rangle$ is the ideal result of applying the XEB sequence to the initial state, and is in general far from any computational basis state. A comparison of the characteristics of these protocols is provided in Table 4.1.

vantage of the full scope of physically-realizable operations. In this chapter, we introduce the *randomized analog verification* (RAV) technique for the task of verifying continuously-parameterized quantum gates. An overview of the various protocols discussed in this chapter is provided in Table 4.1, and a schematic illustration of each protocol is provided in Figure 4.1. We begin with an introduction to verification techniques for gate-based quantum computers, discussing randomized benchmarking (RB) and variants in Section 4.2 and cross-entropy benchmarking (XEB) in Section 4.3. Next, in Section 4.4, we provide an overview of the RAV technique and compare it to XEB. In Section 4.5, we provide numerical simulations demonstrating the conditions under which we expect RAV to provide an efficiency advantage over XEB. In Section 4.6, we report experimental demonstrations of this efficiency advantage on the trapped-ion quantum processor at the Quantum Scientific Computing Open User Testbed (QSCOUT) operated by Sandia National Laboratories [27], as well as on a superconducting quantum processor from the publicly-available IBM Q service [75]. We conclude with additional discussion of these results in Section 4.7, and we summarize in Section 4.8.

4.2 Randomized benchmarking and variants

Randomized benchmarking

The canonical benchmarking technique for gate-based quantum computers is known as *randomized benchmarking* (RB) [40, 91]. The RB protocol consists of generating varying-length random sequences of Clifford operations whose product is equivalent to the identity (where a Clifford operation is defined as in Section 2.3). To generate an RB sequence for an n-qubit system, m Cliffords are drawn at random from the set of all n-qubit Cliffords. Now, to construct an inverse of this m-Clifford sequence, naively one could reverse the sequence and invert each individual Clifford to create an inverse sequence also containing m Cliffords. But because the Cliffords are a group, the product of any sequence of Cliffords is also a Clifford, as is the inverse of the product. And because Clifford sequences can be efficiently simulated [1], both the product and its inverse can be efficiently calculated. Appending the inverse Clifford to the original sequence of m Cliffords, then, gives a sequence of length $m+1$ which is equivalent to the identity. When running this circuit on a device, the *success rate* is the probability that the final measured state is the same as the initial state.

An average error rate can be extracted by running many RB sequences of varying length. Under certain reasonable assumptions about the noise [103, 104], the error incurred by each Clifford operation is independently and identically distributed, and thus the success rate decays exponentially with increasing sequence length. Fitting an exponential decay then allows the average per-Clifford error rate to be calculated.

Implementing RB on hardware requires compilation of each n-qubit Clifford operation into a sequence of native gates. This imposes a severe scalability limit, since compiling arbitrary n-qubit Cliffords results in very deep circuits which become impractical to execute for $n \gg 2$. In a demonstration of RB for $n = 3$, the compilation into native gates required an

average of 7.7 CNOT gates and 18.4 single-qubit gates per 3-qubit Clifford [109]. This scalability limit has spurred the development of efficient RB variants. These variants typically use only a subset of all n-qubit Cliffords which are, by design, easy to compile into native gates.

Cycle benchmarking

The cycle benchmarking (CB) protocol [43] is designed to characterize the performance of a particular n-qubit Clifford which can be natively implemented on a device, for example, an n-qubit Mølmer-Sørensen gate. As shown in Figure 4.1, the procedure involves a sequence of layers in which each n-qubit Clifford is combined with a "twirling" layer, i.e., a layer of single-qubit Paulis which have the effect of depolarizing any errors experienced during the execution of the circuit [155]. The n-qubit Clifford of interest is chosen such that m repetitions of it produce an identity operation, and therefore the final inversion layer is simply a layer of single-qubit gates to ideally return the system to a desired computational basis state.

Because CB sequences consist only of single-qubit Cliffords and a particular native n-qubit Clifford, it does not suffer the scalability limitations of RB and can be scaled to large n. However, since it is designed specifically to measure the average error rate of a particular n-qubit native gate, the scalability depends on the connectivity and native gate set of the system. In a trapped-ion system, an n-qubit Mølmer-Sørensen gate may be achievable for $n = 20$ or more. But in many systems, the native gate set consists only of one- and two-qubit gates, and in these cases CB may not be directly useful for $n > 2$.

Mirror randomized benchmarking

Another efficient variant of RB, known as mirror randomized benchmarking [126], provides scalability that is not limited by the connectivity of the device in the same way that cycle benchmarking is. As depicted in Figure 4.1, mirror RB proceeds by generating a sequence of random layers of one- and two-qubit native Cliffords, which by definition do not require compilation and therefore are efficient to implement even to large system sizes. To generate the inverse, these random layers are reversed and inverted. To avoid unintended cancellation of coherent errors, each inverted layer is then subjected to a randomized compiling procedure [155] which depolarizes the errors experienced during execution of the circuit. The overall sequence is thus equivalent to the identity, and the success rate and analysis proceeds similarly to the standard RB procedure.

4.3 Cross-entropy benchmarking

Cross-entropy benchmarking (XEB) [13] is a fundamentally different approach to benchmarking than RB and its many variants. RB-like protocols require that the sequence ideally

(in the absence of error) leave the system in a particular computational basis state, such that calculating the success rate requires estimating the measurement probability of only that single computational basis state.

In contrast, XEB sequences are made up of layers of random native gates which in general leave the system in an arbitrary state, as depicted in Figure 4.1. Calculating the success rate of an XEB sequence, which requires estimating the overlap of the ideal state with the observed state, therefore requires estimating the entire output distribution, and not just the probability of a single basis state. This is accomplished by using a linear cross-entropy formula (see Equation 4.1) to approximate the fidelity of the resulting state.

Because XEB sequences consist entirely of native gates, there is no compilation cost and therefore no scalability limit on implementing the sequences on a quantum device. However, calculating the linear cross-entropy first requires classical simulation of the ideal circuit, which in general becomes infeasible in the $n \gg 50$ regime. The most well-known application of XEB to date is the so-called "quantum supremacy" experiment carried out by Google [4], in which the classical computation for a 53-qubit XEB run took several hours on a supercomputer, even for a particular random circuit structure which was designed to be easier to simulate classically.

4.4 Randomized analog verification for gate-based devices

Protocol overview

The verification technique introduced in this chapter is a gate-based adaptation of the *randomized analog verification* (RAV) protocol for analog quantum simulators, which is described in detail in Chapter 5. When applied to analog quantum simulations, the RAV protocol consists of running randomized analog sequences of subsets of terms of a target Hamiltonian. In particular, a set of unitary operators is chosen consisting of short, discrete time steps of each of the terms of the target Hamiltonian. A randomly-generated sequence of these operations is then applied, which evolves the system to some arbitrary state. Next, an approximate inverse of this sequence, generated using the same set of unitary operators by using a stochastic compilation protocol (see Section 3.4), is applied to the system, which returns it to the initial state with high probability.

Because current gate-based, non-error-corrected quantum computers are realized by carefully tuning the underlying analog interactions to implement quantum gates with the highest fidelity possible, it is natural to adapt the RAV protocol for use in verifying the behavior of gate-based devices with continuously-parameterized native gates. To accomplish this, we use an approach similar to that used in cross-entropy benchmarking (XEB) by constructing random sequences of *layers*, each of which consists of some fixed number of each of the device's native gates in some randomly-chosen order. Random parameter values are then provided to each of these gates, which allows the protocol to verify the behavior of the de-

vice across the range of allowable parameter values for each gate. But unlike XEB, which proceeds by sampling directly from the output of these random sequences, RAV appends a compiled sequence of layers which approximately inverts the initial sequence. A schematic of the RAV protocol is displayed in Figure 4.1, which illustrates the fact that the primary difference from XEB is this inversion sequence which returns the system nearly to the initial state.

XEB and RAV fidelity estimates

Given a single XEB sequence on an n-qubit system, we can approximate the resulting fidelity as

$$\hat{F}_{\text{XEB}} = \frac{\sum_x P(x)Q(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{4.1}$$

where we have simplified the linear cross-entropy fidelity formula [13] for the case of a single circuit. Here, $P(x)$ represents the classically-computed ideal output probability distribution for the sequence, $Q(x)$ is the observed sample probability of obtaining measurement result x, and $N = 2^n$ is the dimension of the system. $\hat{F}_{\text{XEB}}$ is constructed such that its observed value for a single circuit might not fall within the range $[0, 1]$. But in general, the expected fidelity of the ideal output state (i.e., if $P(x) = Q(x)\ \forall\ x$) is 1, and the expected fidelity of a maximally-mixed output state is 0.

To derive a formula for the fidelity of a single RAV sequence on an n-qubit system, we start with the XEB fidelity estimate in Equation 4.1, where $P(x)$ represents the classically-computed ideal output probability distribution for the sequence, $Q(x)$ is the observed sample probability of obtaining measurement result x, and $N = 2^n$ is the dimension of the system. The RAV sequence is constructed to return nearly all of the population to the initial state x_0. Therefore, we let $P(x_0) = 1 - \epsilon$ for some small $\epsilon \ll 1$, which represents the approximation error of the inversion sequence. As a further simplification, we assume that the remaining probability is spread evenly among the remaining states, such that $P(x) = \frac{\epsilon}{N-1}$ for each $x \neq x_0$.

We can then derive the RAV fidelity from the XEB fidelity formula as follows:

$$\hat{F}_{\text{RAV}} = \frac{\sum_x P(x)Q(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{4.2}$$

$$= \frac{P(x_0)Q(x_0) + \sum_{x \neq x_0} P(x)Q(x) - \frac{1}{N}}{P(x_0)^2 + \sum_{x \neq x_0} P(x)^2 - \frac{1}{N}} \tag{4.3}$$

$$= \frac{(1-\epsilon)Q(x_0) + \frac{\epsilon}{N-1}(1 - Q(x_0)) - \frac{1}{N}}{(1-\epsilon)^2 + \frac{1}{N-1}\epsilon^2 - \frac{1}{N}} \tag{4.4}$$

$$= \frac{NQ(x_0) - 1}{N-1} + \epsilon \, \frac{N(NQ(x_0) - 1)}{(N-1)^2} + \epsilon^2 \, \frac{N^2(NQ(x_0) - 1)}{(N-1)^3} + \cdots \tag{4.5}$$

$$= \frac{NQ(x_0) - 1}{N-1} \left[1 + \frac{N\epsilon}{N-1} + \left(\frac{N\epsilon}{N-1} \right)^2 + \cdots \right] \tag{4.6}$$

where in the next-to-last step we have performed a Taylor expansion around $\epsilon = 0$. Then, in the final step we note that the expression inside the square brackets is just a geometric series in $\frac{N\epsilon}{N-1}$, and therefore we have

$$\hat{F}_{\text{RAV}} = \frac{NQ(x_0) - 1}{N-1} \left[\frac{1}{1 - \frac{N\epsilon}{N-1}} \right] \tag{4.7}$$

$$= \frac{Q(x_0) - \frac{1}{N}}{(1-\epsilon) - \frac{1}{N}} \tag{4.8}$$

which, using the fact that $P(x_0) = 1 - \epsilon$, gives the following expression for the approximate RAV sequence fidelity:

$$\hat{F}_{\text{RAV}} = \frac{Q(x_0) - \frac{1}{N}}{P(x_0) - \frac{1}{N}} \tag{4.9}$$

We use the hat on the symbols $\hat{F}_{\text{XEB}}$ and $\hat{F}_{\text{RAV}}$ to emphasize that they are only estimates of fidelity based on a single circuit instance. To obtain reliable information about the fidelity, these results must be aggregated over many circuit instances. This is especially true in the case of smaller qubit count, where the variance across different random circuit instances is most significant [99].

It is also worth clarifying that the fidelity estimated by $\hat{F}_{\text{XEB}}$ and $\hat{F}_{\text{RAV}}$ is the *depolarization fidelity* [4], which is not equivalent to the typical *state fidelity* that is used in many contexts when discussing fidelity (see Equation 2.41). In particular, if we assume that a circuit's ideal output state is $|\psi\rangle\langle\psi|$ and its execution is subject to purely depolarizing errors, then we can define the true output state in terms of a depolarization parameter λ as

$$\rho_\lambda = (1-\lambda)|\psi\rangle\langle\psi| + \frac{\lambda}{N} I \tag{4.10}$$

where $\lambda \in [0, 1]$ is the fraction to which the output state is depolarized. The depolarization fidelity of this state is then defined as $F = 1-\lambda$. As derived in [4], the average depolarization fidelity $\overline{F}$ is related to the average state fidelity $\overline{f}$ as

$$\overline{f} = \overline{F} + \frac{1 - \overline{F}}{N}, \tag{4.11}$$

such that a fully-depolarized system has $\overline{F} = 0$ and $\overline{f} = \frac{1}{N}$.

Variance of XEB and RAV fidelity estimates

Intuitively, we expect that measuring the success of RAV sequences should be more efficient than measuring the success of XEB sequences. This is because RAV requires estimating the output probability $Q(x_0)$ of only the initial state (under the assumptions used to derive Equation 4.9), whereas XEB requires estimating the full output probability distribution $Q(x)$. Additionally, in the case where the error is small, we expect the final state of a RAV sequence to be close to a basis state, which minimizes the quantum projection noise associated with the measurement.

More concretely, we can demonstrate this efficiency advantage by calculating the statistical variance associated with measurement of $\hat{F}_{\text{RAV}}$ and $\hat{F}_{\text{XEB}}$ for a single sequence. In particular, we would like to derive formulas for the variance of the fidelity estimates resulting from K independent shots of a single RAV or XEB circuit.

We start by assuming that the errors in our device are purely depolarizing. Under this assumption, we can represent the output state of any n-qubit circuit as a mixture of the circuit's ideal output state $|\psi\rangle\langle\psi|$ and the maximally-mixed state $\frac{1}{N}I$ (where $N = 2^n$) as in Equation 4.10, where $\lambda \in [0, 1]$ is the fraction to which the output state is depolarized.

We can then define $P(x)$ and $Q_\lambda(x)$ as the probabilities of measuring outcome x when measuring the states $|\psi\rangle\langle\psi|$ and ρ_λ, respectively, as:

$$P(x) = \langle x|\psi\rangle\langle\psi|x\rangle = \left|\langle x|\psi\rangle\right|^2 \tag{4.12}$$

$$Q_\lambda(x) = \langle x|\rho_\lambda|x\rangle = (1-\lambda)P(x) + \frac{\lambda}{N} \tag{4.13}$$

We can restate Equation 4.1 and Equation 4.9 to define the fidelity estimates $\hat{F}_{\text{RAV}}$ and $\hat{F}_{\text{XEB}}$ in terms of $P(x)$ and $Q_\lambda(x)$ as follows:

$$\hat{F}_{\text{RAV}} = \frac{Q_\lambda(x_0) - \frac{1}{N}}{P(x_0) - \frac{1}{N}} \tag{4.14}$$

$$\hat{F}_{\text{XEB}} = \frac{\sum_x P(x)Q_\lambda(x) - \frac{1}{N}}{\sum_x P(x)^2 - \frac{1}{N}} \tag{4.15}$$

Now, to calculate the expected variance of our these fidelity estimates, we first need to determine their distribution. To do this, we let $\mathbf{Q}_{\lambda,\mathbf{x}} \sim \text{Multinomial}(K, N, Q_\lambda(x))$ be a

random variable representing the number of times outcome $\mathbf{x}$ is observed when taking K independent shots of a RAV or XEB circuit, where $Q_\lambda(x)$ represents the "true" experimental probability distribution of each of N possible outcomes when measuring the state ρ_λ.

By the properties of the multinomial distribution, then, the variance of $\mathbf{Q}_{\lambda,\mathbf{x}}$ is:

$$\begin{aligned}\mathrm{Var}\big[\mathbf{Q}_{\lambda,\mathbf{x}}\big] &= KQ_\lambda(x)\big[1-Q_\lambda(x)\big] &(4.16)\\ &= K\left[(1-\lambda)P(x)+\frac{\lambda}{N}\right]\left[1-(1-\lambda)P(x)-\frac{\lambda}{N}\right]. &(4.17)\end{aligned}$$

We can now construct random variables corresponding to measurements of $\hat{F}_{\mathrm{RAV}}$ and $\hat{F}_{\mathrm{XEB}}$ by replacing $Q_\lambda(x)$ in Equation 4.14 and Equation 4.15 with the scaled random variable $\frac{1}{K}\mathbf{Q}_{\lambda,\mathbf{x}}$, which represents the sample probability of observing outcome $\mathbf{x}$ when taking K independent shots:

$$\hat{\mathbf{F}}_{\mathrm{RAV}} = \frac{\frac{1}{K}\mathbf{Q}_{\lambda,\mathbf{x_0}}-\frac{1}{N}}{P(x_0)-\frac{1}{N}} \qquad \hat{\mathbf{F}}_{\mathrm{XEB}} = \frac{\sum_x P(x)\frac{1}{K}\mathbf{Q}_{\lambda,\mathbf{x}}-\frac{1}{N}}{\sum_x P(x)^2-\frac{1}{N}} \tag{4.18}$$

Once we have done this, calculating the variance of these random variables is straightforward algebra:

$$\begin{aligned}\mathrm{Var}\big[\hat{\mathbf{F}}_{\mathrm{RAV}}\big] &= \frac{1}{K^2}\left(\frac{1}{P(x_0)-\frac{1}{N}}\right)^2 \mathrm{Var}\big[\mathbf{Q}_{\lambda,\mathbf{x_0}}\big] &(4.19)\\ &= \frac{1}{K}\left(\frac{1}{P(x_0)-\frac{1}{N}}\right)^2 \left[(1-\lambda)P(x_0)+\frac{\lambda}{N}\right]\left[1-(1-\lambda)P(x_0)-\frac{\lambda}{N}\right] &(4.20)\end{aligned}$$

$$\begin{aligned}\mathrm{Var}\big[\hat{\mathbf{F}}_{\mathrm{XEB}}\big] &= \mathrm{Var}\left[\frac{\frac{1}{K}\sum_x P(x)\mathbf{Q}_{\lambda,\mathbf{x}}-\frac{1}{N}}{\sum_x P(x)^2-\frac{1}{N}}\right] &(4.21)\\ &= \frac{1}{K^2}\frac{\sum_x P(x)^2\ \mathrm{Var}\left[\mathbf{Q}_{\lambda,\mathbf{x}}\right]}{\left(\sum_x P(x)^2-\frac{1}{N}\right)^2} &(4.22)\\ &= \frac{1}{K^2}\left(\frac{N}{N\sum_x P(x)^2-1}\right)^2 \sum_x P(x)^2\ K\left[(1-\lambda)P(x)+\frac{\lambda}{N}\right]\\ &\qquad\qquad \times\left[1-(1-\lambda)P(x)-\frac{\lambda}{N}\right] &(4.23)\\ &= \frac{1}{K}\left(\frac{N}{N\sum_x P(x)^2-1}\right)^2 \Bigg[\left(\frac{\lambda}{N}\right)\left(1-\frac{\lambda}{N}\right)\sum_x P(x)^2\\ &\qquad + (1-\lambda)\left(1-\frac{2\lambda}{N}\right)\sum_x P(x)^3\\ &\qquad - (1-\lambda)^2\sum_x P(x)^4\Bigg] &(4.24)\end{aligned}$$

The above formulas are sufficient to predict the variance in fidelity measurements for a particular circuit, assuming we know the ideal probabilities $P(x)$ of measuring the circuit output. But of course, these probabilities will be different for every circuit. To make more general predictions, we need to make additional assumptions.

For RAV, the sequences have been explicitly constructed such that most of the population is returned to the initial state x_0. Therefore, we assume that $P(x_0) \approx 1 - \epsilon$ for some small $\epsilon \ll 1$, which represents the approximation error of the inversion sequence. Substituting this into Equation 4.20 gives

$$\mathrm{Var}\big[\hat{F}_{\mathrm{RAV}}\big] \approx \tfrac{1}{K}\left(\tfrac{1}{(1-\epsilon)-\frac{1}{N}}\right)^2 \big[(1-\lambda)(1-\epsilon)+\tfrac{\lambda}{N}\big]\big[1-(1-\lambda)(1-\epsilon)-\tfrac{\lambda}{N}\big] \tag{4.25}$$

where K is the number of independent experimental shots taken for the given sequence and $\epsilon \ll 1$ is the error in the compiled inversion of the RAV sequence.

For XEB, it is known that for systems with tens of qubits and circuits of depth $\gtrsim 10$, the distribution of ideal output state probabilities can be well-approximated by a Porter-Thomas distribution [13], in which the probabilities follow an exponential distribution $\Pr(Np) = Ne^{-Np}$. By the properties of the exponential distribution, then, we have $\sum_x P(x)^k \approx \frac{1}{k}$, and substituting this into Equation 4.24 gives

$$\mathrm{Var}\big[\hat{F}_{\mathrm{XEB}}\big] \approx \tfrac{1}{K}\left(\tfrac{N}{\frac{1}{2}N-1}\right)^2 \Big[\tfrac{1}{2}\left(\tfrac{\lambda}{N}\right)\left(1-\tfrac{\lambda}{N}\right)+\tfrac{1}{3}(1-\lambda)\left(1-\tfrac{2\lambda}{N}\right)-\tfrac{1}{4}(1-\lambda)^2\Big] \tag{4.26}$$

where K is again the number of independent experimental shots taken for the given sequence.

Comparison of XEB and RAV precision

We plot the standard deviation (i.e., $\sqrt{\mathrm{Var}\big[\hat{F}_{\mathrm{RAV}}\big]}$ and $\sqrt{\mathrm{Var}\big[\hat{F}_{\mathrm{XEB}}\big]}$) of these fidelity estimates in Figure 4.2 for $2 \le n \le 16$ qubits and $0 \le \lambda \le 1$, assuming $K = 100$ and $\epsilon = 0.04$. The mean of the RAV and XEB fidelity estimates are in agreement in all cases, but we observe from these plots that RAV fidelity estimates have a lower standard deviation than XEB in all cases, with the advantage tending to shrink as λ and n increase. This implies that fewer RAV repetitions are necessary in order to get an equivalently-precise estimate of the fidelity of a given sequence. We note that this advantage is largest in the regime where λ is small, since this is exactly the regime where the final state of a RAV sequence is near a basis state, which minimizes the quantum projection noise associated with the computational basis measurement.

As a further illustration, we perform a simulation of RAV and XEB fidelity measurements across various regimes of fidelity decay assuming a purely depolarizing channel. We generate representative RAV and XEB sequences for systems with $2 \le n \le 8$ qubits, and we calculate the standard deviation of fidelity measurements on these sequences for $0 \le \lambda \le 1$. These plots, shown in Figure 4.3, show good agreement between the calculated standard deviation

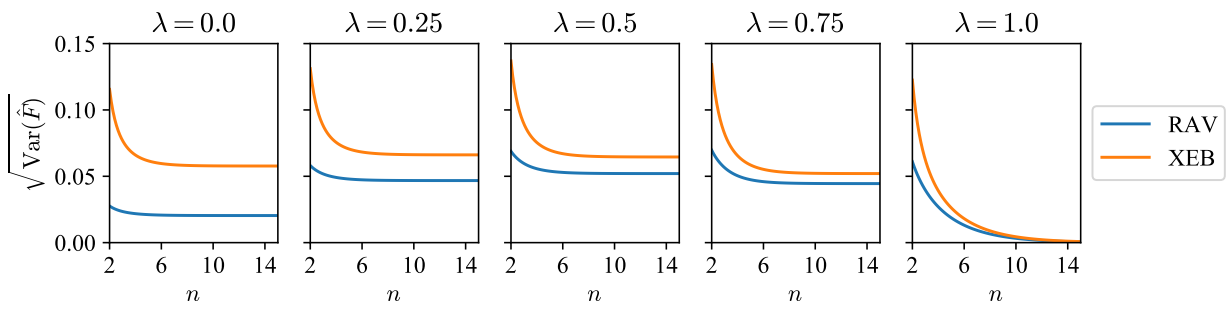


Figure 4.2: Ideal standard deviation of $\hat{F}_{\mathrm{RAV}}$ and $\hat{F}_{\mathrm{XEB}}$ measurements for n-qubit systems as calculated analytically by Equation 4.25 and Equation 4.26, where $\lambda = 0$ corresponds to an ideal output state, $\lambda = 1$ corresponds to a maximally-mixed output state (see Equation 4.10), and we assume a RAV inverse approximation error of $\epsilon = 0.04$. All plots use $K = 100$ independent measurement shots per sequence.

for the RAV circuits and the standard deviation extracted from simulations. We observe some quantitative disagreement for the XEB circuits, which is likely because we are working with simulations of ≤ 8 qubits, and so the Porter-Thomas assumption is not necessarily valid in this regime.

Nonetheless, we observe qualitative agreement between the calculated and simulated results for the XEB circuits as well as for the RAV circuits, and we observe that the variance of the RAV fidelity estimates is lower than the variance of the XEB fidelity estimates in all cases. This supports the existence of the RAV efficiency advantage even in the absence of the simplifying assumptions used to derive Equation 4.25 and Equation 4.26.

We note that the assumption of purely depolarizing errors is not physical, since any real device will be subject to a variety of coherent and stochastic error sources which on their own would not act as a depolarizing channel. However, it is believed that ensembles of random quantum circuits effectively transform even local coherent errors into global depolarizing noise [13, 33], and so it is reasonable to expect that experimental results under various realistic noise sources will show good agreement with the idealized results under the assumption of purely depolarizing noise.

4.5 Numerical demonstrations

To demonstrate the efficiency advantage of RAV, we generated 50 RAV and 50 XEB sequences of varying lengths for a five-qubit system. We generated these sequences using a continuously-parameterized native gate set $\{R(\theta, \varphi), R_Z(\theta), MS(\theta, \varphi)\}$, where these operations are defined as in Equation 2.13, Equation 2.12, and Equation 2.26, respectively. We

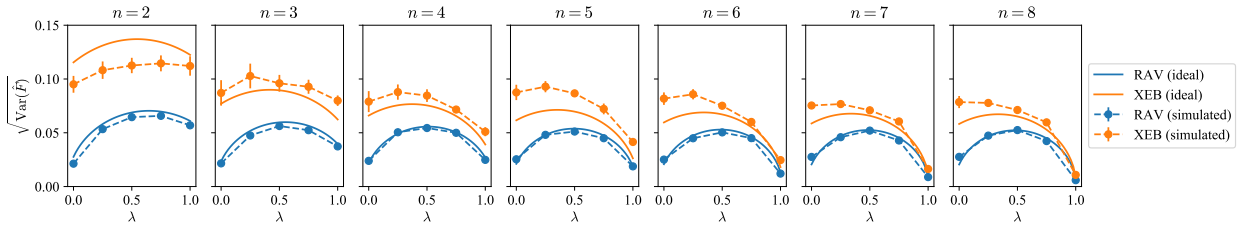


Figure 4.3: Comparison of the precision of fidelity measurements using RAV and XEB at various levels of fidelity decay, where $\lambda = 0$ corresponds to an ideal output state and $\lambda = 1$ corresponds to a maximally-mixed output state (see Equation 4.10). The "simulated" data points, marked by circles, represent the observed standard deviation of 100 simulated measurements of $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for a set of representative RAV (or XEB) sequences on an n-qubit system. Error bars indicate standard error of the mean across the set of sequences. The "ideal" solid curves represent the ideal standard deviation of $\hat{F}_{\mathrm{RAV}}$ and $\hat{F}_{\mathrm{XEB}}$ measurements for the given n and λ as calculated analytically by Equation 4.25 and Equation 4.26, assuming a RAV inverse approximation error of $\epsilon = 0.04$. All plots use $K = 100$ independent measurement shots per sequence.

chose this gate set because it aligns most directly with the native gate set of the QSCOUT trapped-ion processor. Specifically, $R(\theta, \varphi)$ and $MS(\theta, \varphi)$ are implemented by the QSCOUT device as native one-qubit and two-qubit physical operations, and $R_Z(\theta)$ is implemented by the QSCOUT device as a virtual one-qubit operation which requires no physical interaction with the qubits.

We generated the sets of sequences such that the total layer counts of the RAV and XEB sequences are equivalent. Because RAV sequence lengths are unpredictable due to the stochastic inversion compilation process, we first generated 50 RAV sequences over a range of sequence lengths up to 800 layers. For each RAV sequence, we then generated an XEB sequence of exactly the same length.

Each generated layer consists of three $R(\theta, \varphi)$ gates, three $R_Z(\theta)$ gates, and one $MS(\theta, \varphi)$ gate. The target qubit(s) for each gate are chosen uniformly at random from the set of all qubits in the system. A layer is constructed by first choosing random parameter values for each of these seven gates. Values for each θ rotation angle are chosen uniformly at random in the interval $[-\pi/10, \pi/10]$, and values for each φ axis angle are chosen uniformly at random in the interval $[-\pi, \pi]$. After the parameter values are chosen, the order of the gates within the layer is then randomly permuted, which produces the layer that is ultimately used in the sequence.

Figure 4.4 depicts the mean and standard deviation of error rates per layer obtained via both RAV and XEB for sequences under varying simulated depolarization rates. We observe that, over a wide range of depolarization rates, the mean fitted error rates from

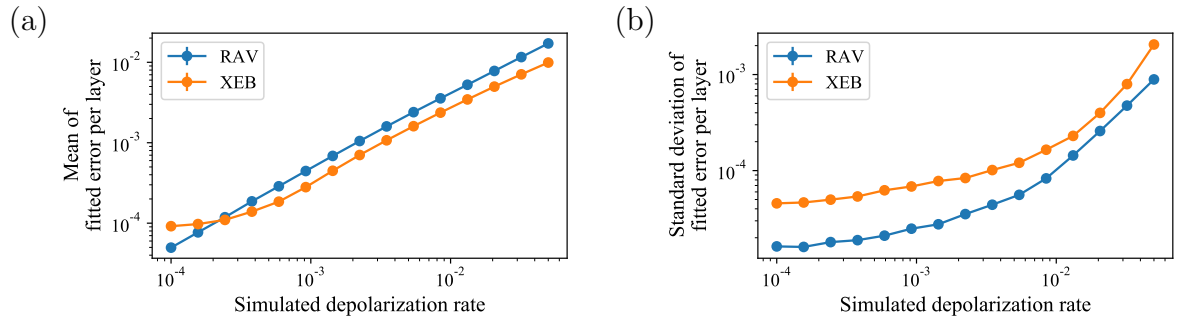


Figure 4.4: Statistics of fitted error rates from simulations of five-qubit RAV and XEB runs. Each data point represents the fitted error rate statistics of 100 independent simulations, using 50 RAV or XEB circuits of up to 800 layers with $K = 100$ shots per circuit, as described in Section 4.5. The sets of RAV and XEB sequences are of matching lengths; i.e., for each of the 50 RAV sequences, an XEB sequence was generated with the same number of layers. Error bars, which are smaller than the data markers, represent the standard error of the mean across 10 independent repetitions. Simulated depolarization rate is implemented as the amount of depolarization per $\theta = \pi/2$ rotation via single-qubit $R(\theta, \varphi)$ gate or $\theta = \pi/20$ rotation via two-qubit $MS(\theta, \varphi)$ gate, where the total amount of depolarization is proportional to the rotation angle θ. Simulated $R_Z(\theta)$ gates are unaffected by depolarization rate. (a) Mean fitted error per layer as a function of simulated depolarization rate. (b) Standard deviation of fitted error per layer as a function of simulated depolarization rate.

RAV and XEB are closely aligned, but the error rates estimated by RAV have a significantly smaller standard deviation than those obtained via XEB, indicating that RAV provides more precise information about the overall error rate of these sequences. We also observe that this advantage is more significant in the regime of lower depolarization rates. For a depolarization rate around 10^{-2}, the RAV error rate estimates are roughly twice as precise as the XEB error rate estimates, whereas around 10^{-4}, they are roughly three times as precise.

We note that because the RAV efficiency advantage comes partially from reduced quantum projection noise due to measurement, we expect that we will find the largest advantage when operating in the early part of the RAV decay curve (which in this simulation is realized by lower depolarization rate). This is the regime where the RAV sequence results remain closest to a basis state, where quantum projection noise is minimized. However, as described previously, RAV continues to have an efficiency advantage in the regime of larger error rates because it requires estimating the output probability of only a single state, rather than estimating the full output probability distribution as required by XEB.

4.6 Experimental demonstrations

QSCOUT experimental results

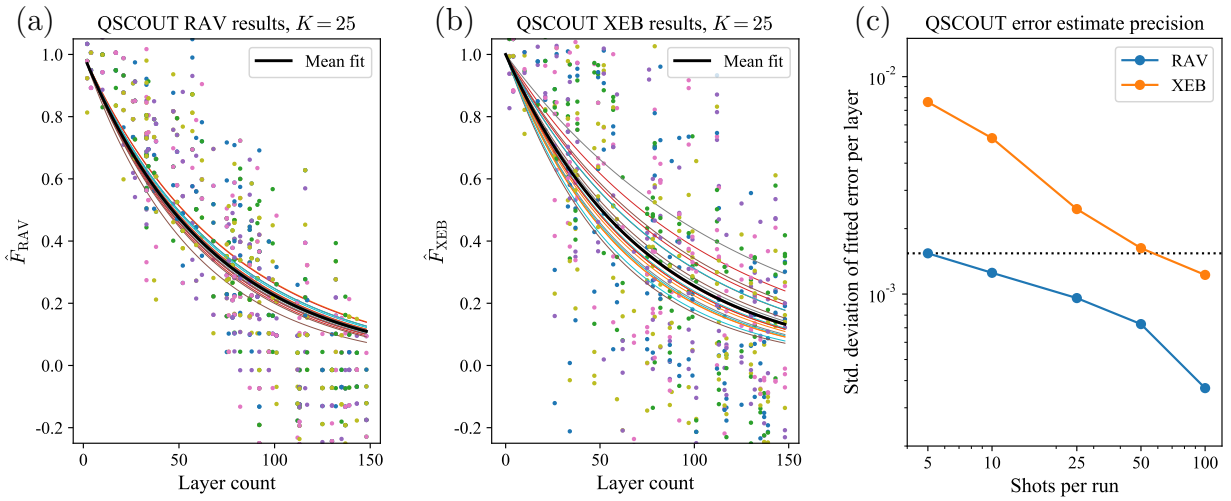
As a first experimental demonstration, we executed 50 RAV and 50 XEB sequences on the QSCOUT two-qubit trapped-ion quantum processor operated by Sandia National Laboratories [27]. We note that this device directly implements the parameterized native gate set $\{R(\theta,\varphi), R_Z(\theta), MS(\theta,\varphi)\}$ that we used to generate these sequences. Additional details on the QSCOUT experimental implementation can be found in [140].

Figure 4.5(a) and Figure 4.5(b) show the results of 20 independent runs of the entire set of RAV and XEB sequences, using $K = 25$ shots per experiment. The same set of sequences was used for each run, but we performed the 20 independent runs in order to be able to calculate the mean and standard deviation of the fitted error rates. (Note that for experimental simplicity, 500 executions were performed for each sequence, and we then separated the shot-level results and interpreted them as $500/K$ independent runs of K shots each.) It is clear visually that the XEB fit curves vary significantly more from the mean than the RAV fit curves. The standard deviations of these fitted error rates for various values of K are plotted in Figure 4.5(c), and the corresponding statistics are tabulated in Figure 4.5(d). As expected, we observe that the error rates obtained via RAV runs have a significantly smaller standard deviation (by a factor of 2.5 to 5) than those obtained from XEB runs. Since the standard deviation of the fidelity estimate goes as $1/\sqrt{K}$ (which is supported by the data in Figure 4.5(c)), this implies that XEB would require approximately 6 to 25 times as many experimental shots as RAV to produce an error rate estimate for this device with equivalent precision. An example of this can be seen visually by the dashed line in Figure 4.5(c), which shows that the RAV $K = 5$ runs provide a more precise error rate estimate than the XEB $K = 50$ runs.

IBM Q experimental results

To provide further experimental results, we executed 72 RAV and 72 XEB sequences on the publicly-available `ibmq_manila` superconducting processor from IBM [75]. This device does not directly implement the parameterized native gate set $\{R(\theta,\varphi), R_Z(\theta), MS(\theta,\varphi)\}$. To adapt the sequences for this device, we translated the sequences from the original parameterized native gate set into the instruction set that is accepted by the IBM Q framework. More specifically, $R(\theta,\varphi) = U(\varphi, \theta - \pi/2, \pi/2 - \theta)$ and $MS(\theta,\varphi) = R_Z(-\varphi)^{\otimes 2} R_{XX}(\theta) R_Z(\varphi)^{\otimes 2}$, where $R_{XX}(\theta)$ is itself a composite gate that can be implemented through multiple native gates. $R_Z(\theta)$ is implementable directly and does not need to be translated.

Figure 4.6(a) and Figure 4.6(b) show the results of 10 independent runs of the entire set of RAV and XEB sequence, using $K = 1000$ shots per experiment. XEB curves are visually more dispersive than RAV curves, which indicates RAV's smaller uncertainty in estimating fidelity. The advantage is verified for different numbers of shots, as is shown in Figure 4.6(c). Figure 4.6(d) tabulates the specific statistics. The standard deviation of the



(d)

Shots per run	*RAV fitted error per layer* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	*XEB fitted error per layer* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	$\frac{\text{XEB } \frac{SD}{Mean}}{\text{RAV } \frac{SD}{Mean}}$
$K = 5$	1.486	0.154	0.104	1.459	0.764	0.524	5.05
$K = 10$	1.484	0.125	0.085	1.410	0.521	0.370	4.38
$K = 25$	1.482	0.096	0.065	1.356	0.246	0.181	2.80
$K = 50$	1.481	0.073	0.049	1.342	0.163	0.121	2.47
$K = 100$	1.480	0.037	0.025	1.334	0.123	0.092	3.68

Figure 4.5: Experimental results from QSCOUT for two-qubit RAV and XEB runs. Each run consists of 50 sequences executed K times each. (a,b) Results of 20 independent RAV (or XEB) runs using $K = 25$ shots per sequence. Colored data points indicate the fidelity estimate $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for K shots of a single sequence, calculated according to Equation 4.9 (or Equation 4.1). Thin colored curves are single-parameter exponential decay fits for the data points from each run. Data points and fit curves from a particular run are assigned the same color. The thick curve is the mean of the individual fit curves. (c) Standard deviation of the fitted error per layer for RAV and XEB runs for various values of K. Smaller standard deviation indicates a more precise estimate of the error rate. Dashed horizontal line through the RAV $K = 5$ data point is a visual guide. (d) Fitted error per layer statistics for RAV and XEB runs for various values of K.

fidelity estimates from RAV experiments is about one to three times smaller than that from XEB experiments, which implies that to achieve the same accuracy of fidelity estimation, XEB would take more experimental resources than RAV.

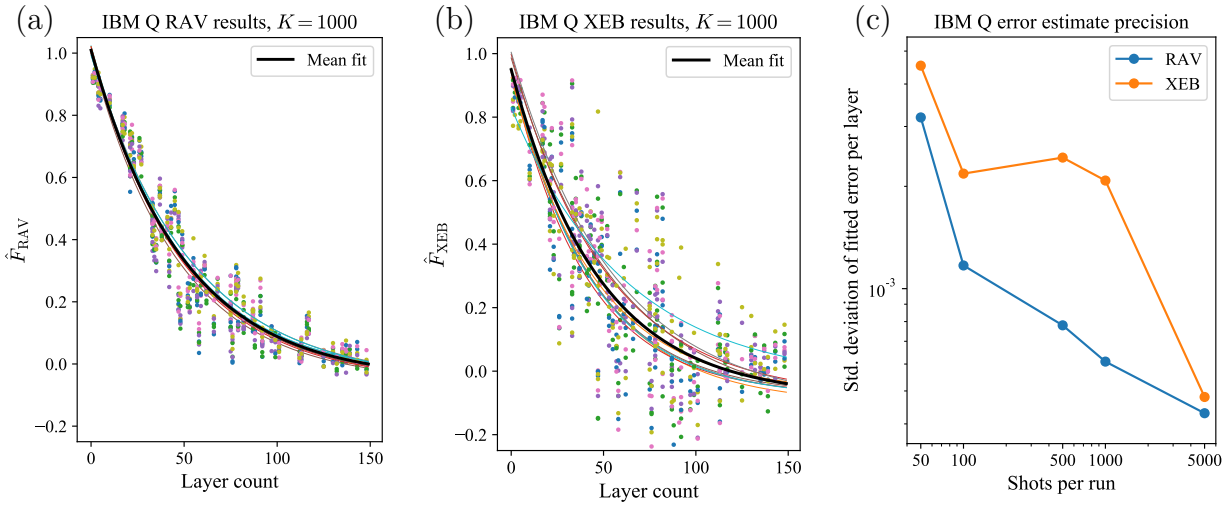
Based on the IBM Q-published gate error rates at the time the experiments were run, we can calculate that the error per layer of our RAV and XEB sequences should be about 1.57×10^{-2}, which corresponds well with the experimental results. We also note that the IBM Q results in Figure 4.6(c) do not demonstrate the expected scaling of the standard deviation of the fidelity estimate as $1/\sqrt{K}$. We believe that this is because the true error rate of the `ibmq_manila` device was fluctuating during the course of the runs, which were spread over the course of several hours. This led directly to increased variance in fitted error rates for some of the experiments; for example, comparing to the QSCOUT results in Figure 4.5(c), we clearly see much higher variance in the IBM Q results despite using the same set of RAV and XEB sequences. Therefore, our estimated error rate variances are including the effects of these physical fluctuations in addition to the inherent variance from the RAV and XEB estimates. We expect that runs in which all of the sequences are executed in rapid succession would help to reduce the effect of such fluctuations.

4.7 Discussion

We note that because of its efficiency advantages, RAV may be particularly useful in the context of frequent calibration runs for devices with continuously-parameterized gates. Minimizing the number of experimental shots per calibration run reduces the downtime of a device due to calibration, and therefore increases its availability for more useful work. Although RAV itself is not a calibration scheme, it can be used to rapidly obtain an estimate of average error over the device's continuously-parameterized gate set (more rapidly than techniques based on cross-entropy benchmarking, as shown in this chapter), which could in turn be used as feedback to a calibration technique or as verification of a completed calibration.

We demonstrate these benchmarking approaches on continuously-parameterized gate sets of both a trapped-ion system, QSCOUT, and a superconducting system, IBM Q. In particular, the gate sets used in this demonstration more closely align with the native physical gates of trapped-ion systems. In QSCOUT, one circuit layer contains a single physical two-qubit gate, where the parameterized θ is directly associated with physical inputs to that gate; however, on the IBM Q system, one circuit layer contains two physical two-qubit gates, and the parameterized θ is instead passed into the surrounding single-qubit gates. As such, the particular RAV construction demonstrated here provides an error rate that may provide more direct assessment of physical two-qubit controls for the trapped-ion QSCOUT system, yet still provides an error rate for the superconducting IBM Q system that is comparable to their published rate. Additionally, RAV could easily be adapted to include gates more closely aligned with the native physical gates of any system, including superconducting systems.

We have demonstrated the generation of RAV sequences on systems of up to $n = 8$ qubits. The bottleneck in the RAV sequence generation is the compilation of the approximate



(d)

Shots per run	*RAV fitted error per layer* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	*XEB fitted error per layer* Mean ($\times 10^{-2}$)	SD ($\times 10^{-2}$)	$\frac{\text{SD}}{\text{Mean}}$	$\frac{\text{XEB } \frac{SD}{Mean}}{\text{RAV } \frac{SD}{Mean}}$
$K = 50$	1.60	0.319	0.19	1.48	0.453	0.31	1.63
$K = 100$	1.49	0.117	0.08	1.32	0.218	0.16	2.05
$K = 500$	1.42	0.078	0.05	1.70	0.243	0.15	3.00
$K = 1000$	1.48	0.061	0.04	1.58	0.208	0.13	3.25
$K = 5000$	1.34	0.043	0.03	1.14	0.048	0.03	1.01

Figure 4.6: Experimental results from IBM Q `ibmq_manila` device for two-qubit RAV and XEB runs. Each run consists of 72 sequences executed K times each. (a,b) Results of 10 independent RAV (or XEB) runs using $K = 1000$ shots per sequence. Colored data points indicate the fidelity estimate $\hat{F}_{\mathrm{RAV}}$ (or $\hat{F}_{\mathrm{XEB}}$) for K shots of a single sequence, calculated according to Equation 4.9 (or Equation 4.1). Thin colored curves are three-parameter exponential decay fits for the data points from each run. Data points and fit curves from a particular run are assigned the same color. The thick curve is the mean of the individual fit curves. (c) Standard deviation of the fitted error per layer for RAV and XEB runs for various values of K. Smaller standard deviation indicates a more precise estimate of the error rate. (d) Fitted error per layer statistics for RAV and XEB runs for various values of K.

inversion sequence via STOQ, which scales poorly with system size (see Section 3.7) and is unlikely to be feasible for $n \gg 10$ qubits. However, this is not an inherent limitation of RAV. If a more efficient technique can be applied to generating the approximate inversion sequence, then RAV sequences could be generated for larger systems. We believe that one promising area of future work would be to adapt the efficient inversion techniques from mirror RB [126] to the context of RAV and verification of continuously-parameterized gates. This would likely require some restrictions on the construction of layers of the generated RAV circuits, such as requiring that each layer approximately implements a Clifford operation, so that the mirroring can be done efficiently.

4.8 Summary

In this chapter, we have motivated and introduced the randomized analog verification (RAV) protocol for the purpose of verifying continuously-parameterized quantum gates. In developing RAV, we have incorporated elements of randomized benchmarking (RB) and cross-entropy benchmarking (XEB) to produce a verification protocol which is both measurement-efficient and applicable to arbitrary native gate sets. A key component is the STOQ approximate compilation procedure introduced in Chapter 3, which is used to compute the inversion layer of RAV sequences. We have demonstrated the efficiency advantage of RAV both numerically and experimentally, and we have discussed its limitations and avenues for future improvement.

So far, we have been describing RAV solely in the context of the gate-based model of quantum computation. In the next chapter, we will move away from this model in order to develop RAV as a tool for the verification of analog quantum simulators.

Chapter 5

Verification of analog quantum simulators

Portions of this chapter were first published in *npj Quantum Information* [141] and are reproduced here with permission of the publisher and the authors.

5.1 Introduction

Background

Quantum simulation has long been proposed as a primary application of quantum information processing [45]. In particular, analog quantum simulation, in which the Hamiltonian evolution of a particular quantum system is directly implemented in an experimental device, is projected to be an important application of near-term quantum devices [125], with the goal of providing solutions to problems that are infeasible for any classical computer in existence. Because the obtained solutions to these problems cannot always be checked against known results, a key requirement for these devices will be the ability to verify that the desired interactions are being carried out faithfully [26, 70, 81]. If a trusted analog quantum simulator is available, then one can certify the behavior of an untrusted analog quantum simulator [158]. But in the absence of a trusted device, provable verification is essentially intractable for systems of interest that are too large to simulate classically [26]. Therefore, in the near-term, we see a need to develop pragmatic techniques to verify these devices and thus increase confidence in the results obtained.

Many experimental platforms have been used to perform analog quantum simulations of varying types, including devices based on neutral atoms [11, 9, 61], trapped ions [10, 54, 92], photons [5], and superconducting circuits [72]. In such works, verification of simulation results is typically performed by comparison to results calculated analytically or numerically in the regime where such calculation is possible. In addition, a technique for self-verification has been proposed and demonstrated [92] which measures the variance of the energy to con-

firm that the system has reached an eigenstate of the Hamiltonian. However, this technique does not verify whether the desired Hamiltonian has been implemented faithfully.

We note here that the task of *verification* of an analog quantum simulation should be distinguished from the task of *validation*. We define *verification* as the process of determining whether the device is operating according to its specification. In the case of an analog quantum simulation, the specification is the particular Hamiltonian of interest. In contrast, we define *validation* as the process of confirming whether the result correctly solves the original problem of interest. For an analog quantum simulation, this problem is likely the behavior of some natural system which is believed to be modeled by the Hamiltonian of interest. For the remainder of this chapter, we focus only on the task of *verification* and therefore consider only whether a device has faithfully implemented the Hamiltonian of interest.

One method which has been proposed for analog simulation verification is to run the dynamics forward and backward for equal amounts of time [26], commonly known as a Loschmidt echo [78, 53], which ideally returns the system to its initial state. Such a method is not able to provide confidence that the parameters of the simulation are correct, nor can it detect some common sources of experimental error such as slow environmental fluctuations or crosstalk between various regions of the physical device. However, it is naturally scalable and is straightforward to implement experimentally, provided that a time-reversed version of the analog simulation can be implemented. An extension of this method similar to randomized benchmarking has also been proposed [36], although this suffers from the same shortcomings just mentioned.

Another natural candidate for verification of analog simulations is to build multiple devices capable of running the same simulation and to compare the results across devices, which is a technique that has been demonstrated for both gate-based devices [59] and analog simulators [39]. This technique has the obvious difficulty of requiring access to additional hardware, in addition to the fact that it may be difficult to perform the same analog simulation across multiple types of experimental platforms.

Experimentalists building analog quantum simulators are in need of practical protocols for verifying the performance of these devices. Ideally such a protocol can be executed on a single device, can provide confidence that the target Hamiltonian is correctly implemented, and can be scaled to large systems. In this chapter, we aim to address these goals by introducing a set of experimentally practical approaches to the task of verifying the performance of analog quantum simulators.

Desired properties of verification protocols

The task of analog quantum simulation involves configuring a quantum system in some initial state, allowing it to evolve according to some target Hamiltonian for a particular time duration, and then analyzing one or more observables of interest. A verification protocol for this process should provide some measure of how faithfully the device implements the target

Protocol	Error sensitivity	Hardware requirements	Scalability limits
Time-reversal analog verification	Fast incoherent noise	Implement time-reversed analog simulation	None inherent
Multi-basis analog verification	Fast incoherent noise, shot-to-shot parameter fluctuation	Implement time-reversed analog simulation in alternate basis and single-qubit rotations	None inherent
Randomized analog verification	Fast incoherent noise, shot-to-shot parameter fluctuation, parameter miscalibration, crosstalk	Implement time-reversed analog simulation and ability to turn Hamiltonian terms on/off individually	Approximate inverse compilation procedure requires simulation of dynamics; protocol must be performed on subsets of larger systems

Table 5.1: Summary of characteristic error sensitivity, hardware requirements, and scalability limits for proposed verification protocols for analog quantum simulators.

Hamiltonian. We claim that a useful protocol for verification of analog quantum simulators should have the following attributes:

Sensitive to many experimental error sources. The main objective of a verification protocol is to measure experimental imperfections. If a protocol is not sensitive to some potential sources of experimental error in the simulation, it cannot give us maximal confidence in the results.

Scalable to large systems. We should not need to rely on comparison of the analog simulation results to numerically-calculated dynamics of the full system, since simulations of interest will be performed in regimes where numerical calculation is infeasible (many tens or hundreds of qubits). A useful verification protocol for such devices should be efficiently scalable to these system sizes, given reasonable assumptions.

Efficient to measure. Verification protocols should ideally leave the system in or near a measurement basis state, rather than in some arbitrary state. Protocols such as cross-entropy benchmarking (XEB) can be used for verification of analog quantum simulators [116, 24], but as discussed in Chapter 4, XEB requires estimation of the full output distribution. Leaving the system near a measurement basis state allows characterization of the final state by estimating the output probability of a single state, which requires fewer measurements.

Applicable to analog quantum simulators. Unlike many benchmarking protocols for digital, gate-based quantum computers, we are not seeking a protocol which can give fine-grained information about the fidelity of individual gates, but rather an approach which can give us information about the reliability of an arbitrary analog simulation.

Overview of verification protocols

In this chapter, we propose a set of three verification protocols for analog quantum simulators which exhibit many of these attributes. These are illustrated in Figure 5.1. The overarching

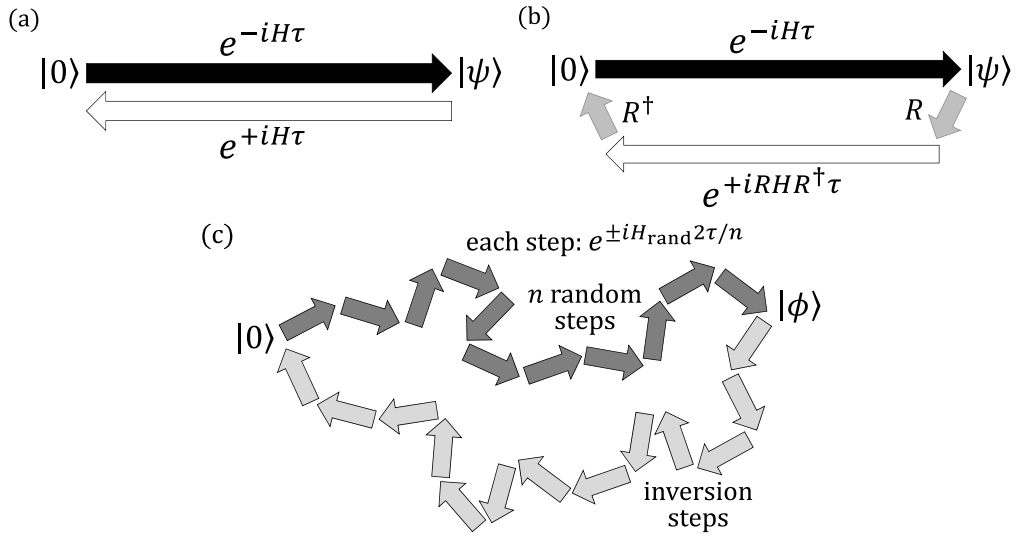


Figure 5.1: Illustration of verification protocols for analog quantum simulators. Various protocols yield information about the accuracy of a quantum simulator by propagating a state along a closed loop and verifying to what degree the system returns to its original state, labeled here as $|0\rangle$. The state $|\psi\rangle$ denotes the state of the system after applying the dynamics of Hamiltonian H for a time τ, whereas the state $|\phi\rangle$ denotes an arbitrary state.
(a) *Time-reversal analog verification:* Running an analog simulation forward in time, followed by the same analog simulation backward in time.
(b) *Multi-basis analog verification:* Running an analog simulation forward in time, rotating the state, performing the backward simulation by an analog version in the rotated basis, and finally rotating the state back.
(c) *Randomized analog verification:* Running a random sequence of subsets of the Hamiltonian terms (denoted as H_{rand}), followed by an inversion sequence of subsets of the Hamiltonian terms which has been calculated to return the system approximately to a basis state.

strategy for each protocol, inspired by the Loschmidt echo procedure, involves asking the simulator to evolve a system from some known initial state through a closed loop in state space, eventually returning to its initial state. By using a basis state as the initial (and final) state, we can efficiently measure the success of this procedure. A number of strategies exist to construct such a closed loop, with varying pros and cons. We use a few of these strategies to construct the proposed verification protocols. These protocols are summarized in Table 5.1, including some types of experimental noise to which each protocol is sensitive, the hardware requirements for implementing each protocol, and the scalability constraints of each protocol.

First, as detailed in Section 5.2, we propose a time-reversal analog verification protocol, in which the simulation is run both forward and backward in time. As illustrated in Fig-

ure 5.1(a), this approach simply performs a Loschmidt echo to reverse the time dynamics of the simulation and then verifies that the system has returned to its initial state. However, because the system traverses the same path in state space in the forward and backward directions, it is insensitive to many types of experimental errors, including systematic errors such as miscalibrations in the Hamiltonian parameters or crosstalk between sites.

To increase the susceptibility to systematic errors, as detailed in Section 5.3, we propose a multi-basis analog verification protocol, as shown in Figure 5.1(b). This is a variant of the time-reversal protocol in which a global rotation is performed on the system after the completion of the forward evolution, and the backward evolution is then performed in the rotated basis. Because this requires a physical implementation of the analog simulation in an additional basis, it will provide sensitivity to any systematic errors that differ between the two bases. For example, errors due to some types of shot-to-shot noise may be enhanced and not cancel out as in the previous protocol.

However, we note that the previous two protocols may still be insensitive to many types of errors, such as miscalibration or the presence of unwanted constant interaction terms, as discussed in Section 5.2 and Section 5.3. To address this, as described in Section 5.4, we introduce a randomized analog verification protocol, which consists of running randomized analog sequences of subsets of the target Hamiltonian terms, as depicted in Figure 5.1(c). In particular, we choose a set of unitary operators consisting of short, discrete time steps of each of the terms of the Hamiltonian to be simulated, which may be in either the forward or backward direction. We randomly generate long sequences of interactions, each consisting of a subset of these unitary operators, which evolves the system to some arbitrary state. We then use a Markov chain Monte Carlo search technique to approximately compile an inversion sequence using the same set of unitary operators, such that after the completion of the sequence, the system is measured to be in a basis state with high probability. This scheme is an adaptation of traditional gate-based randomized benchmarking techniques [40, 91] for use in characterizing an analog quantum simulator. A key difference is that for a general set of Hamiltonian terms, finding a non-trivial exact inversion of a random sequence is difficult, which is why we instead find an approximate inversion sequence. In principle, this approximation is a limitation on the precision with which this protocol can be used to verify device performance. However, in practice, the search technique can be used to produce inversion sequences that return a large percentage (e.g., 99% or more) of the population to a particular basis state, which is enough for the protocol to be useful on noisy near-term devices, since even the most accurate analog quantum simulations typically have fidelities that decay far below this level [101].

We note that the approximate inversion sequence is not strictly necessary, as protocols such as cross-entropy benchmarking (XEB) may be used to estimate the fidelity of an arbitrary state by estimating the output distribution experimentally and comparing to the ideal expected distribution [116, 24]. Such an approach can be used for systems up to around 50 qubits. However, the results from Chapter 4 demonstrate the measurement efficiency advantage that results from applying an approximate inversion sequence, which is why we use this approach here.

Interpreting the results

Each verification protocol can then be executed over a range of simulation times, and the measurement results will provide the success probability of each protocol as a function of simulation time. For a system that implements the target Hamiltonian perfectly, one expects this probability to remain constant, with a small offset from unity due to state preparation and measurement errors, as well as the approximation error for the inversion sequence in the randomized protocol. But if the system dynamics are not perfect, one expects the success probability to decrease as a function of simulation time.

For standard randomized benchmarking protocols, the shape of the decay curve provides additional information about the errors, for example, allowing one to distinguish whether the dominant error source affecting the dynamics is Markovian or non-Markovian. For typical incoherent noise, one expects this to be an exponential decay, but for noise that is non-Markovian [42, 154] or low-frequency [46], the decay curve may be non-exponential.

However, in general, we make no strong claim about the shape of the decay curves resulting from the analog verification protocols. In particular, randomized benchmarking requires that the gate set must form an ϵ-approximate unitary 2-design [34], which requires that the gate set in some sense be "evenly distributed" over the space of all possible unitaries. This is a property that holds not only for the Clifford group but also for any universal gate set, given that the randomly generated sequences are long enough [65]. However, the time-evolution operator generated by a fixed Hamiltonian cannot approach a unitary 2-design without adding a disorder term [36], which means that we cannot directly apply randomized benchmarking theory for any of the verification protocols discussed in this chapter.

Nonetheless, the decay curves still contain potentially useful information about the reliability of the analog quantum simulator. The protocols could be used as a tool to assist in calibrating an analog simulation by attempting to minimize the decay. Also, since each protocol has different sensitivities to errors, comparing decay curves from the various protocols may give clues to an experimentalist about the types of errors that are present.

Detecting errors

In this chapter, we treat noise sources in an analog quantum simulation as modifications of the target Hamiltonian. Physically, these could be caused by variations in quantities such as laser intensity, microwave intensity, magnetic fields, or other terms which could create undesired interactions with the system. We can then represent the full Hamiltonian implemented by the system as

$$\tilde{H}(t) = H + \delta H(t), \tag{5.1}$$

where H is the target Hamiltonian to be simulated, which we assume is time-independent, and

$$\delta H(t) = \sum_k \lambda_k(t)\, \delta H_k \tag{5.2}$$

represents any unwanted time-dependence and other miscalibrations present in the physical system. We assume that each $\lambda_k(t)$ varies on some characteristic timescale t_k. For example, if $\lambda_k(t)$ is a stationary Gaussian process, then t_k may be the decay time of the autocorrelation function $R(t) = \langle \lambda_k(0)\lambda_k(t) \rangle$. We note that there are several distinct regimes:

Miscalibrations. $t_k \gg N\tau$, where N is the number of repetitions performed in a quantum simulation experiment, and τ is the total runtime of each repetition. This regime corresponds to miscalibrations, unwanted interactions, and other noise that varies on a very slow timescale.

Slow noise. $N\tau > t_k > \tau$. This corresponds to noise that causes fluctuations from one run of the experiment to the next, but is roughly constant over the course of a single experiment, i.e., shot-to-shot noise.

Fast noise. $t_k \ll \tau$. This is the type of fluctuation that is most commonly referred to as "noise", i.e., fluctuations in parameters that are much faster than the timescale of a single experiment.

We design verification protocols to detect different subsets of these noise types: the time-reversal analog verification protocol for detecting fast noise, the multi-basis analog verification protocol for additionally detecting some types of slow noise, and finally the randomized analog verification protocol for detecting miscalibrations and other unwanted interactions. These protocols are described and demonstrated in the remainder of this chapter.

5.2 Time-reversal analog verification

Protocol specification

The time-reversal analog verification protocol consists of the following steps, repeated for various values of τ, which should range over the characteristic time scale of the simulation to be tested:

Step 1. Initialize the system state to an arbitrarily-chosen basis state $|i\rangle$.

Step 2. Apply the analog simulation for time τ, that is, apply the unitary operator $e^{-iH\tau}$, which ideally takes the system to the state $|\psi\rangle$. (We use the convention $\hbar = 1$ here and throughout this chapter.)

Step 3. Apply the analog simulation with reversed time dynamics for time τ, that is, apply the operator $e^{+iH\tau}$, which ideally takes the system to the state $|i\rangle$.

Step 4. Measure the final state in the computational basis. Record the probability that the final state is measured to be $|i\rangle$.

After repeating these steps for various values of τ, a decay curve can be plotted which indicates the success probability of finding the system in the desired state as a function of simulation time.

Sensitivity to errors

We first note that this protocol does not provide verification of the values of any time-independent Hamiltonian parameters, because if $\tilde{H}$ is time-independent, $e^{i\tilde{H}\tau}e^{-i\tilde{H}\tau} = \mathbb{1}$ regardless of whether $\tilde{H}$ is actually the desired Hamiltonian. It does, however, provide sensitivity to fast, incoherent noise that affects the system on a timescale shorter than the simulation time, and it also will detect imperfections in the implementation of the time-reversal itself.

More formally, the forward time-evolution operator from time 0 to τ can then be written explicitly in terms of a Dyson series as

$$U_{\text{fwd}}(0,\tau) = \mathcal{T}e^{-i\int_0^\tau dt\,(H+\delta H(t))}, \tag{5.3}$$

where $\mathcal{T}$ is the time-ordering operator. The reverse time-evolution operator from time τ to 2τ is then

$$U_{\text{rev}}(\tau,2\tau) = \mathcal{T}e^{+i\int_\tau^{2\tau} dt\,(H+\delta H(t))}. \tag{5.4}$$

It is apparent that if the noise terms in the Hamiltonian are constant between times 0 and 2τ, i.e., if $\delta H(t) = \delta H$, then we have

$$U_{\text{rev}}(\tau,2\tau)\,U_{\text{fwd}}(0,\tau) = e^{+i\tau\,(H+\delta H)}\,e^{-i\tau\,(H+\delta H)} = \mathbb{1} \tag{5.5}$$

and thus applying the forward and reverse time-evolution operators will return the system to its initial state.

However, this is not true in general if the noise terms have time-dependence. We can illustrate this by making a simplifying assumption that the noise is piecewise constant between times 0 and 2τ as

$$\delta H(t) = \begin{cases} \delta H_1 & 0 \le t < \tau \\ \delta H_2 & \tau \le t < 2\tau \end{cases} \tag{5.6}$$

where δH_1 and δH_2 are non-commuting in general. We then perform a first-order Baker-Campbell-Hausdorff approximation, which shows that

$$\begin{aligned} &U_{\text{rev}}(\tau,2\tau)\,U_{\text{fwd}}(0,\tau) \\ &\qquad = e^{+i\tau\,(H+\delta H_2)}\,e^{-i\tau\,(H+\delta H_1)} &(5.7)\\ &\qquad \approx e^{+i\tau(\delta H_2-\delta H_1+[H+\delta H_1,\,H+\delta H_2]/2)}. &(5.8)\end{aligned}$$

In the general case where $\delta H_1 \neq \delta H_2$, this quantity will not be equal to the identity. A similar argument also holds if the noise terms vary on faster timescales. That is, if $\delta H(t)$ contains one or more noise terms such that $\lambda_k(t)$ has a correlation time $t_k \ll \tau$, then the product of the forward and reverse time-evolution operators will not be equal to the identity in general, and the system will not return to its initial state.

Scalability

The time-reversal analog verification protocol requires only that the analog quantum simulator is capable of implementing the time-reversed dynamics of the desired simulation, that is, the signs of each of the Hamiltonian terms can be negated. Because there are no numerical calculations required, the protocol is independent of the size of the system, and its scalability has no inherent limitations, outside of any physical limitations involved in implementing the analog simulation itself in both directions.

5.3 Multi-basis analog verification

Protocol specification

The multi-basis analog verification protocol consists of the following steps, repeated for various values of τ, which should range over the characteristic time scale of the simulation to be tested:

Step 1. Initialize the system state to an arbitrarily-chosen basis state $|i\rangle$.

Step 2. Apply the analog simulation for time τ, that is, apply the unitary operator $e^{-iH\tau}$, which ideally takes the system to the state $|\psi\rangle$.

Step 3. Apply a basis transformation R to the system to take it to the state $R|\psi\rangle$, with R chosen such that both R and the rotated inverse Hamiltonian

$$H' = RHR^\dagger \tag{5.9}$$

are implementable. For example, if the target Hamiltonian is

$$H = \sigma_x^{(1)}\sigma_x^{(2)}, \tag{5.10}$$

one could choose

$$R = \sqrt{\sigma_y}^{(1)} + \sqrt{\sigma_y}^{(2)} \tag{5.11}$$

if and only if the analog quantum simulator can physically implement the interactions R, H, and

$$H' = RHR^\dagger = \sigma_z^{(1)}\sigma_z^{(2)}. \tag{5.12}$$

Step 4. Apply the analog simulation in the rotated basis and with reversed time dynamics for time τ, that is, apply the operator $e^{+iH'\tau}$, which ideally takes the system to the state $R|i\rangle$.

Step 5. Apply the inverse of the rotation performed in Step 3, that is, apply a global $-\pi/2$ rotation $R^\dagger$ to the system, which ideally takes the system back to the initial state $|i\rangle$.

Step 6. Measure the final state in the computational basis. Record the probability that the final state is measured to be $|i\rangle$.

After repeating these steps for various values of τ, a decay curve can be plotted which indicates the success probability of finding the system in the desired state as a function of simulation time.

Sensitivity to errors

We note that this protocol will detect errors such as miscalibrations or slow fluctuations if the strength of these errors differs in the two bases. Specifically, if $\tilde{H}$ and $\tilde{H}'$ are the implementations in the two bases which contain noise terms $\delta H(t)$ and $\delta H'(t)$, respectively, then the forward and reverse time-evolution operators can be written as

$$U_{\text{fwd}}(0,\tau) = \mathcal{T} e^{-i\int_0^\tau dt\,(H+\delta H(t))}, \tag{5.13}$$

$$U_{\text{rev}}(\tau,2\tau) = \mathcal{T} e^{+i\int_\tau^{2\tau} dt\,(H'+\delta H'(t))}. \tag{5.14}$$

Then, even in the simplest case where we have time-independent noise terms $\delta H(t) = \delta H$ and $\delta H'(t) = \delta H'$, we see that applying the forward and reverse time-evolution operators and the appropriate basis-change operators R and $R^\dagger$, gives

$$\begin{aligned} &R^\dagger\, U_{\text{rev}}(\tau,2\tau)\, R\, U_{\text{fwd}}(0,\tau) \\ &\qquad = R^\dagger\, e^{+i\tau\,(H'+\delta H')} R\, e^{-i\tau\,(H+\delta H)} &(5.15)\\ &\qquad \approx e^{+i\tau(\delta H''-\delta H+[H+\delta H,\, H+\delta H'']/2)}, &(5.16)\end{aligned}$$

where we have defined

$$\delta H'' = R^\dagger\, \delta H'\, R \tag{5.17}$$

as the rotation of $\delta H'$ into the original basis, and where we use the fact from Equation 5.9 that $R^\dagger H' R = H$. We assume here for simplicity that R and $R^\dagger$ are implemented ideally.

We observe again that the resulting quantity is not equal to the identity in the general case where $\delta H \neq \delta H''$, as well as in the cases where δH and $\delta H''$ are non-commuting with each other or with H. So we can conclude that in the case that the noise terms $\delta H(t)$ and $\delta H'(t)$ vary independently of each other, even if their correlation times are much longer than the timescale of a single experiment, the system will not return to its initial state when these time-evolution operators are applied.

Scalability

The multi-basis analog verification protocol requires that the analog quantum simulator implements the desired Hamiltonian in at least two separate bases. For example, a trapped-ion quantum simulator may implement a nearest-neighbor coupling term using both a $\sigma_x\sigma_x$ Mølmer-Sørensen interaction [145] and a $\sigma_z\sigma_z$ geometric phase gate interaction [37], which are equivalent up to a basis change. Likewise, a simulator based on superconducting qubits could implement entangling interactions in multiple bases, for example, bSWAP interactions using different phases of the microwave drive [123]. (Alternatively, if the device cannot implement the analog simulation in a different basis, but does implement a full universal gate set for quantum computation, the Hamiltonian may be implemented in a digital manner in an alternate basis via Trotterization.)

In addition to the multi-basis requirement, the device must also have the ability to perform single-qubit rotations in order to make the necessary basis change. But there are no numerical calculations required in advance, and thus the protocol itself is independent of the size of the system and has no inherent scalability limitations, outside of any limitations in performing the actual analog simulation in the two necessary bases.

5.4 Randomized analog verification

Motivation

It turns out that the previous two protocols cannot detect all types of errors. Most notably, neither protocol verifies that the simulation actually implements the target Hamiltonian H. Errors due to parameter miscalibration or the presence of unwanted constant interaction terms may not be detectable using these schemes.

To address this, we introduce a third protocol, which consists of running randomized analog sequences of subsets of the target Hamiltonian terms. In particular, we choose a set of unitary operators consisting of short, discrete time steps of each of the terms of the Hamiltonian to be simulated. We randomly generate long sequences of interactions, each consisting of a subset of these unitary operators, which evolves the system to some arbitrary state. We then use a stochastic search technique to approximately compile the inverse of these sequences using the same set of unitary operators, which produces another sequence of interactions. When appended to the original sequence the system returns to the initial state (or another basis state) with high probability.

This protocol is inspired by randomized benchmarking (RB) protocols, which are often used for characterization of gate-based devices [40, 91, 103, 105, 104, 49, 47]. Most commonly, RB involves generating many random sequences of Clifford gates and appending to each sequence an inversion Clifford. Ideally, in the absence of errors, the execution of each sequence should return all of the population to a well-known basis state. Measuring the actual population of the desired basis state after the execution of each sequence allows one to calculate a metric related to the average gate fidelity of the device, which can be used to compare the performance of a wide variety of physical devices.

We note that traditional RB has limited scalability due to the complexity of implementing multi-qubit Clifford gates, and has been demonstrated only for up to three qubits [109]; however, RB-like protocols have been demonstrated on larger systems [127, 43].

Figure 5.2 contains an illustration comparing the randomized analog verification protocol with the traditional Clifford-based RB protocol. We note that this protocol significantly differs from a recently-proposed technique for benchmarking analog devices [36] in that we construct the approximate inversion sequence independently of the initial randomly-generated sequence, which in general prevents miscalibrations and constant errors from cancelling out during the inversion step.

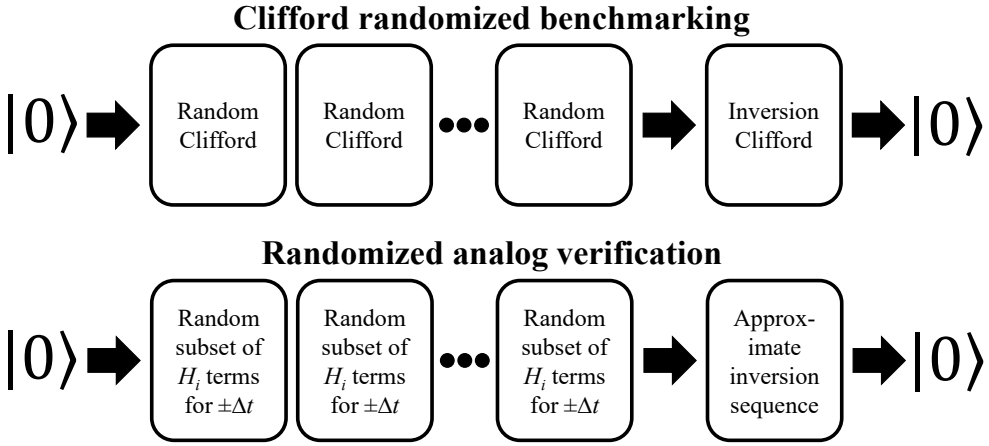


Figure 5.2: High-level comparison of traditional randomized benchmarking and the randomized analog verification protocol. Both protocols involve generating a sequence that starts and ends in a known basis state, which is denoted $|0\rangle$ in this figure for simplicity, and proceed by simply making a series of random choices. For traditional RB, the inversion Clifford is calculated deterministically based on the preceding sequence of random Cliffords. For randomized analog verification, the inversion sequence is compiled approximately via a stochastic search procedure.

Protocol specification

We write the target Hamiltonian as a sum of terms

$$H = \sum_{i=1}^{m} H_i, \tag{5.18}$$

where we assume that the simulator can enable both the forward and time-reversed version of each H_i independently of the others. We note that this protocol, in addition to being sensitive to implementation errors in the time-reversal, will also be affected by experimental errors in the enabling or disabling of the individual Hamiltonian terms.

We then repeat the following steps for various values of τ, which is the time scale on which the sequence will operate and should range over the characteristic time scale of the simulation to be tested:

Step 1. Randomly choose an initial basis state $|i\rangle$.

Step 2. Generate n random subsets (e.g., $n = 100$) of the terms of the target Hamiltonian, and define

$$H_{\mathrm{rand},k} = \sum_{\substack{i \in \text{random subset} \\ \text{of } \{1,2,\ldots,m\}}} H_i \tag{5.19}$$

as the sum of the terms in subset k. To increase the randomness of the resulting path, choose also the direction (forward or time-reversed) of each subset at random. Apply each of the resulting unitary time-evolution operators, i.e.,

$$U_k = e^{\pm iH_{\text{rand},k}2\tau/n} \tag{5.20}$$

for $k = 1$ to n, to the initial state $|i\rangle$, which evolves the system to an intermediate state $|\phi\rangle$.

Step 3. Calculate another sequence of these random unitaries that will approximately invert the process and act on $|\phi\rangle$ to produce a basis state $|f\rangle$ within some target fidelity, e.g., 0.99. Apply the sequence, which ideally will take the system to the final state $|f\rangle$ with probability of at least the desired target fidelity.

Step 4. Measure the final state in the computational basis. Record the probability that the final state is measured to be $|f\rangle$.

After repeating these steps for various values of τ, the resulting decay curve indicates the success probability of finding the system in the desired state after executing the randomized sequences as a function of effective simulation time.

Approximate inversion layer compilation

Calculating an appropriate inversion layer, using only small time steps of the Hamiltonian terms as building blocks, is the most computationally intensive part of this protocol. We cannot directly reverse the random sequence generated, since this would simply be a time-reversal, and errors such as miscalibrations or shot-to-shot noise would cancel out. Instead, we generate a new sequence by explicitly calculating the product of the random sequence of unitaries and then building a sequence which inverts it.

Since compiling an exact inversion layer (outside of simply reversing the random sequence) is likely infeasible, we allow the inversion layer to only approximately invert the original sequence, such that we return nearly all of the population to a basis state. We note that the approximate nature still allows us to assess the quality of the simulation with the targeted precision using a single measurement basis.

To construct the inversion layer, we use the STOQ protocol for approximate compilation introduced in Section 3.4, which is a stochastic Markov chain Monte Carlo (MCMC) search technique using a Metropolis-like algorithm. This is a randomized approach to compiling an arbitrary unitary into a sequence of "gates" drawn from a finite set of allowed unitaries, similar to the approach used in a proposed technique for quantum-assisted quantum compiling [87].

Specifically, since the set of allowed unitaries here consists of all possible random subsets of the Hamiltonian terms, we have the following procedure for approximately compiling the inversion layer (illustrated in Figure 5.3):

1. Generate n randomized layers, each of which determines a unitary operation U_k, as defined in Equation 5.20.

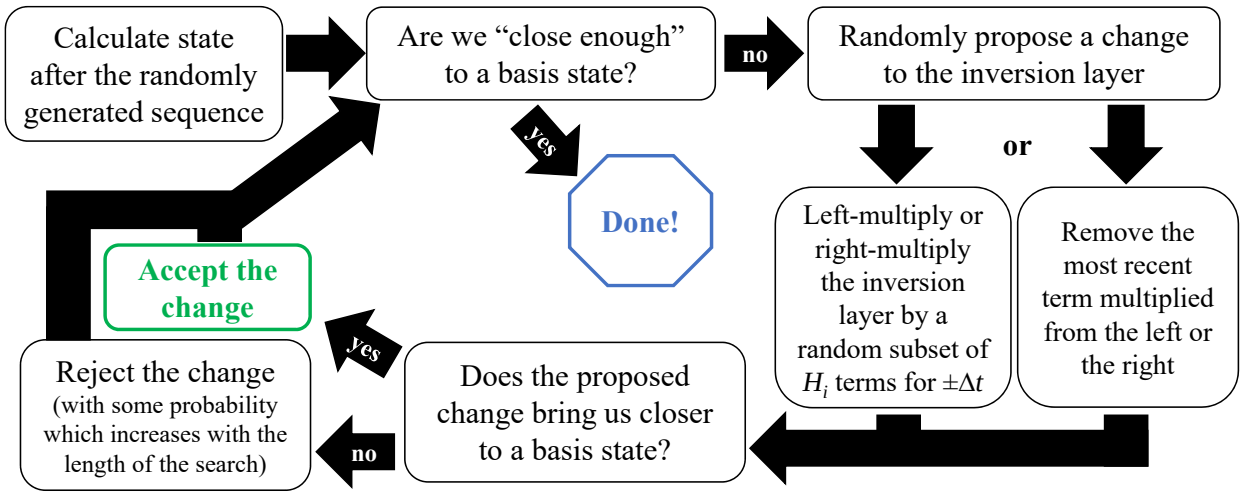


Figure 5.3: Flowchart for the approximate unitary compilation procedure used in randomized analog verification for analog quantum simulators, specialized from the flowchart for STOQ presented in Figure 3.2. The time duration $\pm\Delta t$ refers to the time and direction of each individual step in the sequence, i.e., the desired total simulation time τ divided by n, the total number of steps in the sequence.

2. Calculate the state after applying all n of the randomized layers to the initial state as

$$|\phi\rangle = U_n U_{n-1} \cdots U_2 U_1 |i\rangle. \tag{5.21}$$

3. Build up a new sequence of layers, which will become the inversion layer, by incrementally adding a randomized layer or removing a layer from the beginning or end of the sequence (such that we only have to perform one multiplication per proposed step). Let the product of these layers be U_{inv}.

4. For each proposed addition or removal, look at the basis state of $U_{\text{inv}}|\phi\rangle$ with the largest population fraction to see if it has increased or decreased from the prior state. If it has increased, the system is closer to a basis state, and therefore accept the proposed addition or removal. If it has decreased, usually reject it, but sometimes accept it, based on the value of the MCMC annealing parameter β. (Refer to Section 3.4 for complete implementation details.)

5. Continue until the largest basis state population reaches some desired threshold (e.g., 0.99), which determines the population fraction in the final basis state after executing the compiled sequence.

In order to increase the distinction between this compiled inversion sequence and the original randomly-generated sequence (which seems desirable in order to avoid potentially

cancelling out any systematic errors), we initialize the STOQ algorithm with a large value of the annealing parameter β, which increases the randomness in the early part of the compiled sequence. Over time, we linearly decrease the value of β until the process finally converges toward a basis state.

Notice that because this procedure simply takes us approximately to some basis state (not necessarily the initial state), a true inversion sequence would require a final local rotation of the appropriate qubits to take the system back to the initial state. However, since the intention is simply to measure the resulting state, this final rotation is unnecessary – we can just measure the state and compare the result to the expected final basis state, rather than comparing to the initial basis state.

Because this process is randomized, it is not guaranteed to converge [131]. To account for this, in the implementation used for this chapter, we launch many tens of STOQ compilations in parallel, which in practice typically allows the search to succeed in reasonable time. For example, in the five-qubit numerical simulation described in Section 5.6, when the original sequence has $\sim$100 random layers, one of the compilations will typically converge to the desired accuracy of 98% within a few thousand steps.

Scalability

The scalability of the randomized analog verification protocol is limited by the approximate compilation of the inversion layer. Performing this compilation requires many explicit multiplications of unitary operators acting on the full Hilbert space of the system being simulated, and thus has at least the same complexity as actually simulating the dynamics of the system. Unless a reliable quantum computer is available [87], this must be done on a classical computer, and so it is likely infeasible to apply this protocol directly to systems with more than tens of qubits.

To apply this protocol to large-scale simulations, we can break the full system into subsystems [49, 57] to reduce the exponential scaling to polynomial scaling. Specifically, if the Hamiltonian is k-local, we can decompose the system into subsystems of size $s \geq 2k$ (see Figure 5.4), and then run this protocol on every subsystem. This will test every interaction term, as well as potential errors such as crosstalk that may occur between any two distant interaction terms in the system. The number of such subsystems grows only polynomially with degree s, not exponentially. Since this is equivalent to testing each subsystem of size s independently, the downside of this approach is the loss of sensitivity to errors that may occur only for subsystems of size larger than s; however, in many systems, it is likely reasonable to assume that such errors are small. Additional work will be needed to understand exactly what claims one can make about the performance of the large-scale analog simulation by characterizing the subsystems in this way.

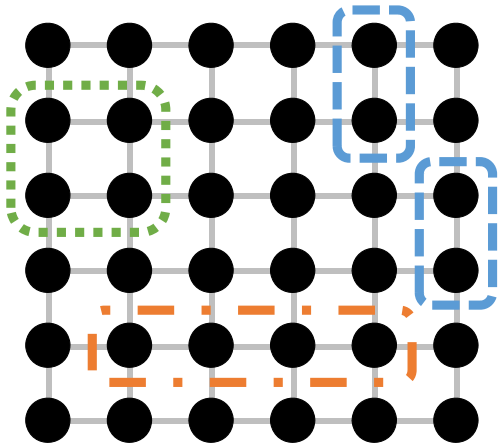


Figure 5.4: Illustration of 6x6 2-D lattice with nearest-neighbor coupling. Here the Hamiltonian is k-local with $k = 2$. Colored dashed outlines show possible subsets of size $s = 4$, each of which is formed from two (possibly distant) pairs of connected neighbors. The total number of ways to choose such subsets from this lattice is 3,540. Running the randomized analog verification protocol on all such subsets (or a selection of randomly-chosen subsets) will test for errors associated with each interaction term in the Hamiltonian, as well as errors that may be caused by unwanted interaction (e.g., crosstalk) between any two pairs of sites in the system.

5.5 Experimental demonstrations

To demonstrate the feasibility of implementing these verification protocols experimentally, we choose a simple two-site Ising model with transverse field

$$H = -\frac{1}{2}b\left(\sigma_y^{(1)} + \sigma_y^{(2)}\right) - \frac{1}{2}J\sigma_x^{(1)}\sigma_x^{(2)} \tag{5.22}$$

with $J = 2\pi \times 139$ Hz and $b = 2\pi \times 227$ Hz. We implement this model in a trapped-ion analog quantum simulator containing two $^{40}Ca^+$ ions. We use the electronic $S_{1/2}$ ground orbital and $D_{5/2}$ metastable excited orbital as the qubit states, and we drive transitions between these states using a 729 nm laser [63]. In particular, we choose $|g\rangle = |S_{1/2}, m_j = -1/2\rangle$ and $|e\rangle = |D_{5/2}, m_j = -1/2\rangle$ as the states of the two-level system.

We prepare the system in the state $|eg\rangle$ or $|ge\rangle$ by optically pumping the ions to the state $|gg\rangle$, using a π-pulse with a laser beam localized to a single ion to prepare the state $|eg\rangle$, and then optionally a π-pulse with a laser beam addressing both ions to prepare the state $|ge\rangle$.

We then implement the Ising model by combining three tones in a laser beam that addresses both ions equally. In particular, we realize the transverse field interaction via a laser tone resonant with the qubit transition frequency with Rabi frequency Ω_C. This

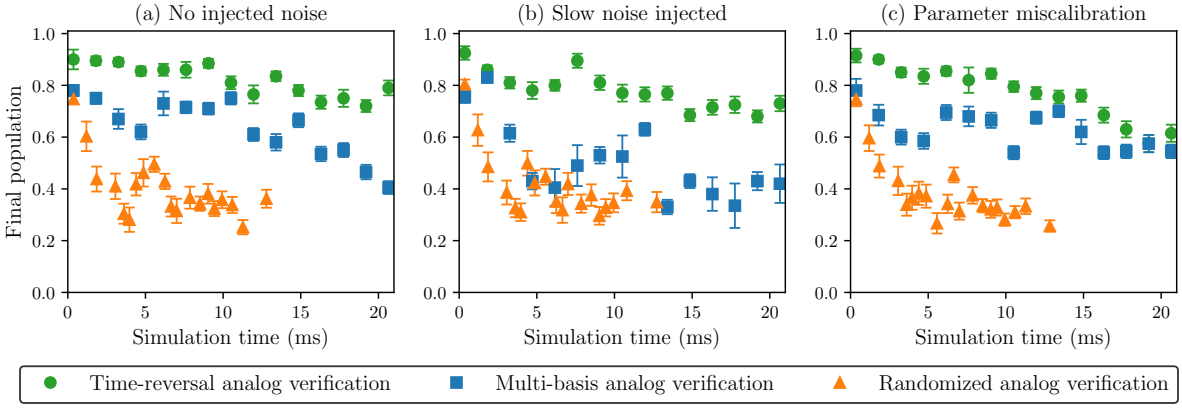


Figure 5.5: Experimental results of verification protocols. Results are for the two-site Ising model from Equation 5.22, with $J = 2\pi \times 139$ Hz and $b = 2\pi \times 227$ Hz. Each plot shows the experimentally-measured population in the expected final state after running each of the verification protocols under the specified type of injected noise. Data points represent raw experimental results and include experimental errors due to state preparation, measurement, and imperfect control. For the time-reversal and multi-basis analog verification protocols, each data point represents the distribution of measured results over 200 independent runs. For the randomized analog verification protocol, each data point represents the distribution of measured results of 10 different randomly generated sequences, with each sequence executed 100 times. Error bars indicate standard error of the mean.

creates the desired $(b/2)(\sigma_y^{(1)} + \sigma_y^{(2)})$ interaction with $b = \Omega_C$. In addition, we implement the site-site coupling via a Mølmer-Sørensen interaction [145] via the axial stretch vibrational mode with $\omega_{\text{ax}} \approx 2\pi \times 1.514$ MHz, where we apply two laser tones detuned from the qubit transition frequency by $\pm(\omega_{\text{ax}} + \delta_{\text{MS}})$, with $\delta_{\text{MS}} = 2\pi \times 80$ kHz, and where each tone has Rabi frequency Ω_{MS}. This creates an effective $(J/2)\sigma_x^{(1)}\sigma_x^{(2)}$ interaction with $J = \eta_{\text{ax}}^2 \Omega_{\text{MS}}^2 / \delta_{\text{MS}}$, where $\eta_{\text{ax}} \approx 0.08$ is the Lamb-Dicke parameter indicating the coupling of the laser beam to the axial mode of the ion crystal, and we tune Ω_{MS} to produce the desired value of the coupling strength J.

In addition to designing the analog simulation itself, we must also implement the time-reversed and rotated versions of the simulation in order to implement the desired verification protocols. For the time-reversal analog verification protocol, we take H to $-H$ by shifting the phase of the resonant tone by π, which takes b to $-b$ in the transverse field interaction, and by changing the Mølmer-Sørensen detuning from δ_{MS} to $-\delta_{\text{MS}}$ (with a small correction to account for a change in AC Stark shift), which takes J to $-J$ in the effective $\sigma_x^{(1)}\sigma_x^{(2)}$ interaction.

For the multi-basis analog verification protocol, we choose the basis rotation

$$R = R_z^{(1)}(\pi/2) + R_z^{(2)}(\pi/2), \tag{5.23}$$

which is a global $\pi/2$ rotation around the z-axis. We implement R physically via a sequence of single-qubit carrier rotations, using the fact that

$$R_z(\pi/2) = R_y(-\pi/2)R_x(\pi/2)R_y(\pi/2). \tag{5.24}$$

We then must implement $RHR^\dagger$, which is the Hamiltonian in the rotated basis. For the transverse field term, we note that

$$R(\sigma_y^{(1)} + \sigma_y^{(2)})R^\dagger = \sigma_x^{(1)} + \sigma_x^{(2)}, \tag{5.25}$$

which we implement by shifting the phase of the resonant tone by $\pi/2$ as compared to the phase used to implement $\sigma_y^{(1)} + \sigma_y^{(2)}$. For the coupling term, we note that

$$R\sigma_x^{(1)}\sigma_x^{(2)}R^\dagger = \sigma_y^{(1)}\sigma_y^{(2)}, \tag{5.26}$$

which we implement by shifting the phase of the blue-sideband Mølmer-Sørensen tone by π with respect to the red-sideband tone [98].

Finally, for the randomized analog verification protocol, we write the target Hamiltonian from Equation 5.22 as

$$H = H_1 + H_2, \tag{5.27}$$

where H_1 and H_2 are defined as

$$H_1 = -\frac{1}{2}b(\sigma_y^{(1)} + \sigma_y^{(2)}), \tag{5.28}$$

$$H_2 = -\frac{1}{2}J\sigma_x^{(1)}\sigma_x^{(2)}. \tag{5.29}$$

We then generate 200 random sequences of subsets of these Hamiltonian terms in either the forward or time-reversed direction, such that each step of each sequence is selected from the set

$$H_{\text{steps}} = \{H_1, H_2, H_1 + H_2, -H_1, -H_2, -H_1 - H_2\}, \tag{5.30}$$

and each sequence consists of $10 \leq n \leq 50$ steps of length 8 μs $\leq t_{\text{step}} \leq 290$ μs. For each sequence, we then compile an approximate inversion sequence consisting of steps from the same set H_{steps}. Each sequence has a randomly-chosen initial state from the set $\{|ge\rangle, |eg\rangle\}$, and each full sequence ideally leaves the system in some basis state with at least 98% probability. The terms in the set H_{steps} are implemented experimentally by enabling or disabling the corresponding laser tones and by time-reversing the analog simulation as necessary.

To test the behavior of each of these protocols, we execute the time-reversal and multi-basis analog verification protocols for varying simulation times and execute all 200 of the randomized analog verification sequences, and we deliberately inject various types of noise in order to demonstrate the behavior of the verification protocols. The results of these experimental runs are shown in Figure 5.5. To produce these results, we executed each protocol under three different sets of experimentally-motivated noise conditions:

1. *No injected noise:* We execute each of the verification protocols after calibrating the individual interactions to approximately match the desired dynamics.

2. *Slow noise injected:* We introduce shot-to-shot fluctuations by intentionally varying the intensity of each of the three tones in the laser beam using parameters drawn from a Gaussian distribution with relative standard deviation of 3 dB. The parameter variations in the original basis are drawn independently from those in the rotated basis, which emulates the case where the system has independent noise sources in the two bases.

3. *Parameter miscalibration:* We intentionally miscalibrate the Mølmer-Sørensen detuning to $\delta_{\mathrm{MS}} = 2\pi \times 60$ kHz, which has the effect of increasing the coupling strength J by a factor of 1/3.

To provide more insight into the results of these protocols, in Figure 5.6 we plot the actual population dynamics of the analog simulation in the absence of injected noise. We observe that the implemented simulation diverges significantly from the ideal simulation after only a few milliseconds, primarily due to miscalibration and dephasing noise. We intentionally allow this divergence as a test case for the various verification protocols, since it is caused by errors that may be typical in experiments. The miscalibration here is due to laser intensities and/or frequencies that have not been optimized to produce the desired dynamics, and the dephasing noise is likely caused by the presence of global magnetic field fluctuations which cause the state to decohere when leaving the subspace $\{|ge\rangle, |eg\rangle\}$, which is a decoherence-free subspace with respect to the global magnetic field.

Also plotted in Figure 5.6 is a curve showing the fidelity between an ideal evolution of the system state and an approximation of the system state obtained experimentally. For the ideal Hamiltonian H, defined in Equation 5.22, we use the target values ($J = 2\pi \times 139$ Hz, $b = 2\pi \times 227$ Hz) and perform unitary evolution under the Schrödinger equation to obtain the dynamics of the ideal state $\rho(t) = |\psi(t)\rangle\langle\psi(t)|$, where $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$. For the experimentally-miscalibrated Hamiltonian $\tilde{H}$, we use parameters that approximately match the observed measurements ($J = 2\pi \times 250$ Hz, $b = 2\pi \times 102$ Hz) with an appropriate dephasing rate ($\gamma_\phi = 2\pi \times 38$ Hz). We then perform non-unitary evolution under the Lindblad master equation, using the Lindblad operator $L = \sqrt{\gamma_\phi/2}\,\sigma_z$ as the dephasing mechanism, to obtain the approximate dynamics of the experimentally-obtained state $\tilde{\rho}(t)$. The approximate fidelity between the ideal state and the experimentally-obtained state is then

$$\tilde{F}(t) = \left[\mathrm{tr}\,\sqrt{\sqrt{\rho(t)}\,\tilde{\rho}(t)\,\sqrt{\rho(t)}}\right]^2. \tag{5.31}$$

The fidelity curve plotted in Figure 5.6 is this approximate fidelity function $\tilde{F}(t)$, and we observe that it decays to 50% in approximately 7 ms.

Despite this fast decay of the fidelity, we note that in the absence of additional injected noise, both the time-reversal and multi-basis analog verification protocols in Figure 5.5(a)

show decay times on the order of tens of milliseconds. Because these protocols are sensitive to fast, incoherent noise, we deduce that the majority of the errors present in the experiment are slower than the timescale of each experiment and are therefore cancelled out by these protocols.

Conversely, we consider the results of the randomized analog verification protocol with no injected noise in Figure 5.5(a). The success probability decays in approximately 3 ms, which is slightly faster than the fidelity decay observed in Figure 5.6. This suggests that the randomized protocol at least detects these experimental miscalibrations or coherent errors that cause the actual simulation dynamics to differ from the ideal dynamics. That is, the randomized analog verification protocol helps to identify imperfections in the simulation with respect to the target Hamiltonian, which is something that the other protocols are unable to do. In addition, the faster decay of the randomized analog verification results as compared to the approximate fidelity curve in Figure 5.6 indicates that there are additional sources of experimental error that are not captured by the population dynamics alone. For example, the experimental procedure involves rapidly enabling and disabling the various interaction terms, which may itself introduce imperfections that cause the success probability to decay more rapidly. Indeed, the difference between the randomized analog verification protocol results with no injected noise in Figure 5.5(a) and with injected noise in Figure 5.5(b) and Figure 5.5(c) indicate that the experimental errors in the simulation dwarf the errors caused by the injected noise.

Finally, we note that a number of the experimental data series in Figure 5.5 show hints of oscillatory behavior, and that in general the shape of each decay curve is non-exponential. This is evidence supporting the claim that these protocols do not fully twirl coherent errors into incoherent errors, and thus do not produce a fully depolarizing channel that would produce an exponential decay in these results.

5.6 Numerical demonstrations

To further test the sensitivity of each protocol to various types of noise, we numerically simulated the dynamics of the verification protocols using the five-site Heisenberg model

$$H = -\frac{1}{2}\sum_{i=1}^{5} b^{(i)}\sigma_z^{(i)} - \frac{1}{2}\sum_{i=1}^{4}\left(J_x^{(i,i+1)}\sigma_x^{(i)}\sigma_x^{(i+1)} + J_y^{(i,i+1)}\sigma_y^{(i)}\sigma_y^{(i+1)} + J_z^{(i,i+1)}\sigma_z^{(i)}\sigma_z^{(i+1)}\right). \quad (5.32)$$

Nominally, we fix all parameter values as $b^{(i)} = J_x^{(i,j)} = J_y^{(i,j)} = J_z^{(i,j)} = 2\pi \times 1$ kHz, but we vary each of these parameters during the simulation according to several different types of potential experimental noise. We simulated the dynamics of each protocol under conditions with several classes of noise sources present individually:

1. *Fast incoherent noise:* The b and J terms in the Hamiltonian have fast noise, modeled as an Ornstein-Uhlenbeck process with a correlation time on the order of τ/n, which is approximately the duration of one step of the randomized analog verification protocol.

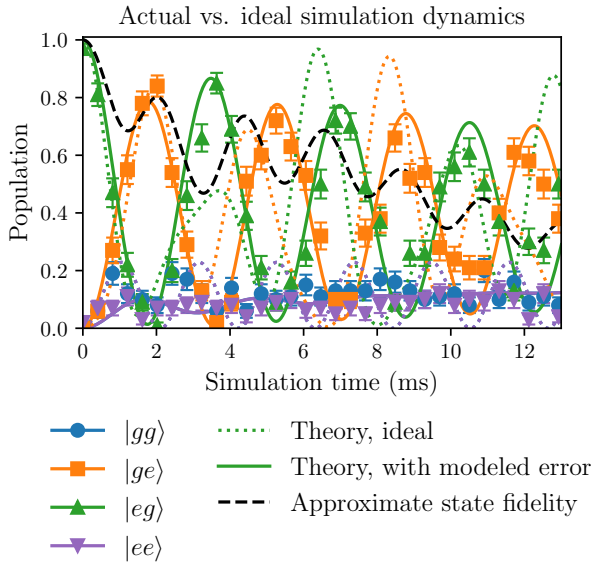


Figure 5.6: Experimentally-measured dynamics of the simulation as a function of time. Each data point is the average of 100 independent runs. Error bars indicate standard error of the mean. The dotted curves represent the ideal dynamics of a perfectly-calibrated analog simulation ($J = 2\pi \times 139$ Hz, $b = 2\pi \times 227$ Hz) in the absence of noise. The solid curves represent the theoretical dynamics of a miscalibrated analog simulation ($J = 2\pi \times 250$ Hz, $b = 2\pi \times 102$ Hz) with a dephasing rate of $\gamma_\phi = 2\pi \times 38$ Hz, where these parameters are chosen empirically as a reasonable approximation of the observed experimental data points. The dashed curve is the fidelity of the state evolved according to the miscalibrated dynamics with the state evolved according to the ideal dynamics, calculated using Equation 5.31.

2. *Slow parameter fluctuations:* The b and J terms in the Hamiltonian have slow noise (modeled as a constant miscalibration that varies from run to run with a Gaussian distribution) that has a typical timescale longer than τ, but shorter than the time between individual experiments.

3. *Parameter miscalibration:* Each of the b and J terms in the Hamiltonian is miscalibrated from the desired value.

4. *Idle crosstalk:* Each of the interaction terms in the Hamiltonian, when disabled, still drives the interaction with some fraction of the intended strength. For example, during steps of the randomized analog verification protocol in which the $\sigma_y^{(1)}\sigma_y^{(2)}$ interaction is intended to be turned off, we still include a fraction of that term in the Hamiltonian being simulated.

The numerical simulation results in Figure 5.7 demonstrate that certain types of noise,

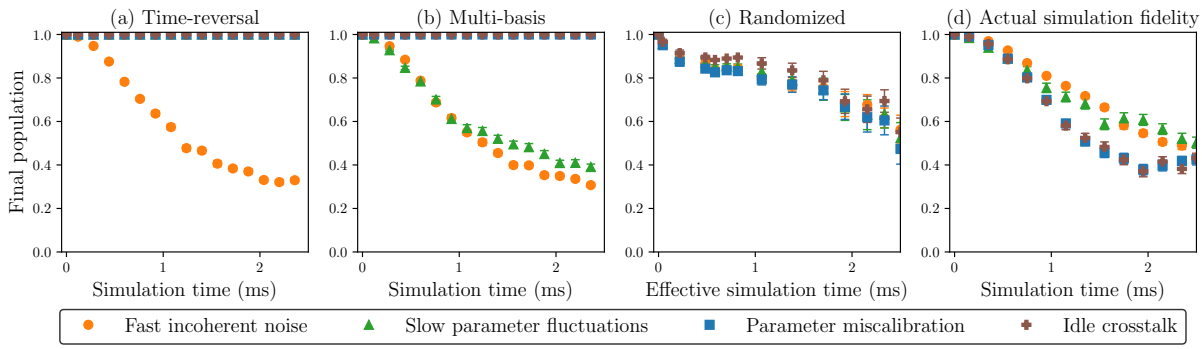


Figure 5.7: Numerical five-qubit simulation results of verification protocols under simulated noise conditions. Five-qubit numerical simulation results showing the sensitivities of each of the three analog verification protocols to four different types of experimental error sources: fast incoherent noise ($\sim\tau/n$) in Hamiltonian parameters with 30% relative standard deviation (RSD), slow fluctuations ($\gg\tau$) in Hamiltonian parameters with 15% RSD, constant miscalibration of Hamiltonian parameters with 10% RSD, and constant idle crosstalk affecting all sites with 10% RSD. The target Hamiltonian is the five-qubit Heisenberg model from Equation 5.32, with $b^{(i)} = J_x^{(i,j)} = J_y^{(i,j)} = J_z^{(i,j)} = 2\pi \times 1$ kHz. The "effective simulation time" is the average time for which each term of the Hamiltonian is enabled. Error bars indicate standard error of the mean. (a) Time-reversal analog verification results. Each data point represents the distribution of results over 50 runs. (b) Multi-basis analog verification results. Each data point represents the distribution of results over 50 runs. (c) Randomized analog verification results. Each data point represents the distribution of 10 different randomly generated sequences with $n = 150$ steps, with each sequence simulated 20 times. (d) Actual fidelity of the analog simulation under each type of noise.

such as fast incoherent noise, can be detected by any of the proposed verification protocols. We see that the multi-basis analog verification protocol is also sensitive to certain slow parameter fluctuations, whereas the randomized analog verification protocol is additionally sensitive to errors such as parameter miscalibration and crosstalk among the interaction terms in the system. Such error sources may cancel out in the forward and backward directions when using more systematic protocols [7, 38], but when using a randomized protocol they are highly unlikely to cancel due to the randomized nature of the sequence and its dependence on the exact parameters of the Hamiltonian. In particular, we see in Figure 5.7(d) that the actual fidelity of the analog simulation is most severely impacted by the parameter miscalibration and crosstalk errors, and only the randomized analog verification protocol is able to detect the presence of these errors.

To gain further insight into the behavior of the randomized analog verification protocol, we also simulated the dynamics under various types of noise using a pair of two-qubit

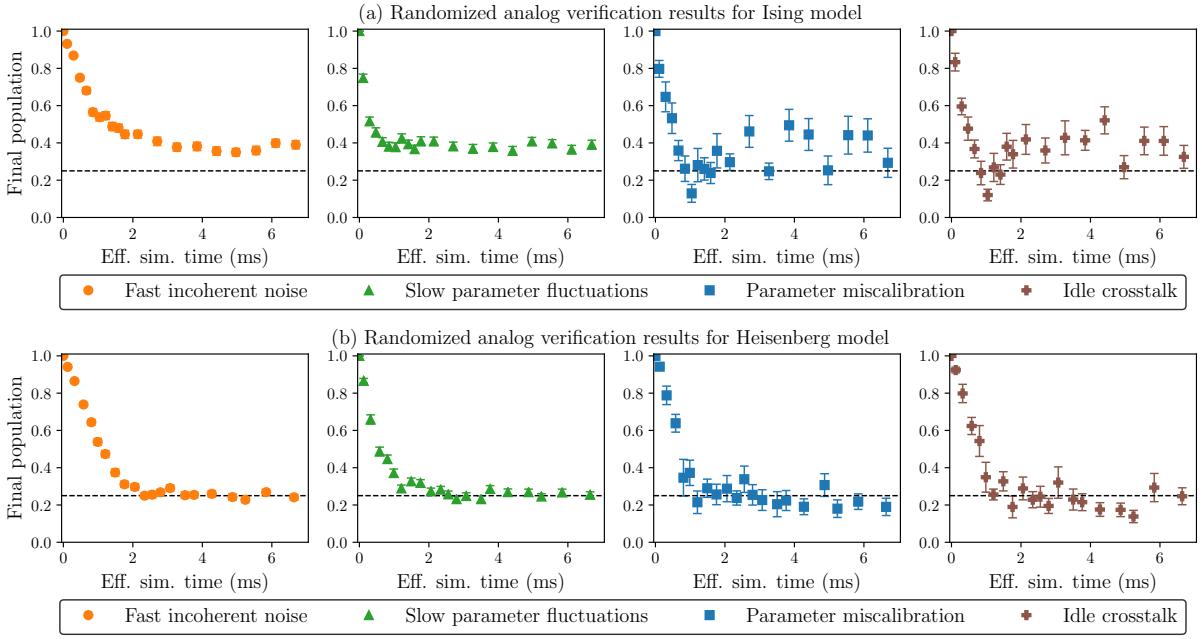


Figure 5.8: Numerical two-qubit simulation results of verification protocols for two different Hamiltonians. Two-qubit numerical simulations showing randomized analog verification results for two Hamiltonians under four different types of experimental error sources: fast incoherent noise ($\sim\tau/n$) in Hamiltonian parameters, slow fluctuations ($\gg\tau$) in Hamiltonian parameters, constant miscalibration of Hamiltonian parameters, and constant idle crosstalk affecting all sites. The "effective simulation time" is the average time for which each term of the Hamiltonian is enabled. Each data point represents the distribution of 10 different randomly generated sequences, with each sequence simulated 15 times. Error bars indicate standard error of the mean. Dashed line added at $y = 0.25$ on each plot as a visual aid.
(a) The target Hamiltonian is the one-dimensional Ising model from Equation 5.33 under fast incoherent noise with 12% relative standard deviation (RSD), slow parameter fluctuations with 6% RSD, parameter miscalibration with 3% RSD, and idle crosstalk with 3% RSD.
(b) The target Hamiltonian is the one-dimensional Heisenberg model from Equation 5.34 under fast incoherent noise with 4% RSD, slow parameter fluctuations with 2% RSD, parameter miscalibration with 1% RSD, and idle crosstalk with 1% RSD.
For both (a) and (b), we choose $b = J_x = J_y = J_z = 2\pi \times 20$ kHz. Note that larger relative errors are used in (a) to compensate for the smaller number of Hamiltonian terms in this simulation, such that the decay times of the plots in (a) and (b) are similar.

Hamiltonians. First, we use a one-dimensional Ising model with transverse field

$$H = -\frac{1}{2}\Big(b(\sigma_y^{(1)} + \sigma_y^{(2)}) + J_x\sigma_x^{(1)}\sigma_x^{(2)}\Big), \tag{5.33}$$

which is identical to Equation 5.22, the Hamiltonian used for the experiment. For the purposes of the randomized analog verification protocol, we treat $b(\sigma_y^{(1)} + \sigma_y^{(2)})$ as a single term, as was also done in the experiment.

Second, we use a one-dimensional Heisenberg model with transverse field terms along each axis

$$\begin{aligned} H = -\frac{1}{2}\Big(&b\sigma_x^{(1)} + b\sigma_y^{(1)} + b\sigma_z^{(1)} + b\sigma_x^{(2)} + b\sigma_y^{(2)} + b\sigma_z^{(2)} + \\ &J_x\sigma_x^{(1)}\sigma_x^{(2)} + J_y\sigma_y^{(1)}\sigma_y^{(2)} + J_z\sigma_z^{(1)}\sigma_z^{(2)}\Big), \end{aligned} \tag{5.34}$$

which is a simplified version of the five-qubit Hamiltonian in Equation 5.32 used for the earlier simulations.

Figure 5.8 contains the numerical simulation results of applying the randomized analog verification protocol to these two Hamiltonians under various types of noise, where we have chosen $b = J_x = J_y = J_z = 2\pi \times 20$ kHz such that the effective simulation times are much longer than the timescale of the system dynamics.

We note that the shape of the decay differs significantly between the two Hamiltonians. In particular, we observe that each of the decay curves for the Heisenberg model in Figure 5.8(b) appears to be nearly exponential in shape and decays to approximately 0.25, which is the expected result for a fully mixed two-qubit state. This is not the case for some of the decay curves for the Ising model in Figure 5.8(a).

As discussed previously, randomized benchmarking protocols produce exponential decay curves in cases where the noise is fully depolarized by the randomized circuits. We note that the "native gate set" obtained from the Heisenberg model in Equation 5.34 is a universal set of quantum gates, which forms an approximate 2-design in the limit of long sequence length. Here we are in fact operating in the limit of "long sequence length", since the dynamics occur at 20 kHz and the protocol is being performed for an effective simulation time of a few milliseconds. So the nearly-exponential shape of the decay curves in Figure 5.8(b) is a good indication that the various noise sources are indeed being depolarized under these conditions.

In contrast, the behavior of the decay curves in Figure 5.8(a), which do not decay to 0.25, can be explained by the fact that the interactions do not fully explore the state space of the system. We also observe non-monotonic behavior of these decay curves in the presence of correlated errors such as miscalibration or crosstalk, which suggests that such errors are not being fully depolarized. Such non-monotonic behavior is also observed in the experimental data in Figure 5.5.

5.7 Discussion

The set of verification protocols for analog quantum simulators introduced in this chapter are experimentally motivated, and we have demonstrated the utility of these protocols both experimentally and numerically. Most notably, we observe that the randomized analog verification protocol is superior in terms of the types of experimental errors to which it is sensitive, but that its scalability to large system sizes requires additional assumptions, such as the ability to verify subsets of the system independently, due to the classical resources required to perform the approximate inverse compilation during the generation of the randomized sequences. We also observe that the randomized analog verification protocol produces results similar to those from traditional randomized benchmarking protocols in cases where the Hamiltonian terms form a universal "native gate set" and where the simulation time is long in comparison to the system dynamics.

It is worth noting that implementing the time-reversed Hamiltonian in the analog quantum simulation device, which is required for all of the discussed verification protocols, is not necessarily trivial for general Hamiltonians that may be simulated. It turns out that the slow Mølmer-Sørensen interaction used to implement the Ising model with trapped ions is easily time-reversible, as demonstrated in the experimental results, which allowed us to demonstrate each of the protocols here without much additional effort. It is likely that many interactions of interest on other physical platforms, such as neutral atoms or superconducting qubits, may have similarly simple physical mechanisms for time-reversing the dynamics.

Ideally, verification protocols are useful not only for verifying the correct behavior of a system, but also for helping to diagnose and fix errors. In particular, an experimentalist may wish to identify not only the existence of errors in the system, but also the types and locations of these errors. Simply running the dynamics of the full simulation and checking the results may not provide the information necessary to diagnose these details. However, the protocols described in this chapter provide additional tools for the experimentalist to help characterize the errors in the system. For example, running each of the protocols and comparing the relative decay curves could help to provide insight into whether the system suffers from fast incoherent noise, slow parameter fluctuations, parameter miscalibration, and/or crosstalk errors. In addition, because each of the protocols can be run on arbitrary subsets of the full system, running each on many different subsets will help to isolate the problematic physical interactions.

5.8 Summary

This chapter has introduced three experimentally-motivated protocols for verification of analog quantum simulators and has demonstrated the feasibility of these protocols both numerically and experimentally. Taken together, these techniques allow for pragmatic evaluation of an analog quantum simulation device in a way that builds confidence that the device is not only operating consistently, but that it is also operating faithfully according to the desired

target Hamiltonian. The decay curves resulting from these protocols may then provide some insight into the type and strength of errors encountered. Such techniques can also be applied to subsets of a larger system to allow an experimentalist to characterize and diagnose the behavior in a scalable way. Future work should pursue a more detailed analysis of the information that these protocols can provide about the types of noise or errors present in the system, as well as the feasibility of applying the randomized analog verification protocol to larger systems. In addition, alternative protocols should be explored that combine the ideas in these protocols with existing techniques from the randomized benchmarking literature, with the goal of producing a practical protocol which depolarizes errors more fully and about which stronger theoretical claims can be made with regard to noise sensitivity and the expected shape of the decay curve.

Chapter 6

Optimization of variational quantum algorithms

Portions of this chapter were first released as part of an arXiv preprint [138] and are reproduced here with permission of the authors.

6.1 Introduction

As the quality and scale of quantum information processors (QIPs) increase, the question of whether they can derive some advantage over conventional (classical) computers, even before reaching the fault-tolerant regime, is becoming increasingly important to the field. Hybrid algorithms that utilize quantum and classical computing are perhaps the most promising route to such an advantage. Variational quantum algorithms (VQAs), which were introduced in Section 2.3, are the prime example of such hybrid algorithms [102, 22]. In a VQA, the QIP evaluates a parameterized cost function that is then optimized by a classical computer.

Conventional implementations of variational algorithms evaluate a parameterized cost function, $V(\boldsymbol{\theta})$, usually representing a parameterized quantum circuit, and then optimize over $\boldsymbol{\theta}$ using off-the-shelf multi-parameter optimization routines like COBYLA, SPSA, and Nelder-Mead [96, 14]. Such an approach only minimally exploits the structure of the underlying problem, and moreover, only minimally utilizes the computational power of the classical computing layer. While this approach has been used to demonstrate variational algorithms with a handful of parameters, $|\boldsymbol{\theta}| \equiv D \leq 10$, it is unclear how its effectiveness and the experimental resources it requires will scale to larger problems, where the number of variational parameters becomes hundreds or thousands.

Motivated by this, we introduce a new approach to optimization in variational algorithms that utilizes modern statistical inference tools to reduce the experimental burden when running variational algorithms. This in effect, moves more of the burden from the QIP to the classical computing layer. The core of our approach is the construction of a *surrogate model* for the variational cost function from QIP experimental data, and performing optimization

with this surrogate model instead of the original data. This is an established approach in optimization theory, and surrogate-based optimization (SBO) has found uses in applications where the optimization cost function is difficult to evaluate due to paucity of data or computational expense [128]. There are a variety of techniques for learning a surrogate model from data, including spline-based fitting, kriging, and neural network models [85]. In this chapter, we demonstrate SBO for VQAs using local kernel approximation techniques. Kernel approximation is particularly useful for building surrogate models for variational quantum circuits for several reasons: (i) the resulting models are explicitly smooth and smooth out unavoidable shot noise in quantum circuit measurements, (ii) the models can be learned with *batches* of circuit outputs, which has practical advantages for quantum computing platforms where circuit loading incurs latency, and (iii) the models allow numerically efficient computation of $V(\boldsymbol{\theta})$ and its derivatives, thus enabling optimization by scalable gradient-based algorithms. Intuitively, surrogate models based on a kernel approximation can be seen as explicitly taking advantage of the fact that the underlying variational cost function is smooth ($V(\boldsymbol{\theta}) \in C^{\infty}$), and thus its value at $\boldsymbol{\theta}$ indicates its value in its neighborhood. We couple this local surrogate model with an adaptive optimization procedure to efficiently find local optima of the variational cost function.

There have been several recent efforts to develop custom optimizers for VQAs, including: variations of stochastic gradient descent that adapt the number of experimental circuit evaluations (shots) to manage the tradeoff between cost function and gradient estimation quality and experimental burden [93, 149, 62], techniques based on Bayesian optimization [133, 150, 74], and machine learning-based optimization approaches for specific VQAs [86]. Most relevant to the work in this chapter is the study of Sung et al. [147], which in the framework of SBO, developed local quadratic models based on experimental data and coupled this with a trust-region optimization algorithm. Our work expands on this result by considering more general, non-parametric surrogate models that are designed to be valid over larger regions in parameter space, where the quadratic model might break down. We note that a related approach based on Gaussian process surrogate models has recently been proposed by Mueller et al. [113].

Finding global optima of parameterized quantum circuits often suffers from the problem of "barren plateaus" [108], wherein the objective function, $V(\boldsymbol{\theta})$, exhibits exponentially-vanishing gradients, both in the absence and presence of hardware noise, making optimization exceeding challenging. Some techniques around this problem are to formulate *local cost functions* [21] and to utilize variational circuit forms that do not exhibit barren plateaus [60]. We emphasize that SBO is not a technique to address the problem of barren plateaus. Instead, it is an approach to increase the performance of classical optimization loops and to reduce the experimental burden in the VQA setting. These issues are orthogonal to the barren plateaus issue – strategies to construct variational circuits that do not possess barren plateaus *and* the use of more advanced classical optimization techniques like SBO will be critical for scaling VQAs.

In this chapter, we introduce an SBO-based approach for optimizing VQAs. We introduce our optimization algorithm in Section 6.2, including an analysis of its theoretical properties

and hyperparameters. Next, in Section 6.3, we present several numerical illustrations of the approach, including comparisons to conventional variational optimization algorithms. We discuss possible extensions of our approach in Section 6.4, and we summarize in Section 6.5.

6.2 Surrogate-based optimization for variational quantum algorithms

The goal of the classical computing layer in quantum variational algorithms is to compute

$$\min_{\boldsymbol{\theta}} V(\boldsymbol{\theta}) \tag{6.1}$$

and, often, also the argument that attains this minimum. Here,

$$\boldsymbol{\theta} = (\theta_1, ...\theta_D) \in [0, 2\pi)^D \tag{6.2}$$

are parameters that dictate the variational quantum circuit ansatz for the problem. The variational cost function is

$$V(\boldsymbol{\theta}) : [0, 2\pi)^D \rightarrow \mathbb{R}, \tag{6.3}$$

which is related to the parameterized circuit, $\hat{U}(\boldsymbol{\theta})$, acting on n qubits, as

$$V(\boldsymbol{\theta}) = \mathrm{tr}[\hat{O}\hat{U}(\boldsymbol{\theta})\hat{\rho}_0\hat{U}^\dagger(\boldsymbol{\theta})], \tag{6.4}$$

for some initial n-qubit state $\hat{\rho}_0$ and observable $\hat{O}$. This quantum expectation must be estimated using many measurements on the circuit output. To do so, we first decompose the observable into a sum of non-commuting operators

$$\hat{O} = \sum_{i=1}^{\nu} \alpha_i \hat{o}_i, \tag{6.5}$$

with $[\hat{o}_i, \hat{o}_j] \neq 0$ for $i \neq j$. For all practical VQAs, $\nu = \mathcal{O}(\mathrm{poly}(n))$. Then, writing the circuit output as $\hat{\rho}(\boldsymbol{\theta}) \equiv \hat{U}(\boldsymbol{\theta})\hat{\rho}_0\hat{U}^\dagger(\boldsymbol{\theta})$,

$$V(\boldsymbol{\theta}) = \sum_{i=1}^{\nu} \alpha_i \mathrm{tr}(\hat{o}_i\hat{\rho}(\boldsymbol{\theta})) = \sum_{i=1}^{\nu} \alpha_i \mathbb{E}\{\mathbf{X}^i(\boldsymbol{\theta})\}, \tag{6.6}$$

where $\mathbf{X}^i(\boldsymbol{\theta})$ is a random variable distributed as $p^i(\boldsymbol{\theta})$ that represents the outcome of measuring $\hat{\rho}(\boldsymbol{\theta})$ in the eigenbasis of $\hat{o}_i$. In practice, the expectation in the final expression is estimated using a sample mean of a number of *shots* (executions of the circuit at $\boldsymbol{\theta}$ and measurements in one of the ν bases). That is, one takes K_i measurements of $\mathbf{X}^i(\boldsymbol{\theta}) : X_1^i(\boldsymbol{\theta}), ..., X_{K_i}^i(\boldsymbol{\theta})$, and approximates

$$\mathbb{E}\{\mathbf{X}^i(\boldsymbol{\theta})\} \approx \frac{1}{K_i} \sum_{j=1}^{K_i} X_j^i(\boldsymbol{\theta}). \tag{6.7}$$

The estimate of the cost function at a given parameter value is then

$$\tilde{V}(\boldsymbol{\theta}) = \sum_{i=1}^{\nu} \alpha_i \left(\frac{1}{K_i} \sum_{j=1}^{K_i} X_j^i(\boldsymbol{\theta}) \right). \tag{6.8}$$

The total number of shots, or circuit executions, necessary to form such an estimate is

$$\mathcal{K} = \sum_{i=1}^{\nu} K_i. \tag{6.9}$$

Since $V(\boldsymbol{\theta})$ must be estimated from a finite number of measurement results, the resulting optimization landscape is noisy and becomes increasingly so as the number of available measurements, $\mathcal{K}$, decreases. This is the impact of *quantum projection noise* or *shot noise*, the irreducible uncertainty of quantum systems that results in indeterminate measurement outcomes in general, on the variational optimization problem. (A more detailed discussion of quantum projection noise can be found in Section 2.4.) The poor performance of most optimization algorithms in such noisy landscapes places a burden on the QIP to produce as many measurements as possible to increase the accuracy of this expectation estimate, and therefore the smoothness of $\tilde{V}(\boldsymbol{\theta})$. In addition to this shot noise, in present and near-future generations of noisy intermediate scale quantum (NISQ) devices [125] there are other sources of noise coming from poor control, measurement, and isolation (decoherence) that produce distortions of the underlying probability distribution over measurement outcomes; i.e., $p^i(\boldsymbol{\theta}) \to \tilde{p}^i(\boldsymbol{\theta})$. We do not directly address this source of noise, although we note that several error mitigation techniques have been developed to address this problem [151, 41, 83, 32], and they can be used in tandem with our optimization approach to achieve some degree of robustness to both sources of noise (shot noise and decoherence).

We now introduce the concept of a local surrogate model to $V(\boldsymbol{\theta})$. This is a function $W : \Theta \to \mathbb{R}$ that is an approximation of $V(\boldsymbol{\theta})$ in a local *patch*, $\Theta \subset [0, 2\pi)^D$. We demand that this surrogate model must be (i) smooth and (ii) efficient to evaluate on a classical computer, requiring no additional measurements from a QIP than those required to construct it. In this chapter, we construct such a surrogate model using a kernel approximation; i.e.,

$$W_\Theta(\boldsymbol{\theta}) = \sum_{j=1}^{\tau} \tilde{V}(\boldsymbol{\theta}_j) \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j), \tag{6.10}$$

where $\tilde{V}(\boldsymbol{\theta}_j)$ are standard estimates of $V(\boldsymbol{\theta})$ (constructed using $\mathcal{K}$ shots) at τ distinct *sample points* $\boldsymbol{\theta}_j \in \Theta$, and $\kappa(\cdot, \cdot)$ is a *kernel function*. Note that the subscript on W_Θ serves to remind us that the surrogate model is valid in some local patch of parameter space, since it is formulated based on data from that local patch.

The choice of κ determines most of the properties of kernel-based surrogate models. In this chapter, we choose a Gaussian kernel,

$$\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j) = \exp\left(\frac{-||\boldsymbol{\theta} - \boldsymbol{\theta}_j||^2}{2\sigma} \right), \tag{6.11}$$

for two reasons. First, it is a simple kernel with only one free parameter, σ, that can be set in a data-driven manner, as we show below. And second, its form allows for easy analytic evaluation of derivatives of $W_\Theta(\boldsymbol{\theta})$, which is a useful property for gradient-based optimization of $W(\boldsymbol{\theta})$.

It is known that this kernel can result in a systematic bias [157]. In the context of VQAs, this is often manifest in an "offset" of the kernel-produced variational cost function values from the experimental values. However, given the prevalence of systematic noise in experimental measurements on current quantum hardware, there is frequently no gain to be made from expending computational resources to get the true experimental surface because it is offset already, and only relative magnitudes matter for optimization. Moreover, in applications where the goal is finding the parameter argument of the minimal objective function, the offset is irrelevant. Finally, in applications where the minimal variational cost function value is desired, it is often possible to fix the offset, both from experimental noise and the kernel, by appealing to special cases when the parameter values simplify the objective function to known values, and shifting the offset of the full surface accordingly.

Figure 6.1 provides an illustration of a true objective function $V(\boldsymbol{\theta})$, interpolated samples $\tilde{V}(\boldsymbol{\theta}_j)$, and surrogate function $W_\Theta(\boldsymbol{\theta})$ constructed using a Gaussian kernel.

Adaptive optimization

As described above, the kernel-based surrogate model is learned over a local patch Θ. In order to find a local optimum of $V(\boldsymbol{\theta})$, we couple this construction with an adaptive optimization procedure that we describe in this section.

We begin with an initial seed for the variational parameters, $\boldsymbol{\theta}^{(0)}$, and define a local patch around it as a D-dimensional hypercube of length ℓ:

$$\Theta^{(0)} = \cup_{m=1}^{D} \left[\theta_m^{(0)} - \frac{\ell}{2}, \theta_m^{(0)} + \frac{\ell}{2}\right]. \tag{6.12}$$

Then we randomly sample τ points in this patch, execute variational circuits defined by each of those sample points, and use the resulting data to form estimates $\tilde{V}(\boldsymbol{\theta}_1), ..., \tilde{V}(\boldsymbol{\theta}_\tau)$. We assume for simplicity that each of the estimates $\tilde{V}(\boldsymbol{\theta}_j)$ is formed using $\mathcal{K}$ shots, i.e., $\mathcal{K}$ does not depend on j, although this is not an essential assumption. The τ samples of $\boldsymbol{\theta}_j$ are sampled sparsely in $\Theta^{(0)}$; to achieve this in practice, we use Latin hypercube sampling over $\Theta^{(0)}$ to choose each $\boldsymbol{\theta}_j$. The number of samples τ and the patch "size" ℓ are important parameters; we develop heuristics for choosing their values and study their scaling with n and D later in this section. The estimates $\tilde{V}(\boldsymbol{\theta}_j)$ are then used to formulate a surrogate model $W_{\Theta^{(0)}}$ for $V(\boldsymbol{\theta})$ on the patch $\Theta^{(0)}$, as defined in Equation 6.10.

Given $W_{\Theta^{(0)}}(\boldsymbol{\theta})$, we perform optimization over this (explicitly smooth) function over the local domain $\Theta^{(0)}$. We do not specify the method to use for this optimization. However, given a smooth objective and easily computable gradients, gradient-based optimizers that incorporate parameter constraints (since the optimization should only be over $\Theta^{(0)}$) are well-suited for this task. In practice, it may be helpful to optimize over a slightly smaller domain

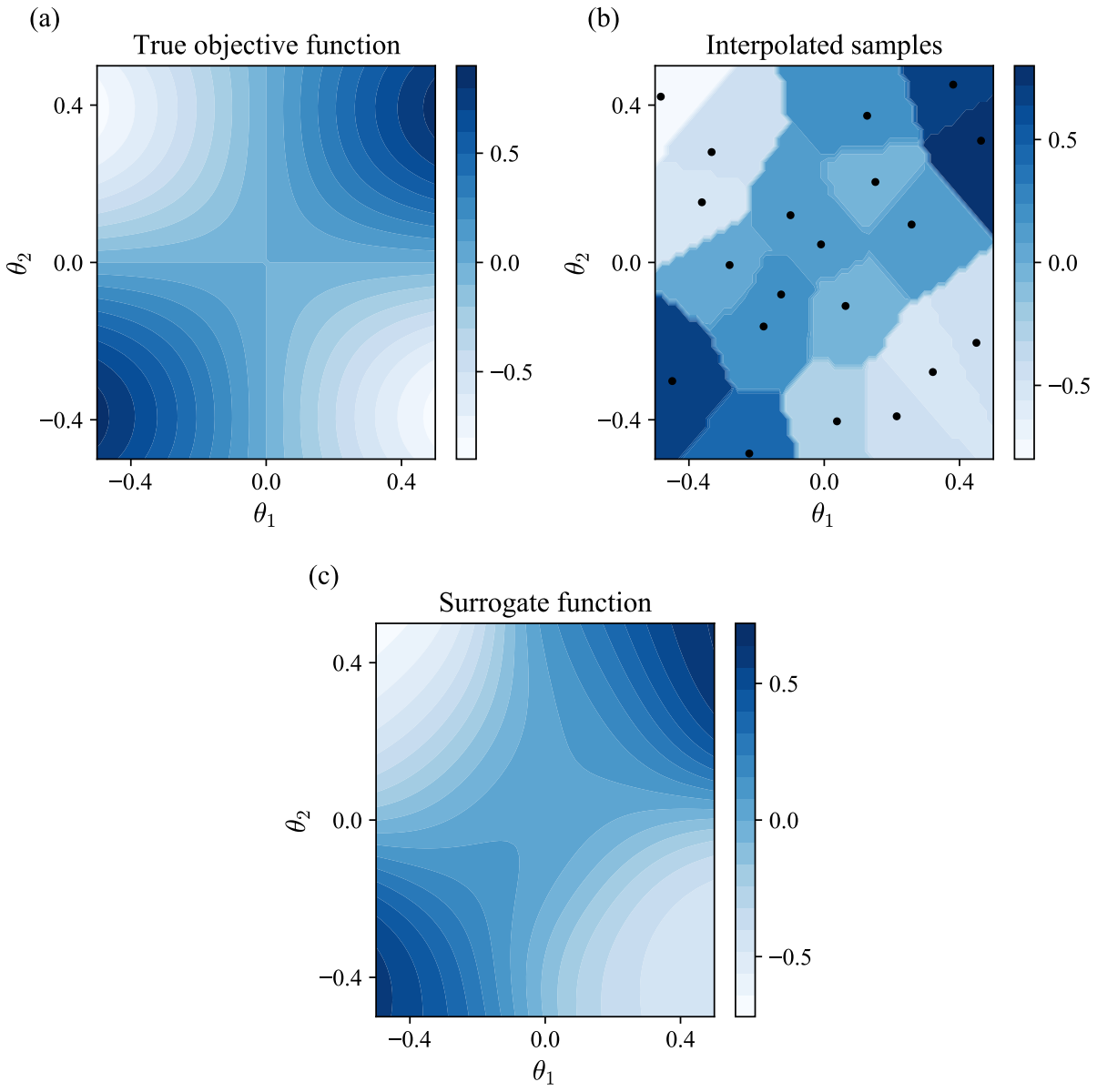


Figure 6.1: An illustration of a local patch of (a) a true objective function $V(\boldsymbol{\theta})$ with dimension $D = 2$ where $\boldsymbol{\theta} = (\theta_1, \theta_2)$, (b) interpolated samples $\tilde{V}(\boldsymbol{\theta})$ using $\mathcal{K} = 100$ shots at each of the $\tau = 20$ sample points, and (c) surrogate function $W(\boldsymbol{\theta})$ constructed using a Gaussian kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j) = \exp(^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_j||^2}/_{2\sigma})$.

to avoid edge effects in the kernel approximation; i.e.,

$$\min_{\boldsymbol{\theta}\in\Theta_\epsilon^{(0)}} W_{\Theta^{(0)}}(\boldsymbol{\theta}), \tag{6.13}$$

where the minimum is over all points in $\Theta_\epsilon^{(0)}$, which is defined as

$$\Theta_\epsilon^{(0)} = \cup_{m=1}^{D}\left[\theta_m^{(0)} - \frac{(\ell-\epsilon)}{2}, \theta_m^{(0)} + \frac{(\ell-\epsilon)}{2}\right]. \tag{6.14}$$

The argument that achieves the above minimum defines the center of the next patch, $\boldsymbol{\theta}^{(1)}$, and this process is repeated.

We refer to the process above as one *iteration* of the optimization run. Each iteration thus requires $\mathcal{K}\tau$ shots. We perform a fixed number of iterations M, giving a total of $\mathcal{K}\tau M$ shots in a full optimization run. To assist in the convergence of the optimization run, we linearly increase ϵ from some initial (small) value ϵ_i in the first iteration to a value near ℓ in the final iteration.

If the minimum $\boldsymbol{\theta}^{(i+1)}$ found after iteration i falls within the interior of the current patch $\Theta^{(i)}$, i.e., if $\boldsymbol{\theta}^{(i+1)} \in \Theta_{\epsilon_{\text{int}}}^{(i)}$ for some small $\epsilon_{\text{int}} \sim \ell/20$ which excludes the boundary of the patch, then we add the minimum $\boldsymbol{\theta}^{(i+1)}$ to a list of local minima Θ_{minima}. After completing M iterations, we calculate the final estimated optimum $\boldsymbol{\theta}_{\text{opt}}$ by taking the coordinate-wise mean of all of the elements of Θ_{minima} that fall within a distance $\ell - \epsilon_f$ (for $\epsilon_f \sim \ell/2$) of the minimum $\boldsymbol{\theta}^{(M)}$ found in the final iteration; i.e., for $\Theta_{\epsilon_f,\text{minima}}^{(M)} = \Theta_{\text{minima}} \cap \Theta_{\epsilon_f}^{(M)}$,

$$\boldsymbol{\theta}_{\text{opt}} = \frac{1}{\left|\Theta_{\epsilon_f,\text{minima}}^{(M)}\right|} \sum_{\boldsymbol{\theta}\in\Theta_{\epsilon_f,\text{minima}}^{(M)}} \boldsymbol{\theta}. \tag{6.15}$$

Figure 6.2 provides a graphical description of the surrogate-based adaptive optimization approach described above.

We note that the optimization approach is decoupled from the surrogate model. Although we have found that the adaptive optimization detailed above is effective, it is by no means unique or optimal. It is possible to modify it or even replace it with another approach while keeping the surrogate model idea intact. In particular, it is likely advantageous to incorporate a memory element that includes information from previous patches into the decisions made at the current patch – this is a promising area for future study.

A reference implementation of the Gaussian kernel-based SBO optimizer, including examples of integration with IBM's Qiskit library, is freely available [137].

Convergence and hyperparameter choices

We now discuss practical considerations for choosing various hyperparameters of SBO and the adaptive optimization technique described above.

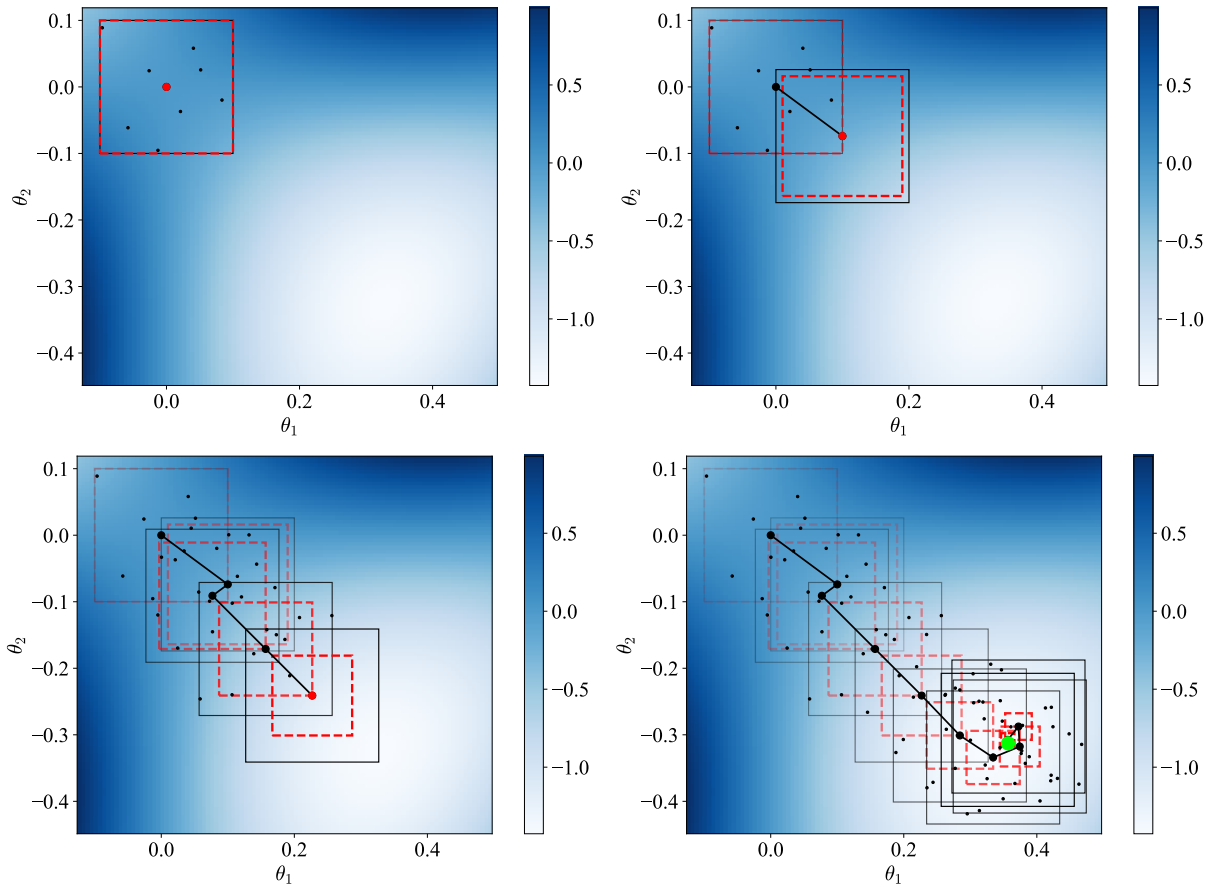


Figure 6.2: A graphical description of the adaptive surrogate-based optimization procedure on a two-dimensional objective function surface parameterized by $\boldsymbol{\theta} = (\theta_1, \theta_2)$, showing snapshots of an optimization run during the first iteration (upper left), after the first iteration (upper right), after the fourth iteration (lower left), and at the completion of the run after $M = 10$ iterations (lower right). The larger, connected points mark the patch centers $\boldsymbol{\theta}^{(i)}$ of each iteration, with the red point indicating the center of the most recent patch. The smaller, unconnected points mark the locations of the $\tau = 8$ samples taken during each iteration. The solid black rectangles mark the boundaries of the sampling region $\Theta^{(i)}$ for each iteration, where each side has fixed length $\ell = 0.2$. The dashed red rectangles mark the boundaries of the optimization region $\Theta_\epsilon^{(i)}$ for each iteration, where ϵ is linearly increased from 0 to $\ell = 0.2$ over the course of the optimization run. The green point in the final plot (lower right) marks the final estimated optimum $\boldsymbol{\theta}_{\text{opt}}$.

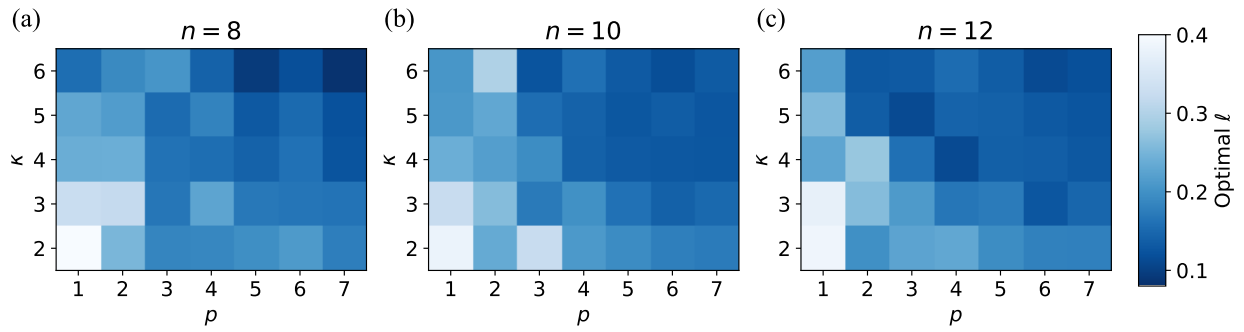


Figure 6.3: Numerical estimation of the optimal SBO patch size ℓ for various instances of n-qubit, p-layer MaxCut QAOA on randomly-generated κ-regular graphs. Each data point is obtained by averaging the final error of 10 independent SBO runs for each $\ell \in \{0.02, 0.04, \ldots, 0.40\}$, and then using cubic splines to fit the results and find the value of ℓ which minimizes the average error. Each run uses $\tau = 30$ sample points per patch, $K = 60$ measurement shots per sample point, and $M = 100$ optimization iterations.

Optimization adjustments ϵ_i, ϵ_{int}, ϵ_f. These parameters are used primarily to avoid boundary effects near the edges of each patch region, since during each iteration we sample from only the interior of the patch. In the demonstrations presented in this chapter, we have used $\epsilon_i = 0$, $\epsilon_{\text{int}} = \ell/20$, and $\epsilon_f = \ell/2$ with good results.

Measurement shots per measurement basis per sample point, K. The choice of K will be primarily driven by experimental considerations. Larger K is always better since it will reduce shot noise and therefore improve the accuracy of the surrogate model, and in turn the performance of the optimization, but at the cost of increased experimental demands (especially run time). As we shall demonstrate in the next section, one of the advantages of constructing a surrogate model is an increased robustness of optimization performance to shot noise, and thus SBO can alleviate the experimental burden without sacrificing optimization performance.

Patch size ℓ and sample points per patch τ. These parameters are intimately related. Intuitively, the larger the patch size, ℓ, the larger the number of sample points per patch, τ, will need to be in order for the surrogate model to be accurate to the true cost function $V(\boldsymbol{\theta})$. Since τ is closely tied to experimental resources, we find it most useful to think in terms of keeping τ fixed at a constant, and varying ℓ. In practice, especially in the near-term, experimental constraints such as device instability and access constraints will dictate how large τ can be, and therefore we think of it as a fixed parameter, independent of variational problem parameters such as n and D. In all of our numerical experiments, including the ones reported in the next section, we have kept $\tau \sim 20$.

Given a fixed, constant τ, the choice of ℓ is dictated by the need to accurately capture the

shape of the objective function $V(\boldsymbol{\theta})$ over the patch in each of the D parameter dimensions. A conservative way to ensure that a fixed number of samples captures the objective function is to demand that this function varies minimally within the patch – i.e., to choose ℓ such that there is likely no more than one critical point of $V(\boldsymbol{\theta})$ in any ℓ^D volume in parameter space. From a loose bound on the number of critical points in a general variational cost function, the optimal patch size scales as $\ell = \Omega(1/\mathrm{poly}(D,n))$, and for MaxCut QAOA on a κ-regular graph with fixed κ, this scaling becomes independent of n and depends only on the dimension D (see appendix of [138]). Figure 6.3 provides numerically-determined optimal patch size ℓ for MaxCut QAOA on κ-regular graphs as we vary the relevant parameters: the number of QAOA layers p (where $D = 2p$), the number of qubits n, and the graph regularity κ. The independence of ℓ from n is supported by this data, as the variation of the surfaces is negligible as n is varied. For the empirical studies reported in this chapter, we have found that patch sizes in the range $0.1 \leq \ell \leq 0.2$ worked well for QAOA problems with $n \leq 12$ and $p \leq 7$ ($D \leq 14$), as well as VQE problems with $n \leq 8$ and $D \leq 8$, using $K \sim 100$.

Perhaps the most robust solution to determining ℓ is to employ an adaptive method that dynamically adjusts ℓ along the optimization path according to a quality of fit metric. This would be possible by moving to a trust-region framework for the optimization [120].

Surrogate model parameters. The procedure used to construct the surrogate model will typically have parameters to set. In this chapter we only consider kernel-based surrogate models, and specifically an isotropic Gaussian kernel $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j)$, which has one parameter, the *Gaussian bandwidth* σ. Intuitively, this parameter describes the volume over which one sample data point influences the behavior of the surrogate model. There is a rich literature on Gaussian kernels and their use in approximation, regression, and smoothing, and as a result, many data-driven heuristics exist for choosing σ. In practice, we have observed good performance using the Silverman bandwidth heuristic [144]

$$\sigma = \left[\frac{4}{\tau(D+2)}\right]^{1/(D+4)}, \tag{6.16}$$

where τ is the number of sample points in the current patch.

6.3 Numerical demonstrations

In this section we demonstrate our SBO procedure on some model VQAs through numerical simulation. We also compare optimization performance with one of the most commonly used and recommended optimization methods for VQAs, simultaneous perturbation stochastic approximation (SPSA) [146]. SPSA is designed to find optima in the presence of noise in the objective function. Key to its popularity in the resource-constrained setting of VQAs is the fact that it estimates gradients using only two evaluations of the (multiparameter) objective function.

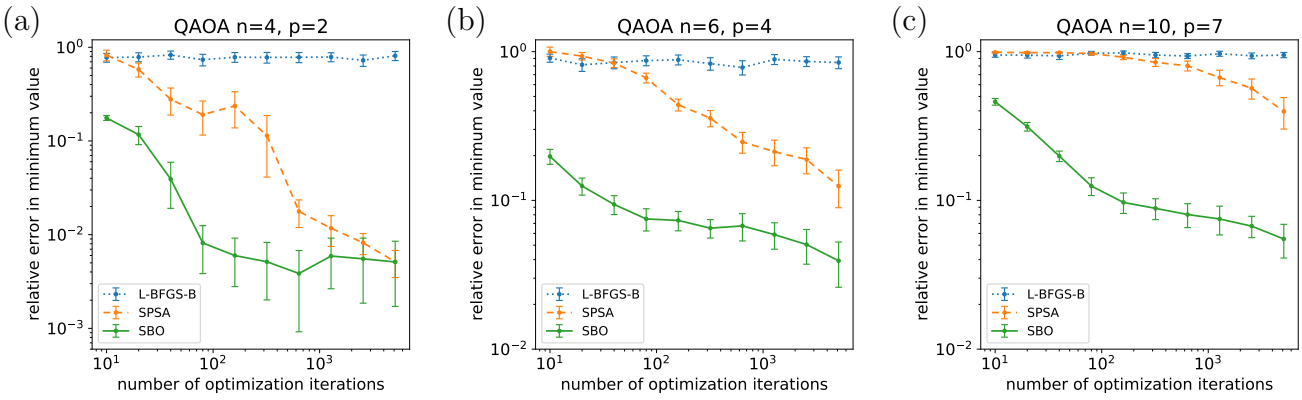


Figure 6.4: Performance of L-BFGS-B, SPSA, and Gaussian kernel-based SBO optimization runs on QAOA applied to unweighted MaxCut problems of various sizes using an ideal simulator which has only shot noise (and no other errors). Each plot displays the results of running p-layer QAOA on a single randomly-generated connected graph with n vertices. The x-axis represents the number of optimization iterations M. The y-axis represents the relative absolute error achieved by the optimization run, i.e., $\left|1 - V_{QAOA}(\boldsymbol{\gamma}_{\text{opt}}, \boldsymbol{\beta}_{\text{opt}})/V_{QAOA,min}\right|$, where $\boldsymbol{\gamma}_{\text{opt}}, \boldsymbol{\beta}_{\text{opt}}$ are the optimal coordinates obtained by the optimization run and $V_{QAOA,min} = \min_{\boldsymbol{\gamma},\boldsymbol{\beta}} V_{QAOA}(\boldsymbol{\gamma}, \boldsymbol{\beta})$ is the true optimum of the objective function. Each run uses $K\tau = 5000$ total shots per iteration, where τ is the number of sample points per iteration and K is the number of shots taken per sample point. Each data point represents the mean of 50 independent optimization runs on one representative problem instance, represented by an Erdős-Rényi random unweighted graph. The initial parameter choice, $\boldsymbol{\theta}^{(0)}$ is the same for all runs; however, the sample points on each patch obtained via Latin hypercube sampling are chosen independently for each run. Error bars indicate standard error of the mean. Additional details on hyperparameter choices and implementation notes can be found in the text.

Quantum approximate optimization algorithm

The quantum approximate optimization algorithm (QAOA) is a variational circuit approach to combinatorial optimization [44], where the optimization problem is encoded in a *problem Hamiltonian*, $\hat{H}_p$, whose ground state encodes the solution to the problem. A commonly studied example is the MaxCut problem, which aims to partition an n-node graph into two sets of nodes, such that the weight of the edges going between the partitions is maximized. A MaxCut problem instance is encoded in an n-qubit Ising Hamiltonian of the form

$$\hat{H}_p = \sum_{(i,j)\in\mathcal{E}} w_{ij} \hat{Z}_i \hat{Z}_j, \tag{6.17}$$

where $\hat{Z}_i$ is a Pauli Z matrix on qubit i tensored with the identity on all other qubits, and $\mathcal{E}$ is the set of edges in the graph, each with weight $w_{ij} \in \mathbb{R}$.

QAOA approaches the goal of preparing low energy eigenstates of $\hat{H}_p$ by p iterated applications of a two-layer ansatz to a product input state to produce the output state:

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{l=1}^{p} e^{-i\beta_p \hat{H}_d} e^{-i\gamma_p \hat{H}_p} |+\rangle^{\otimes n}, \tag{6.18}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $\hat{H}_d = \sum_{i=1}^{n} \hat{X}_i$. The variational parameters $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)$ are optimized such that the energy of the output state is minimized; i.e., the objective function is

$$V_{QAOA}(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) | \hat{H}_p | \psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle. \tag{6.19}$$

The *approximation ratio*, which quantifies how close to the true ground state any $|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$ is, is defined by

$$r = \frac{V_{QAOA}}{E_0}, \tag{6.20}$$

where E_0 is the true ground state energy of $\hat{H}_p$.

If global optima to V_{QAOA} can be found, in the $p \to \infty$ limit QAOA prepares the ground state of $\hat{H}_p$, which encodes the solution to the original combinatorial optimization problem [44]. Moreover, in this case r increases monotonically with p, although the question of what p is required for r to surpass approximation ratios achievable by classical approximation methods is an open one. It is clear that the variational optimization, and even finding good quality local minima of V_{QAOA}, becomes challenging with increasing p.

In terms of the parameters defined in the general description of VQAs in Section 6.2, it is important to note that the QAOA objective is defined through an observable, H_p, that only consists of commuting terms. Therefore, one only needs to measure in the computational basis for QAOA, meaning that we have $\nu = 1$ and we require $\mathcal{K} = K$ total shots per sample point.

Figure 6.4 shows the results of simulated L-BFGS-B, SPSA, and SBO optimization runs of QAOA applied to MaxCut instances on (Erdős-Rényi) random unweighted graphs of (a) $n = 4$ vertices using $p = 2$ layers, (b) $n = 6$ vertices using $p = 4$ layers, and (c) $n = 10$ vertices using $p = 7$ layers. In each case, the objective function is evaluated using an ideal simulator, which has only shot noise and no other errors, to compute the expectation value of the energy using the p-layer QAOA ansatz. We plot the results of each run using $K\tau = 5000$ shots per iteration. We repeated these tests with various values of $K\tau$ ranging from 10^3 to 10^5 and observed qualitatively similar results.

We choose L-BFGS-B here as an example of a gradient-free optimizer. We use a gradient-free optimizer because the gradient of our noisy objective function is not directly available and therefore traditional gradient descent cannot be used. We found that L-BFGS-B, although it still performs very poorly due to the noisy objective function, significantly outperformed Nelder-Mead, another widely-used gradient-free optimizer.

At the smallest problem size, SBO achieves a lower error than SPSA for up to $M \sim 10^3$ iterations, indicating that it converges on a good approximation of the local minimum more efficiently. At the larger problem sizes, SBO achieves a lower error than SPSA for even larger numbers of iterations. For perspective, we note that an optimization run with $K\tau = 5000$ shots per iteration and $M = 10^3$ iterations would require a total of $K\tau M = 5 \times 10^6$ experimental shots. This would require an experimental duration on the order of several minutes using a typical superconducting QIP, or on the order of several days using a typical trapped-ion QIP. Because our results indicate that SBO achieves up to an order of magnitude better result than SPSA in this regime, it appears likely that SBO will outperform SPSA for many QAOA experiments that can be realistically implemented on current and near-future devices.

Variational quantum eigensolver

The first VQA was the so-called variational quantum eigensolver (VQE) [122], which aims to prepare the ground state of an n-qubit Hamiltonian, $\hat{H}_E$, that encodes the energy of a molecule. The variational circuits and parameters, $\boldsymbol{\theta}$, that prepare candidate states vary according to the wavefunction *ansatz* that is used [108]. In all cases, the objective function is defined as

$$V_{VQE}(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | \hat{H}_E | \psi(\boldsymbol{\theta}) \rangle. \tag{6.21}$$

In general, $\nu > 1$ for nontrivial $\hat{H}_E$ and hence measurements in multiple bases are necessary.

Figure 6.5 shows the results of simulated SPSA and SBO optimization runs of VQE for estimating the ground state energy of H_2 and LiH molecules at various interatomic bond lengths using the unitary coupled cluster ansatz with Hartree-Fock initial state. The H_2 ansatz uses four qubits and $|\boldsymbol{\theta}| = 3$ variational parameters, while the LiH ansatz uses four qubits and $|\boldsymbol{\theta}| = 8$ variational parameters. In Figure 6.5(a,b), under shot noise only, we observe that SBO produces a much more accurate estimate of the ground state energy than SPSA using the same number of energy measurements τM, and it remains more accurate even than using SPSA with τM increased by a factor of 2.5 to 5. In Figure 6.5(c,d), using a simulator with a realistic hardware noise model, we observe that SBO achieves consistently lower estimates of the ground state energy than SPSA. In addition, by taking the parameters $\boldsymbol{\theta}$ found by the noisy optimization runs and evaluating the ansatz with those values on an ideal simulator, we observe that the parameter values obtained by SBO correspond to energy values which are much closer to the exact ground state than those obtained by SPSA.

Implementation notes

Here we provide additional implementation details for the QAOA and VQE simulations described in this section. We note that the classical computational cost of the various optimization routines are comparable, and the runtime of the numerical simulations is dominated by the cost of simulating the variational quantum circuit at each of the parameter settings.

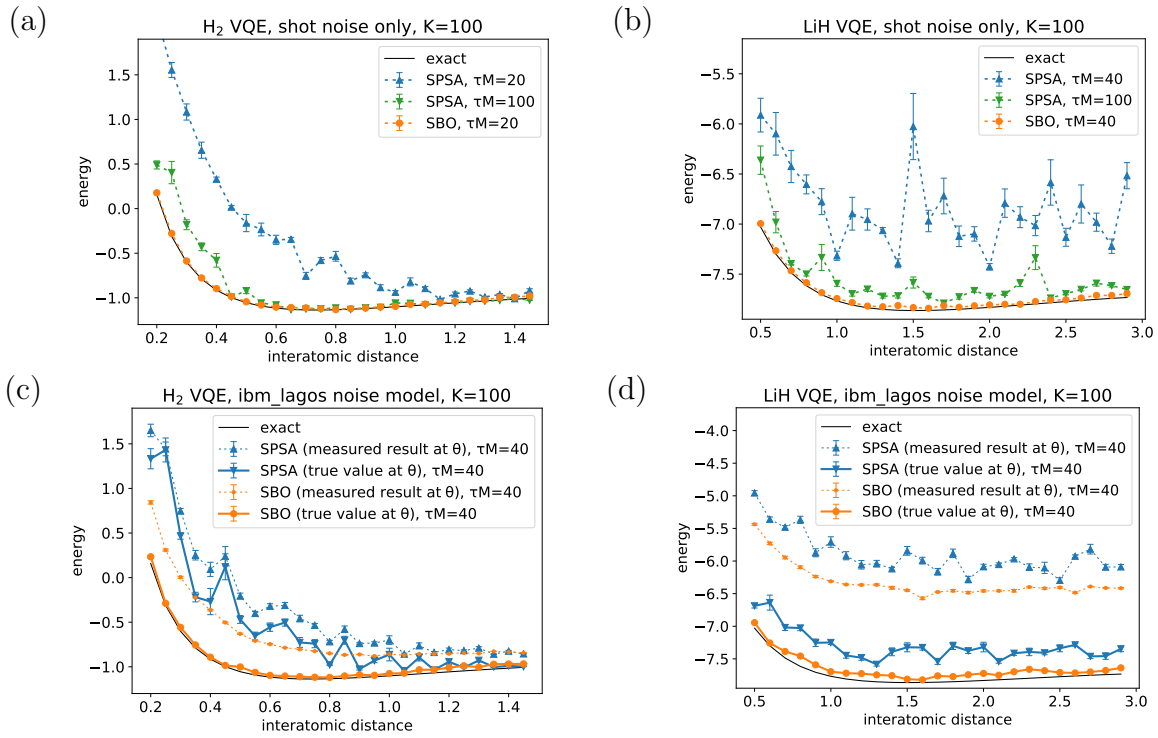


Figure 6.5: Performance of SPSA and Gaussian kernel-based SBO optimization runs on common small-scale VQE problems using simulators with and without realistic hardware noise. Each plot displays the minimum energy obtained for each bond length under the specified optimization conditions. The x-axis represents the interatomic distance (in angstroms) used for the energy calculation. The y-axis represents the energy value (in hartrees) obtained at the conclusion of each optimization run. τM is the total number of energy measurements performed in the optimization run, where τ is the number of sample points per iteration and M is the number of optimization iterations. To measure the energy, $K = 100$ shots are taken per measurement basis per sample point. Each data point represents the mean of five independent optimization runs at the given setting. Error bars indicate standard error of the mean. On each plot, a solid black curve indicates the exact minimum energy value for the given setting. In (a) and (b), we use an ideal simulator which has only shot noise and no other errors. The data points connected by dashed curves represent the energy value obtained by evaluating the ansatz on the ideal simulator at the found optimal parameter values $\boldsymbol{\theta}$. In (c) and (d), we use a noisy simulator implementing a typical noise model and coupling map obtained from the seven-qubit IBM Q Lagos device. The data points connected by dashed curves represent the energy value obtained by evaluating the ansatz on the noisy simulator at the found optimal parameter values $\boldsymbol{\theta}$. The data points connected by solid curves represent the energy value obtained by evaluating the ansatz at $\boldsymbol{\theta}$ using an ideal simulator. Additional details on hyperparameter choices and implementation notes can be found in the text.

For Figure 6.4: The QAOA circuit simulations were implemented using the pyQAOA package [159]. SPSA was implemented using the noisyopt package [107]. L-BFGS-B was implemented using the scipy package [52]. We chose hyperparameters by manual scans to optimize the performance of both SBO and SPSA on these problems. We used $\tau = 20$ for SBO, while $\tau = 2$ for SPSA by definition. In Figure 6.4(a,b,c), we used SBO parameter $\ell = (0.2, 0.2, 0.1)$ and SPSA parameter $a = (0.2, 0.2, 0.1)$, respectively. We used SPSA parameters $c = 0.2$, $\alpha = 0.602$, and $\gamma = 0.101$ for all simulations presented in this figure.

For Figure 6.5: The VQE simulations were implemented via the Qiskit package provided by IBM [152] using a unitary coupled cluster (UCC) ansatz with Hartree-Fock initial state. For SPSA, we used the implementation provided by Qiskit, which includes automatic hyperparameter calibration. In Figure 6.5(a,b,c,d), we used $\tau = (4, 5, 8, 10)$ and $\ell = (0.15, 0.1, 0.15, 0.1)$ for SBO, respectively, while $\tau = 2$ for SPSA by definition.

6.4 Discussion

From both the QAOA and VQE illustrations in Section 6.3, we observe that SBO often achieves a lower error than SPSA for an equivalent number of iterations or experimental shots. From the QAOA results in Figure 6.4, we note that this advantage tends to become more pronounced as the problem complexity increases. We believe these results are a good indication that, for many near-term applications, SBO will achieve better variational parameter estimates with fewer experimental repetitions than existing techniques such as SPSA. Additionally, because the surrogate function smooths out shot noise, SBO often requires fewer shots per sample point than SPSA to produce a result that is equivalent or better.

One unique feature of SBO is that each iteration requires taking samples for many different parameter settings, as opposed to a technique like SPSA which uses only two sample points per iteration. This may provide a particular advantage for experimental platforms that suffer a high latency cost from loading new circuits between each optimization iteration, as well as increased robustness against drift in experimental parameters.

A promising avenue for future work is the application of more powerful surrogate models to the VQA setting, e.g., neural network-based methods for approximation [97] might have better rates of convergence with limited experimental data. In addition, building surrogate models to not only perform smoothing and approximation, as we have done here, but also physics-informed error mitigation to counter decoherence is a potentially fruitful direction.

6.5 Summary

In this chapter, we have introduced a surrogate-based optimization (SBO) technique for variational quantum algorithms (VQAs). We have developed this technique using a Gaussian kernel-based surrogate and an adaptive optimization procedure which iterates over local patches of the objective function landscape to find a good approximation to a local minimum

of the function. We have also provided the results of several numerical simulations of QAOA and VQE problem instances which demonstrate that this SBO technique outperforms existing techniques, most notably SPSA, on these common problem types. The SBO technique outperforms SPSA in a noiseless simulation, where the only error is from quantum projection noise, as well as in a noisy simulation, where a realistic hardware error model is incorporated. Finally, we have discussed future avenues for the use of surrogate-based techniques in VQA optimization.

Chapter 7

Conclusion

This work has introduced several techniques for verification and operation of near-term quantum computers. These techniques are incremental steps toward the broader goal of increasing efficiency in these devices. The randomized analog verification (RAV) protocol, which is developed for gate-based quantum computers in Chapter 4 and for analog quantum simulators in Chapter 5, provides an efficient strategy for verifying the behavior of both continuously-parameterized gates and analog simulations. The RAV technique incorporates the STOQ stochastic compilation technique described in Chapter 3, which is a straightforward and easily parallelizable tool for creating approximate compilations in terms of an arbitrary native gate set.

The goal of increased efficiency extends also to the implementation of quantum algorithms, and the surrogate-based optimization (SBO) technique for variational quantum algorithms developed in Chapter 6 provides a demonstrable operational speedup for a variety of common problems. And increasing the efficiency of the researcher is also an important goal; toward this end, the experimental simulation techniques laid out in Chapter A provide building blocks for developing experimental techniques without requiring physical access to the experimental apparatus.

Improvements in the performance of quantum computers over the coming years will involve breakthroughs at every layer of the stack. Enhancements in the physical qubits and low-level control systems will play a significant role, as will algorithmic advances and discoveries that open up new applications of quantum computing. But just as important are the incremental improvements that will continue to be made at every layer between the hardware and the eventual application: calibration of qubits and gates, quantum compilation, error mitigation, error correction, and classical optimization for variational algorithms – to name just a handful. Physics plays a defining role at every level of the stack, but as devices begin to approach physical limits of performance, engineering of both software and hardware components will become increasingly critical to the continued improvement of quantum computers. The techniques described in this work only scratch the surface of what is likely achievable by the scientific and industrial communities in the coming years as the quest continues to realize the potential of large-scale quantum computing.

Bibliography

[1] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Physical Review A* 70.5 (Nov. 2004), p. 052328. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.70.052328.

[2] Nitzan Akerman et al. "Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser". In: *New Journal of Physics* 17.11 (Nov. 2015), p. 113060. ISSN: 1367-2630. DOI: 10.1088/1367-2630/17/11/113060.

[3] Takafumi Arakaki et al. *JuliaPy/pyjulia: PyJulia*. Nov. 2020. DOI: 10.5281/zenodo.4294939.

[4] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 0028-0836. DOI: 10.1038/s41586-019-1666-5.

[5] Alán Aspuru-Guzik and Philip Walther. "Photonic quantum simulators". In: *Nature Physics* 8.4 (Apr. 2012), pp. 285–291. ISSN: 1745-2473. DOI: 10.1038/nphys2253.

[6] S. H. Autler and C. H. Townes. "Stark Effect in Rapidly Varying Fields". In: *Physical Review* 100.2 (Oct. 1955), pp. 703–722. ISSN: 0031-899X. DOI: 10.1103/PhysRev.100.703.

[7] Harrison Ball et al. "Effect of noise correlations on randomized benchmarking". In: *Physical Review A* 93.2 (Feb. 2016). ISSN: 1050-2947. DOI: 10.1103/PhysRevA.93.022303.

[8] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* (1995). ISSN: 1050-2947. DOI: 10.1103/PhysRevA.52.3457.

[9] Hannes Bernien et al. "Probing many-body dynamics on a 51-atom quantum simulator". In: *Nature* 551.7682 (Nov. 2017), pp. 579–584. ISSN: 0028-0836. DOI: 10.1038/nature24622.

[10] R. Blatt and C. F. Roos. "Quantum simulations with trapped ions". In: *Nature Physics* 8.4 (Apr. 2012), pp. 277–284. ISSN: 1745-2473. DOI: 10.1038/nphys2252.

[11] Immanuel Bloch, Jean Dalibard, and Sylvain Nascimbène. "Quantum simulations with ultracold quantum gases". In: *Nature Physics* 8.4 (Apr. 2012), pp. 267–276. ISSN: 1745-2473. DOI: 10.1038/nphys2259.

[12] Robin Blume-Kohout et al. "Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography". In: *Nature Communications* 8 (2017). ISSN: 2041-1723. DOI: 10.1038/ncomms14485.

[13] Sergio Boixo et al. "Characterizing quantum supremacy in near-term devices". In: *Nature Physics* (2018). ISSN: 1745-2481. DOI: 10.1038/s41567-018-0124-x.

[14] Xavier Bonet-Monroig et al. "Performance comparison of optimization methods on variational quantum algorithms". In: *arXiv* (Nov. 2021). DOI: 10.48550/arxiv.2111.13454.

[15] Sébastien Bourdeauducq et al. *ARTIQ*. Feb. 2021. DOI: 10.5281/zenodo.6619071.

[16] Joseph Broz, Neil Glikin, and Kunal Marwaha. *IonSim.jl*. 2022. URL: https://github.com/HaeffnerLab/IonSim.jl.

[17] Colin D. Bruzewicz et al. "Trapped-ion quantum computing: Progress and challenges". In: *Applied Physics Reviews* 6.2 (June 2019), p. 021314. ISSN: 1931-9401. DOI: 10.1063/1.5088164.

[18] A. H. Burrell et al. "Scalable simultaneous multiqubit readout with 99.99% single-shot fidelity". In: *Physical Review A* 81.4 (Apr. 2010), p. 040302. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.81.040302.

[19] Alice Heather Burrell. "High fidelity readout of trapped-ion qubits". PhD thesis. University of Oxford, 2010. URL: https://www2.physics.ox.ac.uk/sites/default/files/Burrell_Thesis.pdf.

[20] Earl Campbell. "Random Compiler for Fast Hamiltonian Simulation". In: *Physical Review Letters* (2019). ISSN: 0031-9007. DOI: 10.1103/physrevlett.123.070503.

[21] M. Cerezo et al. "Cost function dependent barren plateaus in shallow parametrized quantum circuits". In: *Nature Communications* 12.1 (Dec. 2021), p. 1791. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21728-w.

[22] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9.

[23] Andrew M. Childs, Aaron Ostrander, and Yuan Su. "Faster quantum simulation by randomization". In: *Quantum* 3 (Sept. 2019), p. 182. ISSN: 2521-327X. DOI: 10.22331/q-2019-09-02-182.

[24] Joonhee Choi et al. "Emergent Quantum Randomness and Benchmarking from Hamiltonian Many-body Dynamics". In: *arXiv* (Mar. 2021). DOI: 10.48550/arxiv.2103.03535.

[25] Frederic T. Chong, Diana Franklin, and Margaret Martonosi. "Programming languages and compiler design for realistic quantum hardware". In: *Nature* 549.7671 (Sept. 2017), pp. 180–187. ISSN: 0028-0836. DOI: 10.1038/nature23459.

[26] J. Ignacio Cirac and Peter Zoller. "Goals and opportunities in quantum simulation". In: *Nature Physics* (2012). ISSN: 1745-2473. DOI: 10.1038/nphys2275.

[27] Susan M. Clark et al. "Engineering the Quantum Scientific Computing Open User Testbed". In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–32. ISSN: 2689-1808. DOI: 10.1109/TQE.2021.3096480.

[28] Patrick J. Coles, M. Cerezo, and Lukasz Cincio. "Strong bound between trace distance and Hilbert-Schmidt distance for low-rank states". In: *Physical Review A* (2019). ISSN: 24699934. DOI: 10.1103/PhysRevA.100.022103.

[29] D. Coppersmith. "An approximate Fourier transform useful in quantum factoring". In: *arXiv* (Jan. 1994). DOI: 10.48550/arxiv.quant-ph/0201067.

[30] Marcus Cramer et al. "Efficient quantum state tomography". In: *Nature Communications* 1.9 (2010). ISSN: 2041-1723. DOI: 10.1038/ncomms1147.

[31] George Cybenko. "Reducing quantum computations to elementary unitary operations". In: *Computing in Science and Engineering* (2001). ISSN: 1521-9615. DOI: 10.1109/5992.908999.

[32] Piotr Czarnik et al. "Error mitigation with Clifford quantum-circuit data". In: *Quantum* 5 (Nov. 2021), p. 592. ISSN: 2521-327X. DOI: 10.22331/q-2021-11-26-592.

[33] Alexander M. Dalzell, Nicholas Hunter-Jones, and Fernando G. S. L. Brandão. "Random quantum circuits transform local noise into global white noise". In: *arXiv* (Nov. 2021). DOI: 10.48550/arXiv.2111.14907.

[34] Christoph Dankert et al. "Exact and approximate unitary 2-designs and their application to fidelity estimation". In: *Physical Review A* 80.1 (July 2009), p. 012304. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.80.012304.

[35] Christopher M. Dawson and Michael A. Nielsen. "The Solovay-Kitaev algorithm". In: *Quantum Information and Computation* (2006). URL: https://dl.acm.org/doi/10.5555/2011679.2011685.

[36] E. Derbyshire et al. "Randomized benchmarking in the analogue setting". In: *Quantum Science and Technology* 5.3 (Apr. 2020), p. 034001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab7eec.

[37] L. M. Duan, J. I. Cirac, and P. Zoller. "Geometric manipulation of trapped ions for quantum computation". In: *Science* (2001). ISSN: 0036-8075. DOI: 10.1126/science.1058835.

[38] C. L. Edmunds et al. "Measuring and Suppressing Error Correlations in Quantum Circuits". In: *arXiv* (Dec. 2017). DOI: 10.48550/arxiv.1712.04954.

[39] Andreas Elben et al. "Cross-Platform Verification of Intermediate Scale Quantum Devices". In: *Physical Review Letters* 124.1 (Jan. 2020), p. 010504. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.124.010504.

[40] Joseph Emerson, Robert Alicki, and Karol Zyczkowski. "Scalable noise estimation with random unitary operators". In: *Journal of Optics B* 7 (2005). DOI: 10.1088/1464-4266/7/10/021.

[41] Suguru Endo, Simon C. Benjamin, and Ying Li. "Practical Quantum Error Mitigation for Near-Future Applications". In: *Physical Review X* 8.3 (July 2018), p. 031027. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.8.031027.

[42] Jeffrey M. Epstein et al. "Investigating the limits of randomized benchmarking protocols". In: *Physical Review A* 89 (2014). DOI: 10.1103/PhysRevA.89.062321.

[43] Alexander Erhard et al. "Characterizing large-scale quantum computers via cycle benchmarking". In: *Nature Communications* (2019). ISSN: 2041-1723. DOI: 10.1038/s41467-019-13068-7.

[44] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *arXiv* (Nov. 2014). DOI: 10.48550/arxiv.1411.4028.

[45] Richard P. Feynman. "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6 (1982). DOI: 10.1007/BF02650179.

[46] M. A. Fogarty et al. "Nonexponential fidelity decay in randomized benchmarking with low-frequency noise". In: *Physical Review A* 92.2 (Aug. 2015). DOI: 10.1103/PhysRevA.92.022326.

[47] J. P. Gaebler et al. "Randomized benchmarking of multiqubit gates". In: *Physical Review Letters* 108 (2012). DOI: 10.1103/PhysRevLett.108.260503.

[48] Jay M. Gambetta, Jerry M. Chow, and Matthias Steffen. "Building logical qubits in a superconducting quantum computing system". In: *npj Quantum Information* 3.1 (Dec. 2017), p. 2. ISSN: 2056-6387. DOI: 10.1038/s41534-016-0004-0.

[49] Jay M. Gambetta et al. "Characterization of Addressability by Simultaneous Randomized Benchmarking". In: *Physical Review Letters* 109 (2012). DOI: 10.1103/PhysRevLett.109.240504.

[50] Lukas Gerster et al. "Experimental Bayesian Calibration of Trapped-Ion Entangling Operations". In: *PRX Quantum* 3.2 (June 2022), p. 020350. ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.3.020350.

[51] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. "Verification of Quantum Computation: An Overview of Existing Approaches". In: *Theory of Computing Systems* 63.4 (2019). ISSN: 1433-0490. DOI: 10.1007/s00224-018-9872-3.

[52] Ralf Gommers et al. *scipy/scipy: SciPy*. July 2022. DOI: 10.5281/zenodo.595738.

[53] Thomas Gorin et al. "Dynamics of Loschmidt echoes and fidelity decay". In: *Physics Reports* 435 (2006). DOI: 10.1016/j.physrep.2006.09.003.

[54] Dylan J. Gorman et al. "Engineering Vibrationally Assisted Energy Transfer in a Trapped-Ion Quantum Simulator". In: *Physical Review X* 8.1 (2018). DOI: 10.1103/PhysRevX.8.011038.

[55] Daniel Gottesman. "The Heisenberg Representation of Quantum Computers". In: *arXiv* (July 1998). DOI: 10.48550/arxiv.quant-ph/9807006.

[56] Daniel Gottesman. "Theory of fault-tolerant quantum computation". In: *Physical Review A* 57.1 (Jan. 1998), pp. 127–137. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.57.127.

[57] L. C. G. Govia et al. "Bootstrapping quantum process tomography via a perturbative ansatz". In: *Nature Communications* 11.1 (2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-14873-1.

[58] C. Granade et al. "Robust online Hamiltonian learning". In: *New Journal of Physics* 14.10 (Oct. 2012), p. 103013. ISSN: 1367-2630. DOI: 10.1088/1367-2630/14/10/103013.

[59] C. Greganti et al. "Cross-Verification of Independent Quantum Devices". In: *Physical Review X* 11.3 (Sept. 2021), p. 031049. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.11.031049.

[60] Harper R. Grimsley et al. "ADAPT-VQE is insensitive to rough parameter landscapes and barren plateaus". In: *arXiv* (Apr. 2022). DOI: 10.48550/arXiv.2204.07179.

[61] Christian Gross and Immanuel Bloch. "Quantum simulations with ultracold atoms in optical lattices". In: *Science* 357.6355 (Sept. 2017), pp. 995–1001. ISSN: 0036-8075. DOI: 10.1126/science.aal3837.

[62] Andi Gu et al. "Adaptive shot allocation for fast convergence in variational quantum algorithms". In: *arXiv* (Aug. 2021). DOI: 10.48550/arxiv.2108.10434.

[63] Hartmut Häffner, Christian F. Roos, and Rainer Blatt. "Quantum computing with trapped ions". In: *Physics Reports* 469.4 (2008), pp. 155–203. DOI: 10.1016/j.physrep.2008.09.003.

[64] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009), p. 150502. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.103.150502.

[65] Aram W. Harrow and Richard A. Low. "Random quantum circuits are approximate 2-designs". In: *Communications in Mathematical Physics* 291.1 (2009). ISSN: 0010-3616. DOI: 10.1007/s00220-009-0873-6.

[66] Aram W. Harrow, Benjamin Recht, and Isaac L. Chuang. "Efficient discrete approximations of quantum gates". In: *Journal of Mathematical Physics* (2002). ISSN: 0022-2488. DOI: 10.1063/1.1495899.

[67] T. P. Harty et al. "High-Fidelity Preparation, Gates, Memory, and Readout of a Trapped-Ion Quantum Bit". In: *Physical Review Letters* 113.22 (Nov. 2014), p. 220501. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.113.220501.

[68] W. K. Hastings. "Monte carlo sampling methods using Markov chains and their applications". In: *Biometrika* (1970). ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97.

[69] Naomichi Hatano and Masuo Suzuki. "Finding Exponential Product Formulas of Higher Orders". In: *Quantum Annealing and Other Optimization Methods*. 2005. Chap. 2, pp. 37–68. DOI: 10.1007/11526216.

[70] Philipp Hauke et al. "Can one trust quantum simulators?" In: *Reports on Progress in Physics* 75.8 (2012). ISSN: 0034-4885. DOI: 10.1088/0034-4885/75/8/082401.

[71] Jonas Helsen et al. "Matchgate benchmarking: Scalable benchmarking of a continuous family of many-qubit gates". In: *Quantum* 6 (Feb. 2022), p. 657. ISSN: 2521-327X. DOI: 10.22331/q-2022-02-21-657.

[72] Andrew A. Houck, Hakan E. Türeci, and Jens Koch. "On-chip quantum simulation with superconducting circuits". In: *Nature Physics* 8.4 (2012), pp. 292–299. DOI: 10.1038/nphys2251.

[73] F. Huszár and N. M. T. Houlsby. "Adaptive Bayesian quantum tomography". In: *Physical Review A* 85.5 (May 2012), p. 052120. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.85.052120.

[74] Giovanni Iannelli and Karl Jansen. "Noisy Bayesian optimization for variational quantum eigensolvers". In: *arXiv* (Dec. 2021). DOI: 10.48550/arxiv.2112.00426.

[75] IBM. *IBM Quantum*. 2022. URL: https://quantum-computing.ibm.com/.

[76] W. M. Itano et al. "Quantum projection noise: Population fluctuations in two-level systems". In: *Physical Review A* (1993). ISSN: 1050-2947. DOI: 10.1103/PhysRevA.47.3554.

[77] Abhijith J. et al. "Quantum Algorithm Implementations for Beginners". In: *ACM Transactions on Quantum Computing* 3.4 (2022), pp. 1–92. ISSN: 2643-6809. DOI: 10.1145/3517340.

[78] Rodolfo A. Jalabert and Horacio M. Pastawski. "Environment-independent decoherence rate in classically chaotic systems". In: *Physical Review Letters* (2001). ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.86.2490.

[79] James R. Janesick. *Scientific Charge-Coupled Devices*. SPIE Press, 2009. ISBN: 978-0-81943-698-6. DOI: 10.1117/3.374903.

[80] Ali Javadiabhari et al. "ScaffCC: Scalable compilation and analysis of quantum programs". In: *Parallel Computing* (2015). ISSN: 01678191. DOI: 10.1016/j.parco.2014.12.001.

[81] Tomi H. Johnson, Stephen R. Clark, and Dieter Jaksch. "What is a quantum simulator?" In: *EPJ Quantum Technology* 1.1 (2014). DOI: 10.1140/epjqt10.

[82] Richard Jozsa. "Fidelity for Mixed Quantum States". In: *Journal of Modern Optics* 41.12 (Dec. 1994). ISSN: 0950-0340. DOI: 10.1080/09500349414552171.

[83] Abhinav Kandala et al. "Error mitigation extends the computational reach of a noisy quantum processor". In: *Nature* 567.7749 (Mar. 2019), pp. 491–495. ISSN: 0028-0836. DOI: 10.1038/s41586-019-1040-7.

[84] Ivan Kassal et al. "Polynomial-time quantum algorithm for the simulation of chemical dynamics". In: *Proceedings of the National Academy of Sciences* 105.48 (Dec. 2008), pp. 18681–18686. ISSN: 0027-8424. DOI: 10.1073/pnas.0808245105.

[85] Andy Keane, Alexander Forrester, and Andras Sobester. *Engineering Design via Surrogate Modelling: A Practical Guide*. Washington, DC: American Institute of Aeronautics and Astronautics, Inc., Sept. 2008. ISBN: 978-1-56347-955-7. DOI: 10.2514/4.479557.

[86] Sami Khairy et al. "Learning to Optimize Variational Quantum Circuits to Solve Combinatorial Problems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03 (Apr. 2020), pp. 2367–2375. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i03.5616.

[87] Sumeet Khatri et al. "Quantum-assisted quantum compiling". In: *Quantum* (2019). ISSN: 2521-327X. DOI: 10.22331/q-2019-05-13-140.

[88] A. Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (Dec. 1997), pp. 1191–1249. ISSN: 0036-0279. DOI: 10.1070/RM1997v052n06ABEH002155.

[89] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*. Vol. 47. Providence, Rhode Island: American Mathematical Society, May 2002. ISBN: 978-0-8218-3229-5. DOI: 10.1090/gsm/047.

[90] E. Knill. "Approximation by Quantum Circuits". In: *arXiv* (1995). DOI: 10.48550/arXiv.quant-ph/9508006.

[91] E. Knill et al. "Randomized benchmarking of quantum gates". In: *Physical Review A* 77 (2008). DOI: 10.1103/PhysRevA.77.012307.

[92] C. Kokail et al. "Self-verifying variational quantum simulation of lattice models". In: *Nature* 569 (2019). DOI: 10.1038/s41586-019-1177-4.

[93] Jonas M. Kübler et al. "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms". In: *Quantum* 4 (May 2020), p. 263. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-11-263.

[94] Janusz Kusyk, Samah M. Saeed, and Muharrem Umit Uyar. "Survey on Quantum Circuit Compilation for Noisy Intermediate-Scale Quantum Computers: Artificial Intelligence to Heuristics". In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–16. ISSN: 2689-1808. DOI: 10.1109/TQE.2021.3068355.

[95] B. P. Lanyon et al. "Efficient tomography of a quantum many-body system". In: *Nature Physics* 13.12 (2017). ISSN: 1745-2481. DOI: 10.1038/nphys4244.

[96] Wim Lavrijsen et al. "Classical Optimizers for Noisy Intermediate-Scale Quantum Devices". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Oct. 2020, pp. 267–277. ISBN: 978-1-7281-8969-7. DOI: 10.1109/QCE49297.2020.00041.

[97] Kookjin Lee et al. "Partition of unity networks: deep hp-approximation". In: *arXiv* (Jan. 2021). DOI: 10.48550/arxiv.2101.11256.

[98] P. J. Lee et al. "Phase control of trapped ion quantum gates". In: *Journal of Optics B: Quantum and Semiclassical Optics* (2005). ISSN: 1464-4266. DOI: 10.1088/1464-4266/7/10/025.

[99] Yunchao Liu et al. "Benchmarking near-term quantum computers via random circuit sampling". In: *arXiv* (May 2021). DOI: 10.48550/arXiv.2105.05232.

[100] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (July 2019), p. 163. ISSN: 2521-327X. DOI: 10.22331/q-2019-07-12-163.

[101] Nathan K. Lysne et al. "Small, Highly Accurate Quantum Processor for Intermediate-Depth Quantum Simulations". In: *Physical Review Letters* 124.23 (2020). ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.124.230501.

[102] Alicia B. Magann et al. "From Pulses to Circuits and Back Again: A Quantum Optimal Control Perspective on Variational Quantum Algorithms". In: *PRX Quantum* 2.1 (Jan. 2021), p. 010101. ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.2.010101.

[103] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. "Scalable and robust randomized benchmarking of quantum processes". In: *Physical Review Letters* 106 (2011). DOI: 10.1103/PhysRevLett.106.180504.

[104] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. "Characterizing quantum gates via randomized benchmarking". In: *Physical Review A* 85 (2012). DOI: 10.1103/PhysRevA.85.042311.

[105] Easwar Magesan et al. "Efficient measurement of quantum gate error by interleaved randomized benchmarking". In: *Physical Review Letters* 109 (2012). DOI: 10.1103/PhysRevLett.109.080505.

[106] Esteban A. Martinez et al. "Compiling quantum algorithms for architectures with multi-qubit gates". In: *New Journal of Physics* 18.6 (2016). ISSN: 1367-2630. DOI: 10.1088/1367-2630/18/6/063029.

[107] Andreas Mayer and Svein Ove Aas. *andim/noisyopt: v0.2.2*. May 2017. DOI: 10.5281/zenodo.580120.

[108] Jarrod R. McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (2016). ISSN: 1367-2630. DOI: 10.1088/1367-2630/18/2/023023.

[109] David C. McKay et al. "Three-Qubit Randomized Benchmarking". In: *Physical Review Letters* 122 (2019). DOI: 10.1103/PhysRevLett.122.200502.

[110] Seth T. Merkel et al. "Self-consistent quantum process tomography". In: *Physical Review A* 87.6 (June 2013), p. 062119. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.87.062119.

[111] Francesco Mezzadri. "How to generate random matrices from the classical compact groups". In: *arXiv* (2006). DOI: 10.48550/arXiv.math-ph/0609050.

[112] Klaus Mølmer and Anders Sørensen. "Multiparticle Entanglement of Hot Trapped Ions". In: *Physical Review Letters* 82.9 (Mar. 1999), pp. 1835–1838. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.82.1835.

[113] Juliane Mueller et al. "Accelerating Noisy VQE Optimization with Gaussian Processes". In: *arXiv* (Apr. 2022). DOI: 10.48550/arXiv.2204.07331.

[114] A. H. Myerson et al. "High-Fidelity Readout of Trapped-Ion Qubits". In: *Physical Review Letters* 100.20 (May 2008), p. 200502. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.100.200502.

[115] V. Nebendahl, H. Häffner, and C. F. Roos. "Optimal control of entangling operations for trapped-ion quantum computing". In: *Physical Review A* 79 (2009). DOI: 10.1103/PhysRevA.79.012312.

[116] C. Neill et al. "A blueprint for demonstrating quantum supremacy with superconducting qubits". In: *Science* (2018). ISSN: 10959203. DOI: 10.1126/science.aao4309.

[117] Erik Nielsen et al. "Gate Set Tomography". In: *Quantum* 5 (2021). ISSN: 2521-327X. DOI: 10.22331/q-2021-10-05-557.

[118] Michael A Nielsen. "A simple formula for the average gate fidelity of a quantum dynamical operation". In: *Physics Letters A* 303.4 (Oct. 2002), pp. 249–252. ISSN: 0375-9601. DOI: 10.1016/S0375-9601(02)01272-0.

[119] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, June 2012. ISBN: 978-1-10700-217-3. DOI: 10.1017/CBO9780511976667.

[120] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, 2006. ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5.

[121] Yingkai Ouyang, David R. White, and Earl T. Campbell. "Compilation by stochastic Hamiltonian sparsification". In: *Quantum* (2020). ISSN: 2521-327X. DOI: 10.22331/q-2020-02-27-235.

[122] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5 (2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[123] S. Poletto et al. "Entanglement of two superconducting qubits in a waveguide cavity via monochromatic two-photon excitation". In: *Physical Review Letters* (2012). ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.109.240505.

[124] David Poulin et al. "Quantum simulation of time-dependent hamiltonians and the convenient illusion of hilbert space". In: *Physical Review Letters* (2011). ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.106.170501.

[125] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018). DOI: 10.22331/q-2018-08-06-79.

[126] Timothy Proctor et al. "Scalable randomized benchmarking of quantum computers using mirror circuits". In: *arXiv* (Dec. 2021). DOI: 10.48550/arXiv.2112.09853.

[127] Timothy J. Proctor et al. "Direct Randomized Benchmarking for Multiqubit Devices". In: *Physical Review Letters* 123.3 (July 2019), p. 030503. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.123.030503.

[128] Nestor V. Queipo et al. "Surrogate-based analysis and optimization". In: *Progress in Aerospace Sciences* 41.1 (Jan. 2005), pp. 1–28. ISSN: 0376-0421. DOI: 10.1016/j.paerosci.2005.02.001.

[129] Melissa C. Revelle. "Phoenix and Peregrine Ion Traps". In: *arXiv* (Sept. 2020). DOI: 10.48550/arxiv.2009.02398.

[130] Christian F. Roos. "Ion trap quantum gates with amplitude-modulated laser beams". In: *New Journal of Physics* 10.1 (Jan. 2008), p. 013002. ISSN: 1367-2630. DOI: 10.1088/1367-2630/10/1/013002.

[131] Eric Schkufza, Rahul Sharma, and Alex Aiken. "Stochastic superoptimization". In: *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM Press, 2013, p. 305. DOI: 10.1145/2451116.2451150.

[132] Alireza Seif et al. "Machine learning assisted readout of trapped-ion qubits". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 51.17 (Sept. 2018), p. 174006. ISSN: 0953-4075. DOI: 10.1088/1361-6455/aad62b.

[133] Chris N. Self et al. "Variational quantum algorithm with information sharing". In: *npj Quantum Information* 7.1 (Dec. 2021), p. 116. ISSN: 2056-6387. DOI: 10.1038/s41534-021-00452-9.

[134] A. Shabani et al. "Efficient measurement of quantum dynamics via compressive sensing". In: *Physical Review Letters* 106.10 (2011). ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.106.100401.

[135] Ryan Shaffer. *rmshaffer/pulse-sequence-simulator: v0.1.0*. July 2022. DOI: 10.5281/zenodo.6913408.

[136] Ryan Shaffer. *rmshaffer/stoq-compiler: v0.2.0*. Aug. 2022. DOI: 10.5281/zenodo.7007254.

[137] Ryan Shaffer. *sandialabs/sbovqaopt: v0.1.0*. Sept. 2022. DOI: 10.5281/zenodo.7062967.

[138] Ryan Shaffer, Lucas Kocia, and Mohan Sarovar. "Surrogate-based optimization for variational quantum algorithms". In: *arXiv* (Apr. 2022). URL: https://arxiv.org/abs/2204.05451.

[139] Ryan Shaffer and John Paul Marceaux. *HaeffnerLab/hamiltonian-learning: v0.1.0-alpha*. Aug. 2022. DOI: 10.5281/zenodo.7023483.

[140] Ryan Shaffer et al. "Efficient verification of continuously-parameterized quantum gates". In: *arXiv* (May 2022). DOI: 10.48550/arxiv.2205.13074.

[141] Ryan Shaffer et al. "Practical verification protocols for analog quantum simulators". In: *npj Quantum Information* 7.46 (2021), pp. 1–12. ISSN: 2056-6387. DOI: 10.1038/s41534-021-00380-8.

[142] Kunal Sharma et al. "Noise resilience of variational quantum compiling". In: *New Journal of Physics* (2020). ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab784c.

[143] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172.

[144] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986. DOI: 10.1007/978-1-4899-3324-9.

[145] Anders Sørensen and Klaus Mølmer. "Quantum computation with ions in thermal motion". In: *Physical Review Letters* (1999). ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.82.1971.

[146] James C. Spall. "An Overview of the Simultaneous Perturbation Method for Efficient Optimization". In: *Johns Hopkins APL Technical Digest* 19.4 (1998), p. 11. URL: https://www.jhuapl.edu/spsa/PDF-SPSA/Spall_An_Overview.PDF.

[147] Kevin J. Sung et al. "Using models to improve optimizers for variational quantum algorithms". In: *Quantum Science and Technology* 5.4 (Oct. 2020), p. 044008. ISSN: 2058-9565. DOI: 10.1088/2058-9565/abb6d9.

[148] Masuo Suzuki. "Fractal decomposition of exponential operators with applications to many-body theories and Monte Carlo simulations". In: *Physics Letters A* (1990). ISSN: 0375-9601. DOI: 10.1016/0375-9601(90)90962-N.

[149] Ryan Sweke et al. "Stochastic gradient descent for hybrid quantum-classical optimization". In: *Quantum* 4 (Aug. 2020), p. 314. ISSN: 2521-327X. DOI: 10.22331/q-2020-08-31-314.

[150] Shiro Tamiya and Hayata Yamasaki. "Stochastic gradient line Bayesian optimization for efficient noise-robust optimization of parameterized quantum circuits". In: *npj Quantum Information* 8.1 (Dec. 2022), p. 90. ISSN: 2056-6387. DOI: 10.1038/s41534-022-00592-6.

[151] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. "Error Mitigation for Short-Depth Quantum Circuits". In: *Physical Review Letters* 119.18 (Nov. 2017), p. 180509. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.119.180509.

[152] Matthew Treinish et al. *Qiskit/qiskit: Qiskit 0.36.2*. May 2022. DOI: 10.5281/zenodo.6560959.

[153] Caterina Vigliar et al. "Error-protected qubits in a silicon photonic chip". In: *Nature Physics* 17.10 (Oct. 2021), pp. 1137–1143. ISSN: 1745-2473. DOI: 10.1038/s41567-021-01333-w.

[154] Joel J. Wallman. "Randomized benchmarking with gate-dependent noise". In: *Quantum* 2 (2018), p. 47. DOI: 10.22331/q-2018-01-29-47.

[155] Joel J. Wallman and Joseph Emerson. "Noise tailoring for scalable quantum computation via randomized compiling". In: *Physical Review A* 94 (2016), p. 52325. DOI: 10.1103/PhysRevA.94.052325.

[156] P. Walther et al. "Experimental one-way quantum computing". In: *Nature* 434.7030 (2005). ISSN: 0028-0836. DOI: 10.1038/nature03347.

[157] Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, Dec. 2004. ISBN: 978-0-52184-335-5. DOI: 10.1017/CBO9780511617539.

[158] Nathan Wiebe et al. "Hamiltonian learning and certification using quantum resources". In: *Physical Review Letters* (2014). ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.112.190501.

[159] Greg von Winckel. *pyQAOA: Simulation of Quantum Approximate Optimization Algorithms in Python*. 2021. URL: https://github.com/gregvw/pyQAOA.

[160] J. Zhang et al. "Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator". In: *Nature* 551 (2017), pp. 601–604. DOI: 10.1038/nature24654.

[161] Liudmila A. Zhukas et al. "High-fidelity simultaneous detection of a trapped-ion qubit register". In: *Physical Review A* 103.6 (June 2021), p. 062614. ISSN: 2469-9926. DOI: 10.1103/PhysRevA.103.062614.

Appendix A

Simulating trapped-ion experiments

A.1 Introduction

Trapped ions are among the most established and most promising platforms for building a quantum computer [63, 17]. Ions are trapped in a vacuum chamber by photoionizing an atom of choice, and then tightly confining the resulting positively-charged ions using carefully-tuned DC and RF potentials, ideally forming a one-dimensional crystal, i.e., a linear "chain" of ions. Lasers tuned to specific wavelengths, typically near the energy transitions of the ions' outermost electrons, are used to slow the motion of the ion chain and cool it nearly to its motional ground state.

After the ions have been trapped and cooled, they can be used to perform quantum logic. Typically, two of the electronic energy levels are chosen as the qubit states $|0\rangle$ and $|1\rangle$, and the ions are initialized into the $|0\rangle$ state with very high probability. Next, a quantum circuit of choice is executed on the ion chain by using precisely-timed laser pulses to perform one- and two-qubit gates on the ions. Finally, the states of the ions are measured using a technique that typically requires capturing photons and determining whether each ion is "dark" or "bright", and then mapping that to the corresponding computational basis state.

Building a trapped-ion quantum computer is a complicated endeavor. As an illustration, Figure A.1 is a photograph of only a portion of a trapped-ion quantum computer in the Häffner lab at UC Berkeley. The experimental apparatus includes not only the ion trap itself and surrounding vacuum chamber, but also a table full of fibers and free-space optics, several laser sources, a room full of various electronic equipment, imaging devices for performing state measurement, a programmable control system which can output RF pulses with the precision necessary for high-fidelity gates, and a computer with a user interface which allows the operator to monitor and control the device's overall operation.

Because the device is so complex, designing and testing new quantum programs to run on the apparatus can be tedious and unreliable. First, there is significant overhead to keeping the device calibrated and in proper operation, since lasers must often be re-locked and re-calibrated, optics periodically need to be re-aligned, and ions may occasionally be lost from

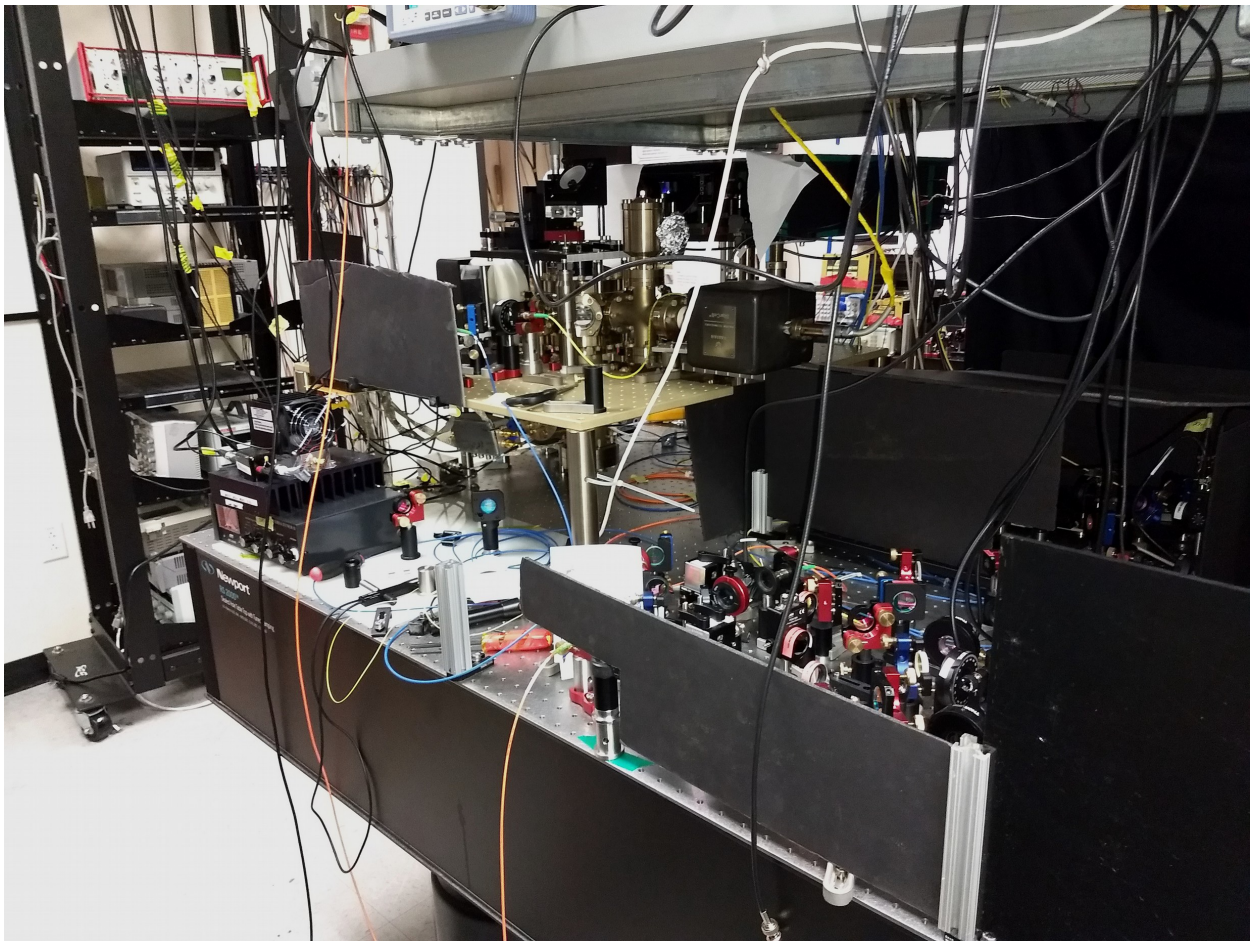


Figure A.1: An experimental trapped-ion quantum computer at UC Berkeley.

the trap. Second, the device is subject to a variety of noise sources, some of which can change significantly over the course of days or even hours. And third, because the apparatus can run only one program at a time, any time spent iterating on the development of a new program prevents others in the lab from performing productive work on the apparatus. On top of all this, it is not always possible to run the experimental apparatus; in the last three years alone, the UC Berkeley experiment has been inoperable on several occasions for days or weeks at a time due to wildfire-induced power outages, building HVAC failures, and pandemic-related lab occupancy restrictions.

For all these reasons, it is extremely advantageous to have a procedure for developing and testing experimental programs without actually requiring a connection to the physical experiment. Ideally, one would be able to write an experimental program, accurately simulate its operation on the real apparatus, debug any problems, and become confident that the

program is performing as intended – all on a computer that is not connected to anything in the lab. Subsequently, the same program can be quickly loaded onto the computer connected to the experiment and tested on the physical apparatus.

In this appendix, we describe several efforts toward such a goal. First, in Section A.2, we discuss software we have built which can run experimental programs independently of the physical apparatus, using the Julia library IonSim.jl [16] to accurately emulate the physics of the laser-ion interactions. Second, in Section A.3, we describe a technique for generating artificial EMCCD camera images of ions which closely resemble real images obtained from the experiment. Third, in Section A.4, we demonstrate a statistical learning procedure for experimental control settings using simulated experimental data.

A.2 Simulating experimental programs

The trapped-ion apparatus shown in Figure A.1 uses the ARTIQ control system [15] to manage the execution of experimental programs. Such programs are often referred to as "pulse sequences", since the control system must deliver precisely-timed TTL and DDS pulses to the various electronic components that control the experiment, which in turn deliver the laser pulses which will drive the desired operations on the ion chain.

In order to run these programs without a physical connection to the ARTIQ system, we developed software [135] to perform the following tasks[1]:

1. Intercept all commands sent to the ARTIQ infrastructure. This includes commands to delay for a certain amount of time, switch a DDS channel on/off, set the frequency or amplitude of a DDS channel, etc. Store this as a list of "pulses", where each pulse specifies the on/off time, amplitude, attenuation, frequency, and phase for a particular output channel.

2. Use knowledge of the experimental setup to combine DDS pulses which affect the same laser tone. For example, if a beam path contains two AOMs which are driven by separate DDS channels, then the pulses for those two channels must be combined by calculating the overlapping times when both channels are on, adding the frequency and phase offsets, adding the attenuation values, and multiplying the amplitude factors.

3. Pass this list of combined pulses from Python to Julia (using PyJulia [3]) and use them to construct a list of `Laser` objects in IonSim.jl.

4. Use the start and stop times for each pulse to construct a time-dependent E-field for each `Laser`. This E-field must have the appropriate amplitude for times when the pulse is on, and zero amplitude for times when the pulse is off.

[1]The implementation of the tasks described in this section can be found within [135]. Specifically, the code resides in the Python script `simulated_pulse_sequence.py` and the Julia script `simulate.jl`.

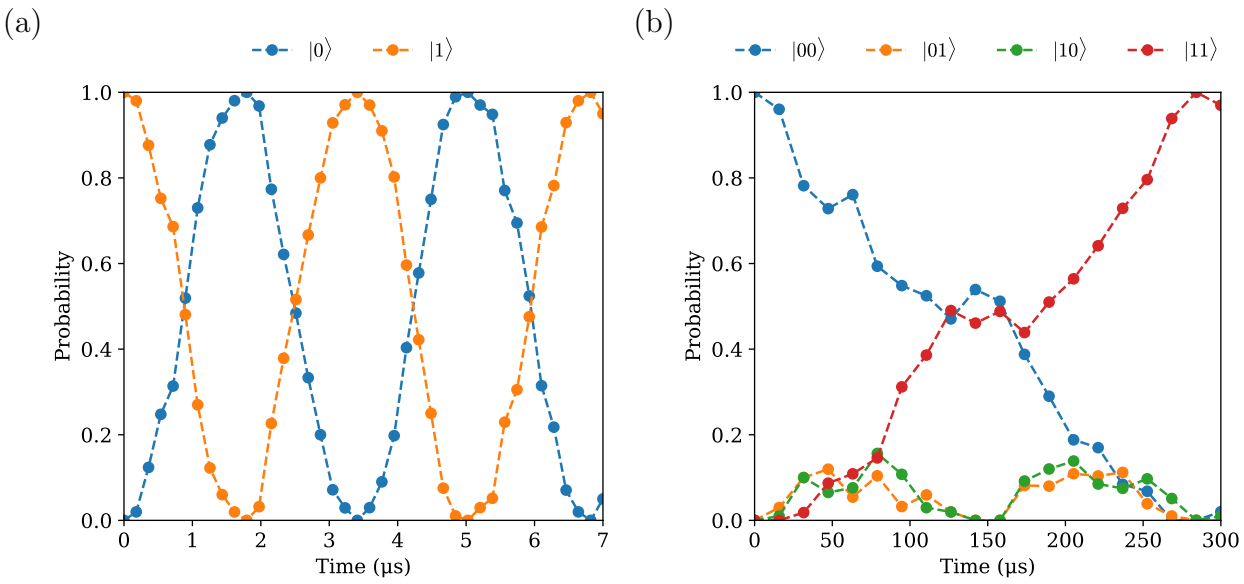


Figure A.2: Simulated results of experimental pulse sequences using IonSim.jl to emulate the physics of the laser-ion interactions. (a) Single-ion Rabi flopping. (b) Two-ion Mølmer-Sørensen interaction.

5. Simulate the system in IonSim.jl over the range of time which contains pulse on/off events, and calculate the ideal measurement probabilities of each output state after the simulation is complete.

6. Simulate quantum projection noise by sampling from this output probability distribution a fixed number of times (e.g., 100), and return the resulting counts.

This technique allows the experimental programs to produce simulated results which closely match what the real experiment should ideally produce. Figure A.2 displays results from this simulation technique for a single-ion Rabi flopping pulse sequence and a two-ion Mølmer-Sørensen interaction pulse sequence. An important note is that the simulation is performed in an ideal controlled environment, i.e., without state preparation or measurement errors, without drift in laser parameters, without heating of the ion chain, etc. Because of this, we observe that the results are nearly ideal, with the exception of the uncertainty created by the quantum projection noise. This is a useful feature for debugging experimental programs, since errors in the pulse sequence logic can be identified and fixed independently of any physical errors that may occur in the real experiment.

Another benefit of the simulation is that the runtime is much faster than that of the physical experiment. The simulation results shown in Figure A.2 took only $\sim$ 1 second to generate in full, whereas a similar experimental run with 100 shots per data point would take closer to $\sim$ 1 minute ($\sim$ 1-2 seconds per data point).

A.3 Simulating EMCCD camera readout

Motivation

Performing state readout of an ion chain is typically done by collecting emitted photons using either a photomultiplier tube (PMT) or a camera with an electron-multiplying charge-coupled device (EMCCD) sensor [17]. One clear advantage of using an EMCCD camera is that it provides two-dimensional spatial information about the collected photons, which enables imaging of the ion chain with single-ion resolution. Readout fidelities of 99.99% have been achieved on chains of four ions using an EMCCD camera with an exposure time of just 400 μs [18].

Improving readout fidelities on longer chains of ions and with shorter exposure times continues to be an active area of research [18, 114, 160, 161]. One approach is to train a machine learning model which can correctly label the state of each ion in an image [132]. In order to train a supervised model capable of >99.99% accuracy, however, obtaining a large quantity (i.e., at least 10^6, and likely much more) of reliably-labeled training images is critical. Collecting such images from experiment may be difficult, not only because it would consume several hours of experimental time, but also because accurately labeling images depends crucially on accurate state preparation. State preparation error for trapped ions as low as 2×10^{-4} has been achieved [67], but even this would result in approximately 200 incorrectly-labeled images in a training set of size 10^6. In other words, the readout accuracy of the model would be limited by the average experimental state preparation error when generating the training set.

One approach to overcoming this limitation is to produce a physically-realistic simulation of the EMCCD image capture for an ion chain in a given state. Because this technique does not rely on experimental state preparation, it would enable the generation of an arbitrarily large number of artificial training images which could be reliably labeled.

Generating artificial EMCCD images of ions

An EMCCD image is the result of a multi-step physical process [19, 79] that converts photons incident on the EMCCD sensor into an amplified signal with per-pixel brightness values which are transferred classically to a readout device, e.g., the experimental control computer.

The first step in generating an artificial image is to simulate the physical process of photons being emitted from a bright ion and being absorbed by a particular pixel on the EMCCD sensor. We model the photon emission as a Poisson process determined by the lifetime of the excited state being used for readout. We model the photon arrival area on the sensor as a two-dimensional Gaussian with mean and covariance determined by the geometry of the experiment and the expected arrival area of photons on the EMCCD sensor. For any given exposure time, then, we can produce a simulation that gives the number of photons incident on each pixel. Such a simulation is implemented the Python function shown in Figure A.3, and an example output of this simulation is shown in Figure A.5(a).

```python
import numpy as np

def generate_ideal_artificial_image(
    image_height: int,       # image height (in pixels)
    image_width: int,        # image width (in pixels)
    photon_rate: float,      # photon incidence rate (per second)
    exposure_time: float,    # exposure time (in seconds)
    mu_x: float,             # center x coordinate of ion
    mu_y: float,             # center y coordinate of ion
    sigma_x: float,          # std dev of Gaussian in x coordinate
    sigma_y: float,          # std dev of Gaussian in y coordinate
):
    mean = [mu_x, mu_y]
    cov = [[sigma_x**2, 0], [0, sigma_y**2]]
    n_photons = np.random.poisson(photon_rate * exposure_time)
    photon_coords = np.random.multivariate_normal(mean, cov, n_photons)
    pixels = np.zeros((image_height, image_width), dtype=np.float64)
    for x, y in photon_coords:
        try:
            pixels[int(round(y))][int(round(x))] += 1.0
        except:
            pass  # photon not incident on sensor area
    return pixels
```

Figure A.3: Python code to generate a simulated EMCCD image of a bright ion without considering EMCCD noise sources.

However, an EMCCD is not simply a photon counter; there are a variety of noise sources which impact the output. Some of these can be ignored in practice. For example, thermal dark counts are negligible when the camera sensor is cooled sufficiently [19]. But other noise sources are more significant and must be considered in order to generate realistic images. In particular, we choose to model three significant noise sources:

1. *Charge transfer efficiency* (CTE) *noise.* Readout of pixels is typically performed by transferring electrons along the rows and columns of the EMCCD grid to the readout electronics which connect to the edges of the sensor. This results in some crosstalk, which means that pixels in the same row or column as a "bright" pixel may experience some error. We model this by adding noise drawn from a gamma distribution with a mean that is some fraction of the sum of the neighboring pixel values (i.e., pixels in the same row or column) weighted by their distance.

2. *Gain noise.* Also referred to as *excess noise factor* (ENF), the effect of gain noise is that

the signal from each photon is not amplified identically by the EMCCD sensor. This essentially results in a "blurring" of each photon as it as absorbed by the detector. We model this by drawing each pixel reading from a gamma distribution with the expected photon count as the mean.

3. *Readout noise.* The readout process itself also experiences some electronic noise, which we model by drawing from a zero-mean Gaussian distribution independently for each pixel and adding this to the pixel reading.

These noise sources can be simulated and added to the ideal image, as implemented in the Python function shown in Figure A.4, and an example image including the simulated noise sources is shown in Figure A.5(b). Note that a "dark" image can be simulated by simply setting `photon_rate = 0`.

Comparison to experimental images

To test the validity of the artificially-generated images, it is necessary to compare them to real EMCCD images obtained from experiment. To do so, we trap a single $^{40}Ca^{+}$ ion in a linear Paul trap and use Doppler cooling to reduce the axial motion to $\overline{n} \sim 10$. We then collect "bright" images of the ion by illuminating it with both 397 nm and 866 nm laser light, which causes resonance fluorescence on the $4^2S_{1/2} \leftrightarrow 4^2P_{1/2}$ dipole transition. While the ion fluoresces, we expose the Nüvü HNü EMCCD camera for the desired time. To collect "dark" images, we repeat the same process, but with no ion in the trap.

Following this procedure, we collected 900 "bright" and 900 "dark" images experimentally using an exposure time of 2.0 ms. We also generated 900 "bright" and 900 "dark" simulated images using parameters that most closely align with the experimental results. Histograms of the resulting image brightness are displayed in Figure A.6. For Figure A.6(b), artificial images with simulated EMCCD noise are generated using the Python function from Figure A.4 with parameters `image_width = 10`, `image_height = 11`, `photon_rate = 6.5e3`, `exposure_time = 2.0e-3`, `(mu_x, mu_y) = (5.71, 4.61)`, `(sigma_x, sigma_y) = (0.7324, 0.8642)`, `gain_noise = 0.1`, `cte_noise = 0.1`, and `readout_noise = 0.05`. These parameters were chosen in order to produce images which exhibited similar statistics to the experimentally-obtained images.

We observe reasonably good agreement between the two histograms. One notable feature of the experimental data is the asymmetric "tail" on the distribution of the dark images, which is missing in the simulated data. One possible cause of this is that our EMCCD experiences some thermal dark counts, which we have not included in our simulation. But more plausibly, this may be an artifact of actual 397nm light which is scattering from the experimental apparatus and reaching the EMCCD sensor.

```python
import copy
from itertools import product
import numpy as np

def generate_artificial_image_with_emccd_noise(
    image_height: int,       # image height (in pixels)
    image_width: int,        # image width (in pixels)
    photon_rate: float,      # photon incidence rate (per second)
    exposure_time: float,    # exposure time (in seconds)
    mu_x: float,             # center x coordinate of ion
    mu_y: float,             # center y coordinate of ion
    sigma_x: float,          # std dev of Gaussian in x coordinate
    sigma_y: float,          # std dev of Gaussian in y coordinate
    cte_noise: float,        # magnitude of CTE noise
    gain_noise: float,       # magnitude of gain noise
    readout_noise: float,    # magnitude of Gaussian readout noise
):
    # first, generate the ideal image without noise
    pixels = generate_ideal_artificial_image(image_height, image_width,
        photon_rate, exposure_time, mu_x, mu_y, sigma_x, sigma_y)

    # add charge transfer efficiency (CTE) noise
    pixels_before_cte = copy.deepcopy(pixels)
    for x, y in product(range(image_width), range(image_height)):
        row_sum = sum([pixels_before_cte[y][x_neighbor] / abs(x_neighbor - x)
                       for x_neighbor in range(image_width) if x_neighbor != x])
        col_sum = sum([pixels_before_cte[y_neighbor][x] / abs(y_neighbor - y)
                       for y_neighbor in range(image_height) if y_neighbor != y])
        pixels[y][x] += np.random.gamma(cte_noise * (row_sum + col_sum))

    # add gain noise
    for x, y in product(range(image_width), range(image_height)):
        if pixels[y][x] > 0:
            pixels[y][x] += gain_noise * (np.random.gamma(pixels[y][x]) - pixels[y][x])

    # add readout noise
    for x, y in product(range(image_width), range(image_height)):
        pixel_gaussian_readout_noise = np.random.normal(0, readout_noise)
        pixels[y][x] += pixel_gaussian_readout_noise

    return pixels
```

Figure A.4: Python code to generate a simulated EMCCD image of a bright ion including realistic noise from various sources. Uses the `generate_ideal_artificial_image` function from Figure A.3.

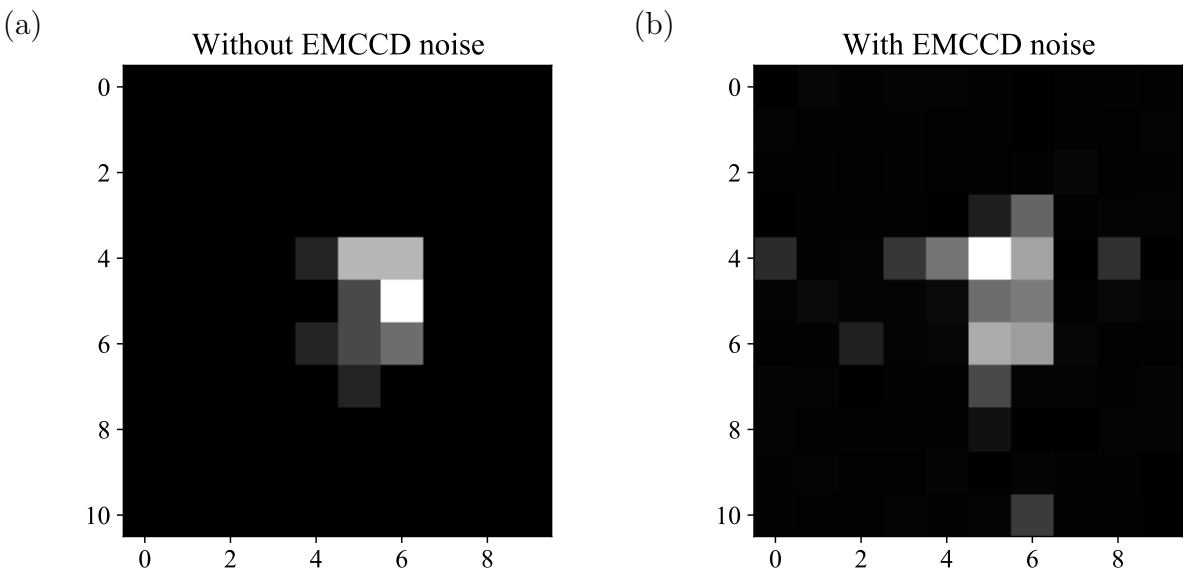


Figure A.5: Simulated EMCCD images produced by the Python functions in Figure A.3 and Figure A.4. (a) Simulated image generated without EMCCD noise, using `image_width = 10`, `image_height = 11`, `photon_rate = 8e3`, `exposure_time = 4e-3`, `(mu_x, mu_y) = (5.71, 4.61)`, and `(sigma_x, sigma_y) = (0.7324, 0.8642)`. (b) The same image with simulated EMCCD noise sources included, using `gain_noise = 1.0`, `cte_noise = 0.1`, and `readout_noise = 0.05`.

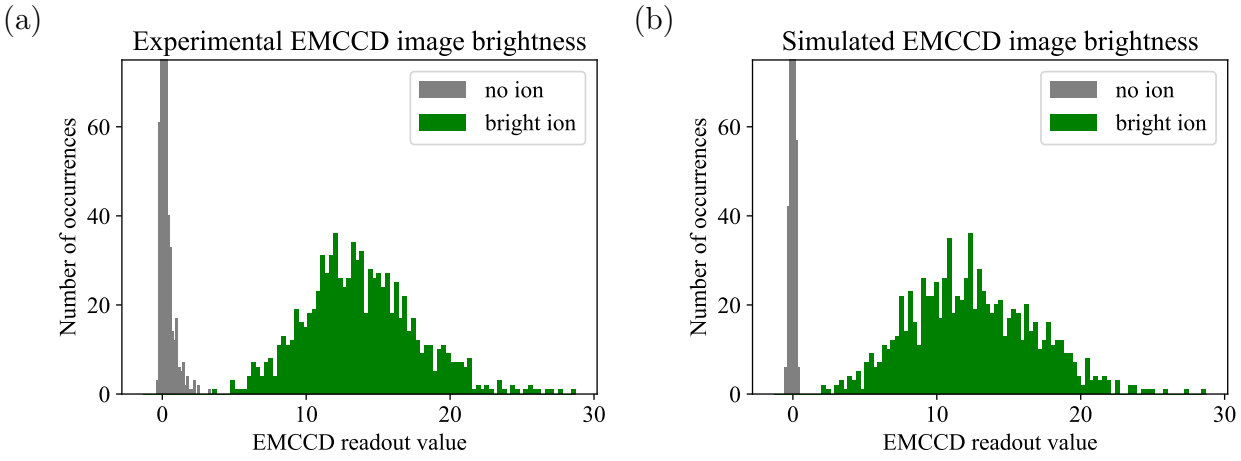


Figure A.6: Histograms of total brightness for experimental and simulated EMCCD images in a fixed 3×3 pixel region of interest. (a) Experimental data using an exposure time of 2.0 ms. (b) Simulated data using generated images with simulated EMCCD noise.

A.4 Simulating experimental control learning

Motivation

Calibration is a necessary and time-consuming step in the operation of any quantum information processor. For example, in the case of a typical trapped-ion processor, laser frequencies and amplitudes must be measured and tuned, optics must be aligned, electronic components such as AOMs must be aligned and calibrated, and readout measurements must be periodically adjusted to account for environmental fluctuations. All of these must be optimized in order to achieve the highest-fidelity gates. Fortunately, in a well-controlled environment, many of these may remain largely stable over the timescale of days, but there are inevitably small fluctuations over shorter timescales that require frequent (sub-hourly) calibration to maintain optimal gate performance. This is essentially a multi-dimensional control optimization problem, in which the task is to find the particular settings of the experimental controls which maximize gate fidelities.

Often the experimentalist performs such calibrations manually by executing a series of one-dimensional parameter scans which attempts to optimize over the multi-dimensional control parameter space. One detailed example of such an approach is provided in [2], in which calibration is performed in order to maximize the fidelity of a two-qubit Mølmer-Sørensen gate.

Because manual calibration is repetitive and time-consuming for a human experimentalist, the development of an automated method for optimizing control parameters could save considerable energy for the experimentalist, in addition to reducing the device time spent on calibration. Indeed, an automated Bayesian calibration technique for trapped-ion devices [50] has been demonstrated which can optimize the calibration of a Mølmer-Sørensen gate in less than a minute, which not only saves considerable time on the part of the experimentalist, but also uses several times fewer measurements than the manual approach while producing a better result.

In general, then, we expect that automating calibration procedures is a valuable avenue to improving the efficiency and reliability of operation of quantum information processors. And as we have discussed previously in this appendix, the ability to develop and test such calibration procedures independently from the experiment is important for both rapid development and for allowing maximal utility of the experiment itself. In this section, we describe efforts toward the simulation of experimental control learning procedures for one-qubit and two-qubit gates, taking advantage of the Julia library IonSim.jl [16] to accurately emulate the experimental dynamics. The code used for the control learning demonstrations in this section is freely available [139].

Single-qubit control learning

One of the simplest single-qubit operations experimentally is driving coherent oscillations between the $|0\rangle$ and $|1\rangle$ qubit states, which is often referred to as Rabi flopping. If the

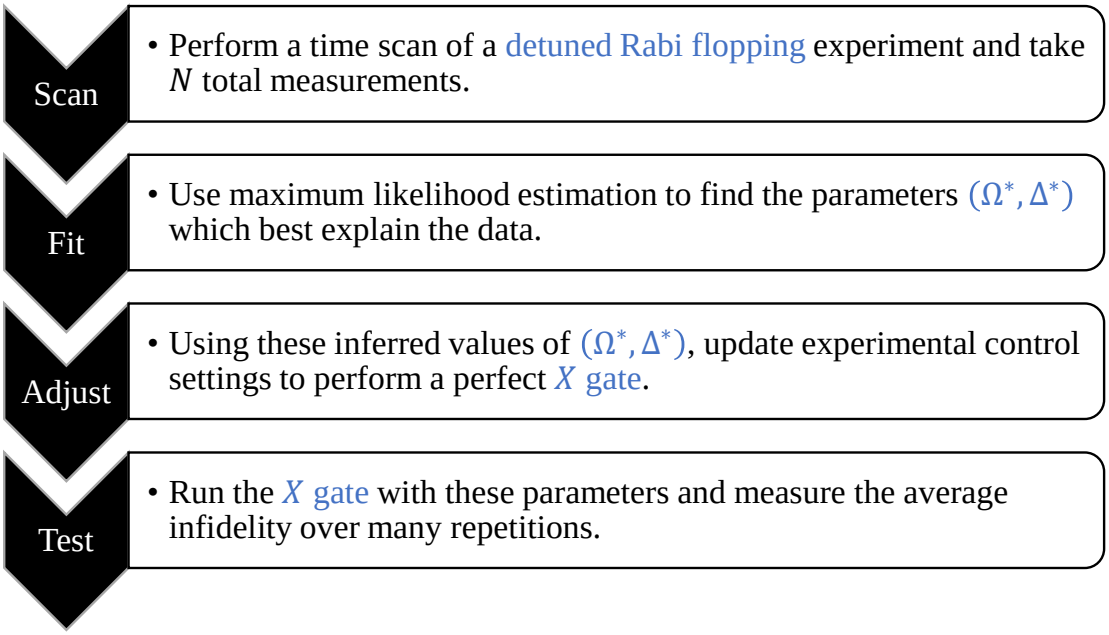


Figure A.7: Illustration of single-qubit control learning procedure.

driving field is exactly in resonance with the qubit frequency, Rabi flopping will cycle the population completely from the $|0\rangle$ state to the $|1\rangle$ state and back. If the field is off-resonant, however, the amplitude of the oscillations decreases. The resulting population dynamics can be described by the Hamiltonian

$$H = \frac{\hbar\Omega}{2}\sigma_x - \frac{\hbar\Delta}{2}\sigma_z \tag{A.1}$$

where Ω is the Rabi frequency (the frequency of the oscillations on resonance), and Δ is the detuning.

In a typical trapped-ion experiment, the Rabi frequency Ω is proportional to the amplitude of the driving laser field, and the detuning Δ is the difference between the laser frequency and the resonance frequency of the qubit. By accurately measuring the empirical values of Ω and Δ, then, we can infer the experimental control settings which would allow us to set Ω and Δ to any desired value.

To simulate this procedure, we follow the process depicted in Figure A.7. We first define "true" values of the parameters (Ω, Δ) to be used in the simulation, which represent the actual (unknown) experimental values that we wish to learn. Importantly, we use a value of Δ that is far from zero, since the symmetric behavior around $\Delta = 0$ would prevent us from learning the correct sign. We then simulate an experimental time scan by using IonSim.jl to take N total measurements over a range of times (in this example, we use 20 points over the range 0 μs to 40 μs). We take the resulting data and use maximum likelihood estimation,

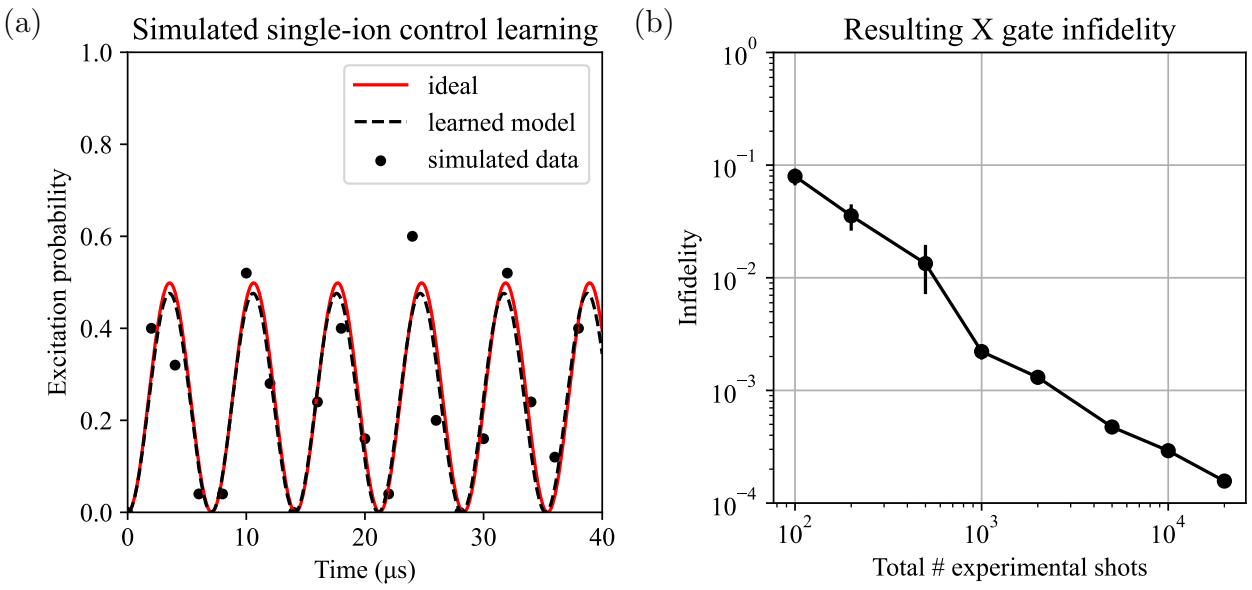


Figure A.8: Simulated single-ion control learning using detuned Rabi oscillations. (a) Illustration of model learned from $N = 500$ total shots obtained from a time scan with 25 shots per point. Data points are obtained by simulating an ideal time evolution of the Hamiltonian from Equation A.1 with $\Omega = 2\pi \times 200$ kHz and $\Delta = 2\pi \times 100$ kHz. (b) Simulated infidelity of an X gate using models learned from varying numbers of experimental shots.

using the dynamics predicted by the time-independent Hamiltonian of Equation A.1 as our objective function, to find the parameters (Ω^*, Δ^*) which best explain the data. Finally, we consider the difference between the learned values (Ω^*, Δ^*) and the ideal values (Ω, Δ) to be our experimental error, and we use IonSim.jl to simulate and measure the experimental infidelity of an X gate under this error. This represents the error that would be incurred if we had adjusted our controls assuming that the learned values (Ω^*, Δ^*) were correct.

Figure A.8 shows the results of using this approach to calibrate an X gate (i.e., a π-pulse). We observe in Figure A.8(b) that using a single time scan consisting of only a few thousand total shots, which would take approximately one minute to execute experimentally, we reach a simulated single-qubit gate fidelity exceeding 99.9%.

Multi-qubit control learning

Calibration must also be performed for multi-qubit interactions, Typically these interactions are significantly more complex than single-qubit interactions and involve several additional experimental control settings. In a trapped-ion experiment, a commonly-used multi-qubit interaction is the Mølmer-Sørensen (MS) interaction [112, 145], which can be used to act on a two-qubit state according to Equation 2.26. When operating on two ions initialized in the

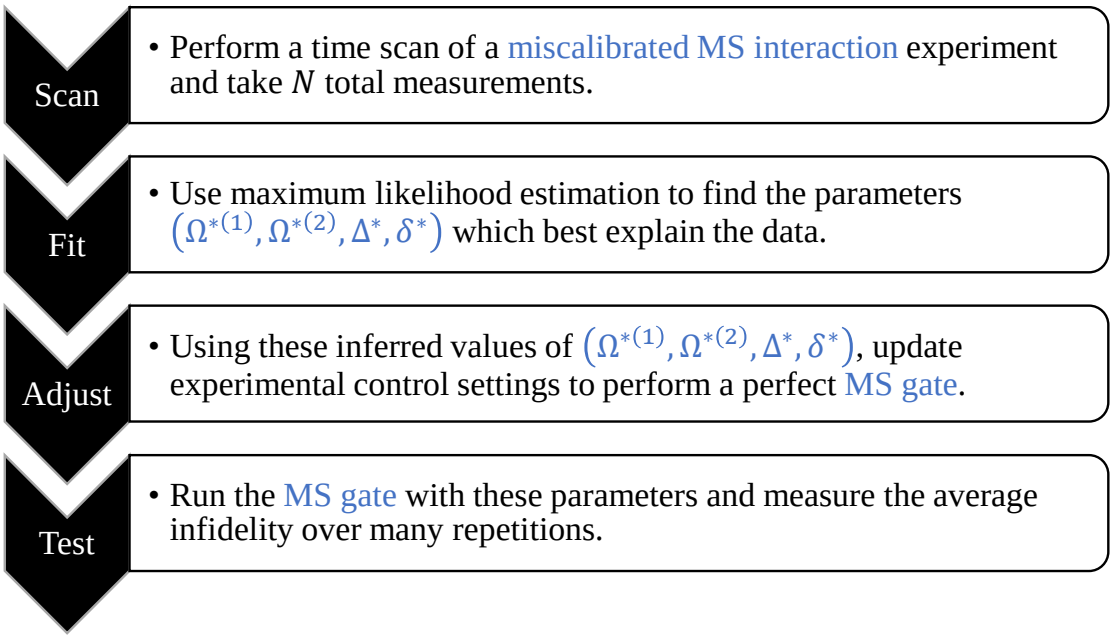


Figure A.9: Illustration of multi-qubit control learning procedure.

$|00\rangle$ state, this interaction creates entangled states of the form $\alpha|00\rangle + \beta|11\rangle$, but the gate parameters must be carefully tuned to avoid leaving unwanted population in the $|01\rangle$ and $|10\rangle$ states.

Fundamentally, the MS interaction works by illuminating an ion chain with two laser tones, often called "blue" and "red" tones, which are ideally detuned from the qubit transition frequency by $(\nu + \Delta)$ and $(-\nu - \Delta)$, respectively. Here ν is the motional frequency of the ion chain and Δ is a small detuning (such that the MS gate time occurs at $1/\Delta$). Additional details on the MS interaction can be found in [63, 130]. To describe the dynamics of the two-ion MS interaction, we perform a rotating-wave approximation with respect to the qubit frequency [130] to arrive at the time-dependent Hamiltonian

$$H = \hbar\Omega \left(\sigma_+^{(1)} + \sigma_+^{(2)}\right) e^{i[\eta(ae^{-i\nu t} + a^\dagger e^{i\nu t}) - (\nu+\Delta)t]} + \text{h.c.} \tag{A.2}$$

where Ω is the single-ion Rabi frequency, Δ is the detuning from the motional sidebands used for the MS gate, ν is the motional frequency of the ion chain, η is the Lamb-Dicke parameter, $\sigma_+ = \frac{1}{2}(\sigma_x + i\sigma_y)$ is a linear combination of the single-qubit Pauli matrices σ_x and σ_y, the superscripts indicate the ion number, and "h.c." indicates the Hermitian conjugate of the preceding terms.

However, the experimental model is more complicated, and we must add a few parameters to account for this. First, the two ions may experience different amplitudes of the driving laser field, meaning that their Rabi frequencies may differ. We therefore separate the Ω

parameter into ion-specific parameters $\Omega^{(1)}$ and $\Omega^{(2)}$. This produces the following modified Hamiltonian:

$$H = \hbar \left(\Omega^{(1)} \sigma_+^{(1)} + \Omega^{(2)} \sigma_+^{(2)} \right) e^{i[\eta(ae^{-i\nu t} + a^\dagger e^{i\nu t}) - (\nu + \Delta)t]} + \text{h.c.}. \tag{A.3}$$

Second, the energy levels of the red and blue sidebands used for the interaction may be slightly shifted due to an AC Stark effect [6]. When considering its effect on the red and blue sidebands, this shift may have both symmetric (i.e., with the same sign) and antisymmetric (i.e., with opposite signs) portions. We therefore wish to incorporate two additional parameters, $\delta_{\uparrow\uparrow}$ (symmetric) and $\delta_{\uparrow\downarrow}$ (antisymmetric), into our model. However, we note that the detuning Δ is also implemented in an antisymmetric fashion (i.e., with opposite signs) on the two frequencies used for the MS gate. Therefore, a model that attempts to learn Δ will include the effect of $\delta_{\uparrow\downarrow}$ in the learned value. We thus ignore the antisymmetric term $\delta_{\uparrow\downarrow}$ and consider only the symmetric term $\delta_{\uparrow\uparrow}$, which we will denote as δ for brevity.

We note that in fact the detuning Δ and AC Stark shift δ may differ from ion to ion, for example, due to a magnetic field gradient along the axis of the ion chain. For simplicity, we ignore this effect for the sake of this demonstration.

Instead of deriving the behavior of the AC Stark shift analytically, we choose to rely on the IonSim.jl library, which allows for the specification of AC Stark effects in its simulations. To do this, we configure an IonSim.jl simulation of a two-ion MS interaction as an objective function with the following model parameters:

1. $\Omega^{(1)}$: the Rabi frequency of ion 1
2. $\Omega^{(2)}$: the Rabi frequency of ion 2
3. Δ: the detuning from the motional sideband
4. δ: the symmetric portion of the AC Stark shift

The blue and red laser tones used for the MS interaction are then detuned from the carrier frequency by $(\nu + \Delta + \delta)$ and $(-\nu - \Delta + \delta)$, respectively. Here we assume that the motional frequency ν is fixed. But note that if ν fluctuates slowly over time, our model may also incorporate this fluctuation into the learned value of the detuning Δ.

In a typical trapped-ion experiment, the Rabi frequencies $\Omega^{(1)}$ and $\Omega^{(2)}$ are proportional to the amplitude of the driving laser field as seen by each ion, the detuning Δ is the difference between the laser frequency and the motional sideband of the qubit, and the parameter δ is the symmetric portion of the AC Stark experienced by both ions. By accurately measuring the empirical values of $(\Omega^{(1)}, \Omega^{(2)}, \Delta, \delta)$, then, we can infer the experimental control settings which would allow us to run an optimal MS gate.

To simulate this procedure, we follow the process depicted in Figure A.9. We start with the experimentally-typical scenario where we have values of $(\Omega^{(1)}, \Omega^{(2)}, \Delta, \delta)$ which produce a recognizable but badly miscalibrated MS interaction. We then simulate an experimental time scan by using IonSim.jl to take N total measurements over a range of times (in this

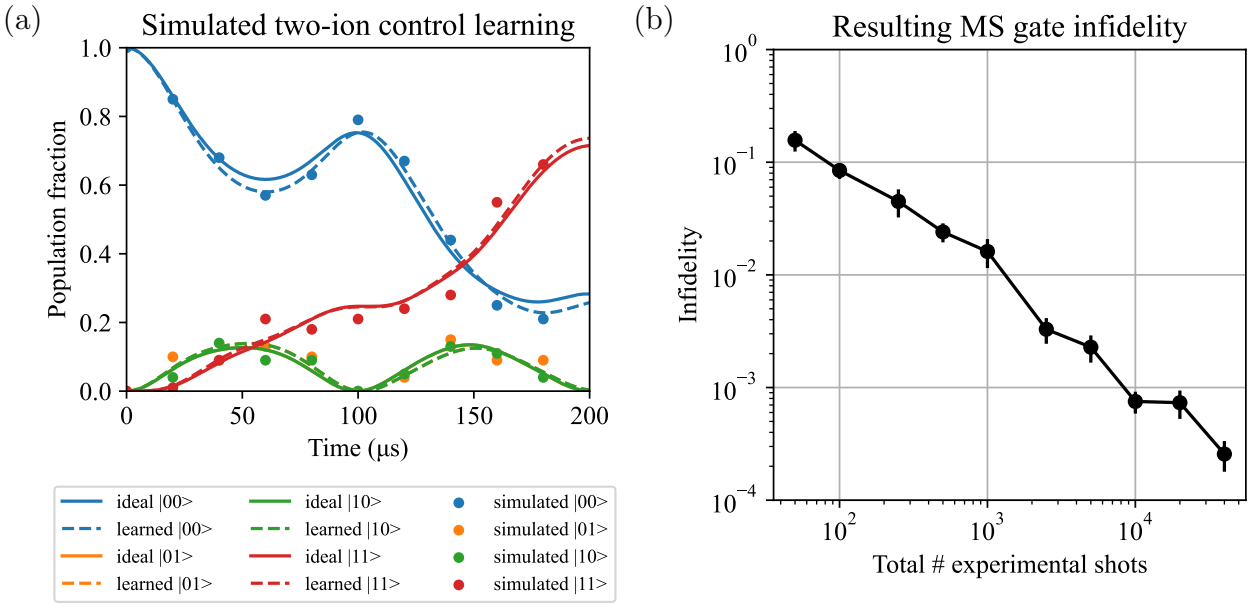


Figure A.10: Simulated two-ion control learning using a Mølmer-Sørensen (MS) interaction. (a) Illustration of model learned from $N = 1000$ total shots obtained from a time scan with 50 shots per point. Data points are obtained by simulating an ideal model of a miscalibrated MS gate interaction. (b) Simulated infidelity of a fully-entangling MS gate using models learned from varying numbers of experimental shots.

example, we use 20 points over the range 0 μs to 200 μs). Then, without knowledge of the initial parameter values, we take the resulting data and use maximum likelihood estimation, using IonSim.jl to simulate our objective function, to find the parameters $(\Omega^{*(1)}, \Omega^{*(2)}, \Delta^*, \delta^*)$ which best explain the data. Finally, we consider the difference between the learned values $(\Omega^{*(1)}, \Omega^{*(2)}, \Delta^*, \delta^*)$ and the ideal values $(\Omega^{(1)}, \Omega^{(2)}, \Delta, \delta)$ to be our experimental error, and we use IonSim.jl to simulate and measure the experimental infidelity of an MS gate under this error. This represents the error that would be incurred if we had adjusted our controls assuming that the learned values $(\Omega^{*(1)}, \Omega^{*(2)}, \Delta^*, \delta^*)$ were correct.

Figure A.10 shows the results of using this approach to calibrate an MS gate, i.e., the $XX(\pi/2)$ gate from Equation 2.24. We observe in Figure A.10(b) that using a single time scan consisting of 10,000 total shots, which would take only a few minutes to execute experimentally, we reach a simulated two-qubit gate fidelity exceeding 99.9%. We note that this result is comparable to but less efficient than a Bayesian calibration technique [50], which has achieved a reported experimental two-qubit gate fidelity exceeding 99.86% using less than 2,000 total shots.

Discussion

We have demonstrated here the simulated calibration of experimental control parameters using a generic maximum likelihood estimation technique, which is a relatively simple approach that can be applied to a wide range of problems. We have demonstrated only the simplest version of this technique, in which an experimental scan is performed using a fixed, predetermined set of control parameters. Likely more efficient would be an iterative approach, in which smaller ranges of parameter space are explored during each iteration, and a classical optimization technique such as SPSA or SBO (see Chapter 6) is used to determine the parameters for each iteration and ultimately converge on optimal parameter settings.

A natural extension of this work would be to extend it to more advanced Hamiltonian learning techniques which have been developed, such as Bayesian parameter estimation [50, 58]. We also note that our approach of using IonSim.jl to simulate the objective function is conceptually similar to the idea of using a trusted quantum device to verify an untrusted quantum device [158], and so incorporating the use of sequential Monte Carlo techniques [58, 73] may also be worthwhile.

A.5 Summary

In this appendix, we have provided an overview of several techniques for simulating trapped-ion experiments. We have discussed the benefits of simulating experimental pulse sequences, and we have demonstrated this by running experimental software from the Häffner lab at UC Berkeley against a simulation of the laser-ion interaction rather than against the physical apparatus. We have also described a technique for generating artificial but realistic EMCCD images of ions, which may be useful for developing and testing readout algorithms that require a large quantity of reliably-labeled training data. Finally, we have demonstrated a simulation of a simple experimental calibration technique, using control learning via maximum-likelihood estimation to calibrate single-qubit and two-qubit gates with a reasonable number of simulated experiments; these results also serve as an experimental guide indicating the quantity of data that must be taken to achieve a desired gate fidelity. Taken together, these techniques begin to form a software toolbox that allow experimental procedures to be developed and tested in isolation from the experimental apparatus, which allows both faster iteration for the developer and less demand for scarce time on the physical experiment.