Spring 5-15-2020

Applying Bayesian Techniques to the Optimization of Parameterized Quantum Circuits

Arthur Rattew
Washington University in St. Louis

Washington University in St. Louis
School of Engineering and Applied Science
Department of Computer Science and Engineering

Thesis Examination Committee:
Roman Garnett, Chair
Ron Cytron
Marco Pistoia

Applying Bayesian Techniques to the Optimization of Parameterized Quantum Circuits

by

Arthur G. Rattew

A thesis presented in partial fulfillment of the
requirements for the degree of
Master of Science

May 2020

Acknowledgements

From the beginning of my academic career, I have been fortunate in that I have been able to work closely with professors and professional researchers on various projects and independent studies. Each experience has played an important role in shaping me as a researcher, and all of my mentors and advisors have been generous and inspiring. A thesis represents the culmination of years of study and mentorship, and so its completion embodies the sum of the efforts of one's mentors. As such, in chronological order, I would like to thank: Professor Angelina Lee, Professor Ron Cytron, Dr. Marco Pistoia, and Professor Roman Garnett. Professor Angelina Lee showed me the value of applying the scientific method to the analysis of algorithms and software. Professor Ron Cytron encouraged my love for learning, and helped me identify the types of questions that must be asked in the pursuit of fundamental knowledge. Dr. Marco Pistoia showed me the value in working as a team, and in rapidly prototyping and testing ideas. Professor Roman Garnett has shared with me his passion for Bayesian methods, and has helped me further appreciate the importance of embedding one's knowledge and beliefs in the construction of models. Finally, I would like to thank my family for their continued support.

Contents

Abstract

In this work, we explore the challenges faced when creating quantum algorithms for near-term quantum computers. We examine the characteristics of problems that are amenable to such advantage, and the limitations of existing approaches. Additionally, we explore the importance of the classical optimizer in the Variational Quantum Eigensolver (VQE), and propose a Bayesian method to optimally configure a single parameter in a given quantum circuit. Experimental testing confirms that our method is significantly more tolerant to noise than the existing analytical approach and its variants ($p < 0.008$).

Chapter 1

Introduction

The potential promise of the theoretically ideal, universal fault-tolerant quantum computers of the distant future is alluring. From exponential speedup in integer factorization, to polynomial speedup in unstructured search, to the simulation of sophisticated natural processes, the potential applications are profound [20, 6]. Yet, the construction of such a device has remained in the realm of the theorist for decades.

Prompted by significant investment from industry, considerable progress has recently been made in the construction of real quantum computers. We are now spectators to the experimental realization of the first generation of potentially useful quantum hardware. IBM and Google have each unveiled 53 qubit quantum devices, with the later having claimed to perform a computation (random circuit sampling) intractable to all classical computers in 2019 [2]. While IBM subsequently demonstrated that a classical supercomputer could exploit its memory hierarchy to perform this task in 2.5 days, the quantum computer was still significantly faster, requiring only 200 seconds [15, 14]. In essence, the algorithm produces a random quantum circuit near the limits of the capabilities of the quantum computer, and samples from the resultant classically intractable probability distribution. Although the benchmark was contrived to specifically exploit the capabilities of Google's quantum device, this process of sampling from classically intractable probability distributions is fundamental to construction of algorithms specifically designed for extant quantum hardware. Yet, as will be discussed shortly, such hardware is not without its faults. In this thesis, we consider the quantum devices of near-future, examine the types of problems they are likely able to offer quantum advantage for, explore their limitations, and present methods to circumvent these limitations. Given current rates of advancement, we can reasonably expect such near-term quantum algorithms to perform tasks of significant commercial and scientific interest on the quantum computers of the 2020s and beyond.

1.1 The Noisy Intermediate Scale Quantum Era

First coined by John Preskill in 2018, the *Noisy Intermediate-Scale Quantum* (NISQ) era characterizes the quantum hardware of the foreseeable future [17]. There currently appear to be two leading approaches in the construction of NISQ devices: superconducting and ion-trap based architectures. Fundamental to any architecture are the implementations of qubits and gates. Superconducting approaches utilize wires cooled to near absolute zero temperatures to implement their qubits. They use precise microwave pulses to implement the gates enacting their transformations on qubits [8]. Ion-trap architectures confine ions in certain regions of a magnetic field to implement qubits, and they use optics to deliver controlled laser pulses to apply gates [8]. While both approaches have their merits, throughout this document we will consider an abstraction equally applicable to all NISQ devices. To design near-term quantum algorithms, it is beneficial to consider the three limitations which define NISQ devices and differentiate them from the theoretically ideal universal fault-tolerant quantum computers of the distant future.

Limitation 1: Qubit Count. A path for scaling NISQ devices to 1000 qubits seems clear, but the challenges in controlling the current leading approaches beyond this boundary are significant [19].

Limitation 2: Coherence Times. Ideal quantum computers operate by enacting a specified sequence of gates on an *isolated* quantum system. Isolating the quantum systems used to currently implement qubits is not an easy task. For example, to isolate the wires used in superconducting approaches from their environment, they must be cooled to nearly a thousandth of a degree above absolute zero. Additionally, the states of these isolated systems must somehow be altered externally if gates are to be applied. Thus, these devices must balance two opposing goals: isolating the systems while also altering their states in some predefined way. We are currently only able to balance these two goals for a limited amount of time. Thus, only a certain number of operations can be applied before the systems lose their quantum properties, rendering any calculations performed invalid. Thus, NISQ devices can only execute shallow circuits.

Limitation 3: Gate Fidelity. Unlike the theoretically ideal quantum devices of the future, NISQ devices are not error corrected. Thus, gate executions have an intrinsically limited fidelity resulting in noisy circuit evaluations. For example, current publicly accessible state-of-the-art devices at IBM have single-qubit gate error rates on the order of 10^{-4}, and two-qubit gate error rates on the order of 10^{-3}.

All together, these limitations prevent the execution of most commonly known quantum algorithms such as Shor's famous encryption-breaking algorithm, or Grover's widely applicable amplitude amplification

algorithm [20, 6]. Thus, if we wish to exploit the potential power of near-term quantum devices, we must investigate new paradigms for the creation of quantum algorithms.

1.2 Hybrid Quantum-Classical Algorithms

One such paradigm was introduced with the Variational Quantum Eigensolver (VQE) by Peruzzo *et al.* in 2014 [16]. By exchanging circuit depth and coherence requirements with an increase in the number of required circuit evaluations, VQE manages to circumvent the limitations of NISQ devices, while ideally capturing their potential. By utilizing parameterized quantum circuits (PQCs), a controlling classical computer alters the parameters to navigate through a region of the device's Hilbert space so as to minimize a given objective function. A formal statement of the problem solved by VQE is offered in a subsequent section of this document. VQE and VQE-like algorithms have already been demonstrated numerous times on small problem instances on real quantum devices [7, 18].

1.3 Optimization of PQCs

An integral component to many near-term quantum algorithms is a classical optimization routine which manipulates the parameters of a given PQC [9, 16, 10, 12, 11]. However, for the reasons already discussed, these optimization routines need to be tolerant to the potentially significant levels of noise present when evaluating circuits of non-trivial depths on extant quantum hardware. Many existing works simply use off-the-shelf optimizers to tune the parameters of their circuits, such as SPSA, COBYLA or BFGS [9, 7]. Given the significant amount of domain specific information available from the structure of the quantum circuits being optimized, and from the unique characteristics of the quantum devices upon which they are optimized, McClean *et al.* identified the creation of quantum circuit-specific optimizers as a potential significant source of improvement for quantum/classical hybrid algorithms.

Chapter 2

Background

2.1 The Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) utilizes the variational method of quantum mechanics to bound the ground state energy of a given Hamiltonian H. It states,

$$\langle\psi(\theta)|H|\psi(\theta)\rangle \geq E_0, \tag{2.1}$$

where E_0 is the lowest-valued eigenvalue of H. Proving the variational method is straightforward. Allow $H|E_i\rangle = E_i|E_i\rangle$ to represent the eigenvalues and eigenstates of H such that $E_i \leq E_{i+1}$. Then,

$$\begin{aligned}
\langle\psi(\theta)|H|\psi(\theta)\rangle &= \sum_{i,j}\langle\psi(\theta)|E_i\rangle\langle E_i|H|E_j\rangle\langle E_j|\psi(\theta)\rangle \\
&= \sum_{i,j} E_i\langle\psi(\theta)|E_i\rangle\langle E_i|E_j\rangle\langle E_j|\psi(\theta)\rangle \\
&= \sum_i E_i|\langle\psi(\theta)|E_i\rangle|^2 \\
&\geq \sum_i E_0|\langle\psi(\theta)|E_i\rangle|^2 \\
&= E_0\sum_i|\langle\psi(\theta)|E_i\rangle|^2 = E_0. \qquad \square
\end{aligned}$$

The last equality follows from the normalization requirement of the state $|\psi(\theta)\rangle$. To apply this method, VQE prepares a parameterized quantum state (called an *ansatz*) by starting with an initial state $|\psi_i\rangle$ and applying a parameterized unitary operator $U(\theta)$, such that $|\psi(\theta)\rangle \equiv U(\theta)|\psi_i\rangle$. It then minimizes the parameter $\theta \in \mathbb{R}^d$

(with d polynomial in the dimension of the total Hilbert space) with the hope that

$$\min_{\theta} \left(\langle \psi(\theta) | H | \psi(\theta) \rangle \right) \approx E_0.$$

In certain circumstances, ansatz may be constructed such that $|\psi(\theta)\rangle$ is guaranteed to contain the ground state. In such circumstances, it is widely expected that VQE will offer an exponential advantage over corresponding classical algorithms when executed on NISQ devices [7, 9].

2.1.1 Quantum Advantage for VQE

Allow H to encode some optimization problem of interest. In order for such a problem to be amenable to quantum advantage, we expect it to have the following characteristics:

1. It must be possible to exploit the underlying problem structure to construct an ansatz that contains a state whose expectation value closely approximates the Hamiltonian's ground state. Alternatively, it must be possible to create an ansatz guaranteed to contain the ground state such that it may be either constructed in time offering a polynomial speedup (or polynomial space saving) over the bottleneck of a corresponding classical algorithm.

2. It must be possible to decompose the Hamiltonian encoding the problem into a sum of tensor products of elementary quantum gates (such as the Pauli gates) so that the expectation value of the operator may be computed. The efficiency of this procedure depends on whether exponential or polynomial quantum advantage is desired.

3. The Hamiltonian must not be diagonal in the standard basis, as this would imply that the Hamiltonian's eigenvectors may be efficiently represented classically.

It is expected that the source of quantum advantage in VQE comes not from a speedup in combinatorial search (as is the case with many Grover-based algorithms), but rather from searching through a tractable volume of states representing probability distributions intractable to classical computers. That is, we anticipate quantum advantage to come from the classical intractability of the states being explored rather than from searching through a larger volume of states more efficiently.

2.1.2 Hamiltonian Decomposition

As already mentioned, VQE computes the ground state energy of a Hamiltonian by iteratively minimizing the expectation value of a parameterized quantum state. To do so, it is essential that the the Hamiltonian

may be decomposed into a sum of Pauli gates, which by linearity of expectation, may be each be computed separately. In some cases, such as for the Hamiltonians' of various molecules, it is possible to perform this decomposition utilizing a polynomial number of Pauli terms in the sum [16]. It is also possible to do so in general (only requiring exponential time and space in the number of qubits), utilizing a decomposition procedure now outlined.

General Decomposition

The method presented below expands on a technique presented by Payne *et al.* in 2019 to decompose an arbitrary Hamiltonian into a sequence of tensor products of elementary quantum gates [13]. As this method runs in $\Theta(N^2 \log N)$ time, any algorithm requiring it is necessarily an exponential time algorithm. However, as the bottleneck for classical determinstic eigenvalue-finding algorithms is matrix multiplication, such algorithms run with a theoretical complexity of $O(N^{2.373})$ (as presented by Virginia Williams in 2014, surpassing the Coppersmith-Winograd algorithm) and with a practical lower-bound of $O(N^{\log_2 7}) \approx O(N^{2.81})$ when using Strassen's algorithm [22].

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}(I + Y)$$
$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \frac{1}{2}(X + ZX)$$
$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \frac{1}{2}(X + XZ)$$
$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{2}(I + XZX).$$

Then, given a Hamiltonian H, we may recursively divide the Hamiltonian as follows,

$$H^{(i)} = \left[\begin{array}{c|c} H_{11}^{(i-1)} & H_{12}^{(i-1)} \\ \hline H_{21}^{(i-1)} & H_{22}^{(i-1)} \end{array}\right] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes H_{11}^{(i-1)} + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes H_{12}^{(i-1)} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes H_{21}^{(i-1)} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes H_{22}^{(i-1)}$$
$$= \frac{1}{2}(I + Z) \otimes H_{11}^{(i-1)} + \frac{1}{2}(X + ZX) \otimes H_{12}^{(i-1)} + \frac{1}{2}(X + XZ) \otimes H_{21}^{(i-1)} + \frac{1}{2}(I + XZX) \otimes H_{22}^{(i-1)}.$$

Where $H^{(i)}$ represents the Hamiltonian on the i^{th} recursive step. This process is repeated until all

remaining matrices are 2×2 (and can then be implemented by an appropriately parameterized U3 gate or weighted sum of Pauli gates). At each level of this recurrence, the four $N \times N$ sub-matrices are added for a total of $\Theta(N^2)$ work. Thus, this decomposition has the recurrence $T(N) = 4T(\frac{N}{2}) + O(N^2)$, and has running time $\Theta(N^2 \log N)$ (from case three of the master method).

Diagonal Hamiltonian Decomposition

It is also worth examining the complexity of decomposing a diagonal Hamiltonian into a sum of 2×2 operators, as such may prove illuminating in determining the types of block-diagonal operators which may be decomposed most efficiently. Assume you are given an H such that H is diagonal in the standard basis. Then,

$$\begin{aligned} H^{(i)} &= \left[\begin{array}{c|c} H_{11}^{(i-1)} & 0 \\ \hline 0 & H_{22}^{(i-1)} \end{array}\right] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes H_{11}^{(i-1)} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes H_{22}^{(i-1)} \\ &= \frac{1}{2}(I + Z) \otimes H_{11}^{(i-1)} + \frac{1}{2}(I + XZX) \otimes H_{22}^{(i-1)}. \end{aligned}$$

Given the sparse nature of this matrix, the recurrence for this decomposition is given by $T(N) = 2T(\frac{N}{2}) + O(N)$, and thus from case three of the master method, runs in $\Theta(N \log N)$ time.

2.1.3 Parameterized Quantum Circuits as Ansatz

A parameterized quantum circuit consists of a sequence of gates acting on qubits, such that at least some of the gates in the circuit contain parameters. For instance, given a single qubit in the state $|0\rangle$, we may apply an $R_y(\theta)$ gate as follows (with $\hbar = 1$),

$$\begin{aligned} R_y(\theta)|0\rangle &= e^{-i\frac{Y}{2}\theta}|0\rangle \\ &= \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \cos(\frac{\theta}{2})|0\rangle + \sin(\frac{\theta}{2})|1\rangle, \end{aligned}$$

where Y is the Pauli-Y gate, $Y = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$.

2.1.4 Single Parameter Optimization

In 2019, Ostaszewski *et al.* proposed an algorithm specifically designed to optimize the parameters of a quantum circuit called Rotosolve [12]. Rotosolve optimizes the parameters of a PQC by iteratively optimally setting one parameter at a time (while holding all other parameters constant) and repeating until some stopping condition has been met. While this approach is not without its limitations, some of which are discussed in the next chapter, the subroutine used to optimally configure one parameter at a time is nevertheless an interesting analytical result, which may be conducive to generalization. We will now discuss this analytical subroutine.

This subroutine, henceforth called Rotosolve Subroutine (RS), assumes an objective function is encoded in a Hermitian operator M whose expectation value is to be optimized with respect to a quantum circuit corresponding to a unitary, $U(\theta)$, generated by a Hermitian and unitary operator H with parameter θ. That is, for some fixed, easy to produce, starting state $|\psi\rangle$ they wish to compute (for $\hbar = 1$):

$$\begin{aligned}\theta^* &= \arg\min_{a\in A}\left(\langle\psi_i|U(\theta)^\dagger M U(\theta)|\psi_i\rangle\right)\\ &= \arg\min_{a\in A}\left(\langle\psi_i|e^{\frac{i}{2}\theta H} M e^{-\frac{i}{2}\theta H}|\psi_i\rangle\right).\end{aligned}$$

We will briefly outline their derivation for the closed-form analytical expression for this minimum. We may explicitly show the sinusoidal nature of $U(\theta)$ with a Taylor expansion, obtaining,

$$U(\theta) = \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})H.$$

Allow the notation $\langle M\rangle_\theta$ to represent the evaluation of the objective function with the parameter set to θ. Allowing $\rho = |\psi_i\rangle\langle\psi_i|$, we may equivalently write the expectation value of the circuit as,

$$\begin{aligned}\langle M\rangle_\theta &= \mathrm{tr}\left(MU(\theta)\rho U(\theta)^\dagger\right)\\ &= \cos^2(\frac{\theta}{2})\mathrm{tr}(M\rho) + i\sin(\frac{\theta}{2})\cos(\frac{\theta}{2})\mathrm{tr}(M[\rho,H]) + \sin^2(\frac{\theta}{2})\mathrm{tr}(MH\rho H).\end{aligned}$$

Applying trigonometric identities and further algebraic manipulation, they finally obtain the expression,

$$\begin{aligned}
\langle M\rangle_\theta &= A\sin(\theta + B) + C, \\
A &= \frac{1}{2}\sqrt{(\langle M\rangle_0 - \langle M\rangle_\pi)^2 + (\langle M\rangle_{\frac{\pi}{2}} - \langle M\rangle_{-\frac{\pi}{2}})^2}, \\
B &= \arctan2\left(\langle M\rangle_0 - \langle M\rangle_\pi, \langle M\rangle_{\frac{\pi}{2}} - \langle M\rangle_{-\frac{pi}{2}}\right), \\
C &= \frac{1}{2}(\langle M\rangle_0 + \langle M\rangle_\pi).
\end{aligned}$$

Finally, they show that θ^* may be evaluated exactly with three samples of the objective function,

$$\theta^* = -\frac{\pi}{2} - \arctan2\left(2\langle M\rangle_\phi - \langle M\rangle_{\phi+\frac{\pi}{2}} - \langle M\rangle_{\phi-\frac{\pi}{2}}, \langle M\rangle_{\phi+\frac{\pi}{2}} - \langle M\rangle_{\phi-\frac{\pi}{2}}\right),$$

where ϕ is an arbitrary real parameter.

2.2 Bayesian Optimization

Given a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, and a domain $X \subseteq \mathcal{X}$, we are asked to find,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in X} f(\boldsymbol{x})$$

such that $f(\boldsymbol{x}^*)$ is a global optimum. Bayesian optimization is a general technique for finding such an $\boldsymbol{x}^*$. Maintaining a surrogate model for the underlying function (in the form of a Gaussian process prior), the algorithm iteratively selects the next values at which to evaluate the objective. Evaluation locations are selected in accordance with an acquisition function that estimates the expected utility of gaining additional information about the objective at various locations in its domain. In so doing, the algorithm gathers evidence and updates its posterior belief regarding the underlying function. The updated surrogate model is then used to make progressively improved selections of evaluation points, yielding progressively improved estimates for the optimum of the objective. Bayesian optimization algorithms tend to perform most favorably when the dimension of the parameter space is less than 20, and tend to be quite resilient to noise in the underlying objective function [3]. Given an acquisition function $\alpha(\boldsymbol{x}; \mathcal{D})$, Algorithm 1 presents the pseudo-code for Bayesian optimization.

Algorithm 1 Bayesian Optimization

procedure BAYESIANOPTIMIZATION(Objective Function f)
  Create an initial data set $\mathcal{D} = (X, Y)$.
  while budget remains **do**
    Condition the Gaussian process on the data.
    Optimize the kernel's hyperparameters (e.g. by maximizing the log marginal likelihood).
    Compute $\boldsymbol{x}_{new} = \max_{\boldsymbol{x}} \alpha(\boldsymbol{x}; \mathcal{D})$.
    Query the objective function to obtain, $f(\boldsymbol{x}_{new})$.
    Update the data-set with $\mathcal{D} = \mathcal{D} \cup (\boldsymbol{x}_{new}, f(\boldsymbol{x}_{new}))$.
  return The maximum value observed so far (or the maximum of the conditioned posterior mean).

2.2.1 Gaussian Processes

Gaussian processes are infinite collections of random variables, such that any finite subset of those variables follows a multivariate normal distribution. Consequently, a Gaussian process represents a distribution over functions. As stated by Williams *et al.*, a Gaussian process is fully specified by a mean function and a covariance function (a kernel) [21]. For an $n \times 1$ input vector, $\boldsymbol{x}$, they give the following formal definition,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}')),$$

where,

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$
$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))].$$

Kernels

The kernel is perhaps the most important part of a Gaussian process. By specifying the "similarity" between arbitrary pairs of points in the input domain, the kernel enables the Gaussian process to fit an underlying function by restricting the probability distribution over possible functions to only those which may explain a given data-set. At a more abstract level, a kernel defines an inner product in some feature transformed space of two input vectors. That is, for some feature transform $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}) | \phi(\boldsymbol{x}') \rangle.$$

Thus, if a kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ is positive semi-definite, it may be considered a valid kernel. When the kernel is computed on all pairs of two sets of points, we form a Gram matrix. That is, given an input $d \times n$ input matrix X and and input $d \times m$ matrix Z such that $\boldsymbol{x}_i$ represents the i^{th} column of X, and $\boldsymbol{z}_j$ represents the

j^{th} column of Z, we define an $n \times m$ Gram matrix as,

$$K(X,Z) = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{z}_1) & \dots & K(\boldsymbol{x}_1, \boldsymbol{z}_m) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{z}_1) & \dots & K(\boldsymbol{x}_n, \boldsymbol{z}_m) \end{pmatrix}.$$

As a computational optimization, kernels which only depend on the difference of their inputs (i.e. are of the form $K(\boldsymbol{x}, \boldsymbol{x}') = f(\sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}')})$ may be written as,

$$K(X,Z) = f(D)$$

where f is applied element-wise to the elements of the matrix D, which is defined as,

$$D^2 = S + R - 2X^T Z,$$

and $S_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_i$ and $R_{ij} = \boldsymbol{z}_j^T \boldsymbol{z}_j$.

2.2.2 Acquisition Function

Acquisition functions serve to estimate expected utility of acquiring a given marginal data point $\boldsymbol{x}_{new}$. Thus, we will first discuss the notion of utility. Consider the simple reward utility function (in the absence of noise),

$$u(\mathcal{D}) = \max_{y \in Y} y$$

It is important to note that when the objective function is noisy, as it is when optimizing parameterized quantum circuits in practice, this utility function is inadequate. As the value of a data set comes from computing the maximum value observed, a data set may be considered unfairly good if noise results in a given observation having a greater value than it would otherwise have (and similarly may result in an unfairly poor judgement of a data set). Thus, a noise resistant utility function should avoid directly using the output values observed from the objective function when judging the quality of a data set. One possible way to do so is by conditioning a Gaussian process on the data set, and evaluating the maximum of the posterior mean (either over the entire domain, or at the observed locations). The *global simple reward* utility function is defined by considering the maximum of the posterior mean over the entire domain. Formally, allowing $\mu_{\mathcal{D}}(\boldsymbol{x})$ to represent the posterior mean of the Gaussian process conditioned on the data $\mathcal{D}$ and allowing X_*

to represent the predictive domain, we may write this utility function as,

$$u(\mathcal{D}) = \max_{\boldsymbol{x} \in X_*} \mu_{\mathcal{D}}(\boldsymbol{x}).$$

Utilizing this utility function, and performing a one-step-lookahead approximation yields the *knowledge gradient* acquisition function [4]. The knowledge gradient acquisition function computes the expected utility of adding a given $\boldsymbol{x}$ to the data set by creating a new data set $\mathcal{D}' = \mathcal{D} \cup (\boldsymbol{x}, y)$ for each possible value of y, and computing the weighted sum of the utility (i.e. the maximum of the posterior mean conditioned on $\mathcal{D}'$) weighted by the probability of y obtaining that value (obtained from the normal distribution marginalized from the Gaussian process). Formally, we may define the knowledge gradient acquisition function as,

$$\alpha_{KG}(\boldsymbol{x}; D) = \int u(\mathcal{D} \cup (\boldsymbol{x}, y)) p(y \mid \boldsymbol{x}, D) dy - \max_{\boldsymbol{x}' \in X_*} \mu_{\mathcal{D}}(\boldsymbol{x}'),$$

where the subtracted term $\max_{\boldsymbol{x}' \in X_*} \mu_{\mathcal{D}}(\boldsymbol{x}')$ term represents the utility of the data set without the addition of the new point. Thus, we obtain,

$$\alpha_{KG}(\boldsymbol{x}; D) = \int \left(\max_{\boldsymbol{x}' \in X_*} \mu_{\mathcal{D}'}(\boldsymbol{x}') \right) p(y \mid \boldsymbol{x}, D) dy - \max_{\boldsymbol{x}' \in X_*} \mu_{\mathcal{D}}(\boldsymbol{x}'). \tag{2.2}$$

In general, computing this integral can be rather computationally expensive, especially given that the acquisition function is computed for each $\boldsymbol{x}$ in the predictive domain. As such, an important optimization utilized in the custom implementation of Bayesian optimization created for this paper, was the Gauss-Hermite quadrature approximation. In general, Gauss-Hermite quadrature offers an approximation for integrals of the form,

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^{n} w_i f(x_i),$$

where n is the order of the approximation, and w_i are real-valued weights. We found that using an order 7 approximation yielded results of precision significantly exceeding our requirements.

2.2.3 Optimization of Entire PQCs with Bayesian Methods

Some work has already been performed exploring the use of Bayesian techniques in the optimization of PQCs. For instance, as presented by Moseley *et al.* in 2018, Bayesian optimization was employed to optimize the parameters of Ansatz to minimize the expectation values of the molecules H_2 and LiH. When compared against the noise-resistant optimizer SPSA, they demonstrate a reduction in the required number of cost

function evaluations by over an order of magnitude when computing the ground state of both molecules in a noiseless setting. In the estimation of the ground state of H_2 in presence of simulated hardware noise, they demonstrate that their Bayesian algorithm converges to the optimal parameter values with a higher probability than SPSA. Their implementation is tested utilizing three acquisition functions: expected improvement (EI), maximum probability of improvement (MPI), and lower confidence bound (LCB). The utility function used is not specified. In the experiments with simulated noise, they only use the EI acquisition function. If the simple reward utility function was used, it is likely that their results could be further improved by utilizing a more noise tolerant utility function, such as the one discussed in the preceding section. Their algorithm uses the Matérn-$\frac{5}{2}$ kernel:

$$C_{\frac{5}{2}}(d) = \frac{1}{2^{\frac{3}{2}}\Gamma(\frac{5}{2})}\left(\frac{2\sqrt{\frac{5}{2}}d}{\theta}\right)^{\frac{5}{2}} H_{\frac{5}{2}}\left(\frac{2\sqrt{\frac{5}{2}}d}{\theta}\right),$$

where d is the distance between the inputs such that $K(\boldsymbol{x}, \boldsymbol{z}) = C_\nu(||\boldsymbol{x} - \boldsymbol{z}||)$, Γ is the gamma function, $H_{\frac{5}{2}}$ is the modified Bessel function of the second kind of order $\frac{5}{2}$ [5].

Chapter 3

Details of the Work

The ultimate goal of this document is to work towards enabling quantum advantage on NISQ devices. As the optimization subroutine in VQE is an important component towards achieving this goal, this work aims to lead to the creation of an optimizer customized for PQCs utilizing techniques from Bayesian optimization. The approach already discussed, presented by Moseley *et al.* in 2018, demonstrates the potential of using Bayesian optimization at a high-level to configure the parameters in a PQC. However, the advantage of such an approach could be compounded if a customized kernel could be developed that accurately modelled the impact of each parameter in a given quantum circuit to the overall energy evaluation of that circuit. While it may certainly prove to be hard to derive such a kernel in general, it is worth further exploration. As a first-step towards this goal, this document constructs and experimentally demonstrates a custom kernel for optimally configuring a single parameter in a quantum circuit at a time, and presents some ideas towards generalizing such a kernel to support optimal concurrent configuration of multiple parameters.

3.1 Single Parameter Optimizer

As already discussed, in 2019, Ostaszewski *et al.* presented a closed form expression for the expectation value of a Hermitian and unitary Hamiltonian depending on only one parameter:

$$f_\theta \equiv \langle H \rangle_\theta = A \sin(\theta + B) + C. \tag{3.1}$$

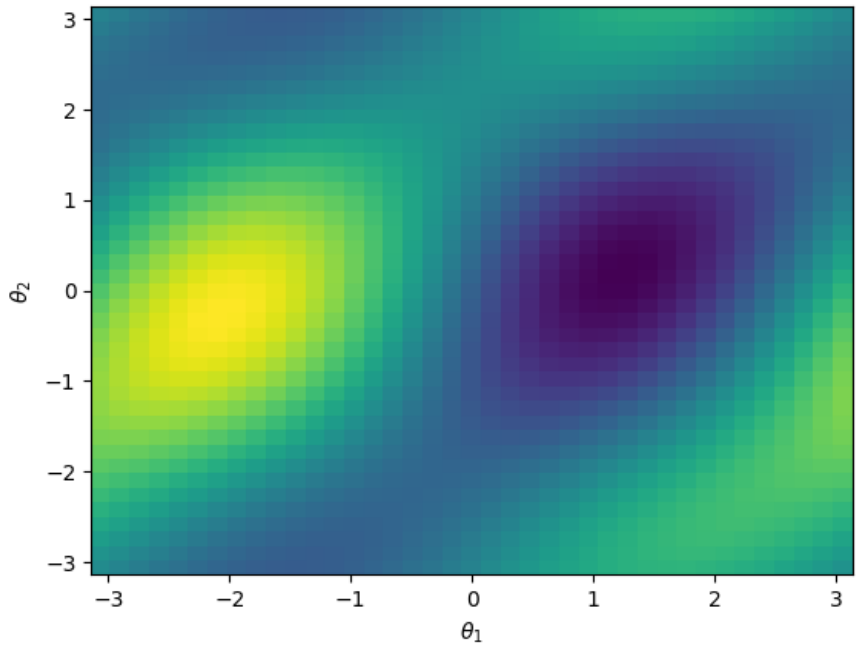

Figure 3.1: **Expectation Value of Two-Parameter Circuit**. Yellow colors represent a maximum, and blue/purple colors represent a minimum. Here a random 4×4 Hermitian matrix is generated as the objective function, and a quantum circuit consisting of two R_y gates (one on each of the two qubits) followed by a CX gate between the qubits is created. The result of sampling the objective function uniformly on the grid of all parameter values is shown.

They then derive the following closed form analytical solution to obtain the optimal value of that parameter,

$$\begin{aligned}\theta^* &= \arg\min_{\theta} \langle H \rangle_\theta \\ &= -\frac{\pi}{2} - \arctan2(2\langle H \rangle_{\theta=\phi} - \langle H \rangle_{\theta=\phi+\frac{\pi}{2}} - \langle H \rangle_{\theta=\phi-\frac{\pi}{2}}, \langle H \rangle_{\theta=\phi+\frac{\pi}{2}} - \langle H \rangle_{\theta=\phi-\frac{\pi}{2}}),\end{aligned}$$

for any $\phi \in \mathbb{R}$. However, such an analytical solution is vulnerable to noise in the evaluation of the objective function. In the absence of noise, both the Bayesian approach and this analytical solution obtain the optimal value utilizing three evaluations of the objective function. In the presence of noise, the benefits of our approach become evident.

3.1.1 Limitations of Rotosolve

While an interesting start to the design of an optimizer for parameterized quantum circuits, any optimizer which functions by optimally configuring on parameter of a quantum circuit at a time necessarily cannot converge to a global optimum in the solution space in general. For instance, consider the simple example shown in Figure 3.1. If your optimization scheme consisted of optimally configuring θ_1 and then optimally

configuring θ_2, and your initial starting point was $(1, -3)$, even though θ_1 would originally have the value at which the global optimum is, the algorithm would first set $\theta_1 \approx -1.5$, and then set $\theta_2 \approx 3$, converging to a sub-optimal space in the search space. Nevertheless, we believe that exploring this single-parameter optimization case may prove a fruitful step toward arriving at a more general, and potentially in some cases, globally optimal optimizer.

3.1.2 Deriving the Gaussian Process Prior for the Single Parameter Optimizer

As is standard in Bayesian optimization, we place a Gaussian process prior on the objective function,

$$p(f_\theta) = \mathcal{GP}(f_\theta; \mu(\theta), K(\theta, \theta')),$$

We will now derive the mean and covariance functions. To the best of our knowledge, the covariance function we are about to derive, $K(\theta, \theta') = \alpha^2 \cos(\theta - \theta') + s^2$, was first proposed by Adler *et al.* in 2009 [1]. First, the Rayleigh distribution with the parameter α is:

$$\mathcal{R}(\alpha) = \sqrt{X^2 + Y^2} \text{ such that } X, Y \sim \mathcal{N}(\theta, \alpha^2).$$

From trigonometric identities, we have:

$$f(x) = A\sin(x + B) + C = A\cos(x - \tau) + C = X\cos(x - \tau) + Y\sin(x - \tau) + b$$

where $A = \sqrt{X^2 + Y^2}$ and $\tau = \arctan(\frac{X}{Y})$. Thus, $A \sim \mathcal{R}(\alpha)$, and $\tau \sim \mathcal{U}(-\pi, \pi)$ (where the uniform distribution follows from X and Y being identically and independently distributed). Defining $f(x) = g(x) + b$, given that τ is uniform, and $g(x)$ is even:

$$\mathbb{E}[g(x) \mid x] = 0.$$

Now to obtain the covariance function:

$$\begin{aligned}\operatorname{cov}[g(x), g(x') \mid x, x'] &= \mathbb{E}[(X\cos x + Y\sin x)(X\cos x' + Y\sin x')] \\ &= \mathbb{E}[X^2\cos x\cos x' + XY\cos x\sin x' + XY\sin x\cos x' + Y^2\sin x\sin x'].\end{aligned}$$

As X and Y are iid, $\mathbb{E}[X^2] = \mathbb{E}[Y^2] = \alpha^2$ and $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y] = 0$, so:

$$\begin{aligned} &= \mathbb{E}[X^2]\mathbb{E}[\cos x \cos x' + \sin x \sin x'] \\ &= \alpha^2 \cos(x - x'). \end{aligned}$$

By linearity, the mean of $f(x)$ is thus given by,

$$\mathbb{E}[f(x) \mid x] = \mathbb{E}[b] = C.$$

By independence,

$$\begin{aligned} \operatorname{cov}[f(x), f(x') \mid x, x'] &= \operatorname{cov}[g(x), g(x') \mid x, x'] + \operatorname{cov}[b, b] && (3.2) \\ &= \alpha^2 \cos(x - x') + s^2. && (3.3) \end{aligned}$$

While this calculation is laborious, a similar approach based on a power series approximation may prove fruitful when generalizing to kernel functions over multidimensional circuit parameter vectors. Such an exploration is left as future work.

Chapter 4

Experiments and Discussion

4.1 Single Parameter Optimization

We assume that our Bayesian Single Parameter Optimization (BSPO) algorithm has direct knowledge of the level of additive Gaussian noise in the underlying system, σ^2. As mentioned by Mitari *et al.* in 2018, sampling the expectation value of an operator $\langle H \rangle$ may be emulated with additive Gaussian noise with the following standard deviation:

$$\sigma = \sqrt{\frac{2}{N_s}} \frac{(\langle H \rangle^2 - 1)}{4},$$

where N_s represents the number of shots taken, and $\langle H \rangle$ is the calculated expectation value of the operator [10]. As this value may be directly computed from training data, BSPO places a delta prior on σ^2 equal to the computed estimate. This work does not model the hardware noise caused by random errors during execution; modelling such errors may prove to be a worthwhile extension.

Throughout the remainder of this paper, the analytical calculation for the minimum of the sine wave, as presented by Ostaszewski *et al.*, will be referred to as Rotosolve Subroutine Standard (RSS). It is important to note that while RSS computes the optimal parameter value for the minimum of the sinusoidal objective utilizing only three queries, predicting the function value at that minimum requires an additional query. While it may be possible to only utilize three evaluations to also compute the value at the minimum, as they have presented it, such a calculation would require estimating the values of A, B and C, which require evaluations of the objective function at $\theta = 0, \pi, \frac{\pi}{2}$ and $\frac{-\pi}{2}$. Thus, in all tests where a value of the function at its minimum is desired, all algorithms use a minimum of four function evaluations, so that the optimal

value of the objective function may also be estimated along with the corresponding parameter. The ability to provide an estimate for the value of the minimum of the function, as well as the corresponding parameter, while only requiring three evaluations is an advantage of BSPO.

Originally, RSS sampled the objective function at three locations (as described in the previous section), computed the value of θ^*, and then evaluated the objective function at $f(\theta^*)$. However, as this process requires four samples, we found RSS performed best when we computed the objective function at $\theta = 0, \pi, \frac{\pi}{2}$ and $\frac{-\pi}{2}$, computed the values A, B and C, and then analytically determined the corresponding θ^* and $f(\theta^*)$ utilizing the calculated sine function. Moreover, when BSPO is allowed to collect a number of samples exceeding four, to maintain a fair comparison, a new variant of the Rotosolve subroutine that utilizes an equivalent number of samples is created. We call this algorithm Rotosolve Subroutine Input Averaging (RSIA). Here, each input is sampled x times, and the average of each input is used to compute the optimal value for θ. The optimal θ value is then evaluated in the objective function, for a total of $3x + 1$ function evaluations.

4.1.1 Noiseless Demonstration of Fitting GP

In this experiment, we demonstrate the process of fitting a Gaussian process to a sinusoidal objective function. As discussed at length, the kernel for a Gaussian process modelling an unknown sinusoidal function of the form,

$$\langle H \rangle_\theta = A \sin(\theta + B) + C,$$

is given by,

$$K(\theta_1, \theta_2) = \alpha^2 \cos(\theta_1 - \theta_2) + s^2.$$

Assume the objective function is noiseless, and that we have an $1 \times n$ matrix $X = \begin{pmatrix} \theta_1 \dots \theta_n \end{pmatrix}$ containing observed inputs, and and associated $1 \times n$ matrix $Y = \begin{pmatrix} y_1 \dots y_n \end{pmatrix}$ containing the corresponding outputs. Moreover, suppose we have an $1 \times m$ input matrix X_* containing points at which we would like to predict the value of the objective function. As already discussed, we may condition the Gaussian process on the observed training data to obtain the following predictive distribution,

$$Y_* \sim \mathcal{N}\left(K(X_*, X)K(X, X)^{-1}(Y - m\mathbb{I})^T, K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\right),$$

where m is the parameter corresponding to the mean of the objective. The process of conditioning the GP on a set of input and output pairs to make a prediction at unseen points is shown in Figure 4.1. This figure

clearly demonstrates that the algorithm accurately predicts the parameter and function value corresponding to the minimum of the given sinusoidal objective function. This is consistent with RSS, which also requires three evaluations of the noiseless objective function to optimally configure the parameter.

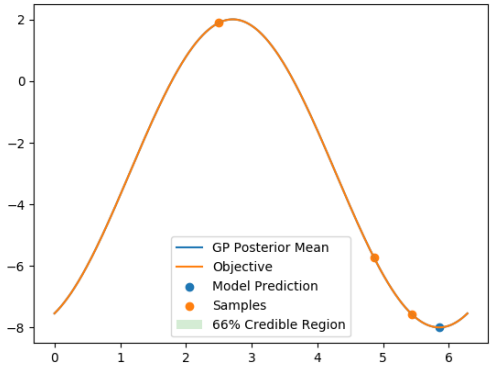

Figure 4.1: **Fitting a Gaussian Process to Training Data and Making Predictions**. In this experiment, the objective function has values $A = -5$, $B = 2$, and $C = -3$. X_* consists of 500 points uniformly spaced on the interval $[0, 2\pi]$. The three training points were selected uniformly at random on the interval $[0, 2\pi]$. The variance on the additive Gaussian noise in the objective function was set to 0. The curve corresponding to the posterior mean is not visible on the figure as the posterior mean is hidden below the objective function. The 66% credible region is not visible as the variance at the predicted points is essentially zero.

It is also interesting to observe the behavior of the algorithm when it does not have sufficient information to perfectly learn the objective function. As expected, the uncertainty collapses at regions where the samples are collected, and grows at parameter values further from the observed samples. Figure 4.2 shows such an experiment. This figure visually illustrates that the proposed kernel successfully captures the periodicity of the parameter, noting that uncertainty decreases as θ approaches 2π as a sample was collected at $\theta \approx 0.5$.

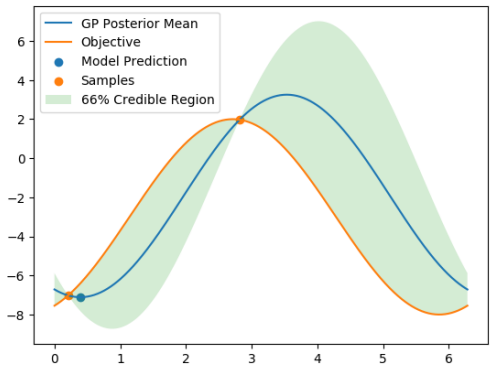

Figure 4.2: **Fitting a Gaussian Process to Insufficient Training Data and Making Predictions**. In this experiment, the objective function has values $A = -5$, $B = 2$, and $C = -3$. X_* consists of 500 points uniformly spaced on the interval $[0, 2\pi]$. The two training points were selected uniformly at random on the interval $[0, 2\pi]$. The variance on the additive Gaussian noise in the objective function was set to 0.

4.1.2 Selecting Subsequent Experiments with an Acquisition Function

A variety of acquisition functions were tested. First, expected improvement was used, and as expected, it tended to prefer exploiting its existing knowledge of the objective function and continued to acquire points near the minimum of the objective. When noise was present, as acquired samples would accumulate in the minimum, the density of these samples could result in the algorithm missing the overall shape of the objective function, resulting in sub-optimal predictions. In contrast, knowledge gradient acquires points which are believed to result in the lowest-valued minimum in the posterior mean after conditioning on the new point. Such samples would usually not be at the minimum of the function, as learning the overall shape of the function tended to result in more accurate estimates for the minimum posterior. However, by almost entirely avoiding collecting samples in the actual minimum of the objective function, this acquisition function occasionally made sub-optimal predictions for the value of the function at the optimal θ. Thus, in our optimization routine, we alternated between acquiring points utilizing expected improvement and knowledge gradient.

In this section, we demonstrate the behaviour of our acquisition routine on the order of points selected when the objective function has additive Gaussian noise. Figure 4.3 demonstrates this behaviour, and is seeded with an initial randomly selected training point. As discussed, it is assumed that an accurate estimate for the variance of the additive Gaussian noise may be obtained in practice, and thus a delta function prior with the correct variance is placed on σ^2.

In the first panel of Figure 4.3, one sample has been collected at approximately $\theta = 2$. As there is no prior placed on α, the maximization of the negative log marginal likelihood determines that an $A = 0$ is the most probable explanation for the single data point, and so it learns a flat line. This may be considered undesirable behaviour, and could be easily remedied by placing a prior that penalizes small values of α, or simply by seeding the algorithm with at least two training points. In practice, we found the best results when seeding the algorithm with three training points evenly spaced on the interval $[0, \frac{3\pi}{2}]$. Nevertheless, once the algorithm acquires its first training point, it starts to learn the shape of a more reasonable sine curve (as shown in the second panel). While the posterior mean in the first panel appears flat, it is actually slightly curved, and so when the expected improvement acquisition function acquires the first additional sample (as shown in the second panel), it selects the point essentially at the minimum of the posterior mean (as opposed to completely randomly selecting it if α was exactly zero). Clearly, the belief maintained by the algorithm in the first panel is incorrect, as the point selected by expected improvement actually has a function value greater than the previous best point observed; such is part of the learning process. As the

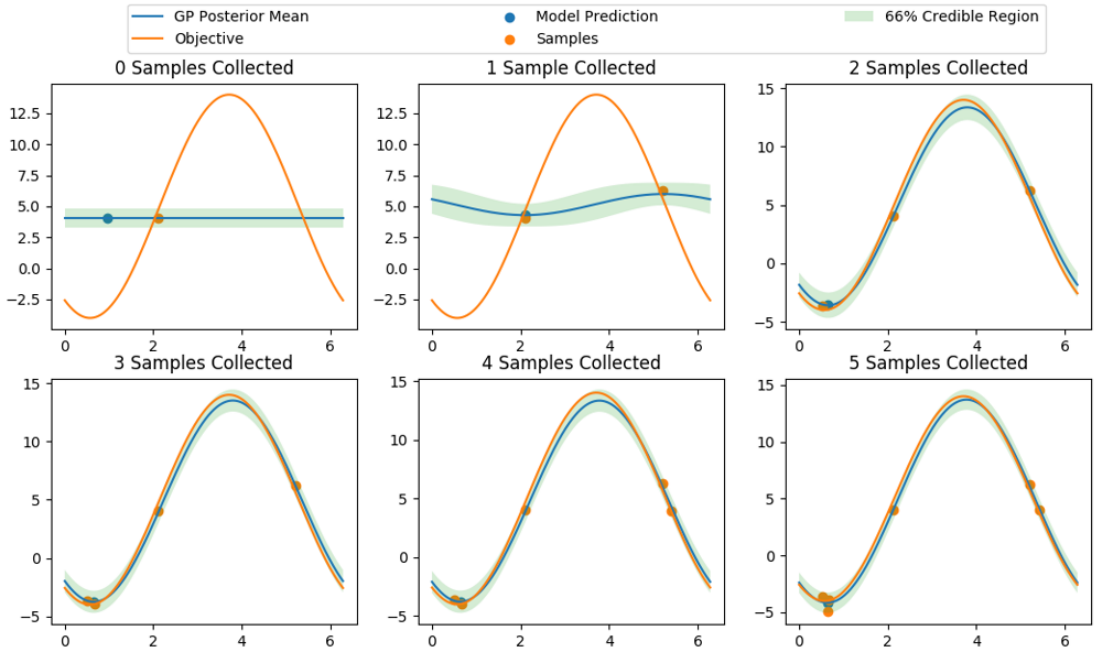

Figure 4.3: **Demonstration of Expected Improvement Acquisition Function with Additive Gaussian Noise**. In this experiment, the objective function has values $A = -9$, $B = 1$, and $C = 5$. X_* consists of 500 points uniformly spaced on the interval $[0, 2\pi]$. The initial training point was selected uniformly at random on the interval $[0, 2\pi]$. The variance on the additive Gaussian noise in the objective function was set to 0.6. A delta function prior is placed on the noise variance of 0.6. Expected improvement is used as the acquisition function.

algorithm alternates between using the expected improvement and knowledge gradient acquisition functions, in the third panel, a point is selected with knowledge gradient. As the belief maintained by the algorithm is still incorrect, the point selected actually happens to be near the minimum of the objective function's mean (selecting such a point is usually not the preference of the knowledge gradient acquisition function, and was a coincidence). In the subsequent panel, the expected improvement function acquires a point, and as the shape of the objective function is now being accurately modelled by the algorithm, as expected, it selects a point maximizing the expected utility by selecting the minimum of the posterior mean. In panel 5, we see the final selection made by knowledge gradient, which as expected, elects to choose a point which it expects to result in the posterior mean being minimized; in this case the point selected happens to contribute more information regarding the overall shape of the sine function. Finally, expected improvement acquires one last point near the minimum of the objective.

4.1.3 Noiseless Evaluation

In the absence of noise, BSPO, RSIA, and RSS all learn the optimal value of the objective function using only three evaluations of the cost function. Here, the implementation of the Bayesian single-parameter

optimization algorithm is experimentally validated by comparing it to the analytical results of 100 randomly generated problem instances. The details of the experiment may be found in Table 4.1.

	Average Absolute Error	Absolute Error Sample Std Dev	Median Absolute Error
θ	2.082424×10^{-6}	9.313339×10^{-6}	2.07786×10^{-7}
$f(\theta)$	9.220666×10^{-5}	234391×10^{-4}	1.302902×10^{-5}

Table 4.1: **Noiseless Experimental Validation of BSPO**. 100 objective functions are randomly generated by selecting values for A and C uniformly at random over the arbitrary interval $[-100, 100]$, and by selecting B uniformly at random in the interval $[0, 2\pi]$. X_* contains 800 elements. BSPO is seeded with an initial point at $\theta = \pi$, and is allowed to collect a total of two additional samples (for a total of three function evaluations) using the alternating acquisition function system.

The errors obtained are all close to the exact analytical solution, and the small deviations from the exact value are well explained by the resolution in the predictive domain. Uniformly dividing the interval $[0, 2\pi]$ into 800 sections yields a step size of approximately 0.0079. As such, it may appear surprising that the algorithm manages to obtain an average absolute error for θ that is multiple orders of magnitude smaller. This apparent discrepancy is explained by the mechanism through which the algorithm both acquires new points to test, and through which it calculates θ^* and $f(\theta^*)$ utilizing the posterior. First, as the acquisition function is evaluated on this discrete grid, the values will tend to be either too small or too large depending on how far from the center of the grid cell a preferable value would have been. The distribution of this bias would be expected to be random in either direction, and thus when collecting multiple samples the impact of the discrete grid would, at least to some extent, cancel out. Moreover, when computing θ^* and $f(\theta^*)$ using the posterior, the analytical minimum of the posterior is computed, thus circumventing any issues associated with discretization.

4.1.4 Comparison to Rotosolve with Additive Gaussian Noise

We now compare the algorithm's performance to various configurations of the analytical algorithm when additive Gaussian noise is present. Table 4.2 summarizes the results of 100 randomly generated problem instances with a relatively small expected signal-to-noise ratio of $\frac{9}{1.25}$. Table 4.3 summarizes the results of 100 randomly generated problem instances with a large expected signal-to-noise ratio of $\frac{9}{3}$. A representative run comparing the two algorithms is shown in Figure 4.4.

		Average Squared Error	Squared Error Sample Std Dev	Median Squared Error
BSPO	θ	0.015767	0.021522	0.006044
	$y(\theta)$	0.707609	0.941278	0.304337
RSIA	θ	0.031043	0.056788	0.008820
	$y(\theta)$	1.686753	2.305758	0.986737
RSS	θ	0.010618	0.013453	0.005267
	$y(\theta)$	1.478234	2.018851	0.600289

Table 4.2: **Performance Evaluation of Single Parameter Optimization Algorithms, Moderate Gaussian Noise**. 100 objective functions are generated and tested by setting $A = 9$, $\sigma^2 = 1.25$, selecting B uniformly at random on the interval $[0, 2\pi]$, and selecting C uniformly at random over the arbitrary interval $[-100, 100]$. The grid of the predictive distribution is discretized to contain 800 elements in the interval $[0, 2\pi]$. BSPO, RSIA, and RSS all use four samples of the objective function. BSPO is seeded with three initial samples at $\theta = 0, \frac{2\pi}{3}, \frac{3\pi}{2}$. As only one sample is acquired during optimization, BSPO is configured with the knowledge gradient acquisition function.

As the level of noise relative to the amplitude of the signal is relatively low in Table 4.2, we would expect all three algorithms to perform fairly comparably, and indeed, in computing the optimal vaue of θ, this is the case. All algorithms utilizes a total of four samples of the objective function. As such, it is is interesting to see that RSS somewhat significantly outperformed RSIA. As RSS utilizes all four samples collected in modelling the underlying objective function, as oppose to RSIA which when collecting four samples, uses three of them to compute $\hat{\theta}^*$, and one of them to compute $f(\hat{\theta}^*)$. Given this difference in modelling, when such a small number of samples are collected, this result makes sense. Moreover, it is also worth noting that BSPO significantly outperformed both RSIA and RSS when modelling the actual true value of the underlying objective function, $f(\theta^*)$, obtaining a mean squared error that was $2\times$ less than either of the other models. BSPO also had a substantially smaller standard deviation and median than the other two models in predicting $f(\theta^*)$.

		Average Squared Error	Squared Error Sample Std Dev	Median Squared Error
BSPO	θ	0.02609	0.04533	0.01109
	$y(\theta)$	3.16508	4.98232	1.65333
RSIA	θ	0.06212	0.12083	0.02716
	$y(\theta)$	9.90465	16.73820	3.37815
RSS	θ	0.06724	0.09900	0.02184
	$y(\theta)$	8.81747	10.93441	4.25875

Table 4.3: **Performance Evaluation of Single Parameter Optimization Algorithms with Large Gaussian Noise**. 100 objective functions are generated and tested by setting $A = 9$, $\sigma^2 = 3$, selecting B uniformly at random on the interval $[0, 2\pi]$, and selecting C uniformly at random over the arbitrary interval $[-100, 100]$. The grid of the predictive distribution was discretized to contain 800 elements. BSPO and RSIA both use seven samples of the objective function, and BSPO is seeded with three samples at $\theta = 0, \frac{2\pi}{3}, \frac{3\pi}{2}$. RSS used four samples of the objective function. BSPO is configured with the alternating acquisition function system.

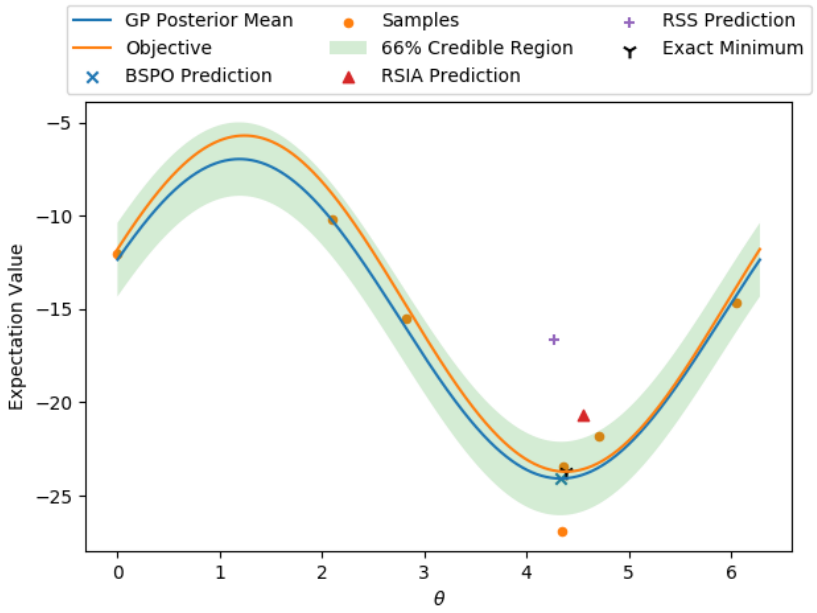

Figure 4.4: **Representative Comparison of Single Parameter Optimization Algorithms with Large Additive Gaussian Noise**. In this experiment, the objective function has values $A = 9$, $B = 2$, and $C = -3$. The variance on the additive Gaussian noise in the objective function is $\sigma^2 = 3$. The grid of the predictive distribution is discretized to contain 800 elements over the interval $[0, 2\pi]$. BSPO is seeded with three initial samples at $\theta = 0, \frac{2\pi}{3}, \frac{3\pi}{2}$. BSPO is configured with the alternating acquisition function system. BSPO and RSIA both use seven samples of the objective function, while RSS uses four.

Table 4.3 gives the results from the same experiment just discussed, only with a much lower signal-to-noise ratio. Indeed, as the magnitude of the noise begins to grow relative to the magnitude of the amplitude of the objective, BSPO begins to significantly outperform any of the configurations of the analytical solution. BSPO outperforms both RSIA and RSS by a factor of at least 1.5× in every metric collected. Assume that both BSPO and RSIA have the same performance. Using a two-tailed test, we find that we may reject this hypothesis with $p < 0.008$ in calculating θ^*, and we may reject it with $p < 0.0003$ in calculating $f(\theta^*)$. When we assume that BSPO and RSS have the same performance, the two-tailed test finds that we may reject this hypothesis with $p < 0.0003$ in calculating θ^*, and we may reject it with $p < 0.000006$ in calculating $f(\theta^*)$. Moreover, it is also worth emphasising the importance of the consistency (the relatively lower standard deviation in error) of BSPO relative to the other two models. When using these single parameter optimizers in a system such as Rotosolve, if some of the parameters are configured sub-optimally during optimization, the rest of the optimization procedure may lead to unpreferable results.

Finally, Figure 4.4 shows a representative of the 100 experiments conducted for the data presented in Table 4.3. As expected given the alternating acquisition function system used by the algorithm, samples are collected for the objective function throughout the domain of the sine curve. These samples allow the

model to essentially disregard the outlier located at approximately $\theta = 4.4$ and $f(\theta) = -27$, as the prior mean is computed utilizing the average f values observed, and as the average f values vary greatly and are not entirely concentrated in the minimum of the objective (as would occur with the expected improvement acquisition function alone), the model manages to quite accurately capture the underlying sine function, and thus gives the best prediction out of all of the models.

Bibliography

[1] Robert J Adler and Jonathan E Taylor. *Random fields and geometry*. Springer Science & Business Media, 2009.

[2] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510.

[3] Peter I Frazier. "A tutorial on bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[4] Roman Garnett. *Bayesian Optimization*. Vol. In preparation. To be published by Cambridge University Press, 2020.

[5] Marc G Genton. "Classes of kernels for machine learning: a statistics perspective". In: *Journal of machine learning research* 2.Dec (2001), pp. 299–312.

[6] Lov K Grover. "Quantum mechanics helps in searching for a needle in a haystack". In: *Physical review letters* 79.2 (1997), p. 325.

[7] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". In: *Nature* 549 (Sept. 2017), 242 EP –. URL: https://doi.org/10.1038/nature23879.

[8] Norbert M Linke et al. "Experimental comparison of two quantum computing architectures". In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3305–3310.

[9] Jarrod R McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (2016), p. 023023.

[10] Kosuke Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (2018), p. 032309.

[11] B Moseley, M Osborne, and S Benjamin. "Bayesian optimisation for variational quantum eigensolvers". In: *quantum* 3 (2018), p. 4.

[12] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. "Quantum circuit structure learning". In: *arXiv preprint arXiv:1905.09692* (2019).

[13] Josh Payne and Mario Srouji. "Approximate Graph Spectral Decomposition with the Variational Quantum Eigensolver". In: *arXiv preprint arXiv:1912.12366* (2019).

[14] Edwin Pednault et al. "Leveraging secondary storage to simulate deep 54-qubit sycamore circuits". In: *arXiv preprint arXiv:1910.09534* (2019).

[15] Edwin Pednault et al. *On "Quantum Supremacy"*. 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/.

[16] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5 (July 2014), 4213 EP –. URL: https://doi.org/10.1038/ncomms5213.

[17] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

[18] Arthur G Rattew et al. "A Domain-agnostic, Noise-resistant Evolutionary Variational Quantum Eigensolver for Hardware-efficient Optimization in the Hilbert Space". In: *arXiv preprint arXiv:1910.09694* (2019).

[19] Engineering National Academies of Sciences, Medicine, et al. *Quantum computing: progress and prospects*. National Academies Press, 2019, p. 124.

[20] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[21] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[22] Virginia Vassilevska Williams. "Multiplying matrices in O (n2. 373) time". In: *preprint* (2014).