Quantum Algorithms for Atomic Clocks
by
Michael Mullan
B.S., University of Illinois, 2002
M.S., University of Illinois, 2006

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Physics
2013
This thesis entitled:
Quantum Algorithms for Atomic Clocks
written by Michael Mullan
has been approved for the Department of Physics

______________________
Emanuel Knill

______________________
Ana Maria Rey

______________________

Date ________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
We describe quantum interrogation schemes for passive atomic clocks. During any given interrogation period, the optimal interrogation algorithm depends on the state of the clock - specifically on the frequency deviation of the flywheel (classical oscillator) from the atomic standard. As a clock runs, it is possible to estimate this deviation. Nonetheless, traditional schemes use the same, fixed algorithm for each interrogation period, which is necessarily independent of this prior knowledge. Here we present a dynamic scheme, tailoring our algorithms to the clock’s state before each interrogation. These strategies are derived by constructing a complete model of a passive clock – specifically, a probability distribution describing the estimated average offset frequency of the flywheel during both the upcoming interrogation period and interrogation periods in the past is updated via appropriate noise models and by measurements of the atomic standard.

To reduce the deviation from an ideal clock we optimize the next interrogation algorithm by means of a semidefinite program for atomic state preparation and measurement whose objective function depends on the updated state. This program is based on the semidefinite programming formulation of quantum query complexity, a method first developed in the context of deriving algorithmic lower bounds. The application of semidefinite programming to an inherently continuous problem like that considered here requires discretization; we derive bounds on the error introduced and show that it can be made suitably small.

Finally, we implement a full simulation of a passive clock with power-law noise models and find significant improvements by applying our techniques.
## Contents

### Chapter

1. **Introduction** ............................................ 1

2. **Quantum Computing** .................................. 5
   2.1 From Classical to Quantum Computation ............... 5
      2.1.1 Classical States ......................................... 6
      2.1.2 Quantum States .......................................... 8
   2.2 Postulates of Quantum Mechanics ....................... 9
      2.2.1 States .................................................... 9
      2.2.2 Composite Systems ...................................... 10
      2.2.3 Evolution ............................................... 11
      2.2.4 Measurement ............................................ 15
   2.3 Useful Properties and Techniques ...................... 19
      2.3.1 Schmidt Decomposition ................................. 19
      2.3.2 Unitary Freedom ........................................ 20
   2.4 Quantum Algorithms .................................... 20
      2.4.1 Search .................................................. 20
      2.4.2 Other Quantum Speedups ............................... 27
      2.4.3 Query Model of Computation ........................... 28
   2.5 Computational Complexity ............................... 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Generalizing the Adversary Method</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Classical Lower Bounds</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>Quantum Lower Bounds</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Modified Spectral Adversary Method</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Semidefinite Programming</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Operator Constraints</td>
<td>37</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Measurement Constraints</td>
<td>41</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Cost Function</td>
<td>43</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Original Spectral Adversary Method</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Other Variants of the Adversary Method</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Noise</td>
<td>49</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Introducing the Environment</td>
<td>49</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Examples</td>
<td>53</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Quantum to Classical Interpolation</td>
<td>56</td>
</tr>
<tr>
<td>3.6</td>
<td>Reduced Quantum Resources</td>
<td>58</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Results</td>
<td>63</td>
</tr>
<tr>
<td>4</td>
<td>Atomic Clocks</td>
<td>65</td>
</tr>
<tr>
<td>4.1</td>
<td>Timekeeping</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Basics of Atomic Clocks</td>
<td>67</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Active vs. Passive Clocks</td>
<td>67</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Modern Atomic Clock Technologies</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Quantum Metrology and Interferometry</td>
<td>70</td>
</tr>
<tr>
<td>4.3.1</td>
<td>State Evolution</td>
<td>70</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Ramsey Spectroscopy</td>
<td>71</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Interferometry</td>
<td>74</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Parameter Estimation</td>
<td>75</td>
</tr>
</tbody>
</table>
4.3.5 Standard Quantum Limit ........................................ 76
4.3.6 Heisenburg Limit ................................................. 77
4.3.7 Prior Work .......................................................... 78

5 Optimal Interrogation .................................................. 82
5.1 Interrogation Algorithms ............................................ 82
5.2 The Clock SDP ........................................................ 84
  5.2.1 Dicke States ..................................................... 84
  5.2.2 Oracles and Probabilities ....................................... 87
  5.2.3 Querier Error ..................................................... 88
  5.2.4 Oracle Error ..................................................... 94
  5.2.5 Size of the SDP ................................................... 103
5.3 Results ................................................................. 104

6 Passive Clock Optimization .......................................... 112
6.1 Introduction .......................................................... 112
6.2 Passive Clock Operation ............................................ 112
  6.2.1 Post Interrogation Strategies .................................. 112
  6.2.2 Modeling a Clock ............................................... 114
6.3 Flywheel Noise ....................................................... 117
6.4 Optimization Strategy ............................................... 118
  6.4.1 Objective ......................................................... 118
  6.4.2 Probability Updates .......................................... 119
  6.4.3 Cost Function ................................................... 122
6.5 Sources of Error ..................................................... 124
6.6 Performance Metrics ............................................... 127
6.7 Results ................................................................. 129
**Tables**

**Table**

3.1 Entropy of 8-element SEARCH (where there may not be a marked element) and 4-element PARITY as a function of the number of independent computational paths, after the first query. ......................................................... 63

5.1 Initial states that minimize the cost function $C(\omega, a) = 4 \sin^2 \left(\frac{\omega - f_a}{2} \right)$ for two atom, one-query protocols assuming different-width Gaussian priors. We used 20 unrefined frequency estimates. ............................................................. 108

6.1 Percent improvement of our method over those of Ramsey and those of Buzek. We average the gains in average frequency variance over the last twenty (81-100) timesteps and those in overlapping Allan variance across all averaging times. ............. 131
Figures

Figure

3.1 4 element SEARCH, with the possibility of no marked element. Here, $|O| = 5$ and $|Q| = 4$  .................................................. 54

3.2 4 element Deutsch-Jozsa problem, $|O| = 8, |Q| = 4$  ........................................ 55

3.3 4 element PARITY. $|O| = 16, |Q| = 4$  .................................................. 56

3.4 Quantum/Classical Interpolation in the 4 element SEARCH problem, where there is always a marked element. The blue line depicts one qubit decohering with probability $p$, whereas the red line depicts two qubits decohering independently with probability $p$.  .................................................. 57

4.1 Light coming from the left is incident on a beam splitter which divides the light into the shorter, upper path and the longer, bottom path. The light is then recombined at the second beamsplitter. Afterwards, a measurement of relative phase provides some information about the difference in path length.  ........................................ 75
5.1 Comparison of the protocol derived in Ref. [18] and those derived by our method for one and four atoms and two different cost functions. We use a 15-point oracle discretization and simulate Gaussian priors of various widths. The figures on the top plot the cost computed in Ref. [18] (green, dashed line) and our numerical upper bound (blue, solid line), which as discussed, is equivalent to the minimum continuous cost obtained by one of our extracted algorithms. The blue solid line in the figures on the bottom correspond to the difference between our lower bound and upper bound, $c_l - c_u$, illustrating both the strength of our bounds and how much lower the continuous cost could potentially be. The lower bound was computed by averaging 100 discretizations; error bars show the estimated standard error of the average thus obtained. Our querier discretization bounds cannot be applied to the periodic cost function. Therefore in (a) the lower bounds are for the discretizations chosen, and in (b), we plot a green, dashed line corresponding to $c_l - c_u - \epsilon_q$, which is a lower bound for the continuous problem. We use 20 frequency estimates; in (b) these are chosen by minimizing equation (5.24). We do not optimize the estimates via the iterative technique of Sect. 5.2.3. The lines connecting the data points are to guide the eyes.

5.2 Comparison of the protocol in Ref. [18] and those obtained by our method for Gaussian priors of various widths. Here we do not discretize the oracle; rather, we change basis and solve SDP $\tilde{S}_C$. The blue, solid line is the cost computed by our method and the green, dashed line is the cost computed by the method of Ref. [18]. The inset in (b) is a magnification of the lower left side of the main plot and shows the cost for both 10 (top line) and 11 (middle line) atoms of the former protocol; these are indistinguishable on the main plot. The frequency estimates in (b) for our method are chosen by minimizing equation (5.24), and are not refined via the iterative technique of section 5.2.3.3.
5.3 Comparison of the optimal protocol derived in Ref. [23] and the one obtained by our SDP for two atoms. The top, red line depicts the difference between our upper bound and the cost computed in Ref. [23], and the bottom, blue line depicts the difference between our lower bound and the cost computed in Ref. [23]. Error bars again correspond to the estimated standard error in the lower bound. We used a 15-point prior discretization and set $F$ to the optimal set of frequency estimates derived in [23] so there is no querier discretization error. The lower bound was computed by averaging 500 random oracle discretizations.

5.4 Upper ($c_u$, dotted line) and lower ($c_l$, solid line) bounds on cost for oracle discretizations with varying numbers of points. We are simulating the last point of figure 5.1(b) - that is, 4 atoms, a standard deviation of 1.875, and the quadratic cost function. The lower bound is computed by averaging 32 random oracle discretizations. Note that the values in figure 5.1 are for a 15 point oracle discretization.

5.5 Comparison of the optimal protocol derived in Ref. [23] (upper, green line) and the one obtained by our SDP (lower, blue line) for two atoms, three frequency estimates chosen, in our case, by refinement according to section 5.2.3.3, and $C(\omega, a) = (\omega - f_a)^2$.

5.6 Average cost after one query (top, green line), two queries combined classically (middle, red line), and two coherent quantum queries (bottom, blue line). We simulated two atoms, and used 100 frequency estimates and the cost function $C(\omega, f_a) = (\omega - f_a)^2$.

6.1 Evolution of the flywheel’s probability distribution in a clock protocol. Noise decreases our knowledge of the flywheel’s frequency, widening $p(\omega)$. The prior probability distribution on the right is used in our optimization procedure, and describes the average frequency of the flywheel over the probe time $T$. A measurement of the atomic standard then yields a measurement outcome $a$, which can be used to compute the posterior distribution $p(\omega|a)$. 
6.2 A high level overview of the strategy developed here to optimize a passive clock. In order to assign estimates of true time to events (left side of figure), flywheel cycles are counted in the context of an estimate of the flywheel’s deviation. This deviation is described by a posterior probability distribution which is computed based on the result of an atomic interrogation. This interrogation is implemented via an algorithm derived by the semidefinite program from chapter 5, which uses the prior probability distribution of the average flywheel deviation during the interrogation period of interest. This distribution is obtained by computing how the relevant noisy process affects the previous posterior distribution.

6.3 Evolution of the flywheel’s probability distribution in a clock protocol. The probability distribution at the top corresponds to the average frequency of the flywheel during interrogation period 1. A measurement of the atomic standard at time \( t_1 \), with \( F \) element POVM \( \{P_a\}_1 \) then yields measurement outcome \( a_1 \) with probability \( p(a_1) \). This can be used to compute the posterior distribution \( p(\omega_1|a_1) \). (In the figure, we imagine that we obtain measurement result \( a_1 = 1 \).) Noise then affects the flywheel for duration \( T_2 \); this decreases our knowledge of the flywheel’s frequency, widening the probability distribution. The resulting prior probability distribution corresponds to the average frequency of the flywheel over interrogation period 2. This procedure is repeated indefinitely.

6.4 A comparison of our method (bottom blue solid line), standard Ramsey spectroscopy (middle red dotted line), and Buzek’s method [18] (top green dashed line) on a two qubit clock subject to fractional Brownian motion with \( H = .5 \) and \( h_{-1} = .03 \). We fix the length of the interrogation period to \( T = 1 \). Figure (a) is a log-log plot of the standard deviation of the frequency deviation averaged over cumulative time, while (b) is a log-log plot of the overlapping Allan deviation. These are computed via 200 iterations of our method, 1000 iterations of Ramsey spectroscopy, and 1000 iterations of Buzek’s method.
6.5 A comparison of our method (bottom blue solid line), standard Ramsey spectroscopy (middle red dotted line), and Buzek’s method (top green dashed line) on a three qubit clock subject to $1/\omega$ noise with $h_{-1} = .05$. Plots (a) and (b) are as described in Fig. 6.4 and are computed via 400 iterations of our method, 800 iterations of Ramsey spectroscopy, and 800 iterations of Buzek’s method.
Chapter 1

Introduction

In recent years, research into quantum computation has grown enormously - so much that, even outside the scientific community, the idea of new machines with enormous, yet mysterious computational power is no longer an unfamiliar one. And while this research has progressed quickly, at the time of this writing, a complete, scalable quantum computer is still decades away. However, while the construction of a quantum computer is an important goal, the field of quantum computation is not so limited; it is far richer and offers deep insights into the physical world.

Quantum computation can perhaps be more broadly defined as the study of the interplay between the physical and the computational sciences. The synergy between the two has led to novel discoveries, revelations, and challenges. Indeed, there are now many physical results that have been obtained, in large part, via tools from computer science. Work by Abrams and Lloyd [4] shows that if quantum mechanics was nonlinear, then nature would be capable of solving problems in the complexity classes #P and NP in polynomial time. There is strong evidence in the computational sciences that these problems are inherently difficult; the ability to solve them efficiently would give nature, and thereby quantum computers, computational capabilities that are considered unrealistically powerful. While this certainly doesn’t constitute a proof of linearity, it is a distinctly different way of analyzing nature. Works by Aaronson [1] and Deutsch [26] take a similar approach. In the former, it is shown that if quantum mechanics is altered either via deviations from the Born rule (that the probability of an outcome is given by the square of the wavefunction), or with the addition of post-selection, then it also gains extraordinary computational power. The latter, by
way of a novel model of time travel, shows that many of the paradoxes traditionally associated with traveling to one’s past are resolved when quantum mechanics is taken into account, but are then forbidden by computational arguments. Furthermore, it is possible to draw a parallel between the execution of an algorithm and the evolution of a physical system via the laws of quantum mechanics. Thinking of such evolution as computation has led to novel ways of solving problems. This is perhaps most vividly demonstrated in adiabatic quantum computation [34], where a computational problem is encoded in a physical Hamiltonian.

Perhaps even more surprisingly, the study of quantum mechanics has led to novel results in classical computer science. Ref. [53] proves an upper bound on the length of a locally decodable code using a quantum argument. From what they call “dequantization” of a quantum result, Ref. [5] proves that approximating the shortest and closest vector in a lattice is in the complexity class \( \text{NP} \cap \text{coNP} \). Ref. [2], adapts the quantum adversary method [8] for classical problems and uses it to prove classical randomized lower bounds for local search. Lower bounds and the adversary method will be discussed in great detail in chapter 3. There, in fact, we will begin with the classical adversary method presented along the lines of Ref. [2] as a first step in the development of the full quantum method. Moreover, recently, the study of quantum complexity theory [14] has led to the emergence of a comprehensive set of quantum complexity classes. Relations between these classes, as well as between quantum and classical complexity classes, have been studied in depth.

This thesis focuses on a concrete example of the interplay between computer science and physics, and the parallel between computation and physical evolution. Specifically, we develop and take advantage of a connection between the operation of a passive atomic clock and the execution of a quantum algorithm. Note that the relationship between atomic clocks and quantum computers is important historically. While quantum computation was in its infancy, it was noticed that atomic clock technology could provide the control [94] needed to implement Shor’s algorithm [87]. This was an early impetus for the subsequent explosion of research in the field. Now, optical clocks, the most accurate clocks in development [84], operate in ion traps, borrowing advances developed first for quantum computation. Thus, ion trap quantum computers and many modern clocks are
essentially identical technologies. In our case, thinking of them as such is critical, as it allows us to apply techniques from the computational sciences to these clocks.

Specifically, our goal is to improve atomic clock performance by developing quantum algorithms that optimize clock interrogations. We then develop a way of bridging a series of such interrogations which maximizes the information gained by each. Unlike prior work, this approach is necessarily dynamic. The optimal interrogation algorithm may vary from one interrogation to the next; these can be constructed as the clock runs. Along the way, we make contributions to both quantum and classical complexity theory, including a novel way of analyzing the performance of optimal classical algorithms that, in the spirit of the work discussed above, is developed purely by quantum arguments. While these developments are often discussed in the context of clocks, we expect them to be useful outside this domain.

We begin by discussing quantum computation, focusing on quantum algorithms and quantum complexity theory. Chapter 2 reviews relevant background material from computer science and quantum mechanics. We develop the principles of quantum computation by extending classical computation; from here, we discuss several well known results in quantum information that will play an important role later. Chapter 3 discusses classical and quantum lower bounding techniques, specifically the adversary method. We briefly discuss the original method due to Ambainis [8] before turning to a detailed discussion of the spectral adversary method. We discuss several alterations and extensions of this method that will be useful in clock optimization as well as in other contexts. In chapter 4 we turn to atomic clocks, giving an overview of their operation and review prior work focusing on optimizing interrogations by taking advantage of atomic entanglement and other quantum effects. Chapter 5 then adapts our variation of the spectral adversary method to optimize a single clock interrogation. Here, this optimization is done in a greedy sense; we are concerned only with individual interrogations and do not consider long term clock stability. In chapter 6, we overcome this limitation and illustrate how our method can be extended to improve long term clock performance. We also implement a full simulation of a clock subject to various power-law noise models and use it to compare our technique to prior work.
Much of the thesis is derived from Refs. [68, 69]. While some text and a number of proofs are taken directly from these references, new, more detailed explanations are added in many important areas. Background material provided in chapters 2 and 4 assumes no knowledge of quantum computation or atomic clocks. The hope is that this makes this thesis accessible to audiences from theoretical quantum information, classical computational complexity, and experimental atomic clocks, as it may find use in each field.

A significant component of this work is a large code base which implements the techniques described in this document. The SDPs derived in chapters 3 and 5 are implemented and solved in Matlab; this includes code to accomplish complementary tasks, such as algorithm and POVM extraction. The passive clock optimization scheme described in chapter 6 is implemented in C++, which at present, calls the Matlab SDP solver as a subroutine. Our code is far too extensive to reproduce here. However in the appendix, we provide code which solves the arguably most useful SDP discussed in this thesis.
Chapter 2

Quantum Computing

2.1 From Classical to Quantum Computation

Formalisms of classical computation often come in the form of either Turing machines or circuits. The latter, in particular, lends itself to an intuitive if somewhat naive glimpse at what could be gained with quantum computers. In the circuit formalism, and indeed, in a real, physical computer, we have a set of classical registers that can each hold one or more bits; each bit can take the values 0 and 1. These registers are operated on by a set of gates arranged in such a way as to compute some desired function.

In the quantum circuit model, we can still speak of registers and gates, but now, each bit can be not only 0 or 1, but also a superposition of both 0 and 1. This means that, in a certain sense, the bit can be in both the 0 state and 1 state at the same time. While classically, given a set of $n$ bits, we could hold any number between 0 and $2^n - 1$, now we can, in this same sense, hold all of these numbers simultaneously, and our circuit of gates computes the desired function on every number in this quantum superposition simultaneously.

At the moment, this seems immensely powerful, but the true power of quantum computation is much more subtle than this. While we can evaluate a function on a superposition as described, we cannot readout more than a single answer. When measured, we say that the quantum superposition collapses, and we obtain one of the $2^n$ possible answers - each with some probability. Thus, in order to take advantage of potential quantum speedups, we cannot naively apply classical algorithms and must be significantly more clever.
While moving from classical computation to quantum computation in this manner may provide intuition about potential gains, it does not provide any insight as to why quantum mechanics operates in such a seemingly strange manner. That is, why should we be able to store such strange superpositions, and why does measurement destroy them? In an attempt to make headway here, and to better match developments later in this thesis, we do not proceed any further with the circuit model. Instead, we will move from classical to quantum computation by thinking of quantum mechanics as a generalization of classical probability theory along the lines of Hardy and Aaronson [44, 3].

First we note that in the classical circuit model discussed above, we did not allow for randomness. While there is no way of generating completely random numbers classically, nor has it been proven that randomness adds no power to deterministic computation, i.e. $P = BPP$\footnote{P and BPP are examples of computational complexity classes [11]. The class P contains problems that are solvable in polynomial time using a deterministic algorithm, while BPP contains problems that are solvable in polynomial time with a randomized algorithm. It is not known whether these two classes are identical.}, randomness is considered a fundamental part of classical computation, and we will make use of it here.

### 2.1.1 Classical States

We represent the state of a classical system consisting of $n$ bits (or registers) by a vector $\vec{s}$ of length $N = 2^n$. In a deterministic computation, these registers can be in $N$ different states, which are represented by vectors with a single 1 in the appropriate position and 0’s elsewhere. For example if the registers are holding the bit string 0…011, the vector $\vec{s}$ has a 1 in position 3,

$$
\vec{s} = \begin{pmatrix}
0 \\
0 \\
0 \\
1 \\
0 \\
\vdots \\
0
\end{pmatrix}.
$$

(2.1)
Label the vector with a 1 in the \(i\)th position \(\vec{s}_i\). Now we allow for randomness and write \(\vec{s}\) as a convex combination of the \(\vec{s}_i\),

\[
\vec{s} = p_0\vec{s}_0 + p_1\vec{s}_1 + \ldots + p_{N-1}\vec{s}_{N-1} = \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{pmatrix},
\]

where \(0 \leq p_i \leq 1\). The \(p_i\)'s are the probabilities for the set of classical registers to be in the state \(i\), so we must have \(\sum_i p_i = 1\). Throughout, the \(s_i\) will be referred to as extremal or pure states while the convex combination \(\vec{s}\) will be referred to as a mixed state. One can think of pure states as physical states; that is, states that correspond to a realizable configuration of the classical registers. A mixed state, on the other hand, then represents a lack of knowledge about the true underlying state. That is, in reality, the computer is in some pure state, \(\vec{s}_i\); however, we do not know which, and can only specify a probability distribution over potential \(\vec{s}_i\). This is often caused classically by the presence of randomness in an algorithm. While this way of distinguishing pure and mixed states can be carried over to quantum mechanics, there, it is often useful to think of mixed states as just as “real” as pure states.

A classical computation corresponds to a sequence of operations \(O(1), O(2), \ldots O(M)\) applied to the state \(\vec{s}\),

\[
\vec{s} \rightarrow \vec{s}' = O(1)O(2)\ldots O(M)\vec{s} = p'_0s_0 + p'_1s_1 + \ldots + p'_{N-1}s_{N-1},
\]

followed by a measurement. These operations are given by stochastic matrices. Stochastic matrices consist of nonnegative real numbers, where entry \(O_{ij}\) corresponds to the probability that state \(j\) will transition to state \(i\), and therefore, \(\sum_i O_{ij} = 1\). \(O\) must be of this form in order to ensure that the new set of probabilities sum to 1, \(\sum_i p'_i = 1\).
2.1.2 Quantum States

Moving from classical probabilistic computation to quantum computation now requires three adjustments:

1. Allow the entries of \( \vec{s} \) to be arbitrary complex numbers, \( a_i \), instead of positive reals, \( p_i \). These entries are referred to as amplitudes.

2. Replace the one-norm with the two-norm. That is, require \( \| \vec{s} \|_2 = \sum_i |a_i|^2 = 1 \). Now, \( |a_i|^2 \) instead of \( p_i \) represents the probability that, when measured, the system will be found in state \( i \).

3. For a space of dimension \( N \), treat all \( \vec{s} \) as pure states. These states correspond to a real, physical configuration of a quantum system. Mixed states are convex combinations of these states in a space of dimension \( N^2 \).

Hardy [44] shows that each of these modifications can be derived starting with a set of axioms consistent with classical probability theory and asserting that there must exist a continuous reversible transformation between any two pure states of a system. It is easy to see that such a continuous transformation cannot exist in classical probability theory, as there are only a finite number, \( N \), of pure states in a system of dimension \( N \). However, this is possible in quantum mechanics, given the three adjustments above.

Next, we examine the consequences of these changes in depth. The next four subsections describe details of what are sometimes referred to as the postulates of quantum mechanics [70].
2.2 Postulates of Quantum Mechanics

2.2.1 States

Instead of writing out a complete column vector, we express the state of a quantum system using Dirac bra-ket notation in the following way:

\[
\vec{s} \rightarrow |\psi\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix} = a_0 |0\rangle + a_2 |1\rangle + \ldots a_{N-1} |N-1\rangle.
\] (2.4)

The \(|i\rangle\) are known as kets and correspond to basis vectors. This particular choice of basis, where the ket \(|i\rangle\) is equivalent to the column vector with a 1 in the \(i\)th position and 0 elsewhere, is known as the computational basis. We suppress writing the ket \(|i\rangle\) if \(a_i = 0\). A sum of states like that above is referred to as a superposition. The bra \(\langle \psi |\) is the complex conjugate transpose of the ket \(|\psi\rangle\). Therefore, the inner product of \(|\psi_i\rangle\) and \(|\psi_j\rangle\) is given by \(\langle \psi_i | \psi_j \rangle\) and the outer or tensor product is given by \(|\psi_i\rangle \langle \psi_j |\). This complex vector space, with associated inner product, is an example of a Hilbert space. All quantum states and operators live in Hilbert spaces.

Mixed states are represented by so-called density matrices. As in the classical case, mixed states, and therefore density matrices, are convex combinations of pure states. Since a pure quantum state of dimension \(N\) is already a linear combination of basis vectors, density matrices are expressed in a space of dimension \(N^2\). The state which is \(|\psi_i\rangle\) with probability \(p_i\), \(i \in 0 \ldots N - 1\) is given by the matrix

\[
\rho \equiv \sum_{i=0}^{N-1} p_i |\psi_i\rangle \langle \psi_i |.
\] (2.5)

These matrices are positive and have trace one. Note that states of classical systems can be represented by diagonal density matrices, as these are equivalent to vectors of probabilities.

From now on, if a quantum system lives in a Hilbert space \(H\), and if a density matrix \(\rho\) describes the state of that system, we will write \(\rho^H\). \(H\) can be taken to be a label of either the Hilbert space itself, or of the system that lives in that space. This notation can be extended to
operators that act on particular systems. For example if a unitary, $U$, acts on systems $H_1$ and $H_2$, we can write $U^{H_1 H_2}$. We will omit writing such a superscript if the state space of $\rho$ is clear. We will also write $|H|$ to indicate the dimension of the Hilbert space $H$.

### 2.2.1.1 Qubits

A qubit is the quantum analog of a classical bit. It is a two-state superposition which can be written as

$$|\psi\rangle = a|0\rangle + b|1\rangle,$$

where $a$ and $b$ are complex numbers and $|a|^2 + |b|^2 = 1$. It is often convenient to express $|\psi\rangle$ in the form

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle,$$

as $\theta$ and $\phi$ can be interpreted as polar and azimuthal angles in a spherical coordinate system where the states $|0\rangle$ and $|1\rangle$ are antipodal along the $z$-axis. This geometrical representation is known as the Bloch sphere. Later, we will speak of rotations around the axes of this sphere.

### 2.2.2 Composite Systems

Composite quantum states can be formed via the tensor product of individual states. That is, if we have $n$ individual states, $\rho^{H_1}, \rho^{H_2}, \ldots, \rho^{H_n}$, a joint (product) state is given by $\rho^{H_1} \otimes \rho^{H_2} \ldots \otimes \rho^{H_n}$. The tensor product of two pure states written in bra-ket notation, $|\psi\rangle$ and $|\phi\rangle$, is $|\psi\rangle|\phi\rangle$.

While this may seem foreign, the tensor product can also be used to combine classical states. Indeed, the state of size $N = 2^n$ of $n$ classical bits (2.2) is simply the tensor product of the states of the $n$ individual bits.

Note, however, that there exist composite quantum states that cannot be written as products of their constituent substates. In particular, an $M$ state system is called entangled if it cannot be written in the form

$$\sum_i p_i \rho_i^{H_1} \otimes \rho_i^{H_2} \ldots \otimes \rho_i^{H_M}$$

(2.8)
We will often be interested in computing the density matrix describing the state of a subsystem given the density matrix describing the state of a composite system. We do so via the partial trace, defined as the unique operation on states of a composite system that acts on product states as

$$\text{tr}_{H_2}(\rho_{H_1 H_2}) = \text{tr}(\rho_{H_2})\rho_{H_1}.$$  \hspace{1cm} (2.9)

where $H_1 H_2$ refers to the composite Hilbert space of $H_1$ and $H_2$. Let the set of states $|i\rangle$ where \{i $\in$ 0 . . . N\} form an orthonormal basis for Hilbert space $H_1$, and the set of states $|x\rangle$ where \{x $\in$ 0 . . . M\} form an orthonormal basis for $H_2$. Throughout, primed labels are taken to be members of the same orthonormal basis as the corresponding unprimed kets. Then a general density matrix of two systems can be expressed as

$$\rho_{H_1 H_2} = \sum_{i, i'} \sum_{x, x'} p_{i i' x x'} |i\rangle\langle i'| \otimes |x\rangle\langle x'|,$$  \hspace{1cm} (2.10)

and the partial trace can be computed as

$$\text{tr}_{H_2}(\rho_{H_1 H_2}) = \sum_{i, i'} \sum_{x, x'} p_{i i' x x'} |i\rangle\langle i'| \otimes \text{tr}|x\rangle\langle x'|)$$

$$= \sum_{i, i'} \sum_{x, x'} p_{i i' x x'} |i\rangle\langle i'| \otimes \sum_{x''} \langle x''|x\rangle\langle x'|x''\rangle$$

$$= \sum_{i, i'} \sum_{x, x'} p_{i i' x x'} |i\rangle\langle i'| \delta_{x'' x} \delta_{x' x''}$$

$$= \sum_{i, i'} \sum_{x} p_{i i' x x} |i\rangle\langle i'|.$$  \hspace{1cm} (2.11)

In the second line of equation (2.11), $\sum_{x''} \langle x''|x\rangle\langle x'|x''\rangle$ is used to sum over the diagonal entries of the matrix $\sum_{x, x'} p_{i i' x x'} |x\rangle\langle x'|$. Note that it is not necessarily true that $\text{tr}_{H_1}(\rho_{H_1 H_2}) \otimes \text{tr}_{H_2}(\rho_{H_1 H_2}) = \rho_{H_1 H_2}$ as $\rho_{H_1 H_2}$ may be entangled.

\subsection*{2.2.3 Evolution}

In this section, we describe how a quantum state changes in time. We begin by discussing how isolated states, or closed systems, evolve. Then we generalize and show how the evolution of an isolated, composite system is related to the evolution of its subsystems.
2.2.3.1 Closed Systems

In quantum mechanics, transformations between states are given by unitary operators, \( |\psi'\rangle = U|\psi\rangle \) In terms of density matrices, if \( |\psi'\rangle = U|\psi\rangle \), and \( \rho = |\psi\rangle\langle\psi| \), then \( \rho' = U\rho U\dagger \). Unitary operators are defined by the requirement that \( U\dagger U = I \), where \( I \) is the identity matrix. This implies that \( U\dagger U|\psi\rangle = |\psi\rangle \); we therefore say that quantum mechanical computation is reversible, as the effect of any unitary can be undone by its transpose. While stochastic matrices preserve the one-norm of a state, unitary matrices preserve the two-norm as required by classical to quantum adjustment 2) above.

Unitary operators evolve states in discrete temporal jumps. Consider the operator \( U(t_1 \rightarrow t_2) \) that evolves the state \( |\psi\rangle \) from some time \( t_1 \) to some later time \( t_2 \). Often this operator is expressed in terms of another operator, the Hamiltonian \( H \)

\[
U(t_1 \rightarrow t_2) = e^{-\frac{i}{\hbar}H(t_2-t_1)},
\]

(2.12)

where \( \hbar \) is Plank’s constant divided by \( 2\pi \). If \( U \) is unitary, \( H \) must be Hermitian. It can be shown that \( H \) and \( |\psi\rangle \) satisfy the famous Schrodinger equation

\[
i\hbar \frac{\partial}{\partial t}|\psi\rangle = H|\psi\rangle.
\]

(2.13)

In contrast to discrete unitary evolution, this differential equation describes infinitesimal temporal changes in \( |\psi\rangle \).

More traditional presentations of quantum mechanics begin by using experimental evidence to assert equation (2.13). Discrete unitary evolution and equation (2.12) then follow by solving the differential equation.

By analyzing the units of equation (2.13), or alternatively, by observing that \( H \) is the generator of time translations, we can conclude that the eigenvalues of \( H \) must have units of energy. Then if \( U \) acts on an energy eigenstate \( |E\rangle \), we have

\[
U(t_1 \rightarrow t_2)|E\rangle = e^{-\frac{i}{\hbar}H(t_2-t_1)}|E\rangle \\
= e^{-\frac{i}{\hbar}E(t_2-t_1)}|E\rangle.
\]

(2.14)
Thus, the action of $U$ alters the phase of $|E\rangle$ in a way that is proportional to both the energy of the state and the time over which it acts. This observation will be critical to our discussion of timekeeping with atomic clocks later.

### 2.2.3.2 Composite States

We now explore the effect of the unitary evolution of a composite state on its substates. Without loss of generality, assume that $\rho$ consists of two subsystems, one of which starts in the state $|0\rangle\langle 0|$. Then we have $\rho^{H_1H_2} = \rho^{H_1} \otimes |0\rangle\langle 0|$. Consider acting on $\rho$ with an arbitrary unitary and tracing out system $H_2$.

$$\text{tr}_{H_2} \left( U^H_1H_2 \rho^{H_1} \otimes |0\rangle\langle 0| U^\dagger_{H_1H_2} \right) = \sum_x \langle x| U^H_1H_2 |0\rangle \rho^{H_1} \langle 0| U^\dagger_{H_1H_2} |x\rangle = \sum_x E^H_{x} \rho^{H_1} E^\dagger_{x} \quad (2.15)$$

where the $|x\rangle$ form an orthonormal basis of $H_2$ and $E^H_x \equiv \langle x| U^H_1H_2 |0\rangle$. The $E_x$ are referred to as Kraus operators [56]. These operators satisfy

$$\sum_x E_x^\dagger E_x = I. \quad (2.16)$$

We abbreviate the complete action of $U^H_1H_2$ on $\rho^{H_1}$ in equation (2.15) as $\xi(\rho^{H_1})$. That is,

$$\xi(\rho^{H_1}) = \sum_x E_x^{H_1} \rho^{H_1} E_x^{\dagger H_1}. \quad (2.17)$$

$\xi$ is referred to as a quantum operation. It can be shown [70] that the following are satisfied if and only if $\xi$ acts as in equation (2.17) with Kraus operators that satisfy equation (2.16).

1. $\text{tr}(\xi(\rho)) = 1$

2. $\xi$ is a convex-linear map on the set of density matrices

3. $\xi$ is completely positive

Note that according to item one above, $\xi$ preserves the trace of $\rho$. In section 2.2.4, we consider non-trace-preserving quantum operations where $\text{tr}(\xi(\rho)) \leq 1$. 
The discussion above demonstrates that unitary evolution of a product state where subsystem two is in a fixed state, is equivalent to a quantum operation on the state of subsystem one. These subsystems are often referred to as open systems, as they evolve not in isolation, but in tandem with some secondary system. Often this secondary system is taken to be some external environment. Quantum operations are the most general type of evolution in quantum mechanics.

### 2.2.3.3 Useful Unitaries

A few unitary operators will be especially useful in what follows. First, consider the Hadamard transform defined as

\[
H = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}.
\]

(2.18)

\(H\) acts as

\[H|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle),\]

(2.19)

and

\[H|1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).
\]

(2.20)

For convenience, we define \(|+\rangle \equiv \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)\) and \(|-\rangle \equiv \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)\) so that \(H|0\rangle = |+\rangle\) and \(H|1\rangle = |-\rangle\).

Second, consider the following three operators

\[R_x(\theta) \equiv e^{-i\sigma_x \theta} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i \sin(\frac{\theta}{2}) \\ -i \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},\]

(2.21)

\[R_y(\theta) \equiv e^{-i\sigma_y \theta} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},\]

(2.22)

\[R_z(\phi) \equiv e^{-i\sigma_z \phi} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.
\]

(2.23)

These operators each rotate a qubit around a particular axis of the Bloch sphere by the given angle.
The $\sigma$ are the Pauli matrices defined as

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

(2.24)

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

(2.25)

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

(2.26)

2.2.4 Measurement

As above, we will describe quantum measurement on isolated systems and then generalize to composite systems. A measurement of an isolated system corresponds to a projective measurement, whereas a more general measurement, known as a positive operator valued measure (POVM) can be implemented by adjoining a supplementary quantum system, known as an ancilla, to the system of interest, and measuring the joint state.

2.2.4.1 Projective Measurements

To better understand quantum measurement, first consider a measurement of a classical mixed state, such as that in equation (2.2). If we measure the values of the $n$ registers that comprise this state, measurement outcome $i$, corresponding to the state $\vec{s}$ being in the state $\vec{s}_i$, is obtained with probability $p_i$. It is instructive to construct projectors onto classical states [44]. Define $Q_i$ as the operator that projects onto state $\vec{s}_i$. The operator $Q_i$ could be expressed as a vector with a 1 in position $i$ and 0’s elsewhere. Then we have $p_i = Q_i \cdot \vec{s}$. We can say that a set of such projectors corresponds to a measurement. That is, if we measure $\vec{s}$ with the set of projectors $\{Q_i\}_i$, we obtain outcome $i$ with probability $p_i$. After the measurement, the system is left in state (or arguably, always was in the initially unknown state) $\vec{s}_i$.

We can define a larger class of projectors by taking linear combinations of the $Q_i$. For
example, we could define $R_{\text{even}} = Q_0 + Q_2 + \ldots$ and $R_{\text{odd}} = Q_1 + Q_3 + \ldots$. A measurement with this set of projectors will produce outcome “even” with probability $p_0 + p_2 + \ldots$ and outcome “odd” with probability $p_1 + p_3 + \ldots$. Note that the probability that we obtain some outcome (in this case “even” or “odd”) must still sum to 1. Now, if we obtain outcome “even”, for example, the post measurement state will be

$$\vec{s} = \begin{pmatrix} p_0 \\ 0 \\ p_2 \\ \vdots \end{pmatrix}$$

which must then be suitably normalized so that its 1-norm is 1.

In quantum mechanics we construct a measurement out of a set projectors in a similar way. A set of projectors, $\{\Pi_i\}$ which correspond to a quantum measurement must satisfy:

1. Orthogonality: $\Pi_i^\dagger \Pi_j = \delta_{ij} \Pi_i$. Therefore, $\Pi_i^2 = \Pi_i$. This defines a projector and implies that its effect is independent of the number of times it is applied.

2. Normalization: $\sum_i \Pi_i^\dagger \Pi_i = I$. This is a consequence of the requirement $\sum_i p_i = 1$.

The probability of obtaining outcome $i$ after a measurement of $|\psi\rangle$ is given by

$$p_i = |\Pi_i |\psi\rangle|^2 = \langle \psi | \Pi_i^\dagger \Pi_i |\psi\rangle.$$  (2.28)

or, expressed in terms of density matrices,

$$p_i = \text{tr}(\Pi_i^\dagger \Pi_i \rho).$$  (2.29)

If outcome $i$ is obtained, then the post measurement state will be

$$\frac{\Pi_i |\psi\rangle}{\sqrt{p_i}},$$  (2.30)

or in terms of $\rho$

$$\frac{\Pi_i \rho \Pi_i^\dagger}{p_i}.$$  (2.31)
The denominators are normalization factors. Note that equations (2.28) and (2.30) immediately imply that the overall phase of $|\psi\rangle$ cannot affect the outcome of a measurement.

Any observable in quantum mechanics corresponds to a Hermitian operator. Physically, a set of projectors is built from the spectral decomposition of this operator, $\sum_i \lambda_i \Pi_i$, where $\Pi_i$ is a projector onto the $i$th eigenspace. If after a measurement with this set of projectors, outcome $i$ is obtained, then we say that the measured value of the observable is equal to the operator’s $i$th eigenvalue, $\lambda_i$.

### 2.2.4.2 POVMs

We now generalize this concept by adjoining an ancilla, $|0\rangle\langle0|^H_2$, to the pure state of interest, $\rho^{H_1}$, and allowing the joint state $\rho^{H_1H_2} = |\psi\rangle\langle\psi|^H_1 \otimes |0\rangle\langle0|^H_2$ to evolve via a unitary transformation, $U^{H_1H_2}$, just before the measurement. Let $U$ act as

$$U|\psi\rangle|0\rangle = \sum_{im} E^{H_2}_{im} |\psi\rangle |i,m\rangle,$$

(2.32)

where the complete set of $|i,m\rangle$ form a basis for $H_2$. Measure $H_2$ with the set of projectors

$$\Pi_i^{H_2} = \sum_m |i,m\rangle \langle i,m|.$$

(2.33)
After the measurement, if outcome \( i \) is obtained, the state of system \( H_1 \) becomes

\[
\text{tr}_{H_2} \left( \Pi_i \Pi_i U \rho \otimes |0\rangle \langle 0| U^\dagger \right)
\]

\[
= \sum_{i' m'} \sum_{i'' m''} \text{tr}_{H_2} \left( E_{i' m'} |\psi\rangle \langle \psi| E_{i'' m''}^\dagger \Pi_i |i', m'\rangle \langle i'', m''| \Pi_i^\dagger \right)
\]

\[
= \sum_{i' m'} \sum_{i'' m''} \sum_{m} \sum_{m''} \text{tr}_{H_2} \left( E_{i' m'} |\psi\rangle \langle \psi| E_{i'' m''}^\dagger |i, m\rangle \langle i, m'| i', m'\rangle \langle i', m'| i, m''\rangle \langle i, m''| i, m'''\rangle \right)
\]

\[
= \sum_{m} \sum_{m''} \text{tr}_{H_2} (E_{i m} |\psi\rangle \langle \psi| E_{i m''}^\dagger |i, m\rangle \langle i, m'| i, m''\rangle)
\]

\[
= \sum_{m} E_{i m} |\psi\rangle \langle \psi| E_{i m''}^\dagger \langle i, m| i, m''\rangle
\]

\[
= \xi_i (\rho).
\]

where each \( \xi_i \) is a non-trace-preserving quantum operation with Kraus operators \( E_{i m} \) that satisfy

\[
\sum_{x} E_{x}^\dagger E_x \leq I. \tag{2.34}
\]

The probability of obtaining outcome \( i \) is then

\[
p_i = \text{tr} (\xi_i (\rho)). \tag{2.35}
\]

Define the operator \( P_i \equiv \sum_{m} E_{i m} E_{i m}^\dagger \). We can then rewrite equation (2.35) as

\[
p_i = \text{tr} \left( \sum_{m} E_{i m} \rho E_{i m}^\dagger \right)
\]

\[
= \text{tr} \left( \sum_{m} E_{i m}^\dagger E_{i m} \rho \right)
\]

\[
= \text{tr} (P_i \rho). \tag{2.36}
\]
Since probabilities must sum to 1,

\[ 1 = \sum_i p_i \]
\[ = \sum_i \text{tr} (P_i \rho) \]
\[ = \text{tr} \left( \left( \sum_i P_i \right) \rho \right). \tag{2.37} \]

Since \( \text{tr}(\rho) = 1 \), we obtain \( \sum_i P_i = I \). The complete set \( \{P_i\}_i \) is referred to as a positive operator valued measure (POVM) and is the most general type of measurement possible in quantum mechanics. As demonstrated above, POVMs can be implemented by the introduction of an ancilla, unitary evolution, and projective measurement.

### 2.3 Useful Properties and Techniques

In this section we present but do not prove several standard, but important results in quantum computation. We begin with the Schmidt decomposition.

#### 2.3.1 Schmidt Decomposition

**Theorem 1.** Given a pure, joint state of two systems, \( |\psi\rangle^{H_1H_2} \), where \( |H_1| = N \) and \( |H_2| = M \), there exists a set of orthonormal states \( |\alpha_i\rangle \in H_1 \) where \( i \in \{1 \ldots N\} \) and \( |\beta_j\rangle \in H_2 \) where \( j \in \{1 \ldots M\} \) such that \( |\psi\rangle^{H_1H_2} \) can be written as

\[ |\psi\rangle^{H_1H_2} = \sum_k \lambda_k |\alpha_k\rangle^{H_1} |\beta_k\rangle^{H_2}, \tag{2.38} \]

where the coefficients \( \lambda_k \) are non-negative real numbers that sum to 1.

This is known as the Schmidt decomposition or Schmidt form of \( |\psi\rangle \). The proof of this theorem can be found in many places; see for example Ref. [70].

The Schmidt decomposition can be used in a process known as purification. Here, an arbitrary mixed state is converted into a pure state on a larger Hilbert space. Say we want to purify the
state $\rho^{H_1} = \sum_i p_i |\alpha_i\rangle\langle \alpha_i|$. We simply introduce a second set of orthonormal states $|\beta_i\rangle^{H_2}$ and write the new pure state in Schmidt form

$$|\psi\rangle^{AB} = \sum_i \sqrt{p_i} |\alpha_i\rangle^{H_1} |\beta_i\rangle^{H_2}. \quad (2.39)$$

Here we must have $|H_1| \leq |H_2|$. Tracing out $H_2$ from $|\psi\rangle\langle \psi|$ yields $\rho^{H_1}$ as desired.

### 2.3.2 Unitary Freedom

We now turn to an important theorem regarding the construction of density matrices. The convex sum of states yielding a given density matrix is not unique. Specifically, given $\rho = \sum_i p_i |\psi_i\rangle\langle \psi_i|$ and $\rho' = \sum_i q_i |\phi_i\rangle\langle \phi_i|$, we may have $\rho = \rho'$. While the two ensembles $\{|\psi_i\rangle, p_i\}_i$ and $\{|\phi_i\rangle, q_i\}_i$ correspond to identical density matrices, we can imagine that they result from two distinct physical processes, i.e. the first ensemble results from some process that prepares the state $|\psi_i\rangle$ with probability $p_i$, while the second results from a process that prepares $|\phi_i\rangle$ with probability $q_i$. The following theorem describes when two such ensembles correspond to the same quantum state.

**Theorem 2.** The two density matrices $\sum_i p_i |\psi_i\rangle\langle \psi_i|$ and $\sum_j q_j |\phi_j\rangle\langle \phi_j|$ are equal if and only if

$$\sqrt{p_i} |\psi_i\rangle = \sum_j U_{ij} \sqrt{q_j} |\phi_j\rangle. \quad (2.40)$$

where the $U_{ij}$ are entries of a unitary matrix.

The proof of this theorem can again be found in a number of sources including Ref. [70].

### 2.4 Quantum Algorithms

#### 2.4.1 Search

Now that we have laid out the differences between classical probability and quantum mechanics, we look at how quantum effects can be utilized in computation. We begin by looking at a ubiquitous problem in computer science, SEARCH, and progress from a classical algorithm to a
fully quantum one. This problem will be used as an example to better illustrate complex concepts throughout this chapter and the next.

In the canonical SEARCH problem, we are given a list of items, and seek to find a particular, marked item within this list. We represent this list by a binary function $f(i) : \mathbb{Z} \to 0, 1$ where $i$ behaves as an index into the list. We assume that $f(i) = 0$ at all $i$ except one, $i_0$, where $f(i_0) = 1$. We seek the value of the unknown index $i_0$. We give our computer one extra bit (or qubit) of memory, $b$, which we use to hold the result of a function evaluation.

### 2.4.1.1 Classical Search

Classically, if all relevant $f$ are equally likely, an optimal algorithm for $N$ element SEARCH is as follows:

$$i_0 = \text{SEARCH}(f)$$

$$i = 0$$

$$N = \text{SizeOf}(f) \quad //\text{The “length” of the list represented by } f$$

while $i < N - 1$:

$$b = f(i)$$

if $b = 1$:

return $i$

else:

$i = i + 1$

return $N$

That is, if all $f$ are equally likely, we cannot do better than advancing down the list one element at a time. This algorithm requires at most $N - 1$ function evaluations. Determining how the number of function evaluations, or more generally, the amount of time needed to solve a problem, scales with the size of the problem’s input will be important later.
To make better contact with quantum mechanics, we randomize this algorithm. Now, instead of progressing through the list linearly, we check an index value \( i \) at random. If \( f(i) = 1 \) we are done; if not, we choose a different, unchecked value of \( i \) at random.

The randomized search algorithm can be expressed in terms of state vectors and evolution via stochastic matrices. To see how this works, consider the concrete case \( N = 4 \). Since \( i \) is selected at random, the initial state of the computer is the classically mixed state

\[
\bar{s}(0) = \sum_{i=0}^{3} \frac{1}{4} \bar{s}_i \otimes \bar{b}_0 = \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 \end{pmatrix}.
\]

(2.41)

Each value of \( i \) corresponds to a \( \bar{s}_i \); since \( i \) is selected uniformly at random, we obtain each \( \bar{s}_i \) with probability \( \frac{1}{4} \). To store the result of a function evaluation \( \bar{b} \) is used, as in the deterministic algorithm above; this is necessary as the effect of future operations should depend on the evaluation of \( f \). Notice that \( \bar{b} \) can only take on two values, and its state space is therefore spanned by \( \bar{b}_0 = (1,0)^T \) and \( \bar{b}_1 = (0,1)^T \).
After a function evaluation, \( b_0 \to b_1 \) for \( i \) where \( f(i) = 1 \). Equivalently, \( \bar{s}(0) \) becomes

\[
\bar{s}'(0) = \begin{pmatrix}
\frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4}
\end{pmatrix}
\begin{pmatrix}
1 \oplus f(0) \\ 0 \oplus f(0) \\ 1 \oplus f(1) \\ 0 \oplus f(1) \\ 1 \oplus f(2) \\ 0 \oplus f(2) \\ 1 \oplus f(3) \\ 0 \oplus f(3)
\end{pmatrix}.
\]

Note that the state \( \bar{s}'(0) \) does not correspond to an evaluation of \( f \) in some quantum superposition. It is mixed state where each \( f(i) \) is evaluated with probability \( p_i = \frac{1}{4} \).

We can then apply a stochastic matrix, \( O \). Since our state is now of dimension 8, \( O \) will be \( 8 \times 8 \). It will be designed to implement the classical, randomized algorithm outlined above. To build such a matrix, note that entry \( k \) of \( \bar{s}(0) \) for \( k \) even corresponds to the probability that \( i = \frac{k}{2} \) and \( f(i) = 0 \), whereas entry \( k + 1 \) corresponds to the probability that \( i = \frac{k}{2} \) and \( f(i) = 1 \). Recall now that entry \((i,j)\) of a stochastic matrix gives the probability of transitioning from state \( j \) to state \( i \). We do not wish to transition away from any odd numbered states, since these correspond to successfully finding the marked element. Therefore, for \( j \) odd, \( O_{[0,7],j} \), where \([a,b]\) refers to matrix indices \( a \) through \( b \), will consist of all 0’s, except for a 1 in position \( j,j \). For \( j \) even, we transition to another state chosen uniformly at random; \( O_{[0,7],j} \) will therefore be \( 1/3 \) at all even \( j \) where \( i \neq j \).
and will be 0 elsewhere. That is, $O$ is the matrix

$$O = \begin{pmatrix}
0 & 0 & 1/3 & 0 & 1/3 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
1/3 & 0 & 1/3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.$$  \(2.43\)

Say now that $i_0 = 2$, i.e. the marked element is in position 2. After one function evaluation and one application of $O$ we have

$$\vec{s}(1) = O\vec{s}(0) = \begin{pmatrix}
1/6 & 0 & 1/6 & 0 & 1/4 & 1/6 & 0
\end{pmatrix}^T.$$  \(2.44\)

If this state is measured, there is a 50% chance of obtaining $i_0 = 2$ and a 50% probability of obtaining some other index which does not contain the marked item. This probability can be understood by thinking of this algorithm as a two step procedure. First, $f$ is evaluated at a particular index, $i$, chosen at random. If $i = i_0$ the algorithm is successful. If $i \neq i_0$, the algorithm can simply declare that the marked item is at some other unchecked index $j$. Therefore, the algorithm is correct if the marked item is at either $i$ or $j$ and obtains a 50% probability of success. This is the best we can do after a single function evaluation.

To find $i_0$ with certainty, the algorithm can continue in this manner, alternating function evaluations with applications of stochastic matrices. Note that the optimal stochastic matrix applied after the second function evaluation will look different, as we do not wish to revisit any checked indices. In fact, we will need to expand our state space in order to store which indices our algorithm has visited. Optimally, after a second evaluation, our probability of error drops to 25% as we will have covered three out of the four possible indices.
2.4.1.2 Quantum Search

We now illustrate how to solve \( N \)-element SEARCH quantumly using Grover’s algorithm \[42\]. Switching to Dirac notation, the initial state of the computer is \( \mathbf{s}(0) \to |\psi(0)\rangle = \sum_i \frac{1}{\sqrt{N}} |i\rangle |b_0\rangle \). All operations are now given by unitary matrices instead of stochastic matrices. As before, a function evaluation affects this state as

\[
|\psi(0)\rangle \to \sum_i \frac{1}{\sqrt{N}} |i\rangle |b_0 \oplus f(i)\rangle. \tag{2.45}
\]

Note that this looks similar to the classical state in equation (2.42); however, here this is treated as a pure state and not a probabilistic mixture. Therefore, in this case, the application of \( f \) is a nonclassical operation.

We alter our approach slightly, and rather than applying \( f \) as above, we begin by flipping \( b_0 \to b_1 \) and applying a Hadamard transform to this bit. Then, \( |\psi(0)\rangle \) becomes

\[
|\psi(0)\rangle = \sum_i \frac{1}{\sqrt{N}} |i\rangle \left( |0\rangle - |1\rangle \right). \tag{2.46}
\]

Now we apply \( f \) and obtain

\[
|\psi(0)\rangle' = \sum_i \frac{1}{\sqrt{N}} |i\rangle \left( |0 \oplus f(i)\rangle - |1 \oplus f(i)\rangle \right). \tag{2.47}
\]

Note that because \( f \) is now a quantum operation, \( f \) must be implemented reversibly. Its application as defined above is reversible as a second application yields

\[
|\psi(0)\rangle' \to \sum_i \frac{1}{\sqrt{N}} |i\rangle \left( |0 \oplus f(i) \oplus f(i)\rangle - |1 \oplus f(i) \oplus f(i)\rangle \right) \\
= \sum_i \frac{1}{\sqrt{N}} |i\rangle \left( |0\rangle - |1\rangle \right). \tag{2.48}
\]

Equation (2.47) is equivalent to

\[
|\psi(0)\rangle' = \sum_i \frac{1}{\sqrt{N}} |i\rangle (-1)^{f(i)} \left( |0\rangle - |1\rangle \right). \tag{2.49}
\]

The second system therefore remains unchanged as a result of the function evaluation, and we can think of the evaluation as simply altering the phase of \( |\psi\rangle \). This is known as phase kickback.
Afterwards, we apply the following unitary to the index register

\[ U = \frac{2}{N} \left( \sum_i \sum_{i'} |i\rangle \langle i'| \right) - I. \tag{2.50} \]

This operation is known as “inversion about the mean”, as it flips each individual amplitude about the mean amplitude.

Let’s see how this works for \( N = 4 \) with, once again, \( i_0 = 2 \). We begin with

\[ |\psi(0)\rangle = \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{pmatrix} \otimes \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}. \tag{2.51} \]

An application of \( f \) results in the state

\[ |\psi(0)\rangle = \begin{pmatrix} 1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{pmatrix} \otimes \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}. \tag{2.52} \]

We then apply the inversion about the mean operation. For a \( 4 \times 4 \) system, \( U \) looks like

\[ U = \begin{pmatrix} -1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 & -1/2 \end{pmatrix}. \tag{2.53} \]

which, when applied to \( |\psi(0)\rangle \), results in

\[ |\psi(1)\rangle = U |\psi(0)\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}. \tag{2.54} \]
A measurement of $|\psi(1)\rangle$ yields $i_0 = 2$ with certainty. For $N > 4$, we would iterate this procedure, alternating function evaluations with applications of $U$. In general, with a quantum algorithm, we can solve SEARCH with only $\sqrt{N}$ function evaluations, whereas we needed $N - 1$ classically.

Simplistically, we can think of this apparent quantum speedup as resulting from so called quantum interference. Because amplitudes can be negative, unlike classical probabilities, they can interfere and cancel in useful ways. The unitary operator $U$ is chosen to implement this interference in a specific way, such that undesirable answers interfere destructively and cancel, whereas correct answers interfere constructively. Manipulating quantum interference in this manner is the essence of quantum-algorithm design.

**2.4.2 Other Quantum Speedups**

While the quantum speedup in SEARCH is significant, it is only quadratic, and pales in comparison to what is possible in other problems. It is outside the scope of this thesis to discuss these problems at length; however, we briefly note a few problems of particular importance.

Shor’s algorithm for factoring integers [87] was perhaps the original impetus for more extensive research in quantum algorithms and quantum information. It offers a super-polynomial speedup over the best known classical algorithm. While this problem arguably has no physical significance, the difficulty of factoring is at the core of RSA (Rivest, Shamir, Adleman) encryption [78] - a public key cryptography protocol that is ubiquitous in internet security and other areas.

Quantum factoring and many other important quantum algorithms can be expressed in terms of phase estimation [21]. The goal of phase estimation is to determine a particular eigenvalue of a unitary operator. (All eigenvalues of unitaries have magnitude 1 and are therefore simply phase factors.) The algorithm at the heart of our clock optimization procedure is related, but not precisely equivalent to, this problem.

Finally, one of the original and perhaps also one of the most important proposed applications of quantum computers is quantum simulation [35]. Large quantum systems are impossible to simulate on classical computers due to the exponential time and space needed to store and manipulate
quantum states. A quantum computer, however, would allow us to simulate any quantum system efficiently, and may therefore find application in many areas where precision modeling of atomic systems is required.

2.4.3 Query Model of Computation

In section 2.4.1, we inserted function evaluations in between stochastic and unitary operations in our classical and quantum algorithms, respectively. We did not discuss how such a function would be implemented. In SEARCH, this function was simple, and so it is perhaps plausible that it could be implemented easily on a classical or quantum computer. However, it will be useful to encapsulate arbitrarily complex computations in a similar manner. We refer to these computations as “oracles”, the application of such an oracle as a “query”, and the use of such a paradigm as the “query model of computation”.

More concretely, we allow an oracle to compute an arbitrary function $f(x_1, x_2, \ldots, x_n) : \{0,1\}^n \rightarrow \{0,1\}^m$. We would like a systematic way of implementing such oracles on a quantum computer. We begin by implementing $f$ classically via a set of Toffoli gates. A Toffoli gate has a three bit input and a three bit output and, written as a stochastic matrix, takes the form

$$
\mathbf{Y} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}.
$$

(2.55)

Thus we see, for example, from the 1 in entry $Y_{6,7}$, that the Toffoli gate encodes the transition $111 \rightarrow 110$ with probability 1. Toffoli gates are universal for reversible classical computation, and therefore, an appropriate series of such operations can implement any function $f$ in a reversible
way.

Notice that $\Upsilon$, in addition to being stochastic, is also unitary, and is therefore a valid quantum operation. Transitions between quantum states are encoded in complete correspondence to those between classical states. For example, $\Upsilon_{6,7}$ now encodes the quantum transition $|111\rangle \rightarrow |110\rangle$, instead of the classical transition $111 \rightarrow 110$. Thus, the same collection of operators used to implement $f$ classically can be used to implement $f$ on a quantum computer. (This may, of course, not be the most efficient way of implementing $f$.) The resulting collection of operators evaluates $f$ in superposition according to equation (2.45) as desired. Note that a more formal version of this argument can be used to show that quantum computers are strictly more powerful than classical computers as they can implement any classical computation in this manner.

From now on, we designate the product of matrices that represent an oracle by the operator $\Omega$. Throughout, we will only be interested in the final form of $\Omega$ and not the sequence of operators used to construct it.

In Grover’s algorithm, the oracle was designed to index into a binary list. This oracle, written using the phase kickback trick, looks like

$$
\Omega = \begin{pmatrix}
(-1)^{f(0)} & 0 & 0 & \ldots \\
0 & (-1)^{f(1)} & 0 & \ldots \\
0 & 0 & (-1)^{f(3)} & \ldots \\
\vdots & \vdots & & & \\
\end{pmatrix}
\otimes \begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}.
$$

(2.56)

Clearly, $\Omega$, depends on $f$. From now on, we assume that $f$ is chosen from some indexed set, $\{f_x\}_x$. The index $x$ may correspond to $f$ is some meaningful way. For example, in SEARCH, the $f$ corresponding to $i_0 = 2$ can be given index $x = 2$. The oracle corresponding to $f_x$ is written as $\Omega(x)$. Often, and in all cases discussed in chapter 3, $x$ is a discrete label. In chapter 5 we consider oracles with continuous labels. The function $f_x$ and its label $x$ will be referred to as the algorithm’s “input”. Hereafter, the word “oracle” refers specifically to the operator $\Omega(x)$ that depends on this input. To clarify, we can think of an input as some data (e.g. a list in SEARCH) that an algorithm is given, while an oracle corresponds to an operation that the algorithm executes based on that
data (e.g. accessing an element in that list).

Grover’s algorithm can now be expressed as a product of matrices,

\[ \vec{s} = |\psi\rangle \rightarrow U_M \Omega(x) U_{M-1} \ldots \Omega(x) U_0 |\psi\rangle. \] (2.57)

The classical algorithm can be written in the same form, \( \vec{s} \rightarrow O_M \Omega(x) O_{M-1} \ldots \Omega(x) O_0 s(0) \). Of course, in this case, \( \Omega \) is a classical operation. Notice that in this form, we have clearly separated the oracle, which has access to the input, from the piece of the algorithm that doesn’t. This is useful as often, the details of \( \Omega \) are not of theoretical interest; rather, we are interested in designing an algorithm given a particular set of oracles \( \{\Omega(x)\}_x \). For example, in SEARCH, we do not care about how a particular element is extracted from a given list; we care only about how to efficiently find the marked element given a set of potential lists. Therefore, in this context, our goal is to choose the \( U_j \) as well as possible. The \( U_j \) cannot depend on \( x \) as the algorithm must work for all inputs, as we do not know which input it will be given. As we will see, this formalism is still useful even if \( \Omega \) is more complicated and does more than access a list.

### 2.5 Computational Complexity

Determining the resources needed to solve problems is of utmost importance in classical and quantum computer science. In particular, given a problem \( P \), we are often interested in analyzing how the time and space required to solve \( P \) scales with the size \( n \) of \( P \)’s input. (Note that “size” here refers to the size of the input’s domain, not to the number of potential inputs.) Generally, we are concerned only with the functional form of this scaling and not with any associated constants, as such constants are often negligible in the limit of large \( n \).

Formally, let the function \( f \) correspond to the quantity of a particular resource (e.g. time, space) needed to solve \( P \). This function naturally depends on the size of \( P \)’s input. We say that \( f = O(g) \) if there are positive constants \( c_u \) and \( k \) such that \( 0 \leq f(n) \leq c_u g(n) \) for all \( n \geq k \). That is, \( f = O(g) \) if asymptotically, the function \( g \) is larger than the function \( f \). Similarly, we say that \( f = \Omega(g) \) if there are positive constants \( c_l \) and \( k \) such that \( 0 \leq c_l g(n) \leq f(n) \) for all \( n \geq k \). If
\( f = O(g) \) and \( f = \Omega(g) \); that is, if \( 0 \leq c_l g(n) \leq f(n) \leq c_u g(n) \) then \( f(n) = \Theta(g) \). This is generally referred to as “big-O” notation.

Indicating that a problem \( P \) requires \( O(g) \) of some resource is equivalent to providing an upper bound. In practice, we can obtain this upper bound by describing an algorithm that solves \( P \) using at most \( g \) of a particular resource. This does not preclude the existence of a more efficient algorithm that solves \( P \) using fewer resources. A lower bound, \( \Omega(h) \), is often obtained via a sophisticated analysis of the problem itself. The adversary method is one well known method for performing such an analysis and deriving lower bounds; the quantum adversary method will be discussed at length in chapter 3.

In the query model of computation, we are often interested in analyzing the minimum number of queries to an oracle required to solve a given problem. Here, oracle queries are, in a sense, used as a proxy for time, and we assign one query one unit of complexity. For example, we say SEARCH has oracle complexity \( \Theta(N) \) classically and \( \Theta(\sqrt{N}) \) quantumly.

Problems are often grouped into classes by their time or space complexity. Extensive classical and quantum complexity hierarchies have been developed, and a great deal of research has been devoted to deriving relationships within and across these hierarchies [14].
Chapter 3

Generalizing the Adversary Method

In this chapter, we develop a generalization of the well known quantum adversary method [8]. To date, this has been the most successful method of lower bounding quantum query complexity and has been applied to many important problems [48]. Of greatest interest to us is the variant of this method known as the “spectral adversary method” [12], which formulates quantum query complexity in terms of a semidefinite program. We generalize this variant to include continuous problems, computation on noisy quantum computers, and computers with restricted quantum resources. We begin simply, by discussing the classical adversary method in the context of SEARCH.

3.1 Classical Lower Bounds

In chapter 2 we gave a classical algorithm for solving SEARCH that used $N - 1$ queries. We therefore say that SEARCH has query complexity $O(N)$. We wish now to show that this is optimal. To do so, we need a lower bound; that is, we wish to show that SEARCH also has query complexity $\Omega(N)$ - matching the upper bound to within a constant factor.

To this end, consider a malicious adversary [32] which has full access to an arbitrary algorithm for a particular problem. Based on an analysis of this algorithm, the adversary selects an input for which the algorithm will perform its worst; in the context of query complexity, the selected input maximizes the required number of queries. In this way, the adversary is used to establish worst case performance. By showing that such an adversary can stifle any algorithm for a particular problem
in this way, we can prove a lower bound.

Returning to SEARCH, consider all possible deterministic algorithms. All such algorithms query input elements in some predetermined order. An adversary can force each of these algorithms to query their input $N - 1$ times by analyzing this ordering and selecting the input where the marked element is in the last position queried. We therefore obtain a lower bound for deterministic SEARCH of $\Omega(N)$.

More generally, we can think of the adversary as giving our algorithm the worst possible probability distribution over inputs. For example, as above, deterministic algorithms for SEARCH are stifled by selecting the trivial probability distribution where the marked element is always in the last position checked and never elsewhere. Given this particular distribution, the algorithm succeeds with probability 0 after a single query. However, notice that such an adversary cannot do as well against the randomized algorithm for SEARCH discussed in section 2.4.1. This algorithm chooses which position to check at random; assuming that the algorithm’s random source is inaccessible to the adversary, it cannot choose a probability distribution for which the randomized algorithm will perform particularly poorly. Indeed, that algorithm will have a nonzero probability of success after a single query for any given distribution of inputs. Of course, depending on its random choices and the input given, the randomized algorithm may still need $N - 1$ queries; however, the point remains that no matter which input an adversary gives us, the randomized algorithm, on average, performs equally well. We therefore say that the randomized algorithm has superior worst-case expected running time.

We now formalize this method along the lines of Refs. [2, 57]. Here we restrict our attention to algorithms whose inputs are lists of individual bits. We denote the $i$th bit, or element, of input $x$ by $x(i)$. We consider functions $F$ over inputs; $F$ has a one-bit output, $F(x) : \{0, 1\}^N \rightarrow \{0, 1\}$. We think of such binary functions as “decision problems”, as they answer a yes/no question about a given input. For the purpose of deriving lower bounds, we can reformulate many interesting problems in this form after a suitable simplification. SEARCH, for example, can be reformulated from “Find the marked element” to “Is there a marked element?”. The latter problem is known as
OR, as it is equivalent to taking the logical OR of all the input bits. That is, we have \( F(x) = 1 \) if \( x \) contains one or more 1’s and \( F(x) = 0 \) if \( x \) contains only 0’s. Notice that the decision problem, OR, can be solved with less information about the input, and therefore, any lower bound derived for OR will apply to SEARCH. It is therefore reasonable to seek lower bounds on the number of queries needed to correctly compute such binary functions.

To compute these bounds, we divide the input into two sets, \( X = \{x_1, \ldots, x_M\} \) and \( Y = \{y_1, \ldots, y_{N-M}\} \), where \( F(x) \neq F(y) \) if \( x \in X \) and \( y \in Y \). The goal of any algorithm is then to decide whether it was given an input from \( X \) or \( Y \). Define the relation \( P = X \times Y \). \( P \) contains pairs of inputs \((x, y)\) which must differ at one or more positions. Given a particular pair \((x, y)\), because \( F(x) \neq F(y) \), if an algorithm does not know whether it was given \( x \) or \( y \) it risks returning the wrong answer. By querying a position at which \( x \) and \( y \) differ, the algorithm can exclude one of \( x \) or \( y \). For example, if \( x_i = 0 \) and \( y_i = 1 \), a query at position \( i \) which yields 0 implies that the algorithm was not given input \( y \) but may (or may not) have been given input \( x \). We say that such a query distinguishes \( x \) from \( y \). This query may have distinguished inputs in other pairs of \( P \) as well. Inputs in all pairs of \( P \) must be distinguished if an algorithm is to succeed with probability 1.

Consider a subset of this relation, \( R \subset P \). The lower bound to be determined depends on four quantities associated with \( R \):

1. \( m \): \( \min_x (|\{y|(x, y) \in R\}|) \) (The minimum, over all \( x \), of the number of \( y \)'s associated with \( x \).)

2. \( m' \): \( \min_y (|\{x|(x, y) \in R\}|) \) (The minimum, over all \( y \), of the number of \( x \)'s associated with \( y \).)

3. \( l \): \( \max_{x,i} (|\{y|(x, y) \in R \land x_i \neq y_i\}|) \) (The maximum, over all \( x \) and \( i \), of the number of \( y \)'s associated with \( x \) where \( x \) and \( y \) differ at position \( i \).)

4. \( l' \): \( \max_{y,i} (|\{x|(x, y) \in R \land x_i \neq y_i\}|) \) (The maximum, over all \( y \) and \( i \), of the number of \( x \)'s associated with \( y \) where \( x \) and \( y \) differ at position \( i \).)
\( l \) and \( l' \) quantify the maximum progress that can be made with a single query. That is, for a given \( x \) and \( i \), there are at most \( l \) pairs that are distinguished by a query at position \( i \); there are at most \( |X| \) \( x \)'s for which this query may be useful. The same argument applies for \( l' \). Therefore, classically, we cannot expect to distinguish more than either \( |X|l \) or \( |Y|l' \) pairs per query. Thus, we expect the number of queries required to evaluate \( F \) to go, heuristically, as \( \Omega(\frac{\text{\# of pairs we need to distinguish}}{\text{max pairs distinguished per query}}) \). The numerator is given by \( |R| \). We have \( |R| \geq \max(|X|m, |Y|m') \). Putting everything together, we can lower bound the number of queries required by \( \Omega(\max(\frac{m}{l}, \frac{m'}{l'})) \) [2].

Note that an algorithm may need to distinguish more input pairs than those in \( R \). However, regardless, it certainly must distinguish those in \( R \), and forcing the algorithm to do additional work (i.e. distinguishing additional pairs), cannot decrease the number of queries required. The choice of \( R \) affects the tightness of the resulting bound.

This method and the method described informally at the beginning of this section both construct lower bounds by demonstrating that any algorithm must do a great deal of work when given particular, “hard” inputs. Above, algorithmic performance was analyzed directly given an adversarially chosen probability distribution over such inputs. In the formal method discussed here, such inputs are added to \( R \), and it is shown that a large number of queries are required to distinguish these inputs from others. That is, while the structure of these two methods differ, many of the ideas are the same. In either case, and in other strategies used to derive lower bounds, an analysis of the underlying structure of a problem (i.e. formally, the function \( F \) and the set of potential inputs) is necessary.

### 3.2 Quantum Lower Bounds

Any quantum algorithm for evaluating \( F \) must still decide whether it was given an input from \( X \) or \( Y \). However, a quantum algorithm can potentially make more progress with each query. Such progress is now measured using inner products.

Consider the action of a quantum algorithm given input \( x \). At time \( t \), we write the associ-
ated quantum state as $|\psi_x(t)\rangle$. Now consider the same algorithm with input $y$, where as before, $F(x) \neq F(y)$. At $t = 0$, we must have $\langle \psi_x(0)|\psi_y(0)\rangle = 1$ since the initial state of the algorithm is independent of the input. If $F$ is evaluated correctly with probability 1 after $t_f$ queries, then we must have $\langle \psi_x(t_f)|\psi_y(t_f)\rangle = 0$, for all $(x, y) \in R$ since these states cannot overlap if a measurement is to return a definitive answer. From above, we know there are more than $\max(|X|m, |Y|m')$ pairs present in $R$. Therefore the sum over all possible inner products

$$P(t) = \sum_{(x, y) \in R} \langle \psi_x(t)|\psi_y(t)\rangle,$$

must go from $\max(|X|m, |Y|m') \geq \sqrt{m|X|m'|Y}$ at $t = 0$ to 0 at $t = t_f$. We expect the required number of queries to be $\Omega \left(\frac{P(t_f) - P(0)}{\max_i(P(i+1) - P(i))}\right)$. Ambainis [8] showed that $\max_i(P(i+1) - P(i))$ is given by $\sqrt{l|X|m'|Y}$ from which one can obtain a bound of $\Omega(\sqrt{mm'}).$

### 3.3 Modified Spectral Adversary Method

Since its introduction, the original quantum adversary method [8] has been reformulated and improved upon many times. While many of these methods do not obviously resemble one another, each, at its core, still establishes bounds by showing that certain inputs are hard to distinguish. We briefly discuss a few of these methods in section 3.4; here however, we are interested in a particular reformulation known as the “spectral adversary method,” due to Barnum, et. al. [12, 13]. It expresses the operation of a quantum query algorithm in terms of a feasible solution of a semidefinite program. We require several adaptations to Barnum’s original method, and our presentation here will be with these adaptations already in place. Later however, we will explicitly point out significant deviations from Refs. [12, 13]. We begin with a brief introduction to semidefinite programming.

#### 3.3.1 Semidefinite Programming

Semidefinite programming [36, 92] is a type of mathematical optimization involving positive semidefinite matrices. Specifically, a semidefinite program (SDP) consists of: 1) a set of matrices
constrained to be positive semidefinite, $M_i \geq 0$, with initially unknown entries, 2) a set of linear constraints over these entries, and 3) a linear objective to be minimized or maximized. One standard way of writing such a program is:

$$\begin{align*}
\text{Minimize } \sum_i \text{tr}(CM_i) \text{ subject to:} \\
\forall k, \sum_i \text{tr}(A_{ki}M_i) = b_k \\
\forall i, M_i \geq 0.
\end{align*}$$

The minimization is over the matrices $M_i$. The real numbers $b_k$ and the matrices $A_{ki}$ make up the chosen constraints, and the cost matrix $C$ is used to express the chosen objective function. If the matrices $M_i$ satisfy the constraints of the SDP, they comprise a “feasible” set. A solution is a feasible set that extremizes the objective function. Note that our formulation of the adversary method will not obviously resemble a program in this form; however, the two can be shown to be equivalent.

The field of semidefinite programming has been richly developed and going into further detail is beyond the scope of this thesis. This development has brought with it a wide array of tools and SDP solvers. Our results here were generated with the SDPT3 solver by Toh, Todd, and Tutuncu [89], and YALMIP by Lofberg [60], a tool designed to convert an SDP written using more elegant syntax into a standard form that common solvers can parse. We note in passing that the original spectral adversary method does not use a solver; rather it makes heavy use of the theory of duality and obtains analytical bounds. This is briefly discussed in section 3.3.5.

### 3.3.2 Operator Constraints

Our goal is to represent the execution of a quantum query algorithm by a semidefinite program. That is, we will develop a set of constraints that a sequence of quantum states must satisfy if they are those generated by the execution of a quantum algorithm, and an objective that quantifies the performance of such an algorithm. Recall that the execution of an algorithm is given by equation (2.57), or expressed in terms of the initial density matrix $\rho = U(0)|0\rangle\langle 0|U^{\dagger}(0)$,
\[ \rho' = \Omega(x)U(t_f)\ldots U(1)\Omega(x)\rho\Omega^\dagger(x)U^\dagger(1)\ldots U^\dagger(t_f)\Omega^\dagger(x), \] (3.2)

followed by a measurement of \( \rho' \) with a POVM. Define \( \rho(t) = \Omega(x)\rho\Omega^\dagger(x)U^\dagger(t_f)\Omega^\dagger(x)U(t_f) \). Then, we can think of equation (3.2) as defining a set of constraints on each \( \rho(t) \). Semidefinite programming is a very natural way to formulate these constraints - density matrices are defined to be positive semidefinite and the constraints corresponding equation (3.2) are necessarily linear in the entries of each \( \rho(t) \).

It is useful to treat \( \rho \) as a state of two systems, \( Q \) and \( A \). \( Q \) is the system where the oracle operators act and is known as the “querier” system. This is often the only system needed for an actual computation. For example, in Grover’s algorithm discussed in section 2.4.1, this was the only system considered. We would like to give the querier additional space however, beyond that used by the oracle operators. We therefore introduce a system of ancillas \( A \), and allow the \( U(t_i) \) to act on both \( Q \) and \( A \), whereas \( \Omega \) acts on \( Q \) only.

An SDP with constraints based on equation (3.2) finds \( \rho^{QA}(t) \) that minimize or maximize some chosen objective function. From these \( \rho^{QA}(t) \) we can extract the optimal algorithm, i.e. the unitaries, \( U(t_i)^{QA} \). However, in equation (3.2), the querier is always acted upon with the same oracle, \( \Omega(x)^{Q} \). Therefore, the algorithm extracted will be optimal only for this particular oracle, and the extracted algorithm will be trivial. Consider again SEARCH and an objective function which penalizes returning some element other than the marked element. If \( \rho^{QA} \) is always acted upon by the oracle that corresponds to the marked element being in position 2, an optimal algorithm is simply \textbf{return} 2. We therefore need a way to encode a probability distribution of oracles. That is, we need a way to say that oracle \( \Omega(x)^{Q} \) is applied with probability \( p(x) \). Our goal then, and indeed, our usual goal in the query model of computation, is to identify the applied oracle.

We encode a probability distribution over oracles via the introduction of an auxiliary quantum system, known as the “oracle” system, denoted by \( O \). Initially, the state of this system is given by

\[ \rho_0 = \sum_x \sum_y \sqrt{p(x)} \sqrt{p(y)} |x\rangle \langle y|. \]

By representing the oracle probabilities with a pure superposition,
we encode the fact that there is no information about $x$ available to the querier (or any system other than $O$). We define the fixed, joint operator $\Omega^{OQ} = \sum_x |x\rangle\langle x|\Omega^Q(x)$, which applies $\Omega(x)$ to system $Q$ conditional on the state $|x\rangle$ of the oracle. In this setting, a multi-query quantum computation is given by the composition

$$\rho^{OQA}(t_f) = \Omega^{OQ}U^{QA}(t_f)\ldots U^{QA}(1)\Omega^{OQ}\rho^{OQA}(0)\Omega^{IOQ}U^{QA}(1)\ldots U^{QA}(1)\Omega^{IOQ}, \quad (3.3)$$

followed by a measurement of $QA$ with a POVM. Here $\rho^{OQA}(0)$ is the initial state, with $\rho^O(0) = \rho_0$ pure as defined above, and $\rho^{QA}(0) \equiv U^{QA}(0)|0\rangle\langle 0|U^{QA}(0)^\dagger$ is a state chosen by the querier. While this formulation may appear rather abstract, observe that

$$\rho^Q(1) = \text{tr}_O(\Omega^{OQ}\rho(0)^{OQ}\Omega^{IOQ})$$

$$= \sum_x p(x)\Omega^Q(x)\rho^Q(0)\Omega^{I\!Q}(x). \quad (3.4)$$

Thus, we are simply encoding the fact that, as desired, oracle $x$ is applied to the querier with probability $p(x)$. Here, the joint states $\rho^{OQA}(t)$ remain pure, as they are acted on by unitaries, but the individual subsystem states, e.g., $\rho^{OQ}(t)$ may be mixed, as they undergo more general evolution (see e.g. section 2.2.3).

We will prove that equation (3.3) corresponds to the following set of semidefinite constraints:

**Definition 3.1: SDP $S_E$:**

$$\rho^O(0) = \rho_0, \quad \rho^{OQ}(0) \geq 0, \quad \rho^O(0) = \text{tr}_Q(\rho^{OQ}(0)),$$

and the following for $t \in \{1, \ldots, t_f\}$:

$$\rho^{OQ}(t) \geq 0,$$

$$\rho^O(t) = \text{tr}_Q(\rho^{OQ}(t)),$$

$$\rho^O(t) = \text{tr}_Q(\Omega^{OQ}\rho^{OQ}(t-1)\Omega^{IOQ}).$$

**Theorem 3.** Given a probability distribution over a set of oracles $\{\Omega(x)\}_x$, a quantum algorithm, consisting of a sequence of unitaries $U(t)$ which computes the final state $\rho_f^{OQA} = \rho^{OQA}(t_f)$ in equation (3.3), exists if and only if there is a feasible solution to $S_E \cup \left[ \rho^{OQ}(t_f) = \text{tr}_A(\rho_f^{OQA}) \right]$. 
**Proof.** We begin by proving the forward direction. Assume we have a quantum algorithm that acts as above. The only nontrivial constraints in \( S_E \) are those given by the last line. Consider the progression of the algorithm from step \( t \) to step \( t + 1 \),

\[
\rho^{OQA}(t + 1) = U^{QA}(t) \Omega^{OQ} \rho^{OQA}(t) \Omega^{OQ} U^{QA}(t)^\dagger.
\]  

(3.5)

The constraints are derived by simply tracing out \( QA \),

\[
\rho^O(t + 1) = \text{tr}_{QA} \left( U^{QA}(t) \Omega^{OQ} \rho^{OQA}(t) \Omega^{OQ} U^{QA}(t)^\dagger \right)
\]

\[
= \text{tr}_{QA} \left( U^{QA}(t) U^{QA}(t) \Omega^{OQ} \rho^{OQA}(t) \Omega^{OQ} \right)
\]

\[
= \text{tr}_Q \left( \Omega^{OQ} \rho^{OQ}(t) \Omega^{OQ} \right). 
\]  

(3.6)

To prove the reverse direction, we must extract inter-query unitaries, \( \{ U^{QA}(t) \}_t \) given a feasible solution \( \{ \rho^{OQ}(t) \}_t \) to the SDP. Notice that these density matrices are mixed states over systems \( O \) and \( Q \) only, since system \( A \) is completely traced out in each of the SDP’s constraints.

To extract these unitaries, first purify each \( \rho^{OQ}(t) \) into \( A \), \( \rho^{OQ}(t) \rightarrow \rho^{OQA}(t) = |\psi(t)\rangle\langle\psi(t)| \).

From the constraints of the SDP, we have for \( t \geq 1 \),

\[
\text{tr}_{QA} \left( \Omega \rho^{OQA}(t - 1) \Omega^\dagger \right)
\]

\[
= \text{tr}_Q \left( \rho^{OQ}(t - 1) \right)
\]

\[
= \rho^O(t)
\]

\[
= \text{tr}_{QA} \rho^{OQA}(t). 
\]  

(3.7)

where \( \rho^{OQ}(t - 1) = |\psi'(t - 1)\rangle\langle\psi'(t - 1)| = \Omega \rho^{OQ}(t - 1) \Omega^\dagger \). The second and third lines of this equation tell us that \( \rho^O(t - 1) \) and \( \rho^O(t) \) must be equal. Furthermore, since \( \rho^{OQA}(t - 1) \) and \( \rho^{OQA}(t) \) are pure, theorem 1 tell us that we can write \( |\psi(t)\rangle \) and \( |\psi'(t - 1)\rangle \) in Schmidt form as follows:

\[
|\psi(t)\rangle = \sum_i \lambda_i |\alpha_i^O\rangle |\beta_i^Q\rangle |\gamma_i^A\rangle
\]

\[
|\psi'(t - 1)\rangle = \sum_i \lambda_i |\alpha_i^O\rangle |\beta_i'^Q\rangle |\gamma_i'^A\rangle,
\]
Therefore the unitary $U^{QA}(t)$ that maps,

$$U^{QA}(t)\rho^{OQA}(t - 1)\Omega^{A}U^{QA}(t) = \rho^{OQA}(t),$$

is given by

$$U(t) = \sum_{i} |\beta_{i}\rangle^{Q}|\gamma_{i}\rangle^{AA}\langle\gamma_{i}'|^{Q}\langle\beta_{i}'|.$$ \hspace{1cm} (3.9)

### 3.3.3 Measurement Constraints

We now turn to the post-computation measurement. We seek constraints that correspond to the application of an arbitrary POVM. We construct these constraints via a connection to so-called “Remote state preparation.”

Recall from theorem 2 that a density matrix $\rho^{A}$ can be described by an infinite number of possible ensembles, each of which can be thought of as arising from different physical processes. Say now that we wish to realize a particular ensemble. That is, after a measurement we wish to obtain the state $|\psi_{i}\rangle^{A}$ with probability $p_{i}$. This can be accomplished by purifying $\rho^{A}$ to a joint state $\rho^{AB}$ and making a suitable measurement on system $B$. We call this procedure Remote state preparation and define it more precisely as follows (switching our labeling from $A \rightarrow O$ and $B \rightarrow Q$ in anticipation of how it will be used):

**Definition 3.2: Remote State Preparation** We say that we can remotely prepare $\{\sigma^{O}_{a}\}_{a}$, which for each $a$, is built from an ensemble $\{|\psi_{ai}\rangle, p_{ai}\}_{i}$, from the state $\rho^{OQ}$ of $O$ and $Q$ if there exists a POVM of $Q$ with operators $\{P^{O}_{a}\}_{a}$ such that for all $a$, $tr_{Q}(P^{O}_{a}\rho^{OQ}) = \sigma^{O}_{a}$.

Each $\sigma_{a}$ is a positive operator with trace $0 \leq tr(\sigma^{O}_{a}) \leq 1$ and consists of sub-ensembles of the full ensemble we wish to prepare. That is, if we wish to prepare $\rho^{O} = \sum_{ai}p_{ai}|\psi_{ai}\rangle\langle\psi_{ai}|$, then we have $\rho^{O} = \sum_{a}\sigma_{a}$ and $\sigma_{a} = \sum_{i}p_{ai}|\psi_{ai}\rangle\langle\psi_{ai}|$. If such an ensemble is measured, the state $|\psi_{ai}\rangle$ is
obtained with probability \( p_{ai} \). The normalized state \( \sigma_a^O / \text{tr}(\sigma_a^O) \) can be conditionally prepared with probability \( \text{tr}(\sigma_a^O) \) by means of a fixed POVM on \( Q \).

Theorem 4 characterizes the set of states that can be remotely prepared according to definition 3.2 from a pure state.

**Theorem 4.** Let \( \rho^{OQ} \) be a pure state of systems \( O \) and \( Q \). We can remotely prepare \( \{\sigma_a^O\}_a \) if and only if \( \sum_a \sigma_a^O = \rho^O \).

**Proof.** Assume that we can remotely prepare \( \{\sigma_a^O\}_a \) from \( \rho^{OQ} \), and let \( \{P_a^Q\}_a \) be the required POVM. Then \( \sum_a \sigma_a^O = \sum_a \text{tr}_Q P_a^Q \rho^{OQ} = \text{tr}_Q (\sum_a P_a) \rho^{OQ} = \rho^O \), by the definition of a POVM.

The converse is a generalization of the GHJW theorem [50] to mixed density operators. Suppose that \( \sum_a \sigma_a^O = \rho^O \). To construct the required POVM, first write each \( \sigma_a^O \) as an explicit mixture of pure states

\[
\sigma_a^O = \sum_m p_{am} |\psi_{am}\rangle \langle \psi_{am}|.
\]

By assumption, \( \rho^O = \sum_{am} p_{am} |\psi_{am}\rangle \langle \psi_{am}|. \) By filling in mixture terms with \( p_{am} = 0 \) if necessary, we can assume that the range of the index \( m \) is independent of \( a \). Since \( \rho^{OQ} = |\Phi\rangle \langle \Phi| \) is pure, we can write the \( OQ \) state in Schmidt form

\[
|\Phi\rangle = \sum_j \sqrt{q_j} |\phi_j\rangle |\varphi_j\rangle,
\]

where the \( |\phi_j\rangle \) and \( |\varphi_j\rangle \) are orthonormal in the Hilbert spaces of \( O \) and \( Q \), respectively. Thus \( \rho^O \) can also be written as the mixture \( \rho^O = \sum_j q_j |\phi_j\rangle \langle \phi_j| \). By theorem 2, these two ensembles are related by a unitary transformation,

\[
\sqrt{p_{am}} |\psi_{am}\rangle = \sum_j u_{am,j} \sqrt{q_j} |\phi_j\rangle,
\]

where the \( u_{am,j} \) are the entries of a unitary matrix. We can now define

\[
P_a^Q = \sum_m \sum_{j,j'} u_{am,j}^* u_{am,j'} |\varphi_j\rangle \langle \varphi_{j'}|.
\]
That $\sum_a P_a = I$ follows from the unitarity condition for $u_{am,j}$. To verify the partial trace condition, compute

$$\text{tr}_Q P_a \rho^{OQ} = \sum_m \sum_{j,j'} u_{am,j}^* u_{am,j'} \text{tr}_Q \left( |\varphi_j\rangle\langle \varphi_j| \sum_{l,l'} \sqrt{q_l q_{l'}} |\phi_l\rangle\langle \phi_{l'}| |\varphi_l\rangle\langle \varphi_{l'}| \right)$$

$$= \sum_m \sum_{j,j'} u_{am,j}^* u_{am,j'} \sqrt{q_j q_{j'}} |\phi_{j'}\rangle\langle \phi_j|$$

$$= \sum_m p_{am} |\psi_{am}\rangle\langle \psi_{am}|$$

$$= \sigma^O_a , \quad (3.13)$$

as desired. \qed

Theorem 4 tells us that we prepare an ensemble $\{\sigma^a\}_a$ where

$$\forall a \sigma_a \geq 0,$$

$$\sum_a \sigma_a = \rho^O . \quad (3.14)$$

if and only if the pure, joint system $\rho^{OQ}$ is measured with a POVM $\{P^Q_a\}_a$. Recall that, in our case, the state $\rho^{OQA}(t_f)$ is pure, but the state $\rho^{OQ}(t_f)$ may be mixed. We therefore construct POVM elements in the space $QA$. We can now use equation (3.14) as a set of constraints corresponding to measurement. We label these constraints $S_M$. The complete set of constraints corresponding to the execution of a quantum query algorithm is then given by $S_E \cup S_M$.

### 3.3.4 Cost Function

Notice that $S_M$ does not constrain or optimize over the number of individual POVM operators $P_a$ needed. In addition, the SDP, as written, does not specify the meaning of individual measurement outcomes, $a$. Each of these is determined by the chosen objective function. To be as flexible as possible, we choose an objective which corresponds to the average value of an arbitrary
cost function. Specifically, our SDP will be designed to minimize,

\[ \langle C \rangle = \sum_a \sum_x C(x, a) p(a|x) p(x) \]

\[ = \sum_a \sum_x C(x, a) \text{tr}(P^A_a \rho^Q_A(t_f)) p(x). \]  

(3.15)

The notation \( \rho^Q_A(t_f) \) refers to a state that was queried with the oracle corresponding to the input \( x \). \( C \) is a function of \( x \) and a measurement outcome \( a \). \( C(x, a) \) can be interpreted as the cost of measuring outcome \( a \) when the oracle corresponding to input \( x \) was applied. The objective (3.15) is an average of this function over all possible inputs with probability distribution \( p(x) \) and all possible measurement outcomes with probability distribution \( p(a|x) = \text{tr}(P^A_a \rho^Q_A(t_f)) \).

In order for equation (3.15) to be a valid SDP objective, it must be linear in the variables of the SDP. We can express equation (3.15) explicitly in terms of the variables of \( S_M \) as,

\[ \langle C \rangle = \sum_a \sum_x C(x, a) \text{tr}(P^A_a \rho^Q_A(t_f)) p(x) \]

\[ = \sum_a \sum_x C(x, a) \text{tr}(P^Q_A \rho^Q_A(t_f))_{x,x} \]

\[ = \sum_a \sum_x C(x, a)(\sigma^O_a)_{x,x} \]

\[ = \sum_a \text{tr}(A^O_a \sigma^O_a), \]  

(3.16)

To get from line one to line two, note that block \((x, x)\) of \( \rho^Q_A(t_f) \) is \( \rho^Q_A \sqrt{p(x)} \sqrt{p(y)} \).

Line three comes from the definition of remote state preparation, and in line four, \( (A^O_a)_{x,x'} \equiv \delta_{x,x'} C(x, a) \). Each \( A^O_a \) will be referred to as a cost function operator. By comparing the third line of equation (3.16) to the first line of equation (3.15), we identify \( (\sigma_a)_{x,x} = p(a,x) \) and \( \text{tr}(\sigma_a) = p(a) \).

The last line of equation (3.16) becomes the objective of the full SDP given below.

**Definition 3.3: SDP \( S \):**

Minimize \( \text{tr} \left( \sum_a \sigma_a A_a \right) \) subject to:

\( \forall a \ \sigma_a \geq 0, \)

\( \sum_a \sigma_a = \rho^O(t_f), \)
\[ \rho^O(0) = \rho_0, \quad \rho^{OQ}(0) \geq 0, \quad \rho^O(0) = \text{tr}_Q(\rho^{OQ}(0)), \]

and the following for \( t \in \{1, \ldots, t_f\} \):

\[ \rho^{OQ}(t) \geq 0, \]
\[ \rho^O(t) = \text{tr}_Q(\rho^{OQ}(t)), \]
\[ \rho^O(t) = \text{tr}_Q(\Omega^{OQ}\rho^{OQ}(t-1)\Omega^{\dagger OQ}). \]

The meaning of individual measurement outcomes is now determined by the specific choice of \( C(x, a) \). Since the goal of a quantum query algorithm is to determine the given input, (or equivalently, the applied oracle) we associate measurement outcomes with inputs via a map \( g(a) \) and consider \( C \) which assign costs to pairs of inputs; that is, \( C(x, a) \rightarrow C(x, g(a)) \). Now \( C(x, g(a)) \) can be interpreted as the cost of measuring input \( g(a) \) when the algorithm was given input \( x \). Given this restriction, theorem 5 shows that at most \( |O| \) measurement outcomes are needed, and thus, the optimal POVM consists of at most \( |O| \) elements.

**Theorem 5.** If \( C \) is a function of input pairs, then restricting SDP \( S \) to \( |O| \) measurement outcomes will not increase the optimal cost.

**Proof.** If \( S \) uses more than \( |O| \) measurement outcomes then \( g \) must map some of these outcomes to identical inputs. Consider any two such outcomes \( a_1 \) and \( a_2 \) where \( g(a_1) = g(a_2) \). From the second to last line of equation (3.16) the contribution of these two outcomes to the total cost is given by

\[
\sum_x C(x, g(a_1)) (\sigma^O_{a_1})_{x,x} + C(x, g(a_2)) (\sigma^O_{a_2})_{x,x}.
\]

(3.17)

\( \sigma_{a_1} \) and \( \sigma_{a_2} \) are chosen optimally by the SDP. We must have \( C(x, g(a_1)) = C(x, g(a_2)) \). Then equation (3.17) becomes

\[
\sum_x C(x, g(a_1)) ( (\sigma^O_{a_1})_{x,x} + (\sigma^O_{a_2})_{x,x} )
\]

\[
= \sum_x C(x, g(a_1)) (\sigma^{O}_{a_T})_{x,x},
\]

where \( \sigma_{a_T} \equiv \sigma_{a_1} + \sigma_{a_2} \) can be chosen by an SDP which is restricted to \( |O| \) measurement outcomes. Therefore, additional outcomes that map to the same input will not reduce the optimal cost. \( \square \)
In the context of traditional quantum query algorithms, we often wish to choose an objective whose value is the average probability of error. This can be done with a suitable choice of \( C \). First, observe that given an input \( x \), as above, some measurement outcomes are considered correct, while others are considered errors. For problems where the goal is to correctly identify the applied oracle, e.g. SEARCH, \( a \) is a correct outcome given \( x \) if and only if \( g(a) = x \). On the other hand, consider a decision problem such as OR, which can be encapsulated by a binary function \( F \) as described in section 3.1. Here \( a \) is considered a correct outcome given \( x \) if and only if \( F(x) = F(g(a)) \).

Average probability of error, denoted here by \( \epsilon \), is equivalent to the probability that oracle \( \Omega(x) \) was applied, outcome \( a \) was measured, but \( a \) is an incorrect outcome given \( x \), averaged over all \( x \) and all \( a \). Denote the set of all \( x \) for which \( a \) is an incorrect outcome by \( E(a) \). Since \( (\sigma_a)_{x,x} \) is equal to the probability that outcome \( a \) was measured, and oracle \( \Omega(x) \) was applied, we have

\[
\epsilon = \sum_a \sum_{x \in E(a)} (\sigma_a)_{x,x}. \tag{3.18}
\]

Thus the appropriate cost function is \( C(x, a) = 1_{E(a)}(x) \) and the matrix elements of the cost function operators are given by \( (A_a)_{x,y} = \langle x|A_a|y \rangle = \delta_{x,y}1_{E(a)}(x) \) where \( 1_{E(a)} \) is the indicator function of the set \( E(a) \). With this choice of cost function, the semidefinite program \( S \) computes the minimum average probability of error of a \( t_f \) query quantum algorithm.

Note that for this cost function (and others) it is not always necessary to use the full set of \( |O| \) POVM elements. The optimal algorithm for OR, for example, requires only two elements. To see this, consider two measurement outcomes \( a_1 \) and \( a_2 \). If \( F(g(a_1)) = F(g(a_2)) \), we expect \( C(x, g(a_1)) = C(x, g(a_2)) \). According to the proof of theorem 5, any two measurement outcomes with identical \( C \) for all \( x \) are redundant. Since \( F(x) \) can only be 0 or 1, there are only two measurement outcomes that lead to distinct values of \( C \). We label these outcomes \( a_0 \) and \( a_1 \). \( a_0 \) is considered correct for all \( x \) where \( F(x) = 0 \) and \( a_1 \) for all \( x \) with \( F(x) = 1 \).

The SDP outputs the optimal density matrix at each timestep \( t \). From these density matrices we can compute the optimal set of inter-query unitaries via the arguments in the proof of theorem 3 and the optimal POVM via the arguments in the proof of theorem 4.
3.3.5 Original Spectral Adversary Method

The original spectral adversary method derived similar measurement constraints without using remote state preparation but did not consider arbitrary cost functions. Rather, a lower bound on the number of queries needed to obtain some fixed probability of error was obtained via the dual of a relaxation of $S$. This bound is given by theorem 6 below [12, 75].

**Definition 3.4: Spectral Adversary Bound** Let $A^n$ be a space of inputs with $A$ a finite set. Consider a function $F : C \subset A^n \rightarrow D$, where $D$ is also a finite set. Let $x$ and $y$ be two inputs to this function. Let $\mathcal{G}$ be the set of real symmetric $|C| \times |C|$ matrices $\Gamma$ satisfying that for all $x, y$, if $F(x) \neq F(y)$ then $\Gamma_{x,y} = 0$. For such $\Gamma$, define the matrices $\Gamma_i, i \in \{0, \ldots, n\}$ by

$$(\Gamma_i)_{x,y} = \begin{cases} 0 & \text{if } x(i) \neq y(i) \\ \Gamma_{x,y} & \text{if } x(i) = y(i), \end{cases}$$

where $x(i)$ refers to the $i$th bit of input $x$. Then define

$$ADV(F) = \max_{\Gamma \in \mathcal{G}, \Gamma \geq 0} \frac{||\Gamma||}{\max_i ||\Gamma_i||},$$

where $||\Gamma||$ is the spectral norm of $\Gamma$. The matrix $\Gamma$, like the relation $R$ defined in section 3.1, is chosen to get a tight lower bound.

**Theorem 6.** The number of queries required to evaluate $F$ on a quantum computer with error probability at most $\epsilon$ is

$$Q_\epsilon(F) \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2} ADV(F).$$

Note that in our case, we constructed cost operators $A_a$ such that $S$ exactly minimizes average probability of error for a fixed number of queries.

3.4 Other Variants of the Adversary Method

There are several other variants of the adversary method, including a “weighted” method [9], a “strongly weighted” method [97], and a “Kolmogorov complexity” method [58]. It was shown that
the spectral method and each of these variants are equivalent [88] - that is, they can prove identical lower bounds. These bounds are often tighter than those that could be obtained with Ambainis’s original method (described in sections 3.1 and 3.2). Their superiority is a result of their ability to consider arbitrary probability distributions over inputs. Indeed, Ambainis’s original method implicitly assumes a uniform probability distribution over inputs. Therefore, one might expect the bounds it derives to be equally tight only if the uniform distribution is the most computationally demanding distribution for all possible algorithms.

Bounds of these newer methods are limited by the certificate complexity of $F$. (See [17] for an explanation of certificate complexity.) Specifically, they are at most $\min(\sqrt{C_0(F)n}, \sqrt{C_1(F)n})$ for partial boolean functions and $\sqrt{C_0(F)C_1(F)}$ for total boolean functions, where $C_0$ and $C_1$ are the 0 and 1 certificate complexities.

Recently, a stronger adversary method was developed [47] that, in some cases, can prove stronger bounds than all of the above methods, and can beat the certificate complexity bound. It is sometimes referred to as the “general” adversary method and bounds query complexity as follows.

**Definition 3.5: General Adversary Bound** Let $\Gamma, \Gamma_i$ and $F$ be as in theorem 6. Then define

$$ADV^\pm(F) = \max_{\Gamma \in \mathcal{G}} \frac{||\Gamma||}{\max_i ||\Gamma_i||}.$$ (3.22)

**Theorem 7.** The number of queries required to evaluate $F$ with error probability at most $\epsilon$ is

$$Q_\epsilon(F) \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2}ADV^\pm(F).$$ (3.23)

This method obtains its added power by allowing $\Gamma$ to be negative.

Note that this method still corresponds to a relaxation of SDP $S$. However this relaxation is nearly exact. By relating quantum query algorithms to span programs, Reichardt showed [75] that

**Theorem 8.** For all boolean functions $F : \{0,1\}^n \rightarrow \{0,1\}$

$$Q(F) = \Theta(ADV^\pm(F)).$$ (3.24)
In other words, for boolean functions, the general adversary bound characterizes quantum query complexity up to a constant factor.

3.5 Noise

3.5.1 Introducing the Environment

SDP $S$ and other variants of the adversary method assume that there is no noise during the execution of the algorithm. Here, we adapt $S$ to allow for arbitrary noisy processes.

We describe such processes via joint unitary evolution on the system of interest, $\rho^Q$, and an external system, usually referred to as the environment. Thus we have

$$\rho^Q \rightarrow D^{QE} \rho^Q |\epsilon\rangle \langle |\epsilon| D^{\dagger QE}$$

(3.25)

where $|\epsilon\rangle$ is the initial, “blank” state of the environment and $D^{QE}$ is unitary. Recall from section 2.2.3 that a unitary operation on a joint system initially in a product state acts as a quantum operation on its subsystems. We can therefore describe the evolution of $\rho^Q$ as

$$\rho^Q \rightarrow \xi(\rho^Q) = \sum_i E_i^{Q} \rho^Q E_i^{Q \dagger},$$

(3.26)

where $\xi$ is an arbitrary quantum operation, and the $E_i^{Q}$ are the Kraus operators, $E_i \equiv \langle \epsilon_i | D^{QE} | \epsilon \rangle$, with $|\epsilon_i\rangle$ an orthonormal basis for $E$, that implement the operation.

To map this evolution to an expanded set of semidefinite constraints, we first extend the states $\rho^{OQ}$ to $\rho^{OQE}$. The environment is not accessible by the querier. We allow the unitary describing the noisy process to be oracle dependent; we therefore fold $D^{QE}$ into an oracle-conditional isometry defined by

$$|x\rangle\langle y|\rho^Q \rightarrow |x\rangle\langle y|\Omega(x)^{QE} \rho^Q |\epsilon\rangle \langle |\epsilon| \Omega(y)^{\dagger QE},$$

(3.27)

where $|x\rangle, |y\rangle$ are the standard oracle basis states, and $\Omega(x)^{QE}$ is unitary. Define $\Omega^{OQE} = \sum_x |x\rangle\langle x|\Omega(x)^{QE}$. In the absence of noise, $\Omega(x)^{QE} = \langle x|\Omega^{OQ}|x\rangle \otimes I^E$. For an $O$-independent noisy unitary one can decompose $\Omega(x)^{QE} = D^{QE} \langle x|\Omega^{OQ} \otimes I^E |x\rangle$. 

If the environments involved in different queries are independent, a new version of $E$, $E_i$ is introduced at each step by the isometry. Let $E^t = E_0 E_1 \ldots E_t$. To account for the noisy query, SDP $S$ is modified to $S_D$ as follows:

**Definition 3.6: SDP $S_D$:**

Minimize $tr(\sum_a \sigma_a^{O_{E^f}} A_a^{O_{E^f}})$ subject to

for all $a$, $\sigma_a^{O_{E^f}} \geq 0$,

$\sum_b \sigma_b^{O_{E^f}} = \rho^{O_{E^f}}(t_f)$,

$\rho^{O_{E^0}}(0) = \rho_0$, $\rho^{O_{E^0}}(0) \geq 0$, $\rho^{O_{E^0}}(0) = tr_Q(\rho^{O_{E^0}Q}(0))$,

and the following for $t \in \{1, \ldots, t_f\}$:

$\rho^{O_{E^f}Q}(t) \geq 0$,

$\rho^{O_{E^f}}(t) = tr_Q(\rho^{O_{E^f}Q}(t))$,

$\rho^{O_{E^f}}(t) = tr_Q\left(\Omega^{O_{E^f}E_i} \rho^{O_{E^f-1}Q(t-1)|\epsilon\rangle\langle\epsilon|\Omega^{O_{E^f}E_i}}\right)$.

Note that it is not possible to simply trace out the $E_i$ in the SDP: As can be seen from the method of reconstructing the algorithm from a solution of the SDP, this would be equivalent to giving the querier access to the $E_i$. That is, the querier has implicit access to anything that gets traced out, since traced out systems are not constrained by the SDP. When the noisy query factors, this is equivalent to not having had any noise at all, because the querier can just undo the noise isometries.

Also note from theorem 4 that remote state preparation requires a pure state. Due to the addition of the environment, $\rho^{O_{QAE^f}}$ is pure, but $\rho^{O_{QA}}$ may not be. This again implies that we cannot trace out the environment, and furthermore, that the querier’s measurement must remotely prepare states over $O_{E^f}$. Therefore, the cost function operators must be extended to $A_a^O \rightarrow A_a^O \otimes I_{E^f}$.

Phase decoherence is a particularly interesting example of this more general SDP. In this
case, the query isometry factors, so that

\[ D^{QE}|i\rangle^Q|\epsilon\rangle^E = |i\rangle^Q|\epsilon_i\rangle^E, \]

(3.28)

where the \( |\epsilon_i\rangle \) are orthonormal states. The effect is to perfectly correlate the environment at each step with the standard query basis.

We now look at how phase decoherence affects the first two steps of a query algorithm. We assume we are given input \( x \) and begin with an arbitrary querier state and a blank environment

\[ |\psi(0)\rangle = \sum_i a_i |x\rangle^O|\epsilon_i\rangle^Q|b_0\rangle^Q|\epsilon\rangle^E_1. \]

(3.29)

We decohere after each query; after the first, \( |\psi\rangle \) goes to,

\[ |\psi(0)\rangle' = D^{QE_1}\Omega^{OQ}|\psi\rangle = \sum_i a_i |x\rangle^Q \Omega(x)|i\rangle^Q|b_0\rangle^Q|\epsilon_i\rangle^E_1|\epsilon\rangle^E_2, \]

(3.30)

where as before, \( x(i) \) is the \( i \)th bit of input \( x \), and \( |b\rangle \) is a state used for recording the results of queries. Next, an arbitrary unitary, expressed in the computational basis as \( U = \sum_{j,j'} u_{j,j'} |j\rangle\langle j'| \) is applied,

\[ |\psi(1)\rangle = U^Q D^{QE_1}\Omega^{OQ}|\psi\rangle = \sum_{i,j} a_i u_{j,i} |x\rangle^O|j\rangle|b_0 \oplus x(i)\rangle^Q|b_1\rangle^Q|\epsilon_i\rangle^E_1|\epsilon\rangle^E_2. \]

(3.31)

And finally, after a second query,

\[ |\psi(1)\rangle' = D^{QE_1}\Omega^{OQ} U^Q D^{QE_1}\Omega^{OQ}|\psi\rangle = \sum_{i,j} a_i u_{j,i} |x\rangle^O|j\rangle|b_0 \oplus x(i)\rangle^Q|b_1 \oplus x(j)\rangle^Q|\epsilon_i\rangle^E_1|\epsilon_j\rangle^E_2|\epsilon\rangle^E_3. \]

(3.32)

We can see that the size of the environment grows with every query, and serves in a sense, as a record of previous queries. The querier has no access to \( E \), and cannot undo this record keeping. In physical systems, this corresponds to the leakage of quantum information into the environment. This information cannot be retrieved, and the system remains decohered.

While perhaps not obvious from the form of the evolution above, a quantum computation subject to complete phase decoherence after each and every query is equivalent to a classical
computation. In other words, a quantum computer which is unable to maintain coherence after querying its input, or records queries via some secondary system, can perform no better than a classical computer. We can see this by examining how the state of the computer evolves from one timestep to the next. To this end, we write $|\psi(0)\rangle$ and $|\psi(1)\rangle$ as a density matrices and trace out $E$. Beginning with $|\psi(0)\rangle$,

$$
|\psi(0)\rangle\langle\psi(0)| = \sum_{i,i'} a_{i,i'} |x\rangle\langle y| \otimes |i\rangle\langle i'\rangle \otimes |x(i)\rangle\langle x(i')\rangle \otimes |\epsilon_i\rangle\langle \epsilon_{i'}|^{E_1} \otimes |\epsilon\rangle^{E_2}.
$$

(3.33)

Tracing out $E$ gives

$$
\sum_{i,j} a_{i,i} |x\rangle\langle y| \otimes |i\rangle\langle i| \otimes |x(i)\rangle\langle x(i)|^{Q}.
$$

(3.34)

For $|\psi(1)\rangle$,

$$
|\psi(1)\rangle\langle\psi(1)| = \sum_{j,j',i,i'} a_{i,i'} u_{i,j} u_{i',j'} |x\rangle\langle y| \otimes |j\rangle\langle j'| \otimes |x(i)\rangle\langle x(i')\rangle \otimes |\epsilon_i\rangle\langle \epsilon_{i'}|^{E_1} \otimes |\epsilon_j\rangle\langle \epsilon_{j'}|^{E_2} \otimes |\epsilon\rangle^{E_3}.
$$

(3.35)

Tracing out $E$,

$$
\sum_{j} \sum_{i} a_{i,i} u_{j,i}^2 |x\rangle\langle y| \otimes |j\rangle\langle j| \otimes |x(i)\rangle\langle x(i)|^{Q} \otimes |b_1 \oplus x(j)\rangle\langle b_1 \oplus x(j)|^{Q}.
$$

(3.36)

Notice that in both equation (3.34) and equation (3.36) the querier density matrix is diagonal. These are classical states - they are simply equivalent to vectors of probabilities. Furthermore we evolve from $|\psi(0)\rangle$ to $|\psi(1)\rangle$ via a query and a matrix with entries $|u_{j,i}|^2$. A matrix whose entries are the square of the absolute value of those of a unitary matrix is a particular type of stochastic matrix known as a unistochastic matrix. Thus we see that we transition from a classical state to a classical state via a stochastic matrix just as in classical computation.

Evaluating SDP $S_D$ requires significant computational resources due to the environment added after each query. While we cannot simply trace out the environment for the reasons given earlier, one might still ask whether there is an alternate SDP describing classical computation that requires less memory. While we do not have a definitive answer to this question, we conjecture that,
in fact, we cannot do better. To understand this, consider the randomized algorithm for solving
SEARCH presented in chapter 2. As discussed, it would not be optimal to apply the matrix (2.43)
to the state (2.44) a second time. Since this matrix encodes equiprobable transitions from all states
that do not contain the marked item to all other states, a second application allows the algorithm to
revisit input positions it has already checked. Instead, the optimal algorithm must keep track of all
positions it has already visited. Specifically, with probability $p_i$, the algorithm records that it has
visited position $i$. Therefore, to properly represent the state of such an algorithm, memory of size
at most $|Q|$ is needed after the first query. After the second query, the algorithm must keep track of
which positions were checked during both the first and second query; that is, with probability $p_{i,j}$,
the algorithm records that state $i$ was visited during the first query, and state $j$ was visited during
the second query. Thus, a representation of the state of the algorithm now requires memory of size
at most $|Q|^2$. In general, after $t$ queries, the state of classical randomized algorithm is described in
a space of size at most $|Q|^t$. This is exactly the size of $E^t$. This example may seem artificial, since
the description of deterministic algorithm which progresses down the input one position at a time
does not require such resources. However, recall from section 3.1 that for some input probability
distributions, the worst case expected running time of a deterministic algorithm is worse than that
of a randomized algorithm.

3.5.2 Examples

In this section, we apply both the quantum and classical SDP to various toy problems. Due to
the enormous size of the SDP in the classical case, the examples presented here use a small number
of qubits and queries. Thus, all numerical results reproduce well known information, and these
examples should be taken as demonstrations of SDP’s $S$ and $S_D$. Each of the graphs that follows
depicts the optimal quantum and classical probability of error vs. query number as computed by
SDP’s $S$ and $S_D$ respectively for various canonical problems in quantum computation.

Figure 3.1 examines 4 element SEARCH. In this example, we allow for an additional input
with no marked element and expect the algorithm to report this correctly. Each of the five possible
inputs are chosen uniformly at random. In this case, even without a query, the algorithm can guess the input correctly 20\% of the time. The reduction in the classical probability of error with query number can be understood following the discussion in section 2.4.1.1.

Figure 3.1: 4 element SEARCH, with the possibility of no marked element. Here, |O| = 5 and |Q| = 4

Figure 3.2 examines the 4 bit Deutsch-Jozsa problem. In this problem, we are promised that the input is constant (contains all 0’s or all 1’s) or balanced (contains the same number of 0’s and 1’s); we must distinguish these two cases. Given an N bit input, a classical algorithm can solve this problem in N/2 + 1 queries, whereas a quantum algorithm can solve it with a single query. Notice from figure 3.2, that unlike SEARCH, classically, we obtain no reduction in error probability whatsoever, until the N/2 + 1-st query. Indeed, until step N/2 + 1, the algorithm can just guess "balanced", and obtain an error probability of only 25\%, as there are 6 possible balanced inputs and 2 possible constant inputs, all selected with equal probability.

Figure 3.3 depicts the PARITY problem. Here, we must decide if the given input contains an even or an odd number of 1’s. Given an input of size N, classically we require N queries, whereas a quantum algorithm needs only N/2 queries. Again, classically, we obtain no reduction in error probability until query N, as even after the first N − 1 positions are checked, the input is still equally likely to contain an even or an odd number of ones.
Constructing an optimal quantum algorithm for PARITY is relatively straightforward. Consider the action of a phase kickback oracle on the $|+\rangle$ state,

$$
\Omega(x)|+\rangle = \frac{1}{\sqrt{2}} \left( (-1)^{x_0}|0\rangle + (-1)^{x_1}|1\rangle \right).
$$

(3.37)

If the parity of positions 0 and 1 in input $x$ is even, we have $\Omega(x)|+\rangle = |+\rangle$ and if the parity is odd, we have $\Omega(x)|+\rangle = |-\rangle$ (each up to an irrelevant global phase). This observation allows us to effectively check the parity of two input positions with one quantum query, whereas we could only check one position at a time with a classical query.

Using this observation, we can construct a quantum algorithm for PARITY.

\[
p = \text{PARITY}(x)
\]

\[
i = 0
\]

\[
p = 0
\]

\[
N = \text{sizeOf}(x)
\]

\[
\textbf{while } i <= N/2:
\]

\[
m = P^{+\cdot-}\Omega(x)|+\rangle|i\rangle
\]

\[
p = p \oplus m
\]

\[
i = i + 1
\]
This algorithm steps through the input two positions at a time, checks the parity, and adds the result (mod 2) to the running total. The state $|+\rangle\!\!\langle i| = H|0\rangle\!\!\langle i|$ corresponds to superposition of computational basis states $i$ and $i+1$, and $P^{+,-}$ projects onto the $|+\rangle$, $|\rangle$ basis and returns a 0 or a 1 depending on the measurement result. Notice that, similar to the classical case, even after the first $N-2$ positions are checked, the input is equally likely to contain an even or odd number of ones, so there is no reduction in error probability until query $N/2$. Recent work by Meyer and Pommersheim [66] explores these abrupt transitions in error probability in more depth.

![Figure 3.3: 4 element PARITY. $|O| = 16$, $|Q| = 4$](image)

### 3.5.3 Quantum to Classical Interpolation

We can also consider quantum computers with limited decoherence. That is, we can assume that after each query, a quantum computer decoheres with some probability. Such computers will not behave entirely classically, but will not perform as well as perfect quantum computers. To illustrate this, consider a two qubit computer where, after each query, the first qubit decoheres
with probability $p$. After the first query, the four computational basis states evolve as,

$$
|x\rangle^Q\Omega^Q(x)|0\rangle^Q|b_0\rangle^Q|\epsilon\rangle^{E_1} \rightarrow |x\rangle^Q\Omega^Q(x)|0\rangle^Q|b_0\rangle^Q \left( \sqrt{p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 + \sqrt{1-p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 \right)
$$

$$
|x\rangle^Q\Omega^Q(x)|1\rangle^Q|b_0\rangle^Q|\epsilon\rangle^{E_1} \rightarrow |x\rangle^Q\Omega^Q(x)|1\rangle^Q|b_0\rangle^Q \left( \sqrt{p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 + \sqrt{1-p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 \right)
$$

$$
|x\rangle^Q\Omega^Q(x)|2\rangle^Q|b_0\rangle^Q|\epsilon\rangle^{E_1} \rightarrow |x\rangle^Q\Omega^Q(x)|2\rangle^Q|b_0\rangle^Q \left( \sqrt{p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 + \sqrt{1-p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 \right)
$$

$$
|x\rangle^Q\Omega^Q(x)|3\rangle^Q|b_0\rangle^Q|\epsilon\rangle^{E_1} \rightarrow |x\rangle^Q\Omega^Q(x)|3\rangle^Q|b_0\rangle^Q \left( \sqrt{p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 + \sqrt{1-p}|\epsilon\rangle^E_1|\epsilon\rangle^E_2 \right).
$$

(3.38)

Above, the state of $E_1$ remains in its initial state with probability $1 - p$ and reflects the value of the most significant querier qubit with probability $p$. This can straightforwardly be adapted to the case where both qubits decohere, or to situations with additional qubits. This allows us to effectively interpolate between classical and quantum computation. Figure 3.4 shows an example of such an interpolation. Here, in 4 element SEARCH, we vary the probability of one or two qubits decohering from 0 – 100%.

Figure 3.4: Quantum/Classical Interpolation in the 4 element SEARCH problem, where there is always a marked element. The blue line depicts one qubit decohering with probability $p$, whereas the red line depicts two qubits decohering independently with probability $p$. 
Note that single qubit decoherence is equivalent to an erroneous application of the Pauli Z operator $\sigma_z$ to a qubit [70]. We can easily consider other Pauli errors with the same formalism. Since the Pauli matrices, along with the $2 \times 2$ identity matrix, form a basis for all $2 \times 2$ matrices, any single qubit error can be expressed as a linear combination of Pauli errors. Therefore, this formalism can be used to examine the effects of arbitrary errors on the performance of an algorithm. Indeed, per equation (3.26), the introduction of system $E$ allows us to consider the effects of arbitrary quantum operations.

### 3.6 Reduced Quantum Resources

Recall from the proof of theorem 3, that an additional system of ancillas, $A$, must be introduced to properly extract the inter-query unitaries, $U^{QA}(t)$. Note, however, that the quantum algorithms we have explicitly presented so far, including Grover’s algorithm for SEARCH, and an algorithm for PARITY, do not make use of such an ancilla space and require a Hilbert space of size at most $|Q|$. This implies that the extracted unitaries and associated quantum states may not be space optimal. That is, they may use more qubits than is strictly necessary. Indeed, SDP S optimizes an arbitrary cost function given a fixed number of queries; for many choices of cost function, including probability of error, the SDP does not consider space utilization.

Therefore, given some cost-optimal algorithm derived by SDP S, there may be an alternate algorithm that achieves the same cost with an identical number of queries but that uses fewer qubits. Our goal in this section is to derive such algorithms from those extracted by the SDP. Here, we are only concerned about quantum resources - in particular the number of coherent qubits needed. We are unconcerned with classical space.

To motivate this, consider the quantum algorithm for PARITY presented in section 3.5.2. Regardless of the size of the input, this algorithm requires only a single coherent qubit prepared in the $|+\rangle$ state. However, the algorithm extracted from the optimal solution of the SDP used to generate figure 3.3 is a two qubit algorithm.

Before continuing, we need a way to quantify the number of qubits utilized by an algorithm.
We make use of the Von Neumann entropy defined by

\[ S(\rho) \equiv \text{tr}(\rho \log \rho) \] (3.39)

Consider a length \( n \) sequence of quantum states where each state is drawn from the ensemble \( \{ |\psi_i\rangle, p_i \} \). This ensemble is described by density matrix \( \rho = \sum_i p_i |\psi_i\rangle \langle \psi_i| \). Schumacher has shown \[85\] that the minimum number of qubits required to represent this sequence is given by \( nS(\rho) \) as \( n \to \infty \). Thus \( S(\rho) \) can be used to quantify the quantum information content of \( \rho \).

The Von Neumann entropy has a number of other useful properties:

1. \( S \) is basis independent. That is, for all unitaries \( U \), \( S(\rho) = S(\rho') \) where \( \rho' = U \rho U^\dagger \). Such a property is necessary if \( S \) is to properly describe the information content of states.

2. \( S(\rho) = 0 \) if and only if \( \rho \) is a pure state. This is can be seen from basis independence. Consider an arbitrary pure state \( |\psi\rangle = \sum_i a_i |\phi_i\rangle \). With a suitable unitary, this state can be transformed to \( |0\rangle \), which has no information content, and hence, entropy 0.

3. A pure state \( \rho^{AB} \) satisfies \( S(\rho^A) = S(\rho^B) \). This is proved in \[70\] by way of the Schmidt decomposition.

Our goal is to reduce the entropy of \( \rho^{QA}(t) \) at chosen points in the extracted algorithm. Note that an initial pure state acted on by a quantum algorithm given a specific input remains pure, and therefore, has 0 entropy at all times. Therefore, one might expect that an initially pure \( \rho^{QA}(0) \) remains pure for all \( t \) and there is no need to reduce entropy. However, recall that in the query model of quantum computation, we always consider a probability distribution of potential inputs, and our goal can be thought of as distinguishing inputs, or equivalently, identifying which oracle was applied to a quantum state. If an individual oracle is applied with certainty, this problem is trivial, and indeed, can be solved with a zero qubit, zero entropy algorithm. However, after a single...
application of $\rho^{OQ}$, the state of the querier becomes equation (3.4),

$$
\rho^Q(1) = \text{tr}_O(\Omega^{OQ}\rho(0)^{OQ}\Omega^{\dagger OQ}) \\
= \sum_x p(x)\Omega^Q(x)\rho^Q(0)\Omega^{\dagger Q}(x). \tag{3.40}
$$

This is a mixed state with nonzero entropy. The closer $\rho^Q$ is to a pure state, the smaller the entropy; roughly speaking, from equation (3.40) this corresponds to a narrower probability distribution over inputs, and we expect algorithms which have fewer inputs to distinguish to use fewer resources.

We therefore seek to split our computation into two or more independent subparts, where each subpart considers a portion of the full oracle density matrix. These subparts will then be combined classically. We expect the querier matrix associated with each subpart to have reduced entropy.

To accomplish this, imagine that the querier makes a measurement at a particular timestep $t_s$ within an algorithm. This measurement is described by a two element POVM consisting of $P_1^{QA}$ and $P_2^{QA}$. If classical outcome 1 is obtained, some sub-ensemble $\rho^Q_1(t_s)$ is remotely prepared on $O$. Likewise, if outcome 2 is measured $\rho^Q_2(t_s)$ is remotely prepared. Per equation (3.14) these satisfy

$$
\rho^O(t_s) = \rho^O_1(t_s) + \rho^O_2(t_s). \tag{3.41}
$$

The algorithm then continues until time $t_f$ considering inputs from sub-ensemble 1 with probability $p_1 = \text{tr}(P_1^{QA}\rho^{OQA}(t_s)) = \text{tr}(\rho^Q_1)$ or sub-ensemble 2 with probability $p_2 = \text{tr}(P_2^{QA}\rho^{OQA}(t_s)) = \text{tr}(\rho^Q_2)$. We expect $\rho^Q_1(t \geq t_s)$ and $\rho^Q_2(t \geq t_s)$ to have reduced entropy, and therefore, the new, split algorithm to require fewer quantum resources than the original, joint algorithm. We would like the new algorithm to perform equally as well (i.e. achieve the same cost) as the original algorithm.

We now construct a new SDP, $S_R$, which is designed to split the original optimal algorithm into these independent subparts. In particular, it will try to split $\rho^O(t_s)$ into $\rho^O_1(t_s)$ and $\rho^O_2(t_s)$, where $\rho^O_1(t_s)$ and $\rho^O_2(t_s)$ are as non-proportional as possible. Ideally, $\rho_1(t_s)$ and $\rho_2(t_s)$ would be sums over disjoint sets of inputs.

To this end, $\rho^O_1(t)$ and $\rho^O_2(t)$ for $t \geq t_s$ become variables in $S_R$; they will be constrained via equation (3.41). They and their purifications into $QA$ will also be subject to the same constraints.
that $\rho^O$ and its purifications were subject to in SDP $S_{E}$. This is equivalent to requiring that two quantum algorithms are applied along two independent computational paths after $t_s$. We also require that the cost of the split algorithm matches that of the original algorithm. To this end, we introduce positive matrices $\{\sigma_{1,a}\}$ and $\{\sigma_{2,a}\}$ constrained via remote state preparation

$$
\rho^O_1(tf) = \sum_a \sigma_{1,a} \\
\rho^O_2(tf) = \sum_a \sigma_{2,a}.
$$

(3.42)

Expected cost is then constrained as

$$
\sum_a \text{tr}(\sigma_a A_a) = \sum_a \text{tr}(\sigma_{1,a} A_a) + \sum_a \text{tr}(\sigma_{2,a} A_a)
$$

(3.43)

where $A_a$ are the cost operators optimized by the original algorithm.

Finally, we choose a new cost function designed to maximize the non-proportionality of $\rho^O_1(t_s)$ and $\rho^O_2(t_s)$. We make the choice,

$$
\text{maximize } \text{tr}(\rho^O_{1\bot}(t_s) \tilde{\rho}^O_1(t_s))
$$

(3.44)

where $\rho^O_{1\bot}(t_s)$ is the part of $\rho^O_1(t_s)$ orthogonal to $\rho^O(t_s)$,

$$
\rho^\bot_1 = \rho_1 - \frac{\text{tr}(\rho \rho_1) \rho}{\text{tr}(\rho^2)}
$$

(3.45)

By maximizing non-proportionality, we force the sum, equation (3.41), to be nontrivial.

Unfortunately, the cost function, equation (3.44) cannot be used directly since it is a nonlinear function of constrained variables. We therefore utilize an iterative strategy. Specifically, we run the SDP multiple times, and at iteration $i$, maximize

$$
\text{tr}(\rho^O_{1\bot}(t_s) \tilde{\rho}^O_1(t_s))
$$

(3.46)

where $\tilde{\rho}^O_1(t_s)$ is set to the value of $\rho^O_{1\bot}(t_s)$ derived at iteration $i - 1$. Then we have the following SDP:

**Definition 3.7: SDP $S_R$:**
Maximize \( \text{tr}(\rho_1^{O}(t_s)\tilde{\rho}_1^{O}(t_s)) \) subject to:

\[
\begin{align*}
\forall a \: \sigma_{1,a} &\geq 0, \\
\forall a \: \sigma_{2,a} &\geq 0, \\
\sum_a \sigma_{1,a} &= \rho_1^{O}(t_f), \\
\sum_a \sigma_{2,a} &= \rho_2^{O}(t_f), \\
\sum_a \text{tr}(\sigma_a A_a) &= \sum_a \text{tr}(\sigma_{1,a} A_a) + \sum_a \text{tr}(\sigma_{2,a} A_a) \\
\rho^O(t_s) &= \rho_1^{O}(t_s) + \rho_2^{O}(t_s),
\end{align*}
\]

and the following for \( t \in \{t_s, \ldots, t_f\} \):

\[
\begin{align*}
\rho_1^{OQ}(t) &\geq 0, \\
\rho_2^{OQ}(t) &\geq 0, \\
\rho_1^{O}(t) &= \text{tr}_Q(\rho_1^{OQ}(t)), \\
\rho_2^{O}(t) &= \text{tr}_Q(\rho_2^{OQ}(t)), \\
\rho_1^{O}(t) &= \text{tr}_Q(\Omega^{OQ} \rho_1^{OQ}(t-1) \Omega^{\dagger OQ}), \\
\rho_2^{O}(t) &= \text{tr}_Q(\Omega^{OQ} \rho_2^{OQ}(t-1) \Omega^{\dagger OQ})
\end{align*}
\]

Note that here, \( \rho^O(t) \), the cost optimal density matrix for the problem under consideration, is derived from \( S \) as before and is considered an input to \( S_R \). We iterate until the maximum value of the objective converges. Initially, \( \tilde{\rho}_1^{O}(t_s) \) is set to a random positive matrix.

It may be possible to split the optimal algorithm more than once. Indeed, we can attempt to split the optimal density matrices \( \rho_1^{O}(t) \) and \( \rho_2^{O}(t) \) returned from \( S_R \), by running \( S_R \) a second time, with \( \rho_1^{O} \) and/or \( \rho_2^{O} \) taking the place of \( \rho^O \). We can then, of course, attempt to split the matrices returned from this second run; clearly, we can repeat this procedure to arbitrary depth, or until we no longer make progress.

We quantify the entropy of such a disjoint algorithm at time \( t \) by averaging over the entropy of each subpart as,

\[
\bar{S} = \text{tr}(\rho_1^{O}(t))S\left( \frac{\rho_1^{O}(t)}{\text{tr}(\rho_1^{O}(t))} \right) + \text{tr}(\rho_2^{O}(t))S\left( \frac{\rho_2^{O}(t)}{\text{tr}(\rho_2^{O}(t))} \right) + \ldots + \text{tr}(\rho_n^{O}(t))S\left( \frac{\rho_n^{O}(t)}{\text{tr}(\rho_n^{O}(t))} \right)
\] (3.47)
where this particular algorithm was split into $n$ parts. In this formula, the coefficients $\text{tr}(\rho_i^O(t))$ are the probabilities of obtaining the density matrices $\rho_i^O(t)$ after a measurement and proceeding down the corresponding computational paths. Entropy is calculated with normalized density matrices, $\rho_i^O(t) = \frac{\rho_i^O(t)}{\text{tr}(\rho_i^O(t))}$. Thus, equation (3.47) computes the average entropy along all possible paths that a computation may take, weighted by the probability of taking each path. Here we are calculating the average entropy of the oracle density matrices, when we are really interested in the querier matrices. However, since $\rho^{OQA}$ is a pure state at all timesteps, $S(\rho^O) = S(\rho^{QA})$.

Note that we have not proved that SDP $S_R$ minimizes average entropy (3.47). Rather, the arguments in this section have attempted to provide some justification that $S_R$ can be reasonably expected to reduce $\bar{S}$. Below, we show that this is the case in practice.

### 3.6.1 Results

We apply this procedure to 8-element SEARCH and 4-element PARITY. We derive optimal 2 query algorithms using SDP $S$, then split the resulting density matrices using SDP $S_R$ after the first query. Table 3.1 gives the average entropy calculated when the optimal density matrix is divided into increasingly many subparts.

<table>
<thead>
<tr>
<th>Number of Parts</th>
<th>SEARCH</th>
<th>PARITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0560</td>
<td>2.000</td>
</tr>
<tr>
<td>2</td>
<td>2.0475</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>2.0378</td>
<td>.0001</td>
</tr>
<tr>
<td>8</td>
<td>2.0262</td>
<td>.0001</td>
</tr>
</tbody>
</table>

Table 3.1: Entropy of 8-element SEARCH (where there may not be a marked element) and 4-element PARITY as a function of the number of independent computational paths, after the first query.

While we make little progress with SEARCH, according to our procedure, in 4-element PARITY, almost no quantum resources need to be carried over from the first to the second query.

This result is perhaps not surprising, in that the quantum algorithm for PARITY discussed in section 3.5.2, is a zero entropy algorithm. Observe that the optimal classical algorithm on a problem of size $N$ looks identical to the optimal quantum algorithm on a problem of size $N/2$ up to a choice
of basis. That is, from the quantum algorithm given, by replacing \(|+\rangle \rightarrow |0\rangle\) and \(P^{+,-} \rightarrow P^{0,1}\), the quantum oracle \(\Omega\) with a classical oracle, and \(N/2 \rightarrow N\) we obtain the optimal classical algorithm. Thus, there is nothing obviously “quantum” about the optimal quantum algorithm. Indeed, since entropy is a basis independent quantity, the use of \(|+\rangle\) instead of \(|0\rangle\) is irrelevant; that is, each has entropy 0 and can be represented by a single classical bit. Therefore, the quantum advantage must come from something outside the algorithm itself.

In this case, the advantage comes from the oracle. The oracle, when applied to \(|+\rangle\), effectively reads two bits of the input in superposition. This is impossible classically. Recall, however, that in the query model of computation, the underlying complexity of the oracle is irrelevant. Indeed, we could utilize an oracle of arbitrary time and space complexity, whose implementation requires some enormous number of qubits, and the entropy required by the algorithm would remain the same. By design, the complexity of the oracle is hidden from the algorithm; our measures of resource utilization count all oracle applications identically and ignore the implementations of the oracles themselves.
Chapter 4

Atomic Clocks

We now turn to a discussion of atomic clocks. Eventually, the techniques developed in chapter 3 will be applied to optimize these clocks. Unlike prior work, our scheme is adaptive; algorithms are derived and applied as a clock is running. In order to understand exactly how this works, we must describe atomic clocks from a somewhat different angle than that in much of the literature.

4.1 Timekeeping

Paradoxically, while time is one of the facets of the physical world that is the least well understood, it is also the property that can be measured with the greatest accuracy. That is, even though we don’t really know what time is, we can measure its passage extremely well - often far better than other physical quantities that seem, intuitively at least, more comprehensible, such as mass or size. This is true even though measurements of time are not particularly straightforward; we cannot simply lay out a ruler or place something on a scale. Rather our challenge is to measure something that is continuously passing or “ticking”, and this presents a fundamentally different challenge.

One insightful way of understanding time and how it’s measured is by recognizing that, according to special relativity, up to a sign in the Minkowski metric, time and space are one and the same. Thus, we can think of objects moving through time just as objects move through space, and can equate the passage of time with progression along the temporal axis.

First, consider how to measure such progression in the more familiar spatial setting. Say we
are driving a car with a broken odometer down a highway. We can often measure our progress by way of mile-markers. These mile-markers are typically evenly spaced, often at intervals of exactly one mile, and are labeled with a number that signifies their exact position along the highway. However, even without these labels, we could still measure our progress as long as the markers are evenly spaced. Indeed, we could in principle assign labels to blank markers as we progress down the highway. This is analogous to how time is measured. We seek a source of regularly spaced markers, or events, along the time axis. As we progress along this axis - through time - we will label these events with our best estimate of their true positions in time.

Notice that, in the example above, if the mile-markers are not evenly spaced, our task becomes much more difficult. If the intervals between markers are nearly equal, our labeling may still be fairly accurate, albeit with unavoidable errors. However, if the intervals between markers vary greatly, an accurate labeling becomes almost impossible. Thus, in order to measure time well, it is very important to choose a source with a period that is as consistent and uniform as possible. These sources are often referred to as oscillators.

In the past, these oscillators were astronomical. For hundreds of years, the recurring position of the sun in the sky provided a set of nearly uniformly spaced periodic events. The second was defined as 1/86400 of a mean solar day. Later, the rotation of the Earth around the sun was used, and the second was defined as 1/31556925.9747 of the length of the year 1900. \[7\]. As of 1967, astronomical clocks have been replaced by atomic clocks, and the second has been redefined in terms of the frequency of a hyperfine transition in a cesium atom. Atomic clocks produce a far more consistent periodic signal than older astronomical sources. The reference oscillator is often referred to as a frequency standard.

In addition to an oscillator, any clock also needs a counter. The counter keeps track of oscillator ticks and assigns labels or timestamps to events as required. Unlike in the mile-marker example above, these events may or may not correspond to ticks of the oscillator. The assignment of timestamps is generally difficult, as even the most consistent oscillator is subject to noise and imperfections. It is therefore important to understand the physics of whatever oscillator is being
used, so that timestamps can be assigned which correct for such undesirable effects. Such corrections are at the heart of any timekeeping strategy, and in fact, the goal of this thesis is to design strategies that effectively combat noisy processes in atomic clocks.

4.2 Basics of Atomic Clocks

All atoms possess a discrete set of energy levels that correspond to the state of the atom’s electrons relative to its nucleus. An atom may transition between these levels by emitting or absorbing a photon. If the energy gap between two levels is $E$, the angular frequency (up to quantum uncertainty) of the absorbed or emitted photon is $\omega = E/\hbar$. This radiation can be used as an almost ideal oscillator. The transition most relevant to atomic clocks is the ground state hyperfine transition of cesium 133. The second is currently defined as the time required for the passage of 9,192,631,770 cycles of the radiation corresponding to this transition. This number was selected to match, as closely as possible, the previous astronomical definition of the second. After the second is defined, we can define units of frequency; e.g. we can now say that the frequency of the cesium hyperfine transition is 9,192,631,770 Hz. A frequency corresponding exactly to some atomic transition is known as a resonant frequency.

4.2.1 Active vs. Passive Clocks

We broadly divide atomic clocks up into two categories: active and passive. Active clocks excite an atom or collection of atoms and measure the radiation emitted as the atoms drop to a lower energy state. Passive clocks, on the other hand, make use of an external oscillator or “flywheel” to interrogate an atomic system. This external oscillator may be a laser, maser, or some other source of photons. An interaction with such an oscillator tuned to or near the resonant frequency of the atoms excites the atoms to their higher energy state. A measurement of the atomic system then provides some information about how close the flywheel’s frequency was to the resonant frequency, which can be used to tune the flywheel appropriately. Here then, it is the radiation of the flywheel that is eventually measured by the clock’s counter.
Recall that the goal of any clock is to construct a time scale by time-stamping a set of events \( n \), with initially unknown time coordinates, \( t_n \). This is done differently in active and passive clocks. An active clock has significantly less flexibility, as it has only one choice for an assignment of a \( t_n \) to \( n \); specifically, \( t_n = \frac{2\pi C_n}{\omega_0} \), where \( C_n \) is the number of cycles counted up through event \( n \), and \( \omega_0 \) is the resonant frequency of the underlying oscillator. In contrast, a measurement of a passive clock yields (incomplete) information about the difference in frequency between the standard and the flywheel, often referred to as the frequency deviation or "detuning". We therefore have some flexibility in assigning \( t_n \)'s to \( n \)'s. Our choice of \( t_n \) may be affected by knowledge gained in measurements before and after \( t_n \), and time-stamps may therefore be continuously updated as the clock progresses. Note that none of this means that active clocks are necessarily any more or less accurate than passive clocks, merely that time assignments are more involved in the latter.

4.2.2 Modern Atomic Clock Technologies

The primary U.S. frequency standard is NIST-F1, a passive cesium fountain clock. In this section, we discuss cesium fountain clocks generally, timekeeping with NIST-F1, and lastly, other clocks now in development with accuracies that surpass even the best fountain clocks.

4.2.2.1 Timekeeping with Cesium Fountain Clocks

The resonant frequency of interest in cesium clocks is in the microwave regime. Therefore, a maser (Microwave Amplification by Stimulated Emission of Radiation) is often used as the flywheel. Masers can be active or passive and can be used as flywheels or standalone clocks. Hydrogen masers, which generate a signal at the 21cm line, have been used extensively in timekeeping. For more information about such masers, see Ref. [93].

In fountain clocks, a collection of atoms is shot upward into a long tube and are subsequently left to fall under the influence of gravity. The atoms interact twice with the flywheel at the bottom of the tube: once at the beginning of their trajectory and once at the end. This is in contrast to older beam clocks which shot a collection of atoms horizontally. Again, these atoms would interact
with the flywheel twice: once at the beginning and once at the end of the tube. The atoms in a beam clock necessarily move with a high velocity; this limits the amount of time they spend in the tube. As will be discussed in section 4.3, the longer the time between flywheel interactions, the better the frequency deviation between the flywheel and the standard can be ascertained. Therefore, by potentially increasing this time, newer fountain clocks have a significant advantage over beam clocks. For a more complete overview of these technologies see Ref. [28].

The generation of official U.S. time at NIST is complex and involves more than just the output of frequency standard NIST-F1. In fact, NIST-F1 is used to measure and adjust the frequency of a collection of active hydrogen masers, which, along with a collection of cesium beam clocks, are ultimately used to generate the final timing signal [72]. Thus we see that a clock or clock ensemble may include both active and passive components.

### 4.2.2.2 Optical Clocks

One theoretically straightforward way of making atomic clocks more accurate is to use a higher energy, and therefore, higher frequency transition. Intuitively, a higher frequency signal ticks comparatively more times within a given time interval. Therefore, within such an interval, a counter obtains comparatively more information. Moreover, since each tick corresponds to the passage of less time, missed or extraneous ticks are not as significant.

Many atomic clocks in development are therefore switching to such higher energy transitions - often in optical frequency ranges [90]. Indeed, a clock currently in development at the National Institute of Standards and Technology (NIST) that uses an optical transition in aluminum [84] has much smaller systematic errors than NIST-F1 even though it uses only a single Al$^+$ ion. In fact, a recent experiment [81], comparing an optical clock based on mercury [29] to an aluminum clock demonstrated a frequency ratio uncertainty of $5.2 \times 10^{-17}$, and another, comparing two aluminum clocks, demonstrated a fractional frequency inaccuracy of $8.6 \times 10^{-18}$ in one of the clocks [19]. These clocks and others like them are constructed using ion traps. The ions are manipulated using the same technology and techniques used in ion trap quantum computation. Because of this
correspondence, these clocks are often referred to as “quantum clocks.”

The optimization techniques in this thesis are designed to optimize passive quantum clocks, and therefore, our discussion from now on will focus entirely on these clocks.

4.3 Quantum Metrology and Interferometry

In passive clocks, it is important to accurately measure the frequency deviation of the flywheel. This involves flywheel/standard interaction and is known as a clock “interrogation”. In this section, we discuss interrogations in detail.

4.3.1 State Evolution

To begin, we must revisit quantum mechanical evolution, discussed previously in section 2.2.3. Recall that if an energy eigenstate evolves over total time $T$, its phase is altered according to equation (2.14),

$$U|E⟩ = e^{-\frac{i}{\hbar}HT}|E⟩ = e^{-\frac{i}{\hbar}ET}|E⟩.$$  (4.1)

We now associate computational basis states with states of definite energy. That is, the state $|0⟩$ corresponds to some energy eigenstate $|E_0⟩$ with energy $E_0$ and $|1⟩$ corresponds to an energy eigenstate $|E_1⟩$ with higher energy $E_1 > E_0$. We can also construct tensor product states; the state $|010⟩$, for example, has energy $E_0 + E_1 + E_0$. Here we are assuming that energy is additive for multiple copies of a state. Now consider the evolution of the superposition $|0⟩ + |1⟩$,

$$U (|0⟩ + |1⟩) = e^{-\frac{i}{\hbar}E_0 T}|0⟩ + e^{-\frac{i}{\hbar}E_1 T}|1⟩$$

$$= e^{-\frac{i}{\hbar}E_0 T} \left(|0⟩ + e^{-\frac{i}{\hbar}(E_1 - E_0)T}|1⟩ \right).$$  (4.2)

Define $\omega_0 \equiv (E_1 - E_0)/\hbar$; then $\phi = \omega_0 T$, and we can rewrite the above as

$$U (|0⟩ + |1⟩) = e^{-\frac{i}{\hbar}E_0 T} \left(|0⟩ + e^{-i\omega_0 T}|1⟩ \right)$$

$$= e^{-\frac{i}{\hbar}E_0 T} \left(|0⟩ + e^{-i\phi}|1⟩ \right).$$  (4.3)
In the context of atomic clocks, $\omega_0$ is the resonant frequency we wish to measure; this is the frequency of an absorbed or emitted photon, equivalent to the energy difference $(E_1 - E_0)/\hbar$ between two states of an atom.

As discussed in section 2.2.4, the global phase $e^{-\frac{i}{\hbar}E_0T}$ is irrelevant and does not affect the result of a measurement. Then, from equation (4.3) we see that a measurement of the relative phase of the $|0\rangle$ state and the $|1\rangle$ state is equivalent to a measurement of the number of cycles, $\frac{\phi}{2\pi}$, that have passed during an interval of length $T$. Prior knowledge of the resonant frequency $\omega_0$ then gives us an estimate of the length of the interval, $T$.

Notice that without a clock, we cannot choose the length of such an interval ahead of time. That is, we cannot choose the length of time that an energy eigenstate evolves under its Hamiltonian. If we wish to interrogate an atomic system for time $T$, we can, at best, interrogate for some estimate of $T$, say $T_E$. $T_E$ may be chosen from some secondary, less accurate clock. A measurement of relative phase after time $T_E$ then lets us refine this estimate.

### 4.3.2 Ramsey Spectroscopy

We now formalize this concept into a concrete procedure that we can use to measure the phase $\phi$. Assume we have $N$ qubits. Start with the state $|\psi_0\rangle = |0\rangle^\otimes N$; the notation $|\psi\rangle^\otimes N$ indicates the tensor product of $N$ copies of $|\psi\rangle$. Now apply a $\frac{\pi}{2}$ y-rotation to each qubit (equation (2.22)),

$$
|\psi(0)\rangle = R_y\left(\frac{\pi}{2}\right)^\otimes N|\psi_0\rangle \\
= \frac{1}{2^{N/2}}(|0\rangle + |1\rangle)^\otimes N.
$$

(4.4)

Each qubit then undergoes a $z$ rotation by the angle $\phi$,

$$
|\psi(0)'\rangle = R_z(\phi)^\otimes N|\psi(0)\rangle \\
= \frac{1}{2^{N/2}}(|0\rangle + e^{i\phi}|1\rangle)^\otimes N.
$$

(4.5)
This step is known as “free evolution”, as during this time, there is no external interaction. Next, we apply a second $\frac{\pi}{2}$ $y$-rotation to each qubit

$$|\psi(1)\rangle = R_y\left(\frac{\pi}{2}\right)^{\otimes N}|\psi(0)\rangle$$

$$= \frac{1}{2^N}\left((1-e^{i\phi})|0\rangle + (1+e^{i\phi})|1\rangle\right)^{\otimes N} \quad (4.6)$$

Finally, $|\psi(1)\rangle$ is measured in the $|0\rangle, |1\rangle$ basis. That is, each qubit is measured independently with a POVM with elements $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$. Each measurement yields classical outcome 0 with probability $p_0 = \sin^2(\frac{\phi}{2})$ and outcome 1 with probability $p_1 = \cos^2(\frac{\phi}{2})$. Note that this latter combination of a $y$-rotation and a computational basis measurement is completely equivalent to measuring each qubit independently in the $|-\rangle, |+\rangle$ basis.

Notice that throughout this scheme, the state consists of an $N$-fold tensor product of identical qubits. This implies that, at each step, the $N$ qubits remain independent of one another. In fact, one application with $N$ qubits is equivalent to $N$ applications with a single qubit. Because of this independence, a measurement of all $N$ qubits (or all $N$ runs) is distributed binomially. That is, the probability that a measurement of the first $k$ qubits yields classical outcome 0, and a measurement of the remaining $N-k$ qubits yields classical outcome 1 is given by

$$\binom{N}{k} p_0^k(1-p_0)^{N-k}. \quad (4.7)$$

Therefore, the average number of qubits in the 0 state is

$$\langle n_0 \rangle = Np_0 = N \sin^2(\frac{\phi}{2}), \quad (4.8)$$

with variance

$$(\delta n_0)^2 = Np_0(1-p_0) = N \sin^2(\frac{\phi}{2}) \cos^2(\frac{\phi}{2}). \quad (4.9)$$

By observing the number of qubits in the 0 or 1 state, we obtain some information about $\phi$. Notice that such a measurement is limited by its fundamentally probabilistic nature. This effect is characterized as a type of measurement noise, often referred to as “quantum projection noise”.

This scheme is known as “Ramsey interferometry”, or “Ramsey spectroscopy” [74]. It is ubiquitous in quantum metrology, particularly in interferometry and passive clock interrogation.
4.3.2.1 Implementation and Rotating Reference Frames

In most passive clocks, like the cesium beam and fountain clocks described in section 4.2.2, the flywheel and the standard interact twice. These interactions correspond to the y-rotations in the Ramsey technique above. The period of free evolution occurs when the atoms traverse the tube.

In order to induce the initial transform $|0\rangle \rightarrow |0\rangle + |1\rangle$ to each qubit with high probability, the flywheel must be oscillating at a frequency $\omega$ close to $\omega_0$. Therefore, each qubit must evolve via a Hamiltonian of the form

$$H = H_F + H_I$$

$$H_F = E_F \left( -\sin(\omega t) \frac{\sigma_x}{2} + \cos(\omega t) \frac{\sigma_y}{2} \right)$$

$$H_I = \frac{\hbar \omega_0}{2} \sigma_z,$$

(4.10)

where $E_F$ is a constant. $H_F$ is the piece of the Hamiltonian due to an interaction with a classical field locked to the flywheel, and $H_I$ is the internal Hamiltonian which the atomic system obeys in the absence of such an interaction. Notice that $H_F$ is rotating around the $z$ axis at frequency $\omega$. The implementation of $H_F$ depends on the technology used. In a cesium clock, for example, it may correspond to a maser, while in an optical clock, it may be due to an appropriately tuned laser.

At $t = 0$, $H_F = E_F \frac{\sigma_y}{2}$. Then, for sufficiently large $E_F$ and sufficiently small $|\omega - \omega_0|$, $H_F$ induces a $y$-rotation in time short compared to the free evolution time $T$. During the period of free evolution, $H_F$ is shut off; afterward from $H_I$, the $|1\rangle$ state picks up the phase $\phi = \omega_0 T$ as in equation (4.3). While, during this time, the flywheel doesn’t interact with the atomic system, it is still oscillating at frequency $\omega$. Therefore, when $H_F$ is applied again, after the period of free evolution, it does not induce a $y$ rotation, but rather, a rotation around the axis $-\sin(\omega T)\hat{x} + \cos(\omega T)\hat{y}$.

More concisely, in a passive atomic clock, if each atom starts in the state $|0\rangle$, an interrogation acts as

$$R_{-\sin(\omega T)\hat{x} + \cos(\omega T)\hat{y}}(\frac{\pi}{2}) R_z(-\omega_0 T) R_y(\frac{\pi}{2}) |0\rangle.$$  

(4.11)

Note that the frequency of the flywheel varies over the course of an interrogation. Therefore, here
we treat $\omega$ as the average frequency of the flywheel over time $T$. It can be shown that the sequence of operations (4.11) produces identical measurement statistics to the sequence of operations

$$R_y(\frac{\pi}{2}) R_z ((\omega - \omega_0)T) R_y(\frac{\pi}{2})|0\rangle,$$

which correspond to the theoretical description of Ramsey spectroscopy above with $\phi = (\omega - \omega_0)T$. The latter, (4.12), is how Ramsey spectroscopy is usually described. Note that this is equivalent to a description in a frame rotating at frequency $\omega$ with $H_F$.

### 4.3.3 Interferometry

Atomic interrogation is formally equivalent to interferometry. Consider a Mach-Zehnder interferometer as depicted in figure 4.1. A beam splitter is used to divide a beam of light, which then travels down two paths of differing lengths. A second beam splitter is used to recombine these divergent beams just before a measurement. In this context, the computational basis state $|0\rangle$ represents a single photon that takes path 0, and the state $|1\rangle$ represents a single photon that takes path 1. Given 50/50 beam splitters, the state after the first such splitter is given by $|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$. If a photon of frequency $\omega$ takes time $T_0$ to traverse path 0, such a photon picks up phase $\phi_0 = \omega T_0$. Likewise, if a photon takes time $T_1$ to traverse path 1, it picks up phase $\phi_1 = \omega T_1$. Because the lengths of these two paths are different, $T_0 \neq T_1$. Then just before measurement, $|\psi\rangle$ is

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left( e^{-i\phi_0} |0\rangle + e^{-i\phi_1} |1\rangle \right)$$

$$= \frac{1}{\sqrt{2}} e^{-i\phi_0} \left( |0\rangle + e^{-i(\phi_1-\phi_0)} |1\rangle \right).$$

(4.13)

A measurement of relative phase then provides information about the difference between the lengths of paths 0 and 1. Notice that equation (4.13) is formally equivalent to equation (4.3). Perhaps this is to be expected given the similarity of measuring progression along spatial vs. temporal axes, as discussed in section 4.1.
Figure 4.1: Light coming from the left is incident on a beam splitter which divides the light into the shorter, upper path and the longer, bottom path. The light is then recombined at the second beamsplitter. Afterwards, a measurement of relative phase provides some information about the difference in path length.

4.3.4 Parameter Estimation

It will be useful to generalize this procedure to generic parameter estimation. Here we apply some quantum operation $\Omega(x)$ that depends on the potentially continuous parameter $x$ to a chosen state $\rho$; our goal is to estimate the value of $x$ based on a measurement of $\rho$. We split parameter estimation into the following steps:

1. **Preparation**: Starting from some fixed initial state, $\rho_0$, prepare $\rho$ via the application of a chosen unitary, $\rho = U\rho_0U^\dagger$. Without loss of generality, we can take $\rho_0 = |0\rangle\langle 0|^N$.

2. **Query**: The quantum operation $\Omega(x)$ is applied to $\rho$. $\rho' = \Omega(x)(\rho)\Omega^\dagger(x)$.

3. **Measurement**: $\rho'$ is measured with a chosen POVM. The measurement outcome is processed classically to formulate an estimate of $x$.

We often split $\rho$ into two systems $Q$ and $A$. The unitary $U$ and the POVM can act on both systems, but the quantum operation $\Omega$ is restricted to act only on $Q$. This is known as an entanglement assisted scheme, as it can exploit entanglement between $Q$ and $A$; for some quantum
operations, it can outperform schemes where $\rho$ is restricted to the space $Q$ [40]. We can also iterate this procedure, repeating steps 1 and 2 indefinitely before measuring. Again, for certain quantum operations, this can outperform one-round schemes. The most general procedure then corresponds to applying the operations

$$
\rho' = \Omega(x)^Q U(t_f)^{QA} \ldots U(1)^{QA} \Omega(x)^Q \rho^{QA} \Omega^{QA}(x) U^\dagger(1) \ldots U^\dagger QA(t_f) \Omega^{QA}(x),
$$

followed by a measurement with a POVM. Notice that this is identical to equation (3.2), describing the execution of a quantum algorithm. This equivalence is at the heart of this thesis.

Notice that passive clock interrogation and interferometry are simply special cases of parameter estimation, and Ramsey interrogation corresponds to a particular one round choice of $U$ and $\Omega$. In passive clock interrogation, the parameter of interest is $\omega$, the frequency of the flywheel. While this thesis emphasizes clocks, many of the techniques developed here work for any form of parameter estimation, and we therefore expect many applications beyond those explicitly discussed. Interferometry alone is of great importance; it has applications ranging from the study of the properties of light, to astronomy and the detection of gravitational waves.

### 4.3.5 Standard Quantum Limit

Recently, the study of quantum interferometry and metrology has made great strides. A considerable amount of research has been devoted to beating the so called “standard quantum limit” (SQL), also known as the “shot noise limit”. This is a bound on the statistical error in estimates of the parameter, $x$.

Recall that in Ramsey interferometry, the variance of the number of qubits found in the $|0\rangle$ state after a measurement is $(\delta n_0)^2 = N \cos^2(\frac{\phi}{2}) \sin^2(\frac{\phi}{2})$. Using this, we can compute the standard deviation of $\phi$ via

$$
\delta \phi = \left| \frac{\partial \phi}{\partial \langle n_0 \rangle} \right| \delta n_0 = \frac{1}{\sqrt{N}}.
$$

The scaling of the uncertainty of $\phi$ (or equivalently, $t$) with $N$ as $1/\sqrt{N}$ is the standard quantum limit. This is, in fact, somewhat of a misnomer, as first, this “limit” can be beaten, and second, it
is quantum mechanics that allows us to measure more efficiently. Indeed, this limited $1/\sqrt{N}$ scaling is due to the independence of the $N$ qubits used, and could be anticipated without the analysis above. Specifically, from the central limit theorem, we would expect the standard deviation of a quantity estimated via $N$ independent measurements to go as $1/\sqrt{N}$. Note that often, the standard quantum limit is defined in terms of Ramsey interferometry; that is, we can say that the SQL is defined to be the maximum precision with which phase can be measured via Ramsey interferometry.

We can also compute the variance in $\omega$ as $\delta \omega = \frac{1}{\sqrt{NT}}$. Therefore, longer interrogation times result in more precise measurements of $\omega$. In practice, $T$ is limited by experimental constraints. Furthermore, we must be careful that $T$ is not too large as a measurement of $\langle n_0 \rangle$ or $\langle n_1 \rangle$ is unable to distinguish $\phi$ from $\phi + 2\pi$, due to the ambiguity in the inversion of the trigonometric functions as in equation (4.8). This problem is often referred to as “fringe hopping”.

4.3.6 Heisenburg Limit

While the SQL can be beaten, quantum mechanics does place a bound on the accuracy with which phase and time can be measured, albeit, one much less limiting than the SQL. This bound is derived from the time/energy form of the Heisenburg uncertainty principle,

$$\delta E \delta t \geq \frac{\hbar}{2}.$$  \hspace{1cm} (4.16)

In order to minimize $\delta t$, we can maximize $\delta E$. An $N$ qubit quantum state with maximal energy uncertainty is

$$\frac{1}{\sqrt{2}} (|00\ldots0\rangle + |11\ldots1\rangle).$$  \hspace{1cm} (4.17)

This is a maximally entangled state, also known as a “cat state”, which we abbreviate as $\frac{1}{\sqrt{2}} (|0\rangle + |N\rangle)$. When measured, this state collapses to an eigenstate with energy $NE_0$ with probability $\frac{1}{2}$ and an
eigenstate with energy $NE_1$ with probability $\frac{1}{2}$. Then, the variance of such a measurement is

$$\langle \delta E \rangle^2 = \langle E^2 \rangle - \langle E \rangle^2$$

$$= \frac{1}{2} (NE_1)^2 + \frac{1}{2} (NE_0)^2 - \left( \frac{1}{2} NE_1 + \frac{1}{2} NE_0 \right)^2$$

$$= N^2 \left( \frac{1}{4} E_1^2 - \frac{1}{4} E_0^2 + \frac{1}{2} E_1 E_0 \right).$$

Therefore, $\delta E \propto N$, and from the Heisenberg uncertainty principle, $\delta t \geq k/N$ where $k$ is a constant. This is referred to as the Heisenberg limit, and is fundamental - assuming no additional prior knowledge, time cannot be measured any more precisely.

We can achieve this bound by using a cat state in an interferometric procedure along the lines of Ramsey interferometry. Consider starting with equation (4.17) instead of equation (4.4). As before let this state freely evolve for a time equivalent to that in equation (4.5). Then, up to an irrelevant global phase, we have the state

$$\frac{1}{\sqrt{2}} \left( |0\rangle + e^{iN\phi} |N\rangle \right).$$

Notice that the large energy uncertainty has turned into a large phase differential. Now, measure with projectors $P_0 = \frac{1}{\sqrt{2}} (|0\rangle - |N\rangle)$ and $P_1 = \frac{1}{\sqrt{2}} (|0\rangle + |N\rangle)$. We have $p_0 = \sin^2 \left( \frac{N\phi}{2} \right)$ and $p_1 = \cos^2 \left( \frac{N\phi}{2} \right)$. Following the same analysis as above, we have, via equation (4.15)

$$\delta \phi = \frac{1}{N}.$$ 

Thus, maximally entangled states achieve the fundamental Heisenberg bound.

Note that this does not imply that these states are optimal for timekeeping or quantum interferometry. In particular, these states are not necessarily optimal if some information about $\phi$ is known in advance. Furthermore, these states are known to perform poorly if the flywheel or the atomic system is subject to certain types of noise.

4.3.7 Prior Work

The derivation of high performing states and measurements has been an active and rich area of research. Here we highlight a few results of particular relevance, often skimming over much of
the physics needed to understand these results in depth.

The first proposals to demonstrate an increase in precision used a class of entangled states known as spin squeezed states \([54, 95, 96]\). These states have some degree of reduced uncertainty in spin along one particular axis of the Bloch sphere, and consequently, increased uncertainty along an orthogonal axis. Not long afterward, work by Bollinger et al. \([16]\) demonstrated that the Heisenberg bound could be achieved by the maximally entangled state described in section 4.3.6 above. These ideas were demonstrated in practice by Meyer, et. al. \([67]\) with a set of experiments in which different types of entangled states were shown to beat the standard quantum limit.

Subsequently, Huelga et. al \([49]\) demonstrated that maximally entangled states cannot beat the SQL in the presence of atomic decoherence. Rather, they showed via numerical optimization that, for their choice of noise model, a partially entangled state maximizes performance. These results were generalized and expanded on in Refs. \([86, 33]\) where precision loss was more thoroughly quantified for specific decoherent processes. Studies in optical interferometry using NOON states, \([39, 82]\) demonstrated a similar loss of Heisenberg scaling in the presence of photon loss.

These results spurred research into schemes that could beat the SQL even in the presence of atomic noise. Huver et. al. \([51]\) showed that this can be accomplished with a class of entangled Fock states. Refs. \([63, 61, 45, 15]\) use multi-round strategies, where phase evolution is intertwined with arbitrary unitary operations, as in equation (4.14). Refs. \([61, 45]\) use unentangled states, and Refs. \([45, 15]\) use an adaptive strategy where the unitary that is applied at step \(i\) depends on the measurement result of step \(i - 1\).

Dorner \([30]\) observed that Ref. \([49]\) and subsequent work assumed that the noisy quantum system is subject to uncorrelated dephasing, while it is, in fact, correlated dephasing that is observed in ion traps. He proposes a scheme that can beat this noise by using decoherence free subspaces. An optical experiment by Kacprowicz et. al. \([62]\), using states engineered by Dorner et. al \([31]\), demonstrated performance superior to the SQL and NOON states.

Andre et. al. \([10]\) studied the performance of a family of spin squeezed states in a full sim-
ulation of a passive atomic clock. Unlike prior work, both flywheel noise and atomic decoherence are considered in the more realistic regime where flywheel noise is dominant. They showed that maximally entangled states perform poorly in this regime, and derive a set of optimal, partially squeezed states that achieve phase measurement scaling of $1/N^{3/2}$. Recently, Rosenband [79] evaluated the performance of many promising interrogation schemes on a numerical simulation of a clock and estimated that quantum entanglement would result in a 15-20% decrease in Allan variance. Allan variance is a commonly used measure of clock performance that will be discussed in chapter 6.

Of greatest relevance to us are works that seek to optimize the average value of a cost function

$$\langle C \rangle = \int \int C(\omega, a)p(a|\omega)p(\omega)d\omega da$$

$$= \int \int C(\omega, a)\text{tr}(P_a\rho_\omega(t_f))p(\omega)d\omega da. \quad (4.21)$$

Optimizations of this form were first considered by Holevo [46]. $C$ is arbitrary and is a function of both the frequency of the flywheel, $\omega$ (or equivalently, the deviation of the frequency of the flywheel from that of the standard), and the measurement outcome $a$. With each measurement we associate an estimate of the flywheel’s deviation, $f_a$, which is typically used inside $C$ in place of $a$. Note that here, we usually take $\omega$ to be the average frequency over the interrogation interval of interest. Equation (4.21) is then an average over all possible frequencies and measurement outcomes, where $p(\omega)$ is the prior probability distribution of average flywheel frequencies, and $p(a|\omega)$ is the probability that outcome $a$ is obtained if a flywheel at average frequency $\omega$ is interrogated and subsequently measured. As indicated above, this is given by $\text{tr}(P_a\rho_\omega(t_f))$ where $P_a$ is a POVM element, and the notation $\rho_\omega(t_f)$ indicates that the atomic system $\rho$ was interrogated by a flywheel of average frequency $\omega$. This is entirely analogous of the description of expected algorithmic cost in section 3.3.4. Indeed, equation (3.15) is a discretized version of equation (4.21).

Here we build extensively on work by Buzek et. al. [18], which derives states and measurements that optimize equation (4.21) with the choice of cost function $C(\omega, a) = 4\sin^2\frac{(\omega-f_0)}{2}$. There, only single round strategies are considered. This work was expanded on, however, by Ref. [91],
which considers multi-round strategies. They demonstrate that for all cost functions within a particular class, the optimal measurement consists of a quantum fourier transform in an appropriate basis. They then find optimal initial states for a couple of choices of $C$.

Both Ref. [18] and Ref. [91] optimize equation (4.21) under the assumption that all flywheel frequencies are equally likely. That is, they assume the prior probability distribution, $p(\omega)$ is uniform. While there are some domains where this may be a reasonable assumption, it is not in the context of atomic clocks. As a clock runs, we often know that the flywheel frequency is in a narrow range around $\omega_0$, and as this thesis will show, we can, in principle, know a great deal more. Much of the prior work above implicitly makes the uniform prior assumption as well. Indeed, as mentioned in section 4.3.6 and discussed more completely in Ref. [15], the Heisenberg limit is only valid when there is zero prior knowledge. It can be beaten in other circumstances. Demkowicz-Dobrzanski addressed this problem in Ref. [23]. He optimized the same cost function considered in Ref. [18], $C(\omega, f_a) = 4 \sin^2 \frac{(\omega-f_a)}{2}$, for single round strategies with arbitrary $p(\omega)$. The approach used is largely analytical, but requires the numerical maximization of a trace norm. The periodic cost functions studied in Refs. [18, 23] are convenient for analytical studies of clock optimization or interferometry but do not penalize phase errors greater than $2\pi$, even though they correspond to frequency estimates far from the true frequency of the flywheel. This issue becomes important when multiple interrogations and long-term clock stability are considered.

The optimization technique that we present in the following chapter subsumes many of these optimization schemes. We optimize (4.21) with arbitrary cost function, prior, and for any number of rounds.
Chapter 5

Optimal Interrogation

In this chapter, we present a strategy for passive atomic clock interrogation that is fully optimal up to discretization error. Per the discussion in section 4.3, the techniques presented here will work for any form of parameter estimation. They can also be used to extend the adversary method to continuous problems. However, since in the following chapter we use this strategy to optimize the evolution of a passive clock, our results are presented in that context.

5.1 Interrogation Algorithms

Recall that the most general way to interrogate a frequency standard corresponds to the sequence of operations, equation (4.14), written in terms of $\omega$, the average frequency of the flywheel, which is now the parameter of interest,

$$\rho(t_f) = \Omega(\omega) U(t_f)^Q \ldots U(1)^QA \Omega(\omega)^Q \rho^{QA}(0) \Omega(\omega)^Q U(1)^QA \ldots U(t_f)^QA \Omega(\omega)^Q.$$  \hspace{1cm} (5.1)

Here, $\rho^{QA}(0)$ is the initial state of the frequency standard, which we take to be

$$\rho^{QA}(0) = U^{QA}(0)|0\rangle \langle 0| U^{Q\dagger A}(0).$$  \hspace{1cm} (5.2)

Each $U(t)$ is an arbitrary unitary operator which can act on system $Q$, the Hilbert space of the frequency standard, and system $A$, an additional, optional system of ancillas. $\Omega^Q$ describes the interaction of the flywheel and the frequency standard. As discussed in section 4.3, we can choose $\Omega^Q(\omega) = R_z((\omega - \omega_0)T)$. Recall that the process of applying $\Omega^Q(\omega)$ is referred to as a query.
Without loss of generality, for the remainder of this thesis we take $\omega_0 = 0$; then $\Omega^Q(\omega) = R_z(\omega T)$. Equivalently, we can take $\omega$ to be the average frequency deviation of the flywheel.

Notice again the equivalence of equation (5.1) and equation (3.2). This equivalence lets us identify a clock interrogation with a quantum algorithm. That is, we can say that clock interrogation is a particular type of quantum algorithm. We will refer to such algorithms as “interrogation algorithms.”

Our goal in this chapter is to optimize the performance of interrogation algorithms. Recall that we can quantify the performance of a clock interrogation via the average value of a cost function, equation (4.21),

$$\langle C \rangle = \int \int C(\omega, a) \text{tr}(P_a \rho_\omega(t_f)) p(\omega) d\omega da.$$  

(5.3)

for some choice of $C(\omega, a)$. In chapter 3 we quantified the performance of an algorithm in precisely the same way. Indeed, equation (5.3) is nearly identical to equation (3.15). Here, just as before, with each measurement outcome $a$, we associate a guess as to which oracle was applied, $g(a)$. But now, these oracles correspond to the average frequency of the flywheel, and therefore, our guesses, now denoted by $f_a \equiv g(a)$, are estimates of this frequency. Moreover, in chapter 3 we constructed a semidefinite program designed to optimize just such a functional. As discussed, we can use this program to choose an optimal set of $U$’s and an optimal POVM. In what follows, we show how to apply this program to generate optimal interrogation algorithms.

In this chapter, we focus on the quadratic cost function $C(\omega, a) = (\omega - f_a)^2$. Intuitively, this choice minimizes the error in estimates of the flywheel’s frequency; precise knowledge of this frequency facilitates effective assignments of timestamps.

For this choice of $C(f, a)$, the optimal choice of $f_a$ is $a \rightarrow f_a = E(\omega|a)$ since equation (5.3) is equivalent to

$$\langle C \rangle = \int E(C(\omega, a)|a) p(a) da$$

$$= \int E((\omega - f_a)^2|a) p(a) da,$$  

(5.4)
and
\[ \argmin_{f_a} \mathbb{E} \left( (\omega - f_a)^2 | a \right) = \mathbb{E}(\omega|a). \] (5.5)

Note that the posterior expectation \( \mathbb{E}(\omega|a) \) is not known until after an interrogation, and so this optimal choice cannot be made ahead of time. This issue is discussed further in section 5.2.3.

Thus we see that a minimization of \( \langle C \rangle \), and therefore, the algorithms we construct, minimize the expected posterior variance of \( \omega \):
\[ \mathbb{V}(\omega) = \int \mathbb{V}(\omega|a)p(a)da \]
\[ = \int \mathbb{E}(\omega - \mathbb{E}(\omega|a))^2 p(a)da. \] (5.6)

### 5.2 The Clock SDP

#### 5.2.1 Dicke States

First, we observe that we can restrict the Hilbert space of system \( Q \) to the completely symmetric subspace, consisting of so-called Dicke states. These states are equal amplitude superpositions of all computational basis states with the same Hamming weight (number of 1’s in the state label). For example, in a 2 qubit system the Dicke states are given by \( |0\rangle = |00\rangle \), \( |1\rangle = (|01\rangle + |10\rangle)/\sqrt{2} \), and \( |2\rangle = |11\rangle \). In a system of \( N \) qubits, there are \( N + 1 \) such states.

Within the symmetric subspace we can express the clock oracle as
\[ \Omega(\omega) = e^{-iJ_z \omega T}, \] (5.7)
where \( J_z \) is the total \( z \) angular momentum operator, \( J_z |k\rangle = (k - N/2)|k\rangle \), for Dicke states \( |k\rangle \). By making a global phase change, we can write the evolution of these states as
\[ |k\rangle \rightarrow e^{-ik\omega T}|k\rangle. \] (5.8)

Ref. [18] shows that nothing can be gained by considering other states of two-level systems in one-query algorithms. Below we present a more complete multi-query argument, which can be seen, in a more general context, as the observation that if an oracle acts trivially within a subspace, that subspace plays no role in the optimal algorithm.
Theorem 9. The minimum average cost (equation 5.3) of an $N$ atom clock interrogation applied according to equation (5.1) with $\Omega^Q(\omega) = R_2^\otimes N(\omega)$ is identical to the minimum average cost obtained if the unitaries in equation (5.1) act only on the symmetric subspace of system $Q$ (but can act arbitrarily on $A$).

Proof. The density matrix $\rho^Q$ affects the average cost via the two constraints

$$\rho^Q(t) = \text{tr}_Q(\rho^{OQ}(t))$$

$$\rho^Q(t) = \text{tr}_Q(\Omega^{OQ} \rho^{OQ}(t - 1) \Omega^{OQ\dagger}).$$

Notice that the off diagonal entries of $\rho^Q$ get lost in the partial trace. (For slightly more details about this, see section 5.2.5). Therefore, we can write $\rho^Q$ generally as

$$\rho^Q = \sum_k \sum_{m=1}^{M(k)} p_{k,m} |k,m\rangle \langle k,m|,$$

(5.9)

where the label $k$ corresponds to the Hamming weight of a state, $m$ labels states with that Hamming weight, and $M(k)$ is the number of states with Hamming weight $k$ (i.e. the number of $m$’s corresponding to a particular $k$). Then the $(x,y)$ block of $\rho^{OQ}$ can be written as

$$|x\rangle\langle y|\rho^Q_{x,y} = \sum_{x,y} \sum_{k,m} p_{x,y,k,m} |x\rangle \langle y| \otimes |k,m\rangle \langle k,m|.$$

(5.10)

Note that here, for clarity, we have assumed that the set of oracles have been discretized, but our argument applies equally well in the continuous case.

We will show that, for all $t$, $\rho^O(t)$, and therefore the average cost, is unaffected by the degree of freedom labeled by $m$, and thus, $\rho^Q(t)$ or equivalently, the unitaries in equation (5.1) can be restricted to the symmetric subspace. Specifically, we show that $\rho^O(t)$ remains the same if the querier density matrix in equation (5.10) is replaced such that

$$|x\rangle\langle y|\rho^Q_{x,y} = \sum_{x,y} \sum_k p_{x,y,k} |x\rangle \langle y| \otimes |k\rangle \langle k|$$

(5.11)

where $k$ is now a label for Dicke states,

$$|k\rangle = \sum_{m=1}^{M(k)} \frac{1}{\sqrt{M(k)}} |k,m\rangle$$

(5.12)
and
\[
p_{k,y} = \sum_{m=1}^{M(k)} p_{x,y,k,m}.
\] (5.13)

This is clearly true for the first constraint above since all diagonal entries of \( \rho^Q \) are simply being summed. Expanding the second constraint,

\[
\rho^O(t) = \text{tr}_Q \left( \Omega^T \rho^Q(t-1) \Omega^O \right)
\]
\[
= \text{tr}_Q \left( \sum_{x'} |x'\rangle \langle x'| \sum_{x,y} |x\rangle \langle y| \rho^Q_{x,y}(t-1) \sum_{y'} |y'\rangle \langle y'| \Omega^\dagger (y') \right).
\] (5.14)

If \( \rho^Q_{x,y} \) is expressed generally, according to equation (5.10), we have

\[
\rho^O(t) = \text{tr}_Q \left( \sum_{x,y} |x\rangle \langle y| \Omega (x) \sum_{k,m} p_{x,y,k,m} |k,m\rangle \langle k,m| \Omega^\dagger (y) \right)
\] (5.15)

The oracle acts as \( \Omega(x)|k,m\rangle = e^{-if(x,k)}|k,m\rangle \) where the phase \( f \) depends only on the Hamming weight of the state and the oracle that is applied. Thus,

\[
\rho^O(t) = \text{tr}_Q \left( \sum_{x,y} |x\rangle \langle y| \sum_{k,m} p_{x,y,k,m} e^{-if(x,k)} |k,m\rangle \langle k,m| e^{if(y,k)} \right)
\] (5.16)

If \( \rho^Q_{x,y} \) is restricted to the symmetric subspace, as in equation (5.11), then instead of equation (5.15) we have

\[
\rho^O(t) = \sum_{x,y} \sum_{k} |x\rangle \langle y| p_{x,y,k} e^{-if(x,k)} e^{if(y,k)}
\] (5.17)

which is equivalent to the last line of equation (5.16). Therefore, we obtain an identical cost if \( \rho^Q \) is restricted to the symmetric subspace. As noted above, this is because the oracle acts identically on states with identical Hamming weights.
5.2.2 Oracles and Probabilities

In the context of generic quantum algorithms, the oracle $\Omega$ was parameterized by a label $x$ which corresponded to the algorithm’s input. The optimal algorithm then depended on the probability distribution of possible inputs, $p(x)$. For example, recall that in SEARCH, if the marked element was in position 2 with 100% probability, the optimal algorithm (the one line algorithm, \textbf{return} 2) looked very different than if the marked element had an equal probability of being in all possible positions (Grover’s algorithm).

In the context of clock interrogations, the oracle is now parameterized by $\omega$, the frequency of the flywheel, as the interaction between the flywheel and the standard, equation (5.7) depends on this frequency. Then, just as above, the optimal algorithm depends on a probability distribution, in this case, that of possible flywheel frequencies. That is, as expected, some knowledge of the flywheel’s frequency improves the performance of an interrogation.

In this context, the joint oracle operator takes the form

$$\Omega^{OQ} = \sum_k \int d\omega |\omega\rangle \langle \omega| \otimes |k\rangle \langle k| e^{-ik\omega T}.$$  \hfill (5.18)

Utilizing such a set of clock oracles in an SDP presents a problem, however, as $\omega$ is a continuous variable, and therefore, we have an infinite set of potential oracles. Furthermore, if as in chapter 3, measurement outcomes are mapped to oracles, we require a POVM with an infinite number of elements. Each of these systems must be finite in order to apply semidefinite programming; therefore in what follows, we restrict $\omega$ and $\alpha$ to a finite set.

It then becomes necessary to make an estimate of the difference between the finite SDP’s optimal cost and that of the infimum of the costs of solutions to the continuous problem. In the following two sections, we separately bound the error due to choosing a finite set of $\omega$ (oracle error) and a finite set of frequency estimates (querier error). As we will see later, it is not strictly necessary to use a finite prior; oracle continuity can be handled in an alternative, largely superior way.

Throughout, we use the notation $S_C(p, F)$ to refer both to the clock SDP and the cost
obtained by the clock SDP, given a prior probability distribution of flywheel frequencies $p$ and a set of frequency estimates $F$. We begin with querier error.

### 5.2.3 Querier Error

#### 5.2.3.1 Number of Frequency Estimates

In section 3.3.4, we argued that we needed at most $|O|$ measurement outcomes. Since $|O|$ is now continuous this is no longer tenable. Fortunately, a smaller set will suffice. In this section we examine how many frequency estimates $f_a$ are needed so that $S_C(p, F) = S_C(p, \mathbb{R})$. One can obtain an upper bound by observing that if the rank of the final oracle density matrix is $r$, then the optimal POVM can always be reduced to at most $r^2$ elements [22], implying that a set $F$ with $|F| \leq r^2$ suffices. This is shown by theorem 10.

**Theorem 10.** An SDP with $r^2$ measurement outcomes can achieve the same expected cost as an SDP with greater than $r^2$ measurement outcomes, where $r$ is the rank of the final oracle density matrix, $\rho^O(t_f)$.

**Proof.** If there are $n > r^2$ measurement outcomes, then there is a nonzero linear combination of the $n$ remotely prepared $\sigma_a$ such that $\sum_a \alpha_a \sigma_a = 0$, where one coefficient satisfies $\alpha_0 = -1$ and all other coefficients satisfy $\alpha_a \geq -1$.

Define a new set $\{\sigma_a^{(\lambda)}\}_a$ where

$$\sigma_a^{(\lambda)} = (1 + \lambda \alpha_a) \sigma_a$$  \hspace{1cm} (5.19)

Here $\lambda$ is a constant chosen in the range $\lambda \in [\epsilon_L, \epsilon_H]$. This range is restricted in order to maintain $\sigma_a^{(\lambda)} \geq 0$ for all $a$. The exact values of $\epsilon_L$ and $\epsilon_H$ depend on the $\alpha_a$’s; however, in all cases, $\epsilon_L < 0$ and $\epsilon_H > 0$. Remote state preparation requires that this new set must satisfy $\sum_a \sigma_a^{(\lambda)} = \rho^O(t_f) =$
\[ \sum_a \sigma_a. \] Expanding the \( \sigma_a^{(\lambda)} \), we see that,

\[
\sum_a \sigma_a^{(\lambda)} = \sum_a (1 + \lambda \alpha_a) \sigma_a \\
= \sum_a \sigma_a + \lambda \sum_a \alpha_a \sigma_a \\
= \sum_a \sigma_a,
\]

as required.

Given this new set, the expected cost is

\[
\langle C \rangle^{(\lambda)} = \sum_a \text{tr}(\sigma_a^{(\lambda)} A_a) = \sum_a \text{tr} ((1 + \lambda \alpha_a) \sigma_a A_a) \\
= \sum_a \text{tr}(\sigma_a A_a) + \lambda \sum_a \text{tr}(\alpha_a \sigma_a A_a) \\
= \langle C \rangle^{(0)} + \lambda \Delta.
\]

where \( \Delta = \sum_a \text{tr}(\alpha_a \sigma_a A_a) \). This is a linear function of \( \lambda \). Since \( \sigma_a \) are chosen optimally, it achieves its minimum value at \( \lambda = 0 \). Any linear function whose domain contains negative numbers, positive numbers, and 0, which takes its minimum value at 0 must be the constant function. Therefore \( \Delta = 0 \), and the new set \( \{ \sigma_a^{(\lambda)} \}_a \) achieves the same expected cost. If \( \lambda = 1 \), the set is reduced by one element, as \( 1 + \alpha_0 = 0 \). We can continue reducing the size of the set in this manner until we are left with a set containing at most \( r^2 \) elements.

Note that for \( t_f = 1 \) and some costs and priors, only \( r \) elements are needed [25, 23]. In addition, numerical evidence suggests that \( r \) elements are sufficient for the clock SDP. The rank of the oracle density matrix is clearly bounded by \( |O| \), but as we will show in section 5.2.4.2, it is also bounded by \( (|Q| - 1)t + 1 \) which can be much smaller.

At present, we have no way of choosing the finite set of frequency estimates optimally. However, we have two effective ways of obtaining a set for which the SDP solution is close to optimal. These are described in the following two sections.
5.2.3.2 Discretization

A straightforward strategy is to run SDP $S_C$ with a large number of uniformly spaced estimates. The POVM extracted from $S_C$ will set any POVM elements associated with unnecessary frequency estimates to zero. We show how to bound the difference between the optimal cost and the cost computed by $S_C$ given these or any other chosen set of estimates. The bounds we obtain apply only to cost functions that meet certain criteria. However, for these cost functions, these bounds are quite tight. We hypothesize that this technique does not result in significant loss of algorithmic performance for other cost functions. We will see that for supported cost functions the querier discretization error can be bounded in a way that depends only on the set of frequency estimates $F = \{f_a\}_a$ and can be made to go to zero by increasing the size and resolution of $F$.

Let $F = \{f_1, \ldots, f_N\}$ with $f_j < f_{j+1}$. We must have $S_C(p, F) \geq S_C(p, F_{\text{opt}})$, where $F_{\text{opt}}$ is the set of estimates achieving the minimum cost. Also, $S_C(p, F_{\text{opt}}) = S_C(p, \mathbb{R})$, since as mentioned above, giving the SDP access to additional estimates does not increase the cost. The following theorem bounds the difference between $S_C(p, F)$ and $S_C(p, \mathbb{R})$ for certain choices of $C(f, a)$.

**Theorem 11.** Let $C$ be a second differentiable, non-negative function satisfying $C''(x) \leq b$, $C(0) = 0$ and $C$ is monotone on $[0, \infty)$ and $(-\infty, 0]$. Define $M(\omega)$ by

$$
M(\omega) = \begin{cases} 
C(\omega - f_1) & \text{for } \omega \leq f_1 \\
C(\omega - f_N) & \text{for } \omega \geq f_N \\
0 & \text{otherwise}
\end{cases}
$$

We have the following inequality:

$$
S_C(p, F) - S_C(p, \mathbb{R}) \leq \max_j \frac{b}{8} (f_{j+1} - f_j)^2 
+ \int M(\omega)p(\omega)d\omega.
$$

**Proof.** Consider an arbitrary interrogation algorithm $Q$. Such an algorithm is determined by choices
for initialization, inter-query unitaries and measurements. Define

$$C(p, Q) = \sum_a \int C(\omega - g_a)p(g_a, \omega|Q)d\omega,$$

(5.25)

where the $g_a$ are $Q$’s frequency estimates. Then $C(p, Q)$ is the expected cost of $Q$ given prior $p$.

Let $g(a)$ be defined by

$$g(a) = \text{argmin}_g \left( \int C(\omega - g)p(a|\omega, Q)p(\omega)d\omega \right).$$

(5.26)

Then $g(a)$ is the optimum frequency estimate $Q$ could make given measurement outcome $a$. Let $Q_g$ be $Q$ modified to make the frequency estimates $g(a)$.

Let $B$ be the expression on the right-hand-side of Eq. (5.24). We show that $C(p, Q_g) \geq SC(p, F) - B$ for any algorithm $Q$. Since $SC(p, \mathbb{R}) = \inf_Q C(p, Q) = \inf_Q C(p, Q_g)$, the result follows. We prove the bound in two steps. In the first step we force the frequency estimates to lie in the interval $[f_1, f_N]$ and in the second we change them to lie in $F$.

For the first step, let $\tilde{g}(a)$ be the value in $[f_1, f_N]$ nearest to $g(a)$. If $\tilde{g}(a) = g(a)$, then

$$C(\omega - g(a)) \geq C(\omega - \tilde{g}(a)) - M(\omega)$$

since $M(\omega) \geq 0$. If $\tilde{g}(a) = f_1$, then one of the following holds:

1. $\omega \geq f_1$, in which case $\tilde{g}(a)$ is nearer $\omega$ and on the same side, so that $C(\omega - g(a)) \geq C(\omega - \tilde{g}(a)) \geq C(\omega - \tilde{g}(a)) - M(\omega)$.

2. $\omega < f_1$, in which case $C(\omega - g(a)) \geq 0 = C(\omega - f_1) - C(\omega - f_1) = C(\omega - \tilde{g}(a)) - M(\omega)$.

A similar argument works for $\tilde{g}(a) = f_N$. Substituting the inequalities in the integral for $C(p, Q)$ we get

$$C(p, Q_g) \geq \sum_a \int C(\omega - \tilde{g}(a))p(a|\omega, Q)p(\omega)d\omega$$

$$- \sum_a \int M(\omega)p(a|\omega, Q)p(\omega)d\omega$$

$$= C(p, Q_{\tilde{g}}) - \int M(\omega)p(\omega)d\omega.$$  

(5.27)

For the second step, we modify $Q_{\tilde{g}}$ to $Q_{\tilde{f}}$, where $\tilde{f}(a)$ is one of the elements of $F$ on either side of $\tilde{g}(a)$. That is, because $f_1 \leq \tilde{g}(a) \leq f_N$, there exists a minimum $j$ such that $f_j \leq \tilde{g}(a) \leq f_{j+1}$,
and we set $\tilde{f}(a)$ to either $f_j$ or $f_{j+1}$. Define $\lambda \in [0,1]$ by $\tilde{g}(a) = \lambda f_j + (1 - \lambda) f_{j+1}$. It is convenient to let $Q_f$ be a “mixed” (randomized) algorithm, where $\tilde{f}(a)$ with probability $\lambda$ and $f_{j+1}$ with probability $1 - \lambda$. Note that a mixed algorithm of this sort cannot be better than the optimal one, that is $C(p,Q_f) \geq S_{C(p,F)}$. To bound the cost, we consider a given $a$ and $\omega$ and estimate the quantity

$$c(\omega, a) = \lambda C(w - f_j) + (1 - \lambda) C(w - f_{j+1}) - C(\omega - \tilde{g}(a)) = \lambda (C(w - f_j) - C(\omega - \tilde{g}(a))) + (1 - \lambda) (C(w - f_{j+1}) - C(\omega - \tilde{g}(a))).$$

(5.28)

Define $\omega_0 = \omega - \tilde{g}(a)$, $\omega_l = \omega - f_{j+1}$ and $\omega_u = \omega - f_j$. We can estimate

$$C(\omega) - C(\omega_0) = (\omega - \omega_0) C'(\omega_0) + \int_0^{\omega - \omega_0} \int_0^x C''(\omega_0 + y) dy dx \leq (\omega - \omega_0) C'(\omega_0) + \frac{1}{2} (\omega - \omega_0)^2 \max_y C''(y) \leq (\omega - \omega_0) C'(\omega_0) + \frac{b}{2} (\omega - \omega_0)^2.$$  

(5.29)

Substituting this bound for each summand of Eq. (5.28) gives

$$c(\omega, a) \leq \lambda((\omega_u - \omega_0) C'(\omega_0) + \frac{b}{2} (\omega_u - \omega_0)^2) + (1 - \lambda)((\omega_l - \omega_0) C'(\omega_0) + \frac{b}{2} (\omega_l - \omega_0)^2) = \frac{b}{2} (\lambda(\omega_u - \omega_0)^2 + (1 - \lambda)(\omega_l - \omega_0)^2) = \frac{b}{2} (\lambda(\omega_u - \omega_0)(\omega_u - \omega_l)) \leq \frac{b}{2} \frac{(f_{j+1} - f_j)^2}{4},$$

(5.30)
where we first applied $\lambda \omega_u + (1 - \lambda) \omega_l = \omega_0$. The next identity requires applying $(1 - \lambda)(\omega_l - \omega_0) = -\lambda(\omega_u - \omega_0)$ to the second summand, and the final inequality is obtained by expanding $\omega_u - \omega_l = f_{j+1} - f_j$ and noting that

$$
\frac{\lambda(\omega_u - \omega_0)}{f_{j+1} - f_j} = \frac{\lambda \tilde{g}(a) - f_j}{f_{j+1} - f_j} = \frac{\lambda (\lambda f_j + (1 - \lambda)f_{j+1}) - f_j}{f_{j+1} - f_j} = \frac{\lambda(1 - \lambda)}{}
$$

is maximized at $\lambda = 1/2$. The definition of $c(\omega, a)$ and inequality Eq. (5.30) can be used to obtain a lower bound for $C(w - \tilde{g}(a))$ that we can substitute in the expression for $C(p, Q_{\tilde{g}})$ as follows:

$$
C(p, Q_{\tilde{g}}) = \sum_a \int C(\omega - \tilde{g}(a))p(a|\omega, Q)p(\omega)d\omega
\geq \sum_a \int (\lambda C(\omega - f_j) + (1 - \lambda)C(\omega - f_{j+1})
- \max_j \frac{b}{8}(f_{j+1} - f_j)^2)p(a|\omega, Q)p(\omega)d\omega
= C(p, Q_{\tilde{f}}) - \max_j \frac{b}{8}(f_{j+1} - f_j)^2
\geq SC(p, F) - \max_j \frac{b}{8}(f_{j+1} - f_j)^2.
$$

To finish the proof, we combine Eqs. (5.27) and (5.32).

The quadratic cost function, $C(f, a) \rightarrow C(x) = x^2$ with $x = f - f_a$, satisfies the conditions of theorem 11, and can therefore be bounded by equation (5.24). A strategy for obtaining an initial choice of an $N$ element $F$ is to optimize the right-hand-side of Eq. (5.24).

5.2.3.3 Estimate Refinement

Recall from section 5.1 that, for the quadratic cost function, the optimal choice for frequency estimate $f_a$ is the posterior mean,

$$f_a = E(\omega|a).
$$

While these cannot be computed until after the SDP has been run, we can use this observation to improve the selection of frequency estimates iteratively. Specifically, we begin with any set
of estimates. We then run the SDP and use the solution to compute the posterior probability distribution for each measurement outcome,

$$p(\omega_x|f_a) = \frac{p(f_a,\omega_x)}{p(f_a)} = \frac{(\sigma_a)_{x,x}}{\text{tr}(\sigma_a)}$$

(5.34)

Next, we compute the mean of each of these distributions:

$$E(\omega|a) = \sum_x \frac{(\sigma_a)_{x,x}}{\text{tr}(\sigma_a)} \omega_x,$$

(5.35)

where we have assumed the prior has been suitably discretized (to be discussed in the next section), and $x$ indexes the finite set of frequencies. Replacing the original estimates by their posterior means is guaranteed to improve the cost without having to change the algorithm. We then run the SDP again, replacing the frequency estimate $f_a$ with $E(\omega|a)$. This procedure is repeated until each estimate is numerically close to its posterior mean. The procedure can be adapted to other costs, but the mean must be replaced by a statistic appropriate for the cost. For example, for $C(x) = |x|$ we compute the median instead of the mean.

In practice, the best strategy is often to start with a large number of estimates, compute the relevant bound according to theorem 11, let the SDP zero out any unnecessary estimates, and then refine the remaining estimates according to the scheme above. In most cases tested, the estimates converged to the optimal set after a small number of refinements.

5.2.4 Oracle Error

5.2.4.1 Discretization

The most straightforward way of handling the continuous system $O$ is to discretize the possible oracle frequencies as $\omega \in \{\omega_1, \ldots, \omega_d\}$. Note that now, the set of frequency estimates and the set of oracle frequencies can be disjoint if desired. The oracle initial state is then given by

$$\rho^O(0) = \sum_{x,y} \sqrt{p(\omega_x)p(\omega_y)}|x\rangle\langle y|,$$

(5.36)

The $p(\omega_x)$ are discretized prior probabilities approximating the continuous prior, and $|x\rangle$ is the oracle basis state corresponding to classical oscillator frequency $\omega_x$. The operator $\Omega^{OQ}$ is now
defined by
\[ \Omega^{OQ}|x\rangle|k\rangle = e^{-ik\omega_x T}|x\rangle|k\rangle. \] (5.37)

The cost operators are given by
\[ (A_a)_{x,y} = C(\omega_x, f_a) \delta(x,y). \] (5.38)

Note that the complete SDP, $S_C$, incorporating oracle discretization as well as the results of section 5.2.3 looks formally identical to SDP $S$ in chapter 3.

In contrast to the previous section, handling oracle discretization error is not particularly direct. Here, a lower bound on the optimal continuous cost is estimated statistically by averaging a number of random oracle discretizations.

As before, let $p(\omega)$ be the prior probability density of clock frequencies. We assume that the prior distribution is absolutely continuous with respect to Lebesgue measure. Let $P(\omega)$ be the cumulative distribution function of $p(\omega)$, and $P^{-1}$ its inverse on $(0,1)$. Define $\omega(k,o) = P^{-1}(o + k/d)$. Given an offset $o \in (0,1/d)$, we can define the probability distribution
\[ p_o(\omega) = \frac{1}{d} \sum_{k=0}^{d-1} \delta_{\omega(k,o)}(\omega). \] (5.39)
where $\delta_{\omega(k,o)}$ is the delta function at $\omega(k,o)$. This is an instance of a discretized prior which is designed to approximate $p$ as $d$ goes to infinity. From the theory of inverse transform sampling [27] (pg. 28), sampling from $P^{-1}(u)$ where $u$ is chosen uniformly at random from the interval $[0,1]$, is equivalent to sampling from $p(\omega)$. Therefore, if $o$ is chosen uniformly at random from $(0,1/d)$, $p_o(\omega)$ averages to the distribution $p(\omega)$. Specifically,
\[ \int_{o=0}^{1/d} \frac{1}{1/d} p_o(\omega) do = \int_{o=0}^{1/d} \frac{1}{d} \sum_{k=0}^{d-1} \delta_{\omega(k,o)}(\omega) do = p(\omega). \] (5.40)

where the identity holds in the distributional sense. To estimate the optimum cost $S_C(p,F)$, we estimate the average $\langle S_C(p_o,F) \rangle_{o \in (0,1/d)}$ by solving $S_C(p_o,F)$ for a number of offsets $o$ chosen uniformly at random from $(0,1/d)$. This gives a lower bound on $S_C(p,F)$. Intuitively, each value
of $o$ yields a discretization that is part of the full probability distribution. If we are able to derive an algorithm for each of these individual discretizations of $p(\omega)$ independently, their average cost should be lower than that achievable by a single algorithm that must handle all of $p(\omega)$. The following proof formalizes this notion. Below, we assume that the frequency estimates have already been discretized, $F = \{f_a\}_a$.

**Theorem 12.** The following inequality holds: \( \langle S_C(p_o, F) \rangle_o \leq S_C(p, F) \).

**Proof.** Consider an arbitrary query algorithm $Q$. Given a prior $r(\omega)$, $Q$ results in the measurement-conditional, unnormalized oracle states $\sigma_a(r, Q)$. Because the cost operators are diagonal, we can consider just the diagonals of $\sigma_a(r, Q)$, which define the joint probability distributions $r(a, \omega|Q)$. Because the oracle operators are conditional on the standard oracle basis, $r(a, \omega|Q)$ factors as

\[
r(a, \omega|Q) = q(a|\omega, Q)r(\omega),
\]

where, as indicated, the distribution $q$ does not depend on $r$.

The average cost for $Q$ and $r$ is given by

\[
C(r, Q) = \sum_a \int [C(\omega - f_a)r(a, \omega|Q)]d\omega.
\]

If $Q_{\text{min}}$ is an optimal algorithm for $S_C(r, F)$, then for any algorithm $Q$, it follows that $C(r, Q) \geq C(r, Q_{\text{min}}) = S_C(r, F)$. Let $Q_o$ and $Q_{opt}$ be optimal algorithms for $S_C(p_o, F)$ and $S_C(p, F)$, respectively. Then

\[
\langle S_C(p_o, F) \rangle_o = \langle C(p_o, Q_o) \rangle_o \\
\leq \langle C(p_o, Q_{opt}) \rangle_o,
\]

where the subscript on the expectations indicates that they are taken with respect to the distribution over offsets $o$. The intuition here is that we can obtain a lower cost if we are able to choose a different algorithm $Q_o$ for different choices of $o$, than if we are forced to use the same algorithm $Q_{opt}$ in all
cases. We can continue from the last line of equation (5.43) as follows:

\[
\langle C(p_o, Q_{opt}) \rangle_o = \left\langle \sum_a \int C(\omega - f_a) p_o(a, \omega | Q_{opt}) d\omega \right\rangle_o
\]

\[
= \left\langle \sum_a \int C(\omega - f_a) q(a | \omega, Q_{opt}) p_o(\omega) d\omega \right\rangle_o
\]

\[
= \sum_a \int C(\omega - f_a) q(a | \omega, Q_{opt}) \langle p_o(\omega) \rangle_o d\omega.
\] (5.44)

Combining equations (5.40),(5.43),(5.44) then gives

\[
\langle S_C(p_o, F) \rangle_o \leq \sum_a \int C(\omega - f_a) q(a | \omega, Q_{opt}) p(\omega) d\omega
\]

\[
= C(p, Q_{opt}) = S_C(p, F),
\] (5.45)

proving the claim of the theorem.

We note that theorem 12 generalizes to arbitrary oracle problems where the cost operators \( A_a \) are diagonal. Furthermore, the proof works for any family of probability distributions \( p_o \) such that \( p \) is a mixture of the \( p_o \).

For each discretization above, an algorithm is extracted. Since these algorithms are optimal for the discretization chosen, they are not necessarily optimal for the continuous problem. Therefore, a computation of the expected cost obtained by any such an algorithm is an upper bound on the optimal continuous cost. Formally,

**Theorem 13.** Let \( Q \) be an algorithm for \( S_C(r, F) \). Then

\[
S_C(p, F) \leq \sum_a \int C(\omega - f_a) q(a | \omega, Q) p(\omega) d\omega.
\] (5.46)

**Proof.** The right-hand-side is the expected cost for algorithm \( Q \) given oracle prior \( p \). Since \( Q \) is optimal for prior \( r \) but not necessarily for \( p \), this must be greater than \( S_C(p, F) \), which is the optimum expected cost for prior \( p \).
In view of the results of this section, we adopt the following procedure $A(p, F)$ for estimating a lower bound and calculating an upper bound of $S_C(p, F)$.

1. Choose discretization parameter $d$ and the number of random samples $k$. Large $d$ should tighten the bounds. Large $k$ improves the statistical estimate of the lower bound.

2. Independently choose $o_j \in (0, 1/d)$, $j \in \{1, \ldots, k\}$ uniformly at random.

3. Do the following for each $j \in \{1, \ldots, k\}$:
   
   a. Compute the optimum cost $C_j = S_C(p_{o_j}, F)$, and from the SDP solution, reconstruct an optimal algorithm $Q_j$ achieving this cost.

   b. From $Q_j$ derive an algorithm for evaluating $q(a|\omega, Q_j)$.

   c. Using this algorithm, evaluate
      \[
      \bar{C}_j = \sum_a \int C(\omega - f_a) q(a|\omega, Q)p(\omega) d\omega
      \]
      by numerical integration.

4. Return $\frac{1}{k} \sum_j C_j$ as a statistical estimate of a lower bound (together with its estimated error) and $\min_j(\bar{C}_j)$ as a numerical upper bound.

5.2.4.2 Alternative Oracle Basis

In this formulation of section 5.2.4.1, the SDP must maintain the potentially large set of discretized basis states $|x\rangle$. This comes at significant computational expense and introduces error. However, surprisingly, it is possible to avoid oracle discretization entirely by exploiting the low rank nature of $\rho^O$.

For arbitrary oracles, it can be shown that the rank of $\rho^O(t)$ grows by at most a factor of $|Q|^2$ at each time step. For the clock oracle, the rank at time step $t$ is at most $(|Q| - 1)t + 1$. We lose no generality in expressing the SDP in terms of a set of states spanning the support of $\rho^O(t)$, thus greatly simplifying the SDP and avoiding oracle discretization and its associated errors.
We first show how to construct a spanning set for the support of $\rho^O(t)$ for an arbitrary oracle unitary $\Omega^{OQ}$. Let $|u(0)\rangle = \int \sqrt{p(\omega)}|\omega\rangle d\omega$ be the continuous oracle initial state. It spans the support of $\rho^O(0)$. We inductively construct a spanning set of unnormalized states $|u(t)_{a,b}\rangle$ for the support of $\rho^O(t)$, where $a, b$ are length-$t$ strings of labels for an orthonormal basis $|i\rangle$ of $Q$.

Inductively suppose that

$$
\rho^{OQ}(t-1) = \sum_{i,j} \rho^O(t-1)_{i,j} |i\rangle\langle j|, \quad (5.47)
$$

where the $\rho^O(t-1)_{i,j}$ are linear combinations of operators of the form $|u(t-1)_{a,b}\rangle\langle u(t-1)_{a',b'}|$. Then

$$
\rho^O(t) = \text{tr}_Q \left( \Omega^{OQ} \rho^{OQ}(t-1) \Omega^\dagger_{OQ} \right) \\
= \sum_{i,j} \text{tr}_Q \left( \Omega^{OQ} \rho^O(t-1)_{i,j} |i\rangle\langle j| \Omega^\dagger_{OQ} \right) \\
= \sum_{k,i,j} \langle k| \Omega^{OQ} |i\rangle \rho^O(t-1)_{i,j} |j\rangle \Omega^\dagger_{OQ} |k\rangle \\
= \sum_{k,i,j} \Omega^O_{ki} \rho^O(t-1)_{i,j} \Omega^\dagger_{kj}, \quad (5.48)
$$

where we defined $\Omega^O_{ki} = \langle k| \Omega^{OQ} |i\rangle$. It follows that $\rho^O(t)$ is a linear combination of operators of the form $\Omega_{kl}|u(t-1)_{a,b}\rangle\langle u(t-1)_{a',b'}|\Omega^\dagger_{km}$. The inductive construction is completed by defining $|u(t)_{a_k,b_l}\rangle = \Omega_{kl}|u(t-1)_{a,b}\rangle$.

Let’s see how this construction works for the clock oracle. The oracle system begins in the state

$$
\rho^O(0) = \int d\omega \int d\omega' \sqrt{p(\omega)} \sqrt{p(\omega')} |\omega\rangle\langle \omega'|
= |u_0\rangle\langle u_0|. \quad (5.49)
$$

In this case, the $|u\rangle$ require only a single label because the oracle operators are diagonal. That is, $\langle k| \Omega^{OQ} |i\rangle = \delta_{ik} (\Omega^{OQ})_{k,k}$. We are also suppressing the time parameter $t$ in the state label. The initial state of the joint system is then $\rho^{OQ}(0) = \sum_{i,j} \rho^O(0)_{i,j} |i\rangle\langle j|$ where $\rho^O(0)_{i,j} = \alpha_{i,j} \rho^O(0)$. The $\alpha_{i,j}$ are amplitudes chosen by the querier, and $i, j \in \{0, 1, \ldots N = |Q| - 1\}$ (and all Latin indices that immediately follow) are labels for Dicke states. Applying the oracle $\Omega^{OQ} = \sum_n \int d\omega |\omega\rangle\langle \omega| \otimes$
|n⟩⟨n|e^{-inωT} to this state yields

\[
\rho^O(1) = \text{tr}_Q \left( \Omega^{OQ} \sum_{i=0}^N \sum_{j=0}^N \int dω \int dω' \alpha_{i,j} |i⟩⟨j| \otimes \sqrt{p(ω)} \sqrt{p(ω')} |ω⟩⟨ω'| \Omega^{iOQ} \right)
\]

\[
= \text{tr}_Q \left( \sum_{n=0}^N \sum_{n'=0}^N \int dω \int dω' \alpha_{n,n'} |n⟩⟨n'| \otimes \sqrt{p(ω)} \sqrt{p(ω')} e^{-inωT} e^{in'ω'T} |ω⟩⟨ω'| \right)
\]

\[
= \sum_{n=0}^N \alpha_{n,n} |u_n⟩⟨u_n|
\]

where

\[
|u_n⟩ = \int e^{-inωT} \sqrt{p(ω)} |ω⟩ dω
\]

Observe that the rank of \(\rho^O(1)\) is \(N + 1\).

We do not compute \(\rho^O(2)\) explicitly, and instead, use the inductive construction above to determine its spanning set. First, we have

\[
Ω_{kl}|ω⟩ = ⟨k|Ω^{OQ}|l⟩|ω⟩ = δ_{kl} e^{-ilω}|ω⟩
\]

Notice from the last line of (5.48) that \(\rho^O(2)\) is generated via the action of these operators on linear combinations of the \(|u_n⟩\). This action is given by

\[
= Ω_{kl}|u_n⟩ = Ω_{kl} \int e^{-inωT} \sqrt{p(ω)} |ω⟩ dω
\]

\[
= δ_{kl}|u_{n+l}⟩
\]

Therefore, \(\rho^O(2)\) is spanned by the \(|u_{k+l}⟩\). Since both \(k\) and \(l\) range from 0 to \(|Q| − 1\), \(\rho^O(2)\) has rank \(2N + 1\).

Continuing recursively, \(\rho^O(t)\) is evidently spanned by \(|u_0⟩, \ldots, |u_{tN}⟩\) and has rank \(Nt + 1 = (|Q| − 1)t − 1\). Note that, in general, this set need not always consist of independent (unnormalized) states.

Rather than expressing the SDP directly in terms of the these states, we utilize the orthonormal relabeling \(|u_n⟩ → |n⟩\). This makes implementation substantially simpler. Specifically, let \(O'(t)\) be a \(tN + 1\) dimensional quantum system with standard basis \(|n⟩\) for \(n = 0, \ldots tN\). Define the
linear map $B(t) : O'(t) \mapsto O$ by

$$B(t) = \sum_{n=0}^{tN} |u_n\rangle \langle n|$$  \hspace{1cm} (5.54)

and cost operators $A_a^{O'(t)}$ by

$$A_a^{O'(t)} = B^\dagger(t) A_a B(t).$$  \hspace{1cm} (5.55)

The oracle operator $\Omega(t)$ is defined as the map $\Omega(t) : O'(t-1)Q \mapsto O'(t)Q$ acting according to

$$\Omega(t) |n\rangle |l\rangle = |n+l\rangle |l\rangle.$$  \hspace{1cm} (5.56)

A complete clock SDP $\tilde{S}_C$ can be formulated as follows,

**Definition 5.1: SDP $\tilde{S}_C$:**

Minimize $\text{tr}(\sum_a \sigma_a^{O'(t_f)} A_a^{O'(t_f)})$ subject to

for all $a, \sigma_a^{O'(t_f)} \geq 0$,

$$\sum_b \sigma_b^{O'(t_f)} = \rho^{O'(t_f)}(t_f),$$

$$\rho^{O'(0)}(0) = |0\rangle \langle 0|, \rho^{O'(0)}Q(0) \geq 0,$$

$$\rho^{O'(0)}(0) = \text{tr}_Q(\rho^{O'(0)}Q(0)),$$

and the following for $t \in \{1, \ldots, t_f\}$:

$$\rho^{O'(t)}Q(t) \geq 0,$$

$$\rho^{O'(t)}(t) = \text{tr}_Q(\rho^{O'(t)}Q(t)),$$

$$\rho^{O'(t)}(t) = \text{tr}_Q \left( \Omega(t) \rho^{O'(t-1)}Q(t-1) \Omega(t)^\dagger \right).$$

According to the next theorem, solving $\tilde{S}_C$ is equivalent to solving $S_C$.

**Theorem 14.** There is a linear map from feasible solutions of $\tilde{S}_C$ onto feasible solutions of $S_C$ that preserves the objective function. Conversely, feasible solutions of $S_C$ can be lifted to feasible solutions of $\tilde{S}_C$ with the same value of the objective function.
Proof. We define the linear map from feasible solutions of \( \tilde{S}_C \) to feasible solutions of \( S_C \) by applying \( B(t) \) and \( B(t)\dagger \) as follows:

\[
\rho^O(t) = B(t) \rho^{O(t)} B(t)\dagger \\
\rho^{OQ}(t) = B(t) \rho^{O(t)} Q B(t)\dagger \\
\sigma_a^O = B(t_f) \sigma_a^{O(t)} B(t_f)\dagger.
\] (5.57)

To see that the constraints of \( S_C \) are satisfied, apply this map to both sides of each constraint of \( \tilde{S}_C \). Because the map \( \rho \mapsto B(t) \rho B(t)\dagger \) is linear, preserves positivity, and commutes with \( tr_Q \), all but the last constraint are directly transformed into the corresponding constraints of \( S_C \). The last constraint requires the identity \( \Omega B(t - 1) = B(t) \Omega(t) \), which is a consequence of the definition of \( B(t) \) and \( \Omega(t) \) (equations (5.54) and (5.56)). The transformed right-hand-side of the constraint is

\[
B(t) tr_Q \left( \Omega(t) \rho^{O(t-1)Q(t-1)\Omega(t)}\dagger \right) B(t)\dagger
= tr_Q \left( B(t) \Omega(t) \rho^{O(t-1)Q(t-1)\Omega(t)}\dagger B(t)\dagger \right)
= tr_Q \left( \Omega B(t - 1) \rho^{O(t-1)Q(t-1)B(t - 1)\Omega(t)}\dagger \right)
= tr_Q \left( \Omega \rho^{OQ(t-1)\Omega(t)}\dagger \right).
\] (5.58)

The left-hand-side transforms to \( \rho^O(t) \), so the last constraint of \( \tilde{S}_C \) also transforms to the corresponding constraint of \( S_C \). That the objective is preserved follows from

\[
tr(\sigma_a^O A_a) = tr( B(t) \sigma_a^{O(t)} B(t)\dagger A_a)
= tr(\sigma_a^{O(t)} B(t)\dagger A_a B(t))
= tr(\sigma_a^{O(t)} A_a^{O(t)}).
\] (5.59)

To lift a feasible solution of \( S_C \) to one of \( \tilde{S}_C \), we first reconstruct the algorithm by adding the ancilla system \( A \), purifying \( \rho^{OQ}(t) \) to \( \rho^{OQA}(t) \) and constructing the unitary operators \( U^{QA}(t) \) as in section 3.3.2. Also, let \( P_a^{QA} \) be the POVM operator of the reconstruction that yields \( \sigma_a^O \). Because \( \rho^O(0) \) is pure, \( \rho^{OQA}(0) = \rho^O(0) \otimes \rho^{QA}(0) \). Let \( \rho^{O(0)QA}(0) = |0\rangle\langle 0| \otimes \rho^{QA}(0) \) and recursively define

\[
\rho^{O(t)QA}(t) = U^{QA}(t) \Omega(t) \rho^{O(t-1)QA(t-1)\Omega(t)}\dagger U^{QA}(t)\dagger.
\] (5.60)
The solution of $\tilde{S}_C$ is obtained from the appropriate partial traces of $\rho^{O(t)Q_A}$ and by defining $\sigma^O(t_f) = \text{tr}_{QA}(P_a^{Q_A} \rho^{O(t_f)Q_A}(t_f))$. Checking that the constraints are satisfied is straightforward and directly follows from the observation that the lifted solution is derived from an explicit algorithm.

The lifted solution also satisfies

$$B(t)\rho^{O(t)Q_A}B^\dagger(t) = \rho^{OQA}(t), \quad (5.61)$$

which can be verified inductively by applying $\rho \mapsto B(t)\rho B^\dagger(t)$ to both sides of equation (5.60) and applying the identity $B(t)\Omega(t) = \Omega B(t - 1)$. To see that the cost is preserved, it suffices to note that

$$\text{tr} \left( \sigma^O(t_f) A_a^O(t_f) \right)$$
$$= \text{tr} \left( \sigma^O(t_f) B^\dagger(t_f) A_a B(t_f) \right)$$
$$= \text{tr} \left( B(t_f) \sigma^O(t_f) B^\dagger(t_f) A_a \right)$$
$$= \text{tr} \left( B(t_f) \text{tr}_{QA} \left( P_a^{QA} \rho^{O(t_f)Q_A}(t_f) \right) B^\dagger(t_f) A_a \right)$$
$$= \text{tr} \left( \text{tr}_{QA} \left( P_a^{QA} B(t_f) \rho^{O(t_f)Q_A}(t_f) B^\dagger(t_f) \right) A_a \right)$$
$$= \text{tr} \left( \text{tr}_{QA} \left( P_a^{QA} \rho^{OQA} \right) A_a \right)$$
$$= \text{tr} \left( \sigma_a A_a \right). \quad (5.62)$$

\[\square\]

### 5.2.5 Size of the SDP

The number of variables in our SDPs is a function of the size of the joint density matrix, the number of frequency estimates, and the number of queries. Specifically, the number of variables scales as $O(|\rho^{OQ}|t_f + |\rho^O||F|)$. The complexity can be reduced by observing that the off-diagonal elements of $\rho^Q$ play no role. In the only potentially relevant constraint, $\rho^O(t) = \text{tr}_Q \left( \Omega(t)\rho^{OQ}(t - 1)\Omega(t) \right)$, our choice of $\Omega$ (equation 5.18) does not mix system $Q$’s diagonal elements with its off-diagonal ones, and the latter are subsequently lost in the partial trace over
Q. In other words, we can replace $\rho^{OQ}$ by $\sum_k \langle k | \rho^{OQ} | k \rangle | k \rangle \langle k |$ without affecting feasibility or the objective.

Then, SDP $S_C$ which discretizes the oracle has $O(N|O|^2 t_f + |O|^2 |F|)$ variables. $\tilde{S}_C$, which uses the alternate oracle basis has $O(N^3 t_f^3 + N^2 t_f^2 |F|)$ variables. The specification of each SDP involves $O(t_f)$ matrix-valued constraints, involving matrices of dimension $O(N|O|^2)$ in $S_C$ and of dimension $O(N^3 t_f^2)$ in $\tilde{S}_C$. Solving $\tilde{S}_C$ will therefore be more efficient unless $t_f$ is large. Even then, $S_C$ introduces oracle discretization error. Code used to solve SDP $\tilde{S}_C$ is given in the appendix.

5.3 Results

Combining the ideas of the previous sections, we use the following procedure for approximately solving the clock SDP $S_C(p, \mathbb{R})$:

- **Discretize the Measurement**: Choose a discretization $F$, $|F| = m$, of the frequency estimates. If the cost function is suitable, we can optimize the right-hand-side of equation 5.24 and let $\epsilon_q$ be the corresponding querier discretization error bound. If the cost function is not suitable, we set $\epsilon_q = 0$ and note that any bounds apply only for the chosen discretization $F$.

- **Modify the Oracle System**: Do one of the following:

  * Apply procedure $A(p, F)$ and let $c_l$ be the statistically estimated lower bound with estimated standard error $s_l$, and $c_u$ the numerical upper bound obtained. Give the estimated cost of $S_C(p, \mathbb{R})$ in the form $((c_l - \epsilon_q) \pm s_l, c_u)$.

  OR

  * Map $O \rightarrow O'(t)$ and run SDP $\tilde{S}_C$. Call the cost of $\tilde{S}_C c_e$. Then report an estimated cost of $((c_e - \epsilon_q), c_e)$

- **Refine the Measurement**: Optionally, iterate SDP $S_C$ with a single chosen oracle discretization, or iterate SDP $\tilde{S}_C$, and refine the selection of frequency estimates.
In the results below, we consider both methods of handling the continuous system $O$.

We begin by considering one-query clock protocols ($t_f = 1$). In this case, the protocol consists of an initial query state to be prepared and a final measurement. Figure 5.1 depicts the optimization of a one and four atom clock via SDP $S_C$ which discretizes system $O$. Figure 5.2 depicts the optimization of five and ten atom clocks via SDP $\tilde{S}_C$ which does not discretize system $O$. Each illustrates the importance of taking into account prior knowledge when deriving optimal clock protocols. Our technique is compared to that of Ref. [18], which derives protocols under the assumption that $\omega$ is uniformly distributed on $[-\pi, \pi]$. We consider Gaussian priors of various widths and see that solving the clock SDP can substantially reduce the expected cost. In each figure, (a) is a comparison with the periodic cost function $4 \sin^2(\omega - \omega_a)$ considered in Ref. [18]. For this cost function and in the limit of wide priors, our protocols converge to those in Ref. [18]. This is confirmed by Table 5.1, which lists initial 2-atom states that optimize this cost function; the final row agrees with the state computed in Ref. [18]. In each figure, (b) is a comparison with the quadratic cost function. This cost function satisfies the conditions in theorem 11, and we can bound the discretization errors for the range shown in the plot by $1.96 \times 10^{-5}$ to $7.56 \times 10^{-3}$, increasing monotonically with the prior’s standard deviation. The bounds are invisible on the scale of the plots.

The inset of figure 5.2 (b) compares the protocols for small-width Gaussian priors, showing the costs for 10 and 11 atoms of the protocol of Ref. [18]. To explain the features of these costs, we note that Ref. [18] uses 11 and 12 (respectively) prior-independent frequency estimates. Since the measurement answers are not deterministic when the frequency is 0, there is a cost even when the prior is concentrated at 0. The frequency estimates nearest 0 are $\pm 0.2856$ for 10 and $\{0, \pm 0.5236\}$ for 11 atoms. The flat regions in the cost curves correspond to the region where the width of the Gaussian prior is close to these frequency estimates.

To verify the technique for obtaining oracle discretization error bounds, we compare our upper and lower bounds to Ref. [23], which derives optimal interrogation algorithms for the periodic cost function and arbitrary prior. This is illustrated in figure 5.3. We use the optimal set of classical
(a) One atom, 20 frequency estimates
\[ C(\omega, a) = 4 \sin^2 \frac{\omega - f_a}{2} \]

(b) Four atoms, 20 frequency estimates
\[ C(\omega, a) = (\omega - f_a)^2 \]

Figure 5.1: Comparison of the protocol derived in Ref. [18] and those derived by our method for one and four atoms and two different cost functions. We use a 15-point oracle discretization and simulate Gaussian priors of various widths. The figures on the top plot the cost computed in Ref. [18] (green, dashed line) and our numerical upper bound (blue, solid line), which as discussed, is equivalent to the minimum continuous cost obtained by one of our extracted algorithms. The blue solid line in the figures on the bottom correspond to the difference between our lower bound and upper bound, \( c_l - c_u \), illustrating both the strength of our bounds and how much lower the continuous cost could potentially be. The lower bound was computed by averaging 100 discretizations; error bars show the estimated standard error of the average thus obtained. Our querier discretization bounds cannot be applied to the periodic cost function. Therefore in (a) the lower bounds are for the discretizations chosen, and in (b), we plot a green, dashed line corresponding to \( c_l - c_u - \epsilon_q \), which is a lower bound for the continuous problem. We use 20 frequency estimates; in (b) these are chosen by minimizing equation (5.24). We do not optimize the estimates via the iterative technique of Sect. 5.2.3. The lines connecting the data points are to guide the eyes.

Figure 5.4 illustrates the effect on our bounds of increasing the number of points in the oracle discretization. Here we are reanalyzing the last point in figure 5.1(b). We see a substantial reduction in the gap between the upper and lower bounds by increasing the number of points in frequency estimates derived in Ref. [23]; therefore, in the continuous limit, the clock SDP and that of Ref. [23] should yield identical costs. Consequently, the deviation depicted in Figure 5.3 is due entirely to discretization error and the limitations of our bounds.
the discretization from five to ten, and small but consistent reductions for increases thereafter.

Figure 5.5 illustrates the effect of changing the cost function, showing that the protocols in Ref. [23], optimal for the periodic cost function, are suboptimal for the quadratic cost function. These protocols involve three frequency estimates for the cases considered. We use the same number, choose them by means of the iterative technique discussed in section 5.2.3, and achieve a smaller cost at high standard deviation.

Our procedure can also optimize sequences of two or more clock queries \((t_f \geq 2)\). If these queries are fully coherent, the algorithm is of the form given in equation (5.1), and the SDP implicitly optimizes the initial state, each \(U(t_i)\), and the measurement. Alternatively, we can combine the two queries classically. In this case, we update our knowledge of the clock’s phase using Bayes’ rule between the queries. That is, after the first query, we compute posterior distributions for each measurement outcome, as in equation (5.34). We then run the SDP again, once for each

Figure 5.2: Comparison of the protocol in Ref. [18] and those obtained by our method for Gaussian priors of various widths. Here we do not discretize the oracle; rather, we change basis and solve SDP \(\bar{S}_C\). The blue, solid line is the cost computed by our method and the green, dashed line is the cost computed by the method of Ref. [18]. The inset in (b) is a magnification of the lower left side of the main plot and shows the cost for both 10 (top line) and 11 (middle line) atoms of the former protocol; these are indistinguishable on the main plot. The frequency estimates in (b) for our method are chosen by minimizing equation (5.24), and are not refined via the iterative technique of section 5.2.3.3.

(a) Five atoms, 200 frequency estimates
Number of SDP Constraints: 7704
\[ C(\omega, a) = 4 \sin^2 \frac{\omega - f_a}{2} \]

(b) Ten atoms, 100 frequency estimates
Number of SDP Constraints: 15004
\[ C(\omega, a) = (\omega - f_a)^2 \]
Table 5.1: Initial states that minimize the cost function $C(\omega, a) = 4\sin^2\left(\frac{\omega - f_a}{2}\right)$ for two atom, one-query protocols assuming different-width Gaussian priors. We used 20 unrefined frequency estimates.

| Standard Deviation | |0| |1| |2| |
|-------------------|---|---|---|---|
| .25               | .7071 | 0 | .7071 |
| .75               | .5621 | .6066 | .5621 |
| 1.25              | .5167 | .6827 | .5167 |
| 1.75              | .5025 | .7035 | .5025 |
| 2.25              | .5000 | .7071 | .5000 |

outcome, using the corresponding posterior distribution as the new prior. We compute a new cost by averaging each of the costs obtained, weighted by the probability of obtaining the corresponding measurement outcome, $\text{tr}(\sigma_a)$. Here we assume that there is no noise between sequential queries. Any noise would affect the intermediate prior distributions.

Figure 5.6 compares the performance of a sequence of two queries combined classically and quantumly. We simulate two atoms and use the same set of 100 frequency estimates for each method. We see that, as expected, while we gain information by combining queries classically, fully coherent queries provide the greatest advantage. Since the discretization error for a 100 element POVM is relatively small, we conjecture that fully coherent queries will remain advantageous given the fully optimal set of frequency estimates and POVM.

The idea of combining queries classically by updating the appropriate probability distributions will be greatly expanded on in the next chapter.
Figure 5.3: Comparison of the optimal protocol derived in Ref. [23] and the one obtained by our SDP for two atoms. The top, red line depicts the difference between our upper bound and the cost computed in Ref. [23], and the bottom, blue line depicts the difference between our lower bound and the cost computed in Ref. [23]. Error bars again correspond to the estimated standard error in the lower bound. We used a 15-point prior discretization and set $F$ to the optimal set of frequency estimates derived in [23] so there is no querier discretization error. The lower bound was computed by averaging 500 random oracle discretizations.
Figure 5.4: Upper ($c_u$, dotted line) and lower ($c_l$, solid line) bounds on cost for oracle discretizations with varying numbers of points. We are simulating the last point of figure 5.1(b) - that is, 4 atoms, a standard deviation of 1.875, and the quadratic cost function. The lower bound is computed by averaging 32 random oracle discretizations. Note that the values in figure 5.1 are for a 15 point oracle discretization.

Figure 5.5: Comparison of the optimal protocol derived in Ref. [23] (upper, green line) and the one obtained by our SDP (lower, blue line) for two atoms, three frequency estimates chosen, in our case, by refinement according to section 5.2.3.3, and $C(\omega, a) = (\omega - f_a)^2$
Figure 5.6: Average cost after one query (top, green line), two queries combined classically (middle, red line), and two coherent quantum queries (bottom, blue line). We simulated two atoms, and used 100 frequency estimates and the cost function $C(\omega, f_a) = (\omega - f_a)^2$. 
Chapter 6

Passive Clock Optimization

6.1 Introduction

In chapter 5, we presented a technique that, up to discretization error, optimized an individual clock interrogation. A passive clock operates by stringing together a long series of such interrogations, using each jointly or independently to assign timestamps to events. The performance of such clocks is based on both how much information is extracted from an interrogation and how that information is used to infer time. In this chapter, by making use of the techniques of chapter 5, we show how to optimize this performance.

6.2 Passive Clock Operation

6.2.1 Post Interrogation Strategies

A passive clock operates by alternating between atomic interrogations and processing measurement results. The former has already been discussed at length; here then, we focus on the latter. In particular, we wish to know both how timestamps are assigned based on measurement results, and how these results affect subsequent interrogations.

As discussed in section 4.2.1, in practice, most atomic clocks solve each of these problems in a relatively straightforward way. Recall that in standard Ramsey spectroscopy, the measurement at the end of an interrogation yields incomplete information about the accumulated phase difference between the flywheel and the standard over the interrogation period, $\phi - \omega_0 T$, where $\phi$ is the fly-
wheel’s phase, \( \omega_0 \) is the resonant frequency of the standard, and \( T \) is the length of the interrogation period. This is then used, along with \( \phi \) to estimate frequency deviation. Using this information, the frequency of the flywheel is adjusted up or down in an attempt to align it to frequency of the standard. Cycles of the adjusted flywheel are counted and used as an approximation of elapsed time. Subsequent interrogations proceed in exactly the same manner, using identical interrogation procedures, albeit with an adjusted flywheel.

While this strategy is very effective in practice, clock accuracy can be improved by using a more sophisticated approach. First, notice that standard Ramsey spectroscopy assigns timestamps using relatively little information. Notably, after each interrogation, the same measurement outcome always results in the same estimated phase deviation. That is, the number of qubits in excited states translates directly into an estimate of \( \phi \) via equation (4.8). The resolution of any reasonably sized POVM, including the one used in standard Ramsey spectroscopy is limited. However, additional information, such as an estimate of the frequency of the flywheel just prior to an interrogation, along with a physical description of how this frequency is expected to change over the course the interrogation period, can be used to supplement the measurement. It is therefore desirable to maintain a probability distribution describing the frequency of the flywheel and to update that distribution as a clock progresses. Furthermore, recall that the interrogation algorithms we constructed in chapter 5 were optimal for specific prior probability distributions of frequency deviations. Therefore, this distribution must be maintained in order to derive the best possible algorithms for future interrogations. The optimal operation of a passive clock must then be adaptive, using different algorithms for different interrogation periods.

Finally, note that mathematically, an adjustment of the flywheel is unnecessary. Knowledge of the underlying frequency deviation is all that is necessary to assign timestamps. Therefore, in the simulations we construct here, the flywheel is not adjusted. However, in practice, such a simplification is not possible. This is because, among other reasons, the frequency of the flywheel must be near that of the atomic resonance to effect the \( |0\rangle \rightarrow |1\rangle \) transition.
6.2.2 Modeling a Clock

We assume that the time taken by one round of clock evolution is dominated by the free evolution time, $T$. That is, we assume the time needed to prepare the atomic state, measure, and adjust the flywheel are negligible compared to $T$. Recall from section 4.3 that an interrogation can consist of an interwoven series of atomic system preparations and queries, as in equation (4.14). However, in this chapter we assume that there is only one preparation and query per interrogation. While allowing sequences would result in superior clock performance, they would be more difficult to implement in experiment.

We also assume full quantum control over the atomic system and that the only source of noise is statistical fluctuations of the flywheel. While this latter assumption is sensible for many modern clocks, if necessary, our scheme can be adapted to account for decoherence using the techniques of section 3.5. Noise is assumed to affect the flywheel only during the query, as this typically corresponds to the period of free evolution which dominates the time of an interrogation, and is where the flywheel is not explicitly controlled.

As described above, the frequency of the flywheel is described by a probability distribution. This distribution always has nonzero width, as the frequency of the flywheel will never be known exactly. Figure 6.1 illustrates how this distribution changes as the clock evolves. Roughly speaking, the distribution widens during an interrogation period as noise causes drift, and narrows after a measurement as information is gained.

Figure 6.2 is a high level overview of the scheme described in this chapter, and illustrates how these probability distributions are used and manipulated in a larger context. Notice that knowledge of both the flywheel’s frequency deviation and the relevant noise is used to generate an interrogation algorithm. Using Bayes rule, a posterior distribution is computed from the measurement outcome which is then used, along with a count of flywheel cycles, to assign appropriate timestamps.
Figure 6.1: Evolution of the flywheel’s probability distribution in a clock protocol. Noise decreases our knowledge of the flywheel’s frequency, widening $p(\omega)$. The prior probability distribution on the right is used in our optimization procedure, and describes the average frequency of the flywheel over the probe time $T$. A measurement of the atomic standard then yields a measurement outcome $a$, which can be used to compute the posterior distribution $p(\omega|a)$. 
Figure 6.2: A high level overview of the strategy developed here to optimize a passive clock. In order to assign estimates of true time to events (left side of figure), flywheel cycles are counted in the context of an estimate of the flywheel’s deviation. This deviation is described by a posterior probability distribution which is computed based on the result of an atomic interrogation. This interrogation is implemented via an algorithm derived by the semidefinite program from chapter 5, which uses the prior probability distribution of the average flywheel deviation during the interrogation period of interest. This distribution is obtained by computing how the relevant noisy process affects the previous posterior distribution.
6.3 Flywheel Noise

Before we describe our scheme further, we turn to a description of the noise that has been experimentally determined to affect passive atomic clocks. We do not discuss such experiments or the physical origin of this noise. Such a description would be implementation dependent and a significant detour from our purpose here. Rather, we detail the mathematics used to represent the appropriate noise models.

Much of the relevant noise can be approximated by a continuous multivariate Gaussian random walk characterized by a spectral density, $S(\omega)$. It has been determined experimentally that we can take $S(\omega) = h_\alpha \omega^\alpha$, where $\alpha \in \{-2, -1, 0, 1, 2\}$, and $h_\alpha$ is an $\alpha$ dependent scale factor [6]; these are examples of power law noise. Different values of $\alpha$ are dominant in different clocks and at different frequency ranges. For example, $1/\omega$ noise is dominant in modern optical clocks [71].

We can completely characterize these processes by the multivariate Gaussian’s mean vector, which we assume to be zero, and covariances, $\text{Cov}(\omega(s), \omega(s'))$, where $\omega(s)$ and $\omega(s')$ are Gaussian distributed random variables that correspond to the flywheel’s frequency at times $s$ and $s'$. Here, as in chapter 5, we are interested in time averaged frequencies; specifically, on the average of the flywheel’s frequency over an interrogation period. These averages will also be Gaussian, characterized by a covariance matrix $C_{i,j}$. Moreover, because we are interested in only the change of the flywheel’s average frequency during an interrogation period, we need to consider only the covariances of the departure of these averages from their initial values; we can therefore write

$$C_{i,j} = \text{Cov}(\langle \omega(s) \rangle_{I_i} - \langle \omega(s) \rangle_{I_0}, \langle \omega(s) \rangle_{I_j} - \langle \omega(s) \rangle_{I_0}),$$  \hspace{1cm} (6.1)

where the notation $\langle \omega(s) \rangle_{I_i}$ refers to the time average of $\omega(s)$ over interval $I_i = [t_i, t_{i+1}]$, of length $T_i = t_{i+1} - t_i$. The intervals $I_i$ and $I_j$ correspond to interrogation periods $i$ and $j$, while $I_0$ is an interval of arbitrary length before the first interrogation period. For most models, we can condition
on $\langle \omega(s) \rangle_I = 0$, in which case (6.1) becomes

$$C_{i,j} = \text{Cov}(1_{T_i} \int_{t_i}^{t_{i+1}} \omega(s) ds, 1_{T_j} \int_{t_j}^{t_{j+1}} \omega(s') ds') = \frac{1}{T_i} \frac{1}{T_j} \int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \text{Cov}(\omega(s), \omega(s')) ds ds'.$$

(6.2)

The instantaneous covariances $\text{Cov}(\omega(s), \omega(s'))$ depend on $\alpha$. Since power law noise is a special case of fractional Brownian motion, we take

$$\text{Cov}(\omega(s), \omega(s')) = \frac{h_2}{2} (|s|^{2H} + |s'|^{2H} - |s - s'|^{2H}),$$

(6.3)

where $H \in (0, 1]$ is known as the Hurst exponent. [64]. $H$ characterizes the correlations of the noise process. For power law noise, $H = \frac{-\alpha - 1}{2}$. Note that $1/\omega$ noise must be treated separately. In this case, after a suitable renormalization, we can take [38]

$$\text{Cov}(\omega(s), \omega(s')) = -2h - 1 \ln |s - s'|.$$

(6.4)

Also, for this particular model, because of the singularity at $s = s'$, we cannot condition on $\langle \omega(s) \rangle_I = 0$. Instead, we must integrate equation (6.1) fully; we choose $I_0$ to be an interval of length $T_i$ before the first interrogation period.

### 6.4 Optimization Strategy

In this section, we fill in the details of the strategy outlined in section 6.2.2 and figure 6.2.

#### 6.4.1 Objective

We write the state of the flywheel as the probability distribution

$$p(\langle \omega_1 \rangle, \langle \omega_2 \rangle, \ldots \langle \omega_n \rangle | a_1, a_2 \ldots a_n),$$

(6.5)

where $\langle \omega_i \rangle$ is the average frequency of the flywheel during interrogation period $i$, and $a_i$ is the measurement result obtained at the end of interrogation period $i$. From now on, for brevity, we write the joint distribution above as $p(\omega_n | a_n)$ where, as in chapter 5, an averaging over the relevant
interrogation period is implied, and $\omega_n$ and $a_n$ refer to the sequence of average frequencies and the sequence of measurement outcomes obtained at steps 1 through $n$. Note that because many noise models discussed above have infinite memory, we cannot simply marginalize out $\omega_1 \ldots \omega_{n-1}$. As in the previous chapter, we can assume $\omega_0 = 0$ without loss of generality. In this context, the flywheel’s phase deviation is equivalent to the flywheel’s phase.

The accuracy with which we estimate true time depends on how closely we estimate the phase of the flywheel. We therefore wish to choose phase estimates, $\theta_n^*$, that minimize

$$E\left((\theta_n - \theta_n^*)^2\right), \quad (6.6)$$

where $\theta_n$ is the cumulative phase of the flywheel after interrogation period $n$, $\theta_n = \omega_1 T_1 + \omega_2 T_2 + \ldots + \omega_n T_n$, and $\theta_n^*$ is our estimate of this phase. The expectation is taken over the noise model. That is, equation (6.6) is equivalent to

$$\int (\theta_n - \theta_n^*)^2 p(\theta_n|a_n) d\theta_n, \quad (6.7)$$

where $p(\theta_n|a_n)$ can be computed directly from $p(\omega_n|a_n)$.

The choice

$$\theta_n^* = E(\theta_n|a_n) \quad (6.8)$$

will minimize equation (6.7). Equation (6.7) with $\theta_n^*$ chosen according to equation (6.8) is equivalent to the posterior variance, $V(\theta_n|a_n)$. Our goal is therefore to construct quantum protocols that minimize the expected posterior variance increase during each period $n$.

$$\sum a_n V(\theta_{n-1} + \omega_n T_n|a_1 \ldots a_n)p(a_n|a_1 \ldots a_{n-1}) - V(\theta_{n-1}|a_1 \ldots a_{n-1}) \quad (6.9)$$

6.4.2 Probability Updates

As depicted in figures 6.1 and 6.2, the flywheel’s probability distribution, $p(\omega_n|a_n)$, can be updated as the clock runs. Implementing such updates correctly is critical to the performance of
our interrogation algorithms. Indeed, notice that since these algorithms must minimize equation (6.9), the \( n \)th algorithm must depend on \( p(\omega_n | a_{n-1}) \).

In general, when advancing from just before interrogation period \( n \) to just before interrogation period \( n + 1 \) we must (1) compute the prior \( p(\omega_n | a_{n-1}) \) from the old state of the flywheel and the noise model, (2) Derive and apply an interrogation protocol based on this distribution, and (3) compute the posterior distribution \( p(\omega_n | a_n) \) from the prior and measurement result \( a_n \). More specifically, we proceed as follows.

1. We have the joint probability distribution \( p(\omega_{n-1} | a_{n-1}) \). Use the chosen noise model to compute \( p(\omega_n | \omega_{n-1}, a_{n-1}) \).

2. Compute the prior probability distribution \( p(\omega_n | a_{n-1}) = p(\omega_n | \omega_{n-1}, a_{n-1})p(\omega_{n-1} | a_{n-1}) \).

3. Use \( p(\omega_n | a_{n-1}) \) and SDP \( \tilde{S}_C \) to compute an optimal quantum algorithm consisting of a unitary matrix \( U \) and a POVM \( \{P_a\}_a \).

4. Use this protocol to interrogate the frequency standard for a time \( T_n \). Obtain measurement outcome \( a_n^* \).

5. Used the derived algorithm to fill in the collection of distributions \( p(a_n | \omega_n) \) for all possible measurement outcomes \( a_n \). We have \( p(a_n | \omega_n) = p(a_n | \omega_n, \omega_{n-1}, a_{n-1}) \) since \( \omega_{n-1} \) and \( a_{n-1} \) provide no additional information about \( a_n \) beyond that in \( \omega_n \).

6. Compute the posterior distribution (which is also the state of the clock at the next timestep)

\[
p(\omega_n | a_n) = p(\omega_n, \omega_{n-1} | a_n = a_n^*, a_{n-1}) = p(a_n | \omega_n, \omega_{n-1}, a_{n-1})p(\omega_n, \omega_{n-1} | a_{n-1}) / p(a_n^*).
\]

7. Compute the posterior expected phase \( E(\theta_n = \theta_{n-1} + \omega_n T_n | a_n) \) This is our estimate of the flywheel’s phase after interrogation period \( n \).

Since this procedure will use SDP \( \tilde{S}_C \), we refer to it as \( P(\tilde{S}_C) \). Figure 6.3 is an illustration of \( P(\tilde{S}_C) \) during the first interrogation period. Note that \( P(\tilde{S}_C) \) can be readily generalized if we
have more or less information at every step. That is, we can describe the state of the clock at step $n$ as the distribution $p(s_n|a_n)$ where $s_n$ is not limited to consisting of just a list of frequencies.

Figure 6.3: Evolution of the flywheel’s probability distribution in a clock protocol. The probability distribution at the top corresponds to the average frequency of the flywheel during interrogation period 1. A measurement of the atomic standard at time $t_1$, with $F$ element POVM $\{P_a\}_1$ then yields measurement outcome $a_1$ with probability $p(a_1)$. This can be used to compute the posterior distribution $p(\omega_1|a_1)$. (In the figure, we imagine that we obtain measurement result $a_1 = 1$.) Noise then affects the flywheel for duration $T_2$; this decreases our knowledge of the flywheel’s frequency, widening the probability distribution. The resulting prior probability distribution corresponds to the average frequency of the flywheel over interrogation period 2. This procedure is repeated indefinitely.
The distribution \( p(\omega_n|\omega_{n-1}, a_{n-1}) \) computed at the end of step 1 is independent of the actual history of the clock; it tells us only what average frequency deviations are expected during interrogation period \( n \) given some potential history and a particular choice of noise model. It can be computed from the model’s covariance matrix following a standard technique reproduced here. Assume the multivariate Gaussian \( p(\omega_N) \) has \( N \)-dimensional mean vector \( \mu = (E(\omega_1), E(\omega_2), \ldots E(\omega_N)) \) and \( N \times N \) covariance matrix \( C_{i,j} = \text{Cov}(\omega_i, \omega_j) \). The mean vector and covariance matrix together completely characterize the distribution; in our case, they are constructed entirely from the chosen noise model, as described in section 6.3. Before conditioning, all processes we consider in this paper are assumed to have mean zero, \( \mu = 0 \), but for illustrative purposes, here we allow for nonzero mean.

We wish to compute \( p(\omega_N|\omega_{N-1}) \) from \( p(\omega_N) \). Denote the submatrix of \( C \) containing rows \( r \) through \( s \) and columns \( c \) through \( d \) as \( C_{[r,s],[c,d]} \); do the same for subvectors of \( \mu \). Then the probability distribution \( p(\omega_N|\omega_{N-1}=f) \) is a univariate Gaussian with mean

\[
\mu' = \mu_N + C_{N,[1,N-1]}C_{[1,N-1],1,N-1}(f - \mu_{[1,N-1]})
\]

and variance

\[
C' = C_{N,N} - C_{N,[1,N-1]}C_{[1,N-1],[1,N-1]}C_{[1,N-1],N}.
\]

### 6.4.3 Cost Function

As indicated, the derivation of the optimal quantum interrogation protocol (step 3 above) is done via the SDP \( \tilde{S}_C \) from chapter 5. Recall that this SDP minimizes the value of a chosen cost function,

\[
E(C) = \sum_a \int C(\omega,a)p(\omega,a)d\omega.
\]

Consider the quadratic cost function, \( C(\omega,a) = (\omega - f_a)^2 \), which was examined in depth. Per equation (5.6), a minimization of \( E(C) \) with this choice of \( C \) during period \( n \) results in an algorithm that minimizes the expected posterior variance of \( \omega_nT_n \). Unfortunately, minimizing individual variances is insufficient as \( V(\theta_n|a_n) \neq V(\omega_1T_1|a_1) + V(\omega_2T_2|a_2) + \ldots V(\omega_nT_n|a_n) \) if
there are correlations between the $\omega_i$'s. Many of the noise models considered in section 6.3 contain such correlations. Instead, we require a cost function which results in a minimization of equation (6.9). Theorem 15 below shows that this is

$$C(\omega, a) = (\omega T - f_a)^2 + 2(\omega T - f_a)\mathbf{E}(\theta - \mathbf{E}(\theta) | \omega)$$

(6.13)

**Theorem 15.** Given interrogation time $T$ and a map from measurement outcomes to frequency estimates $g(a)$, a minimization of $\langle C \rangle$ with the cost function $C = (\omega T - g(a))^2 + 2(\omega T - g(a))\mathbf{E}(\theta - \mathbf{E}(\theta) | \omega)$ is equivalent to a minimization of the expected posterior variance increase of the cumulative phase, $\sum_a \mathbf{V}(\theta + \omega T | a)p(a) - \mathbf{V}(\theta)$.

**Proof.** Begin by expanding

$$\sum_a \mathbf{V}(\theta + \omega T | a)p(a) - \mathbf{V}(\theta)$$

$$= \sum_a \int \int (\theta' + \omega' T - \mathbf{E}(\theta + \omega T | a))^2 p(\theta', \omega' | a)p(a)d\omega'd\theta' - \mathbf{V}(\theta).$$

(6.14)

Since, in general,

$$\argmin_y \mathbf{E}\left( (X - y)^2 \right) = \mathbf{E}(X),$$

(6.15)

we can rewrite (6.14) as

$$\min_g \sum_a \int \int (\theta' + \omega' T - g(a))^2 p(\theta', \omega' | a)p(a)d\omega'd\theta' - \mathbf{V}(\theta).$$

(6.16)

where we are taking the minimum over all functions of measurement outcomes $g$. We can subtract any constant from inside the square of equation (6.16) without changing its value, as any constant shift will get absorbed in the minimum over $g$. We choose to subtract the constant $\mathbf{E}(\theta)$, yielding

$$\min_g \sum_a \int \int \left( (\theta' - \mathbf{E}(\theta)) + \omega' T - g(a) \right)^2 p(\theta', \omega' | a)p(a)d\omega'd\theta' - \mathbf{V}(\theta).$$

(6.17)

Expanding the square,

$$\min_g \sum_a \int \int \left( (\theta' - \mathbf{E}(\theta))^2 + (\omega' T - g(a))^2 + 2(\theta' - \mathbf{E}(\theta))(\omega' T - g(a)) \right) p(\theta', \omega' | a)p(a)d\omega'd\theta'$$

$$- \mathbf{V}(\theta).$$

(6.18)
Integrating out $\omega'$ in the first term gives

$$V(\theta) + \min_g \sum_a \int \int \left( (\omega'T - g(a))^2 + 2(\theta' - E(\theta)) (\omega'T - g(a)) \right) p(\theta', \omega'|a)p(a)d\omega'd\theta' - V(\theta).$$

(6.19)

The two $V(\theta)$ terms cancel. We can then factor $p(\theta', \omega'|a) = p(\theta'|\omega', a)p(\omega'|a)$. We know that $\theta'$ is conditionally independent of $a$ given $\omega'$, $p(\theta'|\omega', a) = p(\theta'|\omega')$ since $\omega'$ always provides more information about $\theta'$ than $a$. We can then rewrite (6.19) as

$$\min_g \sum_a \int \left( \int (\omega'T - g(a))^2 p(\theta'|\omega')d\theta' + 2(\omega'T - g(a)) \int (\theta' - E(\theta)) p(\theta'|\omega')d\theta' \right) \times p(\omega'|a)p(a)d\omega'. \hspace{2cm} (6.20)$$

We carry out the $\theta$ integral and obtain

$$\min_g \sum_a \int \left( (\omega'T - g(a))^2 + 2(\omega'T - g(a)) E(\theta - E(\theta)|\omega') \right) p(\omega', a)d\omega'. \hspace{2cm} (6.21)$$

This is of the form required by equation (4.21) and will minimize the posterior expected variance gain of $\theta$. From equations (6.14), (6.15) and (6.16) we see that the optimal choice for $g(a)$ is $g(a) = E(\theta + \omega T|a)$. Observe that the first term looks like $V(\omega T)$ and the second term looks like $2\text{Cov}(\omega T, \theta)$, although if $g(a)$ is chosen optimally as above, these correspondences are not exact.

In practice the optimal map $g(a)$ is approximated via a discretization $g(a) \rightarrow f_a$ selected with the techniques described in 5.2.3. While our discretization bounds cannot be applied to this cost function, we can refine the estimates iteratively as discussed, by setting $f_a = E(\theta + \omega T|a)$ after each iteration.

### 6.5 Sources of Error

There are three sources of error in the derivation of interrogation algorithms using $P(S_C)$:

1. discretization error in the SDP used to construct each unitary and POVM,
2. incomplete knowledge of the true length of each interrogation period, and
3. the inability to discretize and
store the complete state \( p(\omega_1, \omega_2, \ldots, \omega_n|a_1, a_2, \ldots a_n) \). The first issue was discussed in chapter 5; in this section, we discuss the latter two issues.

To address (2), recall that we have indicated that interrogation period \( n \) lasts for time \( T_n \). This is the "real" time for which the flywheel and the atomic system interact, and for which the flywheel is subject to noise. However, the length of each interrogation period is chosen by an experimentalist, who in general, doesn’t know \( T_n \); he can only estimate \( T_n \) from the flywheel or some external clock. We therefore have a mismatch between the chosen length of an interrogation period, which is used by the SDP to generate algorithms, and its true length. To analyze the effect of this mismatch, note that we expect any clock to be measured frequently enough such that, after any interrogation, the difference between its true phase and its predicted phase is less than \( \pi \). (Recall, for example, in Ramsey spectroscopy, for larger phase deviations, the relevant trig functions, equation (4.8), could not be unambiguously inverted.) This introduces a time error of at most \( 2\pi/\omega_0 \). Since \( \omega_0 \) is large, and therefore, \( 2\pi/\omega_0 \) is small compared to the length of a typical interrogation period, an algorithm optimized for such an erroneous time period is not expected to be much different than an algorithm optimized for the correct time period.

To address (3), note that the distribution of each \( \omega_n \) is inherently continuous and must be discretized sufficiently finely. However, if we represent each \( \omega_n \) with \( P \) points, then the size of the joint probability distribution grows by a factor of \( P \) at every step; if \( P \) is large, the strategy described above quickly becomes computationally infeasible. We must therefore store only a limited number of \( \omega_n \), and must marginalize out old distributions as the clock progresses. Fortunately, the SDP does not require the full joint distribution. However, the cost function (6.13), does require \( E(\theta_{n-1}|\omega_n, a_n) \), which is implicitly a function of all past \( \omega_n \). The following shows how this expectation, and more generally, \( E(\theta^k_n|\omega_n, a_n) \) for arbitrary \( k \) can be appropriately updated step by step even if \( p(\omega_n|a_{n-1}) \) is truncated.

Below, we use \( s_n \) to denote the state of the clock at timestep \( n \). Usually, we will take \( s_n = \omega_n \), but for illustrative purposes, here we allow for more general states. We assume we have \( E(\theta^k_n|s_n, a_n) \) at timestep \( n \) and need \( E(\theta^k_{n+1}|s_{n+1}, a_{n+1}) \) after an interrogation. We need \( k = 1 \)
to construct the cost function. \( k = 2 \) can be used to calculate variances, which, if we are able to store the complete, relevant clock history for a particular noise model, can be used to evaluate the performance of the clock. In what follows, we allow for arbitrary \( k \). We are given \( p(s_n|a_n) \) and can compute \( p(\omega_{n+1}, s_n|a_n) \) and all conditionals and marginals from the noise model. Note that \( \omega_n \) is deterministically implied by \( s_n \), and \( \theta_n \) is deterministically implied by all \( \omega_n \) and \( T_n \) up through interrogation \( n \). We assume all probability distributions have been discretized.

We start with

\[
E(\theta_{n+1}^k | s_{n+1}, a_{n+1})
\]

\[
E \left( (\theta_n + \omega_{n+1} T_{n+1})^k | s_{n+1}, a_{n+1} \right)
\]

\[
= \sum_{j=0}^{k} E \left( \binom{k}{j} T_{n+1}^j \omega_{n+1}^j \theta_{n+1}^{k-j} | s_{n+1}, a_{n+1} \right). \tag{6.22}
\]

To evaluate this we need

\[
E(\omega_{n+1}^j T_{n+1}^j \theta_{n+1}^{k-j} | s_{n+1}, a_{n+1})
\]

\[
= \sum_{\omega_{n+1}} \omega_{n+1}^j T_{n+1}^j E(\theta_{n+1}^{k-j} | \omega_{n+1}, s_{n+1}, a_{n+1}) p(\omega_{n+1} | s_{n+1}, a_{n+1}). \tag{6.23}
\]

We have that \( \theta_n \) is conditionally independent of \( \omega_{n+1}, s_{n+1}, \) and \( a_{n+1} \) given \( s_n \) and \( a_n \). Therefore,

\[
E(\theta_{n+1}^{k-j} | \omega_{n+1}, s_{n+1}, a_{n+1})
\]

\[
= \sum_{s_n} E(\theta_{n+1}^{k-j} | \omega_{n+1}, s_{n+1}, s_n, a_{n+1}) p(s_n | \omega_{n+1}, s_{n+1}, a_{n+1})
\]

\[
= \sum_{s_n} E(\theta_{n+1}^{k-j} | s_n, a_{n+1}) p(s_n | \omega_{n+1}, s_{n+1}, a_{n+1}) \tag{6.24}
\]

The last factor comes from the noise model. The desired result is obtained by combining Eq. (6.22), (6.23), and (6.24).
To construct the cost function, we also require $E(\theta_n^k | \omega_{n+1}, a_n)$. This is given by

$$E(\theta_n^k | \omega_{n+1}, a_n) = \sum_{s_n} E(\theta_n^k | \omega_{n+1}, s_n, a_n) p(s_n | \omega_{n+1}, a_n)$$

$$= \sum_{s_n, \theta_n} \theta_n^k p(\theta_n^k | \omega_{n+1}, s_n, a_n) p(s_n | \omega_{n+1}, a_n)$$

$$= \sum_{s_n, \theta_n} \theta_n^k p(\theta_n^k | s_n, a_n) p(s_n | \omega_{n+1}, a_n)$$

$$= \sum_{s_n} E(\theta_n^k | s_n, a_n) p(s_n | \omega_{n+1}, a_n).$$

(6.25)

In the third line we applied the conditional independence of $\theta_n$ on $\omega_{n+1}$ given $s_n$. The factor in the last sum can again be computed from the noise model.

Unfortunately, this truncation means that many noise models will lack their complete history. But since correlations typically fall off as a power law [64], we conjecture that this will not substantially impact the performance of our algorithms.

### 6.6 Performance Metrics

Before demonstrating the utility of our technique, we discuss metrics often used to evaluate the performance of clocks.

In many contexts, when quantifying the error or spread in a particular quantity, the standard variance is the tool of choice. Given a time ordered, discrete set of frequency measurements of size $M$, $\{\omega_i\}_i$, this is empirically estimated as

$$\sigma^2 = \frac{1}{M-1} \sum_{i=1}^{M} (\omega_i - \bar{\omega})^2,$$  

(6.26)

where $\bar{\omega} = \frac{1}{M} \sum_{i=1}^{M} \omega_i$ is the average of the $M$ measurements.

The variance, however, is not suitable when analyzing noisy quantities with correlations. The mean of such quantities will drift over time; therefore, since the variance is defined in terms of departures from the mean, it may be larger at longer time scales. As discussed in section 6.3, the noise affecting atomic clocks has long term correlations. These correlations result from the stationarity of the increments of relevant processes [65].
Because of these difficulties, an alternate metric, known as the Allan variance [6, 76, 77] is used to better assess clock stability. It is defined as

\[ \sigma_A^2(\tau) = \frac{1}{2(M-1)} \sum_{i=1}^{M-1} (\langle \omega_{i+1} \rangle_{(\tau)} - \langle \omega_i \rangle_{(\tau)})^2, \]  

(6.27)

where \( \langle \omega_i \rangle_{(\tau)} \) is the average frequency deviation of the flywheel during interval \( i \) of \( M \) which has length \( \tau \). Here, an interval can be identified with an interrogation period or a sequence of such periods. We assume there is no dead time between these intervals. Equation (6.27) corresponds to the average squared difference of adjacent average frequency deviations. We say that \( \sigma_A^2(\tau) \) is the Allan variance at averaging time \( \tau \).

A more informative estimate of frequency stability can be calculated by averaging frequencies over intervals that overlap. Assume that we have a collection of \( M \) averaged frequencies \( \langle \omega_i \rangle_{(T)} \). We define \( \langle \omega_j \rangle_{(mT)} \) as the average of the \( m \) subsequent \( \langle \omega_i \rangle_{(T)} \); specifically, \( \langle \omega_j \rangle_{(mT)} = \frac{1}{m} \sum_{i=j}^{j+m-1} \langle \omega_i \rangle_{(T)} \). For a given \( m \), construct all possible \( \langle \omega_j \rangle_{(mT)} \); the index \( j \) can take the values \([1, M - 2m + 1]\). Notice that the intervals used in these averages overlap. Then the overlapping Allan variance at averaging time \( \tau = mT \) is defined as the average of the squared difference of all adjacent (separated by \( mT \)) \( \langle \omega_j \rangle_{(mT)} \) [77], and is computed as

\[ \sigma^2(\tau) = \frac{1}{2(M - 2m + 1)} \sum_{j=1}^{M-2m+1} (\langle \omega_{j+m} \rangle_{(mT)} - \langle \omega_j \rangle_{(mT)})^2. \]  

(6.28)

This can be more easily expressed in terms of cumulative phases as

\[ \sigma^2(\tau) = \frac{1}{2(M + 1 - 2m)\tau^2} \sum_{i=1}^{M-2m+1} (\theta_{i+2m} - 2\theta_{i+m} + \theta_i)^2. \]  

(6.29)

Note that technically, an atomic clock is assumed to have some exact Allan variance, which could be computed, in principle, given an infinitely long run of a clock. Equations (6.27) and (6.28) are best considered to be finite estimators of an underlying, true quantity.

In our analysis, we use the overlapping Allan variance rather than the standard Allan variance. In addition, it is also natural for us to analyze frequency stability by computing the sample variance of a clock’s frequency deviation averaged over progressively longer intervals. This variance
is computed via several independent runs of a clock. We expect this to be different, in general, from the variance computed via a single long run of a clock, as the relevant noise processes are not ergodic.

### 6.7 Results

We can test our strategy on arbitrary noise models (the noise models discussed here with unbounded history or other entirely different classes of noise) via a monte carlo simulation of a real clock. In these simulations, the noisy flywheel moves randomly from frequency to frequency. We do not adjust its frequency to that of the standard; rather, as described earlier, we compute an estimate of the flywheel’s phase after every interrogation period. These estimates could subsequently be used to assign timestamps.

Consider such a simulation just before interrogation period \( n \). Assume that the average frequency deviations of the flywheel during interrogation periods \( 1 \ldots n - 1 \) were \( \omega_1 \ldots \omega_{n-1} \), and after these interrogation periods, we obtained measurement outcomes \( a_1 \ldots a_{n-1} \). Then we proceed as follows

1. From the frequency history \( \omega_1 \ldots \omega_n \) and the chosen noise model compute \( p(\omega_n|\omega_{n-1}, a_{n-1}) \) for the specific frequencies in our history. Sample from this distribution to obtain \( \omega_n \).

2. Extract the optimal algorithm for the upcoming interrogation period using procedure \( P(\hat{S}_C) \).

3. Apply this algorithm to a simulated clock at flywheel deviation \( \omega_n \). This yields a probability distribution over measurement outcomes \( p(a_n|\omega_n) \).

4. Sample from this distribution to obtain measurement outcome \( a^*_n \). This outcome is used in step 4 of \( P(\hat{S}_C) \).

5. The flywheel’s phase is estimated by \( P(\hat{S}_C) \) (described in step 7) as \( E(\theta_n|a_{n-1}, a^*_n) \) using the probability distributions maintained by \( P(\hat{S}_C) \).
We can utilize a large number of such runs to compute the sample variance of the cumulative phase. More specifically, the variance at step $n$ is computed as

$$\frac{1}{M} \sum_{i=1}^{M} (\theta_n^{(i)} - E(\theta_n^{(i)}|a_{n-1}^{(i)}, a_n^{(i)}))^2$$

(6.30)

where $\theta_n^{(i)}$ is the phase of the simulated clock after interrogation period $n$ during run $i$ of $M$, and $E(\theta_n^{(i)}|a_{n-1}^{(i)}, a_n^{(i)})$ is $P(\hat{S}_C)$’s estimate of that phase. We can also compute the overlapping Allan variance of each run and average each of these independent estimates. This simulation is used to generate all results that follow. However, it is not actually needed if the noise model used by $P(\hat{S}_C)$ matches the noise that actually affects the clock. In this case, the expectations $E(\theta_n^2|\omega_n, a_n)$ and $E(\theta_n|\omega_n, a_n)$ computed in section 6.5 can be used to compute the metrics discussed in section 6.6 directly.

We compare our method to standard Ramsey spectroscopy, and as in chapter 5, to the algorithms of Buzek, et. al. [18]. Since the algorithms in [18] are optimal for uniform prior probability distributions of flywheel deviations, this comparison demonstrates the advantage in tailoring algorithms to the true distribution and the need for the full strategy described here. For now, we limit our comparisons to clocks with low noise in order to reduce errors of size $2\pi/T$ (fringe hops). Different methods have different heuristics for dealing with this issue and an analysis of optimal strategies is not given here.

As discussed, conventionally, at the end of a clock interrogation, the flywheel is adjusted based on the measurement result in an effort to lock its frequency to that of the frequency standard. While mathematically, this is not strictly necessary if one just wishes to time-stamp a set of events, Ramsey spectroscopy performs best when the prior is centered around zero. Nonetheless, it can be shown that a measurement given prior $p(\omega+\omega_s)$ in the $|+\rangle, |-\rangle$ basis is equivalent to a measurement given prior $p(\omega)$ in the rotated basis $R_z(\omega_sT)|+\rangle, R_z(\omega_sT)|-\rangle$. Therefore, if desired, we can achieve equivalent performance without frequency correcting by using an adjusted algorithm. Note that given measurement outcome $a$, the optimal choice for such a correction (or rotation) is the posterior mean $E(\omega|a)$. Since posteriors have not been available experimentally, an alternate heuristic is used.
However, here we make this optimal choice. The method we compare is therefore slightly better than that used in practice.

Figure 6.4 compares our method to that of Ramsey and to that of Buzek on a two qubit clock subject to Brownian motion ($\alpha = -2$) with $h_{-2} = .03$ for 100 timesteps. Since Brownian motion is memoryless, it can be modeled completely without the need to truncate probability distributions. Figure 6.5 is an identical comparison on a three qubit clock subject to $1/\omega$ noise, ($\alpha = -1$), with $h_{-1} = .05$. We cannot maintain the infinite history required by this noise model and derive our algorithms by truncating the frequency history after only a single step. Note that while Buzek’s algorithms do not perform well here, we expect them to perform better in clocks with large numbers of atoms. Table 6.1 summarizes the improvements achieved by our technique. These results are consistent with those predicted in [79]. We expect these improvements to increase slightly by storing a more complete frequency history and increase significantly in clocks with additional atoms.

<table>
<thead>
<tr>
<th>Noise Type</th>
<th>Ramsey</th>
<th>Buzek</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brownian</td>
<td>46.9</td>
<td>51.7</td>
</tr>
<tr>
<td>Brownian</td>
<td>49.3</td>
<td>50.9</td>
</tr>
<tr>
<td>$1/\omega$</td>
<td>18.9</td>
<td>39.8</td>
</tr>
<tr>
<td>$1/\omega$</td>
<td>21.6</td>
<td>33.0</td>
</tr>
</tbody>
</table>

Table 6.1: Percent improvement of our method over those of Ramsey and those of Buzek. We average the gains in average frequency variance over the last twenty (81-100) timesteps and those in overlapping Allan variance across all averaging times.

While the algorithms derived here already significantly outperform traditional, fixed algorithms, we can obtain further improvements by choosing the interrogation time $T$ optimally at every step. Longer interrogation times will provide more information, however, if $T$ is chosen too large the clock will fringe hop. While this issue is beyond the scope of this thesis, we believe our simulations can be adapted to choose $T$ intelligently. Also, while our algorithms perform well even when derived with a significantly truncated frequency history, additional storage will, nonetheless, always be advantageous. Unfortunately, this often requires a dramatic increase in computational time. We would like to understand this tradeoff more completely, and ideally, develop a systematic
Figure 6.4: A comparison of our method (bottom blue solid line), standard Ramsey spectroscopy (middle red dotted line), and Buzek’s method [18] (top green dashed line) on a two qubit clock subject to fractional Brownian motion with $H = .5$ and $h_{-1} = .03$. We fix the length of the interrogation period to $T = 1$. Figure (a) is a log-log plot of the standard deviation of the frequency deviation averaged over cumulative time, while (b) is a log-log plot of the overlapping Allan deviation. These are computed via 200 iterations of our method, 1000 iterations of Ramsey spectroscopy, and 1000 iterations of Buzek’s method.

Figure 6.5: A comparison of our method (bottom blue solid line), standard Ramsey spectroscopy (middle red dotted line), and Buzek’s method (top green dashed line) on a three qubit clock subject to $1/\omega$ noise with $h_{-1} = .05$. Plots (a) and (b) are as described in Fig. 6.4 and are computed via 400 iterations of our method, 800 iterations of Ramsey spectroscopy, and 800 iterations of Buzek’s method.
way to determine when to cutoff the clock’s history.
Chapter 7

Conclusions and Discussion

Implementation of the method developed here will undoubtedly result in improved clock performance, particularly as the most accurate passive clocks today use standard Ramsey spectroscopy. Furthermore, our method subsumes much of the prior work discussed in chapter 4. Indeed, many of these works derived optimal interrogation algorithms under certain sets of assumptions; given the same set of assumptions, SDP $\tilde{S}_C$ will derive identical, or in many cases, superior algorithms.

However, such performance cannot be achieved with a single, fixed algorithm. That is, we cannot simply write down an analytic formula for a single set of optimal initial atomic states or for an optimal POVM. Instead, a different interrogation algorithm must be derived for each interrogation period. This is, of course, computationally nontrivial, but we obtain acceptable performance on an average desktop at the time of writing, as long as probability distributions are not discretized unnecessarily finely, long histories are not stored, and only clocks with small numbers of atoms are considered.

If, on a particular system, or given a particularly fine discretization, our method is unable to keep up with a physical clock, dead time can be introduced between clock interrogations. This is often done experimentally, and while not explicitly considered here, can be added to our scheme straightforwardly. One could also consider approximating the flywheel’s probability distributions by analytic functions. In many cases, a Gaussian would be a good fit; with this choice, convolutions (used, for example, when the appropriate noise model is applied) can be implemented easily. Other approximations, either to the update procedure, or to the SDP, may be possible. Finally, for a
small number of sequential interrogations on a clock with well known noise properties, one could consider generating all possible interrogation algorithms ahead of time. Of course, the number of potential algorithms for $M$ interrogation steps is given by $|F|^M$, where $|F|$ is the number of measurement outcomes, so such an approach quickly becomes infeasible. However, in practice, the optimal algorithm often does not change substantially from step to step, so it may be possible to prepare a set of near optimal algorithms in advance and simply choose from within this set as the clock advances.

Still, because SDP $\tilde{S}_C$ scales as $O(N^3t_f^3 + N^2t_f^2|F|)$, our method will generally be limited to clocks with small numbers of atoms. However, the most accurate clocks now in development use a suitably small number of atoms [28, 19, 81, 29] and are well within the applicability of our method. Furthermore, full quantum control over systems of comparable size has already been demonstrated in ion traps [43]. Therefore, we expect that it may be possible, albeit difficult, to implement this method in practice soon.

Note that, in contrast, in other domains where quantum algorithms have been proven to offer advantages, solving useful instances of interesting problems requires control over quantum systems of sizes far beyond what is achievable experimentally. Indeed, clocks may be among the first systems where a nontrivial quantum algorithmic gain is realized.

Perhaps it is surprising that even though modern clocks can already measure time with remarkable precision, (it’s estimated that the aluminum optical clock at NIST [84] is accurate to one second in 3.7 billion years) that we still seek to do better. It’s certainly true that accurate clocks are needed for a wide range of applications, from the global positioning system to coordinating activities on microprocessors or the internet. Indeed, the positional accuracy of GPS is dependent foremost on the accuracy of the satellites’ on-board clocks. But one of the most interesting applications of clocks, one that often needs more precision than even the best modern clocks, is probing physics itself. Experiments have been done which test for variation in the fine structure constant [73], and recently, which confirm general relativity on small length scales [20]. It is somewhat remarkable that gravitation can be explored entirely with quantum mechanics, and that such experiments can
be improved upon with ideas from computer science.

No one is sure how precisely time can ultimately be measured. As discussed, in addition to our method here, one can consider optimizing the interrogation time. In fact, a recent result [80], shows that interrogation times can be greatly increased by using ensembles of atoms with scaled resonant frequencies. One could, of course, consider an altogether different method of interrogation, or even more drastically, timekeeping with entirely different technology. As discussed in chapter 4, timekeeping technology has changed in the past and is likely to do so again.

Perhaps, one day, through all these advances, our understanding of what time *is* will match the precision with which we can measure it.
Bibliography


.1 Code

While the code for our complete optimization procedure is far too extensive to reproduce in this document, here we discuss a few of the issues that need to be considered when developing such software.

.1.1 SDP Solver

There are a number of commercial and open source SDP solvers available. These are written for several different languages, with C++ and Matlab being perhaps the most common. As discussed in section 3.3, our results were produced using SDPT3 [89], a Matlab solver.

There are advantages and disadvantages of using a Matlab solver. Generally speaking, SDP’s can be written cleanly, and errors are therefore often easier to catch. This is especially true if YALMIP [60] is used. With this package, one can code SDP’s using syntax that nearly matches their pure, mathematical description. Our measurement constraints can be written, for example, as
\[
C = [ C, \sigma(:,:,i) \geq 0 ];
\]
\[
C = [ C, \text{sum}( \sigma,3 ) == pOracle(:,:,T) ];
\]

Matlab solvers are significantly slower than their C++ counterparts, however. Therefore, if interrogation algorithms are needed in real time, Matlab may not be the best choice. In Matlab, a single run of SDP $\tilde{S}_C$ takes $< 10$ seconds on a modern computer at the time of writing. However, it is usually necessary to refine the selection of frequency estimates. They usually converge rapidly within an 2-3 iterations, but this still substantially increases running time.

.1.2 Probability Updates

The results depicted in chapter 6 are generated using an optimization routine coded in C++ which calls the Matlab solver discussed above. The core of this routine is functionality that manipulates and updates probability distributions. These updates dominate the running time so it is
important that they be implemented efficiently. Our code uses a recursive algorithm to progress through each value of each variable of one or more probability distributions; a function object, which computes a standard probabilistic operation (e.g. marginalization, posterior computation) is passed as a parameter.

Running time is strongly affected by the number of points used to discretize each probability distribution, as well as their maximum dimension (i.e. the number of random variables, generally corresponding to the clock at times in the past). Discretizations of up to 500 points can be simulated reasonably efficiently; this decreases as dimension increases. Distributions of dimension 1 can be simulated quickly, dimension 2 adequately, and dimension 3 very slowly. Distributions of dimension 4 or higher are currently infeasible.

Individual runs of the optimization procedure and the clock simulator are completely independent of one another. Therefore performance analyses like those generated here can be easily run in parallel.

.1.3 SDP $\tilde{S}_C$

Here we provide important pieces of the code used to solve clock SDP $\tilde{S}_C$. The code is written in Matlab and Yalmip.

**SDP $\tilde{S}_C$**

```matlab
function [minimum,pQO1,pQO2] = ClockSDPSeries( Objective, Q, 
    numberOfOutcomes, T, TP, estimates, priorDomain, prior, phases )

oracleSize = 1 + ( Q - 1 ) * T;
% Construct needed SDP vars. For the clock oracle, we can assume a block diagonal pOQ
pOracle = sdpvar( oracleSize, oracleSize, T, 'hermitian','complex' );
```

\[
\text{sigma} = \text{sdpvar( oracleSize, oracleSize, numberOfOutcomes, 'hermitian','complex' );}
\]
\[
\text{for } t = 1:T \\
\quad \text{pJoint}\{t\} = \text{blkvar}; \\
\quad \text{for } i = 1:Q \\
\quad \\n\quad \quad \text{pJoint}\{t\}(i,i) = \text{sdpvar( oracleSize, oracleSize, 'hermitian','complex' );} \\
\quad \text{end}
\]
\[
\text{end}
\]
\[
\text{for } t = 1:T \\
\quad \text{pJoint}\{t\} = \text{sdpvar( pJoint}\{t\} );
\]
\[
\text{end}
\]
\[
\% SDP constraints \\
C = []; \\
\% Initial density matrix \\
initial = \text{zeros( oracleSize, oracleSize );} \\
initial(1,1) = 1; \\
C = [ C, \text{imag( pOracle(:,:,1) ) == zeros( oracleSize,oracleSize ) } ]; \\
C = [ C, \text{real( pOracle(:,:,1) ) == initial } ]; \\
\% Positivity \\
\text{for } t = 1:T \\
\quad C = [ C, \text{pJoint}\{t\} \geq 0 ] ;
\]
\[
\text{end}
\]
\[
\% Measurement constraints \\
\text{for } i = 1:\text{numberOfOutcomes} \\
\quad C = [ C, \text{sigma(:,:,i) } \geq 0 ];
\]
\[
\text{end}
\]
\[
\% Remote state preparation \\
C = [ C, \text{sum( sigma,3 ) } == \text{pOracle( ;,:,T ) } ];
\]
% Partial trace of pOQ = pO

for t = 1:T
    entriesTrace = PartialTraceQ( pJoint{t}, oracleSize, Q );
    for x = 1:oracleSize
        for y = 1:oracleSize
            C = [C, pOracle(x,y,t) == entriesTrace(x,y) ];
        end
    end
end

% Oracle application

for t = 2:T
    entriesQuery = PartialTraceQ( ApplyClockOracleBasis( pJoint{t-1}, oracleSize,Q ), oracleSize,Q );
    for x = 1:oracleSize
        for y = 1:oracleSize
            C = [C, pOracle(x,y,t) == entriesQuery(x,y) ];
        end
    end
end

% Solve the SDP

settings = sdpsettings('solver','sdpt3');
solvesdp(C, Objective( sigma, oracleSize, numberOfOutcomes, TP, estimates,
        priorDomain, prior, phases ), settings );

% Output minimum cost and optimal density matrices

minimum = double( Objective( sigma, oracleSize, numberOfOutcomes, TP, estimates,
        priorDomain, prior, phases ) )
pQO1 = double( pJoint{1} );
pQO2 = double( pJoint{2} );

end

% This is the function passed as objective in solvesdp. It corresponds to the
% cost function in the alternate oracle basis

function objective = ClockObjectiveSeries( sigma, O, sigmaSize, TP,
                                           guesses, priorDomain, prior, phases )

objective = 0;
for a = 1:sigmaSize
    for r = 1:O
        for s = 1:O
            objective = objective + sigma( r,s,a ) .* priorIntegral;
        end
    end
end

function y = priorIntegral()

y = 0;
for x = 1:length( priorDomain )
    y = y + real( prior(x,x) ) .* exp( -1i .* ( r - s ) .* priorDomain(x) .* TP )
        .* ( ( priorDomain(x)*TP - guesses(a) ).^2
            + 2 * ( priorDomain(x)*TP - guesses(a) ) .* phases(x) );
end
% Functions to compute and apply the clock oracle in the alternate basis

function pQOOut = ApplyClockOracleBasis( pQO,O,Q )

oracle = GenerateOracleBasis( O,Q );
pQOOut = oracle * pQO * oracle';

function oracle = GenerateOracleBasis( O,Q )

% The oracle in the alternate basis shifts the entries of system O by i, the state of the querier
LS = zeros( O,O );
for x = 1:O - 1
    LS( x + 1, x ) = 1;
end
oracle = zeros( O*Q, O*Q );
for i = 1:Q
    oracle( ( i - 1 ) * O + 1 : ( i - 1 ) * O + O, ( i - 1 ) * O + 1 : ( i - 1 ) * O + O ) = LS( i - 1 );
end