



Department of Physics
The Blackett Laboratory
Quantum Optics and Laser Science Group

Quantum Coins and Quantum Sampling

Howard Dale

Thesis submitted in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy in Physics
of
Imperial College London

August 2016

Abstract

Quantum computing is rapidly becoming one of the largest research areas in quantum physics and yet we are still unable to prove the superiority of quantum computing over classical with the exception of a few contrived and unrealistic scenarios. Understanding the areas in which quantum computing provides an advantage and, more importantly, where that advantage comes from is the key to making progress in the field. We certainly do not solve this mystery in this thesis but do explore some unconventional routes through which answers may be found in the future.

To this end we explore quantum advantage in a different computational setting. Rather than working with Turing Machines as most research does, we focus on the Bernoulli Factory: a randomness processing scheme studied in the fields of statistics and computer science.

The Bernoulli Factory takes an infinite string of random bits generated with a fixed unknown bias as input and outputs a single random bit, the bias of which is a function of the input bias. This can be seen more intuitively as generating one biased coin from another, given a desired relationship between the two biases.

The Bernoulli Factory is inherently probabilistic, like quantum mechanics, and so, it leads us to questions about the fundamental nature of quantum randomness and quantum sampling.

We find, as we hoped, that the Bernoulli Factory is more tractable than the Turing Machine and that we are able to prove quantum superiority in both resource efficiency and classes of computable functions.

We first give the quantum analogue of the classical Keane-O'Brien theorem which establishes the increased class of computable functions for the quantum case. We then adjust the problem in a variety of ways, such as introducing multiple coins, errors in state preparation, phases and restriction on the bias or operations allowed. These lead to additional results which lend insight into the problems faced in quantum computing as well as the nature of quantum randomness.

Acknowledgements

I would first like to thank my supervisors: Terry Rudolph and David Jennings. I have learnt a lot more from them about life than I have about physics, and I learnt quite a lot of physics. After spending the first two decades of my life coasting the world of research with its quagmires and pitfalls was a big shock to me. The guidance they provided allowed me to bend where I might have broken.

I was lucky to be part of the third cohort of the Controlled Quantum Dynamics DTC. Our group bounced off each other tremendously well and I still look back on my first year at Imperial as one of the best in my life. In particular Jake Iles-Smith, Naomi Nickerson and Mercedes Gimeno-Segovia for their contributions to my personal development and for innumerable fascinating distractions, conversations and whiteboard parties.

I would also like to thank the many friends I made during the PhD itself who made the twelfth floor a fun place to work, including but not limited to: Ben, Antony, Kamil, Matteo, Pete, Chuanqi and Kate. I would also like to thank Kyle McEnery for joining me in our dark corner and helping me realise that there were a lot more routes available to me that weren't physics.

My family have been invaluable through some difficult times and I hope they know how much I appreciate their support. I love you guys and I do not tell you enough.

Most importantly I would like to thank Claudia Parkes for saving my life. Without you I would not be anything worth being. Every time I think I am done learning from you I am swiftly proved wrong. I hope we continue to grow together.

Declaration of Originality

I hereby declare that this thesis, and the work reported herein, was composed by and originated entirely from myself. Information utilised from the published and unpublished work of others has been acknowledged in the text with references.

Howard Dale

Copyright declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

‘Hard work beats talent when talent fails to work hard’

Tim Nolte

Contents

List of Figures	9
Introduction	13
1 Mathematical Preliminaries	22
1.1 Sample Space	22
1.2 Probability Measures	23
1.3 Events and Functions	24
1.4 On discernible events and open events	26
1.5 On discernible events and termination	27
2 The Bernoulli Factory	29
2.1 What is a Bernoulli Factory?	29
2.1.1 Relation to Turing Machines	31
2.1.2 Motivation	31
2.2 Classical Result	32
2.2.1 Formal statement of the problem	33
2.2.2 Classical Proof	33
3 The Quantum Bernoulli Factory	39
3.1 Why make it quantum?	39
3.1.1 How should it be made quantum?	40
3.2 Formal statement of the problem	40
3.3 Quantum result	42

3.4	Discussion	51
3.4.1	On quantum advantage	51
3.4.2	On practicality	52
4	Variants on the Quantum Bernoulli Factory	53
4.1	Changing the domain	53
4.1.1	Wastlund approach	54
4.1.2	Formal statement of the problem	54
4.1.3	Result on reduced domain	55
4.1.4	Discussion	62
4.2	Multiple Parameters	63
4.2.1	Two different coins	63
4.2.1.1	Formal statement of the problem	64
4.2.1.2	Result for two different coins	64
4.2.2	One coin with two parameters	69
4.2.2.1	Formal statement of the problem	69
4.2.2.2	Result for two parameters	70
4.3	Automata	72
4.3.1	Finite Events	73
4.3.2	Finite and Pushdown Automata	77
4.3.3	Coin distinguishing problems	79
4.4	Discussion	81
5	Practicalities of the Quantum Bernoulli Factory	83
5.1	A practical example	83
5.2	Comparisons with the classical case	87
5.3	Errors	89
5.3.1	Mixed States	89
5.3.2	Errors in p -values	90
5.3.3	Dark Counts	91

5.3.4	Combined Errors	91
5.3.5	Discussion	93
5.4	Implementation	94
5.4.1	Photonic Chips	94
5.4.2	Superconducting Qubits	94
5.5	Joint measurements	96
6	Q-Sampling and Quantum Foundations	98
6.1	Sampling and Q-sampling	98
6.1.1	The power of Q-sampling	98
6.1.2	What can be q-sampled?	99
6.2	Foundational discussion	103
	Conclusions	105
	References	108

List of Figures

1.1 Views of Quantum Computing: Three possible views of **BQP**, marked in red, in relation to other complexity classes. We label the views, from left to right, “pessimistic,” “middle-ground” and “optimistic.”

2.1 Approximations used in the classical Bernoulli Factory: The result of the described approximation for 17 (orange), 101 (blue) and 501 (green) to a simple function (red). We note that it does not converge to our function for large n .

3.1 Unsuccessful Bounding Functions We attempt to bound the function from above and below. Here we show $A_{100}(p)$ and $B_{100}(p)$, and their counterparts for very large n , for the piecewise linear function depicted. Note that both fail to be good bounding functions when $f(p)$ gets close to zero or one.

3.2 Our q-constructible “Heaviside” function Here we show an example of an important function for this task, which approaches zero rapidly close to the chosen point. This function is $T_{400,3,0.5}$

3.3 Further Difficulties with Bounding Functions The product function $A_{100}(p)T_{400,5,1/2}(p)T_{400,1,3/4}(p)$ is *nearly* a good lower bounding function for $f(p)$. It struggles at 0.5 from being pushed too flat and at 0.75 from not being flat enough.

4.1 One example of the unusual functions one could construct using Theorem 2 The actual discontinuities are excluded. Effectively we gain access to

piecewise continuous functions but with their discontinuous points excluded, leaving a function which is continuous on its domain.

4.2 Permitted regions for the quantum two coin problem In addition to points, horizontal and vertical lines (shown in blue) where $f(p, q) = 0$ are permitted. Regions, halflines and diagonal lines (shown in red) are not permitted.

4.3 A function constructible with joint measurements. We note that along the line $p = q$ the function is zero. This is not possible with only single qubit measurements.

4.4 Figure 4.4: A selection of quantum finite events Quantum finite events only possible with at least three-qubit measurements (purple) and at least two qubit measurements (blue). Also included is a simple single-qubit measurement finite event which cannot be constructed by the classical factory, finite or non-finite (green).

4.5 An Automaton Finite Automaton, this machine only has a finite number of internal states but still can run indefinitely. A finite *event* would have a similar diagram but with no loops allowed.

4.6 A cube depicting the eight automata considered We note that all stronger models can perform this coin distinguishing operation, but the weaker model cannot. This does not allow us to solve the problem.

5.1 The probability amplification function This shows the function $f_{\wedge}(p) : [0, 1] \rightarrow [0, 1]$ and the first 35 terms of its decomposition.

5.2 The random walk used in the algorithm The algorithm uses a random walk on a semi-infinite ladder with transitions given by Bell-Measurement outcomes on two qubits.

5.3 Efficiency of $2p$ algorithm This graph shows the expected number of flips required to execute the algorithm for a given value of p .

5.4 The effect of decoherence on the observed function For most regions this produces a function close to the target but at points where the function approaches zero or one the deviation is more pronounced.

5.5 The effect of deviations in p -values on the observed function We note that this produces some amount of drift which depends on the derivative of the function.

5.6 The effect of dark counts on the observed function We note that this produces higher deviations the further the function is from a value of one half.

5.7 Simulated data from a “realistic,” although very error-prone, experiment We see that at $p = 0.5$ the deviation is most severe and that the spread of the datapoints is wider towards the endpoints.

5.8 Comparison of quantum and classical finite events to the simulated data We see that both classical (red) and quantum (green) factories are able to get good fits to the data with quite a low amount of coins although the quantum uses fewer.

5.9 Efficiency of single qubits only The number of flips required to generate $4p(1 - p)$ with single qubit operations only using the method described in [Y⁺16]

Parts of this thesis are based on material in the following papers:

- H. Dale, D. Jennings and T. Rudolph. Provable Quantum Advantage in Randomness Processing. Nat. Commun. 6:8203 doi: 10.1038/ncomms9203 (2015) [DJR15].
- H. Dale and T. Rudolph. Practicalities of Quantum Advantage in Randomness Processing. To be submitted for review shortly
- A paper based on the content of Chapters 4 and 6 is currently in preparation

The following work was not included in the thesis:

- Goodwin, J. F., Brown, B. J., Stutter, G., Dale, H., Thompson, R. C., & Rudolph, T. (2015). Trapped-ion quantum error-correcting protocols using only global operations. Physical Review A, 92(3), 032314.

Introduction

How powerful is a quantum computer?

This question has been debated heavily since quantum computing was first posited [F81] and it would be bold to say that a clear consensus has formed. In light of this we will begin with statements that are as uncontroversial as possible [D85].

Quantum computing does not violate the Church-Turing Thesis.

The Church-Turing thesis states that all computable functions can be computed by a Turing machine; it makes no claims about efficiency [C36,T36]. A computable function is any function that could be calculated in finite time on a real device. It is worth introducing computability here as not only does it help us to place quantum computing in context but we will also find that in the non-Turing-complete models we introduce later quantum mechanics can increase the class of computable functions.

There is no question as to whether quantum computing as we currently understand it can be simulated by a classical computer, the only question is whether said simulation can be made efficient [BV97]. This means that quantum computers and classical computers have the same class of computable functions. It also means a quantum computer given polynomial time can be no more powerful than a classical computer given exponential time.

This gives us a very loose upper bound on the power of quantum computing, now we turn to the lower. Quantum computers must be at least as powerful as classical

computers because they can simulate them. Quantum computers also provide their own randomness, and so must be at least as powerful as a probabilistic classical computer [B+97].

Computational Complexity

To speak more clearly about the relative power of a quantum computer we will give very brief definitions of several computational complexity classes

[BV97,L73,C71,ML72,V79,T91,F09,A16,P03,A13].

P, which stands for Polynomial time, is one of the most fundamental complexity classes, it contains all problems solvable by a classical deterministic computer, or equivalently a deterministic Turing machine, in time and space polynomial in the size of the input. All the classes listed below are also restricted to polynomial space.

BPP, which stands for Bounded Probability Polynomial time, is the class of problems solvable by a probabilistic classical computer with bounded probability of success (greater than $2/3$) in polynomial time. It clearly contains **P** as a probabilistic classical computer is simply a classical computer with additional capabilities.

BQP, which stands for Bounded probability Quantum Polynomial time, is the class of problems solvable by a quantum computer with bounded probability of success (greater than $2/3$) in polynomial time. It clearly contains **P** and **BPP** as a quantum computer can simulate a probabilistic classical computer.

NP, which stands for Non-deterministic Polynomial time, is the class of problems the solutions to which can be verified by a classical deterministic computer in polynomial time. It clearly contains **P** as one can check the solution by simply solving the problem. The question of whether **P** is equal to **NP** is considered one of the greatest problems of computer science and mathematics.

PH, which stands for Polynomial Hierarchy, is the union of many classes. If **NP** is the class of problems verifiable in **P** then we can also define **NP^{NP}**, the class of problems verifiable in **NP**. We can keep iterating and define an infinite tower of complexity classes.

#P, or “number-P,” is the class of counting problems for **NP** problems. If an **NP** problem asks “is there a solution?” then its **#P** equivalent asks “how many solutions?”. For the inclusions shown to be precise and based on decision problems the class **P^{#P}**, the class of decision problems solvable in polynomial time by a computer with the ability to instantly solve an **#P** counting problem, should be considered. Toda’s Theorem [T91] tells us that **P^{#P}** contains **PH**.

PP, which stands for Probabilistic Polynomial time the class of problems solvable by a classical computer in polynomial time with error probability of less than one half. This class is equivalent to $\mathbf{P}^{\#\mathbf{P}}$.

PSPACE, which stands for polynomial space is the class of problems solvable by a computer which is polynomial in space but may take exponential time to solve the problem.

This figure shows three positions one might hold on the power of quantum computing. The pessimist believes that the addition of quantum mechanics offers no advantage to computing. The optimist believes that quantum computers can solve the hardest **NP** problems such as the Travelling Salesman or Knapsack problems as well as problems entirely outside the polynomial hierarchy. Unsurprisingly most, but certainly not all, academics occupy the middle ground [A10,FR98,F⁺98].

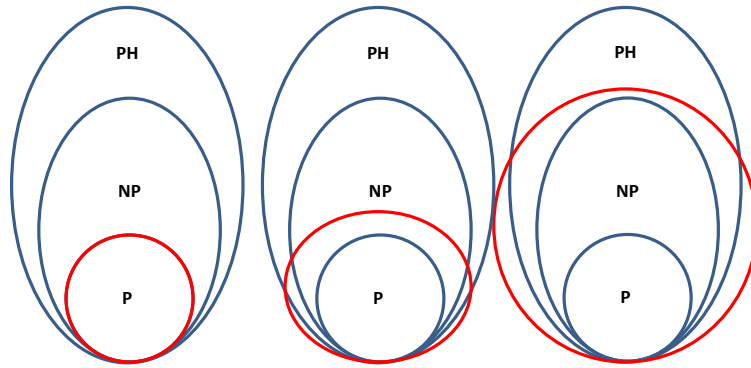


Figure 1: Views of Quantum Computing Three possible views of **BQP**, marked in red, in relation to other complexity classes. We label the views, from left to right, “pessimistic,” “middle-ground” and “optimistic.”

There are two key points of contention here: the relationship between **BQP** and the entire polynomial hierarchy, and the relationship between **BQP** and **NP** specifically.

The Polynomial Hierarchy

To discuss the polynomial hierarchy it is essential to form some intuition for the typical problems at each level. One helpful way to think of it is in terms of games. We will phrase our examples in terms of chess although there are many other games that would work. We must actually consider chess variants on an n -by- n board since complexity theory is about how well our algorithms work for a family of problems of size varying with n .

In chess the objective is to capture the king. If we wish to check if a candidate move is a winning move, we simply check to see if it captures the king and is a legal move. This is an easy problem and will be in $\mathbf{P}$ for all sensible chess variants.

Now imagine we have a board state and we wish to know if there is a winning move. In order to solve this problem we must check all possible moves for both victory and legality, which could be an exponential number of moves. However, if someone tells us what the winning move is then checking that it wins it is in $\mathbf{P}$, so checking if there is a winning move is in $\mathbf{NP}$ since this is the class of problems checkable in $\mathbf{P}$.

Finally imagine we have a board state and we wish to know if there is a move such that my opponent will have a winning move after I make it. Now for each of the possibly exponentially many candidate moves, we must then check the set of possible moves afterwards: a doubly exponential calculation. Whilst this problem is harder we can still see that if someone tells us the move that will lead to our opponent's winning chance then it is in $\mathbf{NP}$ to find our opponent's winning move, since this problem is checkable by an $\mathbf{NP}$ machine, we say that this problem is in $\mathbf{NP}^{\mathbf{NP}}$. We see how each level of the polynomial hierarchy corresponds to one additional move in our problem.

We can now understand intuitively why the polynomial hierarchy is believed to be infinite. If it is not infinite then at some point, say seven, all subsequent levels will be equal; solving nine moves will be no harder than seven. In fact this

would imply that solving the entire game, from start to finish, would be no harder than finding a win in seven moves. So if a result would imply that the polynomial hierarchy stops increasing at level k , usually phrased as “collapses to the k th level,” then we should be as sceptical of this as we would be of someone saying “solving chess is no harder than finding a win in k moves.” Chess, and similar games, have been the subject of a great deal of research and have had the resources of nations levelled against them. The suggestion of a simple k -move solution would be greeted with extreme skepticism by the academic community [FL81,NSS58,V06].

Evidence for the power of quantum computing

We have just explained why the polynomial hierarchy is almost certainly an infinite tower of complexity classes containing a wealth of problems, how then are we justified in claiming that the power of quantum computing might stretch outside of said tower even in our middle-ground position?

Firstly, just because a problem lies outside the polynomial hierarchy does not imply that it is in any sense harder, it may simply be a very different type of problem [A11]. Secondly, if **BQP** is contained in some finite level of the polynomial hierarchy we would expect there to be an algorithm to simulate quantum circuits at that level. At this point it is very hard to imagine what such an algorithm would look like, particularly where the finite element would enter. We do understand how to simulate quantum computers with a classical computer that has the additional capacity to solve counting problems instantly [ADH97,FR98] but this class, $\mathbf{P}^{\#\mathbf{P}}$, actually contains the entire polynomial hierarchy [T91].

We also have some evidence for abilities outside the polynomial hierarchy in the famous “Boson Sampling” problem[AA11]. This problem consists of sampling from the output of a linear optical circuit. It evoked great interest since the circuits required are not universal for quantum computing [KSV02] and yet it was shown, under certain assumptions, that even an approximate classical algorithm

would result in the collapse of the polynomial hierarchy to the third level [AA14]. Now since this is a sampling problem it cannot be compared easily to problems in the polynomial hierarchy (which are decision problems), however, this problem still makes it very difficult to imagine placing **BQP** at some level in the hierarchy without it collapsing. Another problem with almost identical consequences, although less robustness, is classically sampling from the outputs of quantum circuits with only commuting gates [BJ10]. It is also certainly interesting to note that the nature of quantum mechanics lends itself well to sampling and search problems and that exploring these, rather than the decision problems more common in classical complexity theory, might be an avenue worth pursuing [A11].

With what we have discussed of the polynomial hierarchy in mind we now turn to the relationship between **BQP** and **NP**. Almost of all of this discussion is centered around one problem: factoring. Shor’s discovery of a polynomial time quantum algorithm for factoring [S94] remains a strong candidate for the most important result in quantum computing. Factoring is believed to be an intermediate problem, being neither computable in polynomial time nor one of the hardest problems in **NP** [P03,BHZ87]. Factoring has been studied for centuries so there is a strong case that a polynomial time algorithm would have been discovered if one existed. It could be argued that a polynomial time algorithm for primality testing [AKS04] went undiscovered for nearly as long but there were successful primality testing algorithms that were randomized, non-general or not provably sound; factoring does not have any algorithms that are in some sense “close” to being polynomial. This lack of a classical algorithm is currently the strongest evidence for quantum computing exceeding classical computing, but proving that there is no classical algorithm is harder than proving $P \neq NP$ [P03].

Why not go further? Why, given the short time for which quantum computing has been studied, do we assume that there could not be quantum polynomial time algorithms for all problems in **NP**? Firstly we note that almost all problems we have quantum algorithms for have the property that both correct and incorrect

answers can be checked in polynomial time. The set of problems with this property is believed to be strictly smaller than **NP**, if it is not then the polynomial hierarchy collapses to the second level [P03,AR05.]. It is also interesting to note that small changes to the laws of quantum mechanics almost always result in trivial algorithms for **NP** problems [A04,AW09] and yet if there is an algorithm for **NP-hard** problems in the empirically correct version of quantum mechanics it is demonstrably far from obvious.

Oracles and Promises

So it seems we find ourselves in a difficult situation. All the things we wish to prove not only lie along difficult routes but often rest upon conjecture and supposition. How then do we proceed?

If we can't win the game; we must change the rules. Complexity theorists have long made progress by considering how results can be proven in worlds with different rules[BGS75,BG81,B87,CGH04]. The most common examples of this are worlds with oracle machines, some of which we have seen already.

When we write a complexity class like $\mathbf{P}^{\#\mathbf{P}}$, what we are really describing is the class of problems solvable by a **P**-machine with the ability to call upon a $\#\mathbf{P}$ -machine as a sub-routine.

The worlds where most proofs take place are worlds with black-box oracles and worlds with promises.

A black box oracle is an oracle machine the inner workings of which cannot be determined or investigated. In the proof that Grover's quantum unstructured search algorithm is optimal a black box has to be invoked because unstructured search cannot arise naturally [G97]. To clarify, if the problem of finding a specific state does not involve a black box then the known conditions on the state can be turned into a 3-SAT problem [C71] which then provides some structure to the search.

A promise problem is a problem where the possible inputs are restricted by an unbreakable promise. Often determining if an input satisfies the promise can be as hard or harder than solving the problem on the input. This prevents the promise from being absorbed into the algorithm. The Deutsch-Jozsa algorithm shows that quantum computing can outperform classical computing when deciding if a function is balanced or constant [DJ92]. It requires a promise that only balanced or constant functions, nothing inbetween, will be submitted as inputs.

We have already given two very well-known examples of quantum proofs that utilise oracles and promises. We hope this will make our step into an alternate model of computation for the rest of this work more understandable.

Concluding Remarks

We have established that the power of quantum computing has been the subject of much scrutiny and has still evaded a clear answer due to its being surrounded by some of the most difficult problems in mathematics and computer science. We have touched on the fact that quantum mechanics is inherently probabilistic and that much of its interesting computational behaviour can be found in sampling algorithms. We have lastly discussed how progress has been made in both classical and quantum complexity theory by considering alternative rules or computational models.

Where does this lead us? We consider an alternative computational model which is inherently probabilistic and in which all problems are sampling problems. The problem has been studied extensively in classical statistics and so a quantum generalization would have been of interest even had it not synchronised so well with the direction of current quantum computing research.

We will show in this model, the Bernoulli Factory, that quantum mechanics increases the class of computable functions, increases the efficiency with which many functions can be calculated and highlights fundamental ideas about what can and

cannot be sampled. We will then adapt and adjust the Bernoulli factory in a variety of ways in order to pin down exactly what aspects of quantum mechanics are providing this advantage and why.

Chapter 1

Mathematical Preliminaries

We begin by reviewing the mathematical tools required to understand the Bernoulli factory and variations upon it. Many of these may be unfamiliar to readers from a quantum mechanical background. Many of these definitions are based on those given in [W99], with some adaptations to make integration with the quantum case more straightforward.

1.1 Sample Space

The first thing we need to work with coins effectively is a formal way to talk about all the possible outcomes. If each flip is a sample X_i with result 0 or 1. We will call 0 “Tails” and 1 “Heads.”

Since most of our algorithms do not terminate after a fixed number of flips we need to consider infinite sequences. We represent the possible outcomes of this process by the set Ω of all infinite sequences $X = (X_1, X_2, X_3, \dots)$ such that $X_i \in \{0, 1\}$ for every i .

To navigate this space we define the following cylinder sets:

$$V(v_1, \dots, v_k) := \{X \in \Omega : X_i = v_i \text{ for } 1 \leq i \leq k\}. \quad (1.1)$$

These are the sets of all infinite strings of zeroes and ones that share a specific

initial segment. For example, $V(101)$ is the set of all infinite $\{0, 1\}$ -strings beginning with the initial segment 101.

1.2 Probability Measures

A probability measure maps a set of outcomes, in this case strings, to a single real value which is the probability of one of those outcomes occurring. Whilst many results later in the thesis will use more complicated probability measures we will restrict ourselves to that of a simple biased classical coin for this section. Each bias $p \in (0, 1)$, defines a probability measure

$$P_p(V(v_1, \dots, v_k)) = \prod_{i=1}^k p^{v_i} (1-p)^{1-v_i}. \quad (1.2)$$

Many of our definitions work very easily for open subsets of Ω , so it is worth detailing what it means for a subset to be open when it consists of infinite strings. A set is open if for every element in the set there is a finite neighbourhood around the element within which every element belongs to the set. The term neighbourhood simply refers to a set which contains the element in question which is usually chosen to be as small as possible. If we have an infinite string $X = (X_1, X_2, X_3, \dots)$ belonging to a set S then if S is open there exists an integer k such that $V(X_1, \dots, X_k) \subseteq S$.

Any element which cannot have a finite neighbourhood constructed around it is considered to be part of the boundary of a set. An open set does not contain any boundary elements, a closed set contains all of its boundary elements. Interestingly in a mathematical context these terms are not mutually exclusive, allowing for “clopen” sets but these will not be relevant here.

We can see immediately that all of our cylinder sets are trivially open because the entire set can be the neighbourhood for any element. This leads us to the obvious counterpoint: what does a closed set look like in this space? One example would be the set of all strings with more “Heads” than “Tails.” If we consider an element in this set, such as the string 110 repeated endlessly, we can see that it is

impossible to construct a neighbourhood around it. There are no finite strings which will always contain more “Heads” than “Tails” for every possible completing string, since we can always choose an infinite string of “Tails” as our completing string.

For closed sets we cannot use our probability measure in its current form, so we define an outer measure,

$$P_p(S) = \inf\{P_p(E) : E \text{ is open and } S \subseteq E\}. \quad (1.3)$$

This simply finds the open set with the smallest measure that still contains S . In many cases this will not yield a meaningful probability but nonetheless it is important to define it.

For our earlier example of a closed set, that of the set of all strings containing more “Heads” than “Tails,” this outer measure yields an entirely meaningless answer of unity.

1.3 Events and Functions

An *Event* is a set of possible outcomes. For example, when drawing one card from a standard 52-card deck the sample space is the set of all cards. In this scenario we could consider the event “the card drawn is a Diamond”, which would be the set of 13 outcomes which match this criterion.

We formally define an event as a subset S of Ω which, for all $p \in (0, 1)$, satisfies

$$P_p(S) + P_p(S^C) = 1. \quad (1.4)$$

An event is something which happens or does not happen. This is not as useful a definition as it first appears as it may not be clear whether an event occurs until infinite coins have been flipped. We are interested in events for which we can determine their occurrence with a finite number of flips.

Definition 1: An event E is *discernible* if there exist two sets, A and B , of finite

0 – 1 strings such that

(i) for every $p \in (0, 1)$, with probability 1 a string X will belong to $V(K)$ for exactly one string K belonging to A or B .

(ii) if this string K is in A then E occurs; if it is in B then E does not occur. If there is no such string K , which must necessarily be with probability zero, then E may or may not occur.

More intuitively, a discernible event is one for which, if it occurs or does not occur, then we will definitely know after an arbitrarily large but finite number of coin tosses. Discernible events can also be viewed as events for which the boundary, the set of infinite strings for which E may or may not occur, has measure zero.

Another way to think about discernible events is as follows:

- The sample space consists of all possible outcomes
- Each finite string K , whether it belongs to A or B , covers or “tiles” a portion of the sample space
- The sample space is divided into two regions: those where the event occurs and those where it does not occur.
- The strings in A should tile the region where the event occurs completely. Similarly the strings in B should completely tile the region where the event does not occur.
- If an event can be tiled in this way it is discernible; if it cannot then it is not discernible.

We will also occasionally be required to partition the sample space into more than two subsets to represent more than two outcomes. These are *functions* rather than *events*.

Definition 2: A function $\alpha : \Omega \rightarrow \mathbf{N}$ is discernible if $\alpha^{-1}(n)$ is a discernible event for every n .

A discernible function is one for which we will know the value of α after a finite number of coin tosses.

We will also occasionally be required to be more strict and require that our events occur after a set, finite, number of flips. It is possible for discernible event to have an infinite expected number of flips.

Definition 3: A finite event, E is a discernible event for which there exists an integer, k , such that after k flips we always know whether or not E has occurred.

These events are used in proofs throughout and are also discussed in their own right in Chapter four.

1.4 On discernible events and open events

It may seem at this point that open events are always discernible and vice versa; this is mostly, but not always, correct. Since both will be used throughout and are not interchangeable we provide two examples to illustrate the difference.

Consider the set of strings S which have an initial segment of any length which is repeated immediately such as 0111101111..., 1010... or simply 00... Clearly we can tell in finite time when this event does occur, but we can never tell when it does not occur. However the probability that it does not occur is greater than zero since

$$P_p(S) < \sum_{k=1}^{\infty} P_p(\text{Set of strings that repeat first } k \text{ flips}) \quad (1.5)$$

$$< \sum_{k=1}^{\infty} \frac{1}{2^k} \quad (1.6)$$

$$< 1. \quad (1.7)$$

We have here a total being less than the sum of its parts as these events are not mutually exclusive. For example the strings beginning 0000... are counted by

$k = 1, 2, 3$ at minimum.

Imagine we are using the our coin tosses to execute a random walk. Heads we increase our position by one; tails we decrease it. Let us now describe our event, E , as follows: “our position in the random walk reaches 2 before it reaches -1, or it never reaches either.”

We can tell whether this event occurs by observing a value of 2 or a value of -1 . There is one string, however, which will never allow us to determine whether the event occurs: the string $HTHTHTHT\dots$ alternating infinitely. Since this string occurs with probability zero, the event is discernible. However, this single string forms the entire boundary of E and is also contained within E . This means that E is a closed discernible event.

1.5 On discernible events and termination

A discernible event must terminate in finite time. A finite event always terminates after some number of flips, k . Whilst these seem similar it is important to note several key differences.

- There does not necessarily exist a value k after which a given event terminates.
Discernible does not imply finite.
- A discernible event does not necessarily terminate in a finite expected number of flips.
- Certain simple events can be shown to have infinite expected number of flips whilst still being discernible
- Events may have finite expected number of flips for some values of p but infinite for others.

A simple example would be that of an algorithm that operates in rounds. The k th round requires 2^k coins and terminates with probability $1/2$. One quickly see

that while this algorithm terminates in a finite time, since $\lim_{k \rightarrow \infty} 2^{-k} = 0$ it requires an infinite number of coins on average

$$E(\text{coins}) = 2 * \frac{1}{2} + 4 * \frac{1}{4} + 8 * \frac{1}{8} = \dots \quad (1.8)$$

$$= 1 + 1 + 1 + \dots \quad (1.9)$$

$$= \infty \quad (1.10)$$

This example is loosely based on the St. Petersburg Paradox [A77]. This so-called paradox draws attention to the fact that people in general are not willing to pay very much to play a game which has infinite expected value. Similarly we find, both in this work and in other real world applications, people are willing to run algorithms which terminate in infinite expected time. Usually in such cases the infinite expectation is caused by extremely rare events which are not typically observed. For example in the algorithm described above over 99% of runs will terminate within 128 flips. So, paradoxically, an algorithm with infinite expected runtime is not necessarily impractical.

We note also that even a Von Neumann coin [VN51], the most basic example which is discussed at the beginning of the next chapter, has runtime approaching infinity as p approaches either endpoint. Despite this we will still explore algorithms which have a fixed finite runtime (in Chapter 4) and a finite expected runtime (in Chapters 4 and 5).

Chapter 2

The Bernoulli Factory

2.1 What is a Bernoulli Factory?

Imagine you have a coin with unknown bias p , that is, it produces the result “Heads” with probability p which is unknown to you. Using only this coin, which may be flipped as many times as you need, you wish to output a random bit the bias of which is a function of p . This function, $f(p)$, is known to you. Another way to consider this is as producing a new coin from your original coin. We require this sampling to be exact; approximations will not suffice although they form an essential part of many algorithms in practise.

The best known instantiation of this problem is the “Von Neumann coin.” This riddle [VN51], now a very common interview question, asks “Given an unfair coin, how can you produce fair results?” Formally, given a coin of unknown bias $p \in (0, 1)$, can one produce a Bernoulli variable with success probability $1/2$ using only repeated flips?

The solution is well known, first we flip our biased coin twice in succession, then

- if the result is heads then tails, we give the output “Heads” or “1.”
- if the result is tails then heads, we give the output “Tails” or “0.”
- if the results of the two flips are the same then we begin again.

Since the two outputs each occur with probability $p(1 - p)$ they are equally likely. This yields the fair coin we are seeking.

The Bernoulli Factory problem was first formally posed by S. Asmussen [AGH92], albeit it in a less general form, in 1991. Asmussen asked if, given a coin of unknown bias $p \in (0, 1/2)$, one could produce a Bernoulli variable with success probability $2p$ using only repeated flips and auxiliary random variables with known distributions.

This additional access to auxiliary variables in Asmussen’s statement seems to make the “Von Neumann coin” extremely trivial. Whilst this is true, this statement is only added as a shortcut since via the “Von Neumann coin” we can produce our own auxiliary variables even with only our biased coin.

Asmussen’s problem turns out to be much harder than Von Neumann’s. Indeed it can be shown that such an algorithm cannot exist [KOB94].

Let us suppose that an algorithm to produce a $2p$ -coin by flipping a p -coin exists. There must be some number of flips m after which there is some chance of outputting “Tails.” The minimum probability is some constant, $\alpha \in (0, 1]$, multiplied by the probability of the least probable string. The least probable string is the string of all Heads or the string of all Tails depending on the value of p . We have constrained p to be less than $1/2$ so the least probable string is the string of all Heads. This yields

$$1 - f(p) = 1 - 2p \leq \alpha p^m, \quad (2.1)$$

since the probability of outputting “Tails” after m flips must be less than or equal to the total probability of outputting “Tails” at any point in the algorithm. This equation cannot be satisfied for any finite value of m , so the existence of a classical algorithm for a $2p$ -coin leads to a contradiction.

Table 2.1: Comparison between Turing Machines and Bernoulli Factories.

Turing Machine	Bernoulli Factory
Deterministic (not fundamentally probabilistic)	Fundamentally probabilistic
Outputs a single bit	Outputs a single random bit
Input bitstring is pre-determined and finite	Input bitstring is random and infinite in length
Quantum mechanics believed to improve efficiency	Quantum mechanics provably improves efficiency
Quantum mechanics cannot increase the class of computable functions	Quantum mechanics provably <i>does</i> increase the class of constructible functions.

2.1.1 Relation to Turing Machines

The similarities between a Bernoulli Factory and a Turing Machine are clear but there are important differences. In contrast to the finite automata discussed in Chapter 5 both are read/write machines, making them effectively unfettered by memory limitations. They can also both be comfortably thought of within Turing’s original “infinite tape writing” scenario, although in the case of the Bernoulli Factory the input string must also be infinite, whereas a Turing Machine’s input tape is restricted to begin with only a finite amount of symbols on it [K98].

Turing originally conceived of the machine as deterministic [T36] and while the definition has been widened to include random elements [S06] it is believed, according to the popular derandomization hypothesis [F09,FS08], that this addition grants no computational advantage. The Bernoulli Factory, on the other hand, is essentially random; it would make no sense to talk of a derandomized version.

2.1.2 Motivation

The Bernoulli Factory may seem somewhat abstract but it is deeply connected to a very fundamental statistical question: what do you do when you wish to simulate an event which has an unknown probability? This may seem an impossible question

but it is a frequent problem in statistics [L⁺11]. One can imagine many situations where the probability exists but only approximations are available with a finite amount of computational time (our equivalent here would be infinite coin flips in order to obtain the true value of p).

In the Bernoulli Factory, like in many statistical scenarios, the true probability is available but requires infinite computation or sampling. In other cases, we may only have estimators for p , or only a finite portion of an infinite series.

There is still ongoing research into real-life situations where the Bernoulli Factory is relevant, often in situations where approximate sampling would have unpredictable effects on simulations and so exact sampling becomes a necessity. Examples include diffusive and regenerative processes [BR05,BR08,B⁺06].

2.2 Classical Result

The classical result can be understood easily and intuitively from our earlier demonstration that a classical algorithm for a $2p$ -coin is impossible. If there exists a value p_0 for which $f(p_0) = 0$, then in order to ever output “Heads” we must be certain that our coin does not have bias p_0 . The only values of p_0 for which we can achieve this certainty are 0 and 1. We can understand the classical result for allowed functions, albeit simplistically, as “any continuous $f(p)$ which never takes the value 0 or 1 except possibly at $p = 0$ or $p = 1$.”

We can also understand the key equation 2.2 as clearly following from this intuitive description. If we ever want to output “Heads” or “Tails” in finite flips there must be a shortest string after which we will output a result. This string must occur with some finite probability. If the string of flips is of length k then this probability must always be greater than $p^k(1-p)^k$ since the probability will be of the form $p^{k_1}(1-p)^{k_2}$ where $k_1 + k_2 = k$.

2.2.1 Formal statement of the problem

The classical literature uses the terms “simulable” or “observable” for functions which can be outputted by a Bernoulli factory. Given the various other meanings this word can take in discussions of quantum mechanics, particularly quantum computing, we will adopt “constructible.”

Classical Problem: *A function $f : (0, 1) \rightarrow [0, 1]$ is constructible with the classical Bernoulli Factory if there exists a discernible event E such that $P_p(E) = f(p)$ for every $p \in (0, 1)$. Which functions are constructible?*

This leads to the formal result:

Classical Result: *A function $f : (0, 1) \rightarrow [0, 1]$ is constructible with the classical Bernoulli Factory if and only if f is continuous and for some k*

$$p^k(1-p)^k < f(p) < 1 - p^k(1-p)^k \quad (2.2)$$

2.2.2 Classical Proof

Almost all proofs within the sampling paradigm of this work follow a similar form. We show that we can create approximations for a class of functions and then show that the power to approximate grants us the power to sample exactly. This proof is taken from [KOB94].

The proof can be summarised as follows:

- We cannot sample $f(p)$ exactly, but we can approximate it.
- We can define $f(p)$ as a sum of functions which are easy to sample, as in equation 2.4.
- We can recursively obtain these functions which are easy to sample by first approximating $f(p)$, then taking the difference between the true function and the approximation, and then approximating that.

- In summary: from $f_i(p)$ we obtain $g_i(p)$, from the difference between these we obtain $f_{i+1}(p)$, repeat
- $f_0(p)$ is our original $f(p)$ function.

It is interesting to note that while the approximations used later for the quantum cases are complex and multi-faceted, for the classical case a fairly crude approach is sufficient. In fact use of the term “approximation” is debateable since as n increases the function actually deviates further from the function it is approximating.

The formula used is simply

$$g_{m,n}(p) = \sum_{k=0}^n H\left(f_m\left(\frac{k}{n}\right)\right) \binom{n}{k} p^k (1-p)^{n-k}. \quad (2.3)$$

Here H is the Heaviside Step function. We can most easily read this as flipping a coin n times, estimating p from these results and then outputting one or zero depending on whether $f(k/n)$ is greater than one half. Whilst this gives passable approximations for small numbers of flips one can easily see that as the number of flips increases this approach will converge to a function which is very different from $f(p)$ as shown in Figure 2.1. In the limit of large n this approximation will tend to a series of step functions.

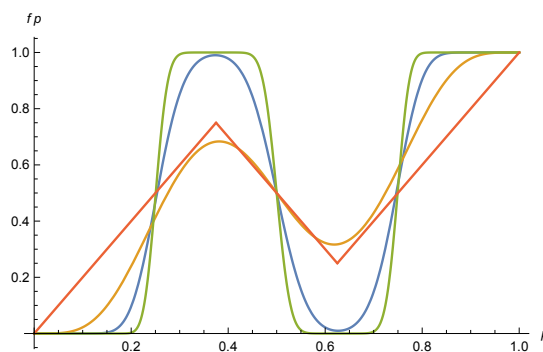


Figure 2.1: Approximations used in the classical Bernoulli Factory The result of the described approximation for 17 (orange), 101 (blue) and 501 (green) to a simple piecewise linear function (red). We note that it does not converge to our function for large n .

We want to be able to construct an event E which occurs with probability $f(p)$,

we begin by writing our target function as

$$f(p) = \sum_{a=0}^{\infty} \frac{1}{4} \left(\frac{3}{4}\right)^a g_{a,n_a}(p). \quad (2.4)$$

The values n_a are chosen such that $g_{a,n_a}(p)$ satisfies the criteria discussed below.

We also show that we can always choose such an n_a .

This will allow us to first produce a value of a variable A according to

$$p(A = a) = \frac{1}{4} \left(\frac{3}{4}\right)^a, \quad (2.5)$$

and then to construct an event which occurs with probability $g_a(p)$. If this event occurs then E occurs. Given how we have defined A and $g_a(p)$ this will ensure E occurs with the desired probability, but we must first prove that our infinite series of $g_a(p)$ exist and are constructible.

We define our initial approximation as

$$g_{0,n}(p) = \sum_{k \text{ s.t. } f(k/n) < 1/2} \binom{n}{k} p^k (1-p)^{n-k}, \quad (2.6)$$

which corresponds to the event where we flip n coins, estimate p , and find that $f(p)$ is great than or equal to one half.

We can rewrite our series as

$$f(p) = \frac{1}{4} g_{0,n_0}(p) + \sum_{a=2}^{\infty} \frac{1}{4} \left(\frac{3}{4}\right)^{a-1} g_{a,n_a}(p). \quad (2.7)$$

We can relabel the series on the right as $f_1(p)$, a new function related to the original by

$$f_1(p) = \frac{4}{3} (f(p) - \frac{1}{4} g_{0,n_0}(p)). \quad (2.8)$$

The problem of sampling $f(p)$ is now reduced to sampling $g_{0,n_0}(p)$ and $f_1(p)$. We can do this iteratively so long as we can show that the $f_i(p)$ will continue to be satisfy the conditions of our theorem. Since they will always be continuous as they

are defined as a combination of continuous functions we only need show that they satisfy

$$p^k(1-p)^k < f(p) < 1 - p^k(1-p)^k \quad (2.9)$$

for some k .

Now if $f(p) < \frac{5}{8}$ for a given value of p then

$$f_1(p) = \frac{4}{3} \left(f(p) - \frac{1}{4}g_{0,n}(p) \right) \quad (2.10)$$

$$< \frac{4}{3} \left(\frac{5}{8} - \frac{1}{4}g_{0,n}(p) \right) \quad (2.11)$$

$$< \frac{4}{3} \frac{5}{8} \quad (2.12)$$

$$= \frac{5}{6} \quad (2.13)$$

$$< 1 - p^3(1-p)^3. \quad (2.14)$$

While if $f(p) > \frac{5}{8}$ for a given p we have

$$f_1(p) = \frac{4}{3} \left(f(p) - \frac{1}{4}g_{0,n}(p) \right) \quad (2.15)$$

$$< \frac{4}{3} \left(f(p) - \frac{1}{4}f(p) \right); \quad (2.16)$$

we can always choose an n such that $g_{0,n}(p) > f(p)$ since $g_{0,n}(p) \rightarrow 1$ as $n \rightarrow \infty$.

This value of n is the n_a value described earlier. So

$$f_1(p) < \frac{4}{3} \left(f(p) - \frac{1}{4}f(p) \right) \quad (2.17)$$

$$< f(p) \quad (2.18)$$

$$< 1 - p^k(1-p)^k \quad (2.19)$$

Repeating these arguments around the value $\frac{3}{8}$ yields the second requirement:

$$f_1(p) > p^k(1-p)^k \quad (2.20)$$

We now see that beginning with a function $f(p)$, labelled as $f_0(p)$, which satisfies the conditions of the classical theorem we can iteratively define $g_i(p)$. As we define each one we will be left with a remainder $f_{i+1}(p)$ which also satisfies the conditions, allowing us to continue. This proof can be considered semi-constructive in that it does not yield the algorithm, which is infinite, but does yield a deterministic procedure by which one could obtain as much of the algorithm as one would require. Unfortunately, because the first step of the algorithm is generating a value of A , one does not know in advance how many steps one would require.

It should be understood that this proof demonstrates that a decomposition in terms of finite events can always be found. It does mean that all Bernoulli Factory algorithms are complex and opaque. We use the example of an algorithm of $f(p) = \sqrt{p}$ from [W99] to demonstrate this.

We recall that the Taylor expansion of $\sqrt{1+x}$ about $x = 1$ is

$$\sqrt{1+x} = \sum_{n=0}^{\infty} \frac{(-1)^n(2n)!}{(1-2n)(n!)^2(4^n)} x^n \quad (2.21)$$

$$\sqrt{p} = 1 - \sum_{n=0}^{\infty} \frac{2n!}{n!^2 2^{2n+1}} (1-p)^{n+1} \quad (2.22)$$

$$= 1 - \sum_{n=0}^{\infty} \frac{C_n}{2^{2n+1}} (1-p)n + 1. \quad (2.23)$$

Here C_n are the Catalan numbers [K08]. The Catalan numbers can be understood in several ways: as the number of ways of partitioning n elements without crossing; the number of ways of writing a well-formed statement with n sets of brackets; and finally the number of ways as the number of strings of n ones and

$n + 1$ zeros such that no proper initial segment contains more zeros than ones. This last one leads us to our algorithm.

If we flip a fair coin until the total number of tails exceeds the total number of heads the probability of this occurring after $2n + 1$ flips is $C_n/2^{2n+1}$. Recall that we can always produce a fair coin from our unfair p -coin. We then flip our p -coin $n + 1$ times and output a result of “one” unless we get no results of “Heads.” We can see that the probability of this event is that of $\sqrt{p}$ with reference to the decomposition above.

Chapter 3

The Quantum Bernoulli Factory

In this chapter we discuss the quantum version of the Bernoulli factory, its power and some of its implications for the additional power granted by quantum mechanics. We begin by describing the most logical extension of the Bernoulli factory and then prove a quantum analogue to the Keane-O’Brien theorem of the classical factory [KOB94]. Much of the work in this chapter is drawn from the paper *Provable quantum advantage in randomness processing* [DJR15].

3.1 Why make it quantum?

The Bernoulli factory has many properties that made it an appealing problem to approach from a quantum perspective. Sampling algorithms in general were already becoming an area of interest in quantum research, as were alternate or restricted models of computation [AA14,A⁺03,FU00].

This problem actually first came to light when Terry Rudolph was trying to patch a foundational model with probability amplification, something he assumed would be easy. Upon seeing the proof that it was impossible with classical bits, he decided to investigate the quantum case.

3.1.1 How should it be made quantum?

Whilst we believe the model for the quantum Bernoulli factory presented in this chapter to be the most natural one, it is certainly possible to argue for others. In this chapter we consider quantum inputs, quantum operations and classical outputs.

If either the inputs or the operations are made classical, then no advantage can be obtained over the wholly classical factory. If the outputs are quantum then the nature of the problem becomes very different (see discussion of q-sampling in chapter 6), and comparisons to the classical factory become very difficult. This leads us to explore this semi-quantum model first. What does a quantum input look like? We chose to define a quantum coin, informally “quoin,” as follows

$$|p\rangle = \sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle. \quad (3.1)$$

Allowing quantum operations might seem like it has no classical analogue but recall that in the problem as originally posed by Asmussen auxiliary random variables are an available resource, which is equivalent to being able to apply the stochastic matrix

$$S = \begin{pmatrix} P_{1,1} & P_{1,2} \\ P_{2,1} & P_{2,2} \end{pmatrix}, \quad (3.2)$$

where $P_{i,j}$ is the transition probability between states i and j . In addition allowing unitary operations is no different to allowing different measurements in the output step and if we were to restrict both we would reduce to the classical factory.

3.2 Formal statement of the problem

Our question is relatively simple, for what functions $f : (0, 1) \rightarrow (0, 1)$ does there exist a constructible event which occurs with probability $f(p)$? This simple question hides some complexity, as the probability measure we use in the quantum Bernoulli factory is much more complicated than that of the classical.

In our quantum factory our ability to apply different unitaries to different coins means defining some new things, many of these are extensions of things introduced in chapter two.

The total sample space is given by

$$\Omega_{tot} = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n \quad (3.3)$$

is the set of all ordered n -tuples of infinite $\{0, 1\}$ strings.

Subsets are then products of cylinder sets.

$$V_{tot} = V(\underline{v}_1) \times V(\underline{v}_2) \times \dots \times V(\underline{v}_n) \quad (3.4)$$

is the set of all ordered n -tuples of infinite $\{0, 1\}$ strings which begin with the finite strings $\underline{v}_1, \underline{v}_2, \dots, \underline{v}_n$ respectively. Again illustrating with an example, the set $V(1) \times V(01) \times V(111)$ is the set of all ordered 3-tuples of infinite $\{0, 1\}$ strings which begin with 1, 01 and 111 respectively.

We also have different probability distributions in the quantum cases, defined by a value p and finite set of unitary matrices. These unitaries describe the algorithmic freedom we have to process the unknown quantum state, and are determined by the target function $f(p)$. For each Ω_i we associate a unitary U_i of the form

$$U_i = H(a_i) = \begin{pmatrix} \sqrt{1-a_i} & \sqrt{a_i} \\ \sqrt{a_i} & -\sqrt{1-a_i} \end{pmatrix}, \quad (3.5)$$

which yields a coin that gives output one with probability

$$h_a(p) = \left(\sqrt{p(1-a)} - \sqrt{a(1-p)} \right)^2 \quad (3.6)$$

when measured in the computational basis. It is interesting to note that even this extremely simple operation leads to an output function which is not constructible classically according to the Keane-O'Brien Theorem.

We call the set of unitaries $G = \{U_i\}$ and so the quantum probability distribution is a function of both p and G .

$$P_{p,G} \left(\prod_{i=1}^{|G|} V(\underline{v}_i) \right) = \prod_{i=1}^{|G|} \prod_{j=1}^{|\underline{v}_i|} |\langle \underline{v}_{i,j} | U_i | p \rangle|^2. \quad (3.7)$$

We are interested in functions where we can construct sets and probability measures such that there is an event that occurs with probability $f(p)$ and is discernible.

It is also noteworthy that while the domain of the classical functions is usually $(0, 1)$ the domain of the quantum functions is usually $[0, 1]$. This is because at $p = 0$ the classical factory input is a coin that always yields Tails, which is deterministic and therefore of limited use. The quantum factory still receives a quantum state, which can still produce random outcomes.

Quantum Problem: *A function $f : [0, 1] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli Factory if there exists an event S and a probability measure $P_{p,G}$ such that S is discernible with respect to $P_{p,G}$ and $P_{p,G}(S) = f(p)$ for every $p \in (0, 1)$. Which functions are constructible?*

3.3 Quantum result

Theorem 1: A function $f : [0, 1] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli Factory if and only if it is Simple and Poly-Bounded.

For this proof we introduce the notions of Simple and Poly-Bounded functions (SPB functions) and bounding functions, which will be used throughout.

Definition 2: A function $f(p) : [0, 1] \rightarrow [0, 1]$ is simple and poly-bounded (SPB) iff it satisfies

1. f is continuous.
2. Both $Z = \{z_i : f(z_i) = 0\}$ and $W = \{w_i : f(w_i) = 1\}$ are finite sets.

3. $\forall z \in Z$ there exists constants $c, \delta > 0$ and integer $k < \infty$ such that

$$c(p - z)^{2k} \leq f(p) \quad \forall p \in [z - \delta, z + \delta].$$

4. $\forall w \in W$ there exists constants $c, \delta > 0$ and an integer $k < \infty$ such that

$$1 - c(p - w)^{2k} \geq f(p) \quad \forall p \in [w - \delta, w + \delta].$$

Definition 3: A function $L : [0, 1] \rightarrow [0, 1]$ is a lower bounding function for a function $f : [0, 1] \rightarrow [0, 1]$ if

$$L(p) \leq f(p) \text{ for } p \in [0, 1]. \quad (3.8)$$

A function $U : [0, 1] \rightarrow [0, 1]$ is an upper bounding function for a function $f : [0, 1] \rightarrow [0, 1]$ if

$$U(p) \geq f(p) \text{ for } p \in [0, 1]. \quad (3.9)$$

Proposition 1: For any SPB function $f(p)$ there exists a pair of bounding functions $L(p)$ and $U(p)$, which represent the probability of a finite event, such that

$$L(p) \leq f(p) \leq U(p)$$

where

$$\max_{p \in [0, 1]} (U(p) - L(p)) < \frac{1}{2}.$$

These two conditions ensure that

$$g(p) = \frac{L(p)}{1 - U(p) + L(p)} \quad (3.10)$$

will be a good approximation to $f(p)$, which will be an important step later on.

The proof can be summarised as follows:

- We show that we can always create upper and lower bounding functions which correspond to finite events for an SPB function
- We show that given bounding functions which correspond to finite events we can create a good approximation to the bounded function which corresponds to a discernible event
- We show we can decompose any SPB function into an infinite set of functions corresponding to discernible events
- We show given this decomposition we can create construct any SPB function with the quantum Bernoulli factory

We will outline an explicit construction of these bounding functions, which are key to Theorem 1.

Consider some candidates for $L(p), U(p)$ which do not quite work, namely $A_n(p), B_n(p)$ defined by

$$\begin{aligned} A_n(p) &= \sum_{k=0}^n \frac{2}{3} f\left(\frac{k}{n}\right) \binom{n}{k} p^k (1-p)^{n-k} \\ B_n(p) &= \sum_{k=0}^n \left(\frac{1}{3} + \frac{2}{3} f\left(\frac{k}{n}\right) \right) \binom{n}{k} p^k (1-p)^{n-k} \end{aligned}$$

These bounding functions have a minimum separation of one third rather than one half. This is because some of the modifications we will use will slightly increase this value, meaning we require an initial buffer.

From Bernstein's proof of Weierstrass' approximation theorem[RF88] we know that $A_n(p), B_n(p)$ converge *uniformly* in n to $\frac{2}{3}f(p), (\frac{1}{3} + \frac{2}{3}f(p))$, and so would be good bounding functions had we the freedom to take n to infinity. They are also manifestly constructible (within even a classical Bernoulli factory). However in the vicinity of the zeros and ones of $f(p)$ we have a problem, as easily illustrated with the example in Figure 3.1.

The strategy for dealing with this will be to multiply $A_n(p)$ by a generalized

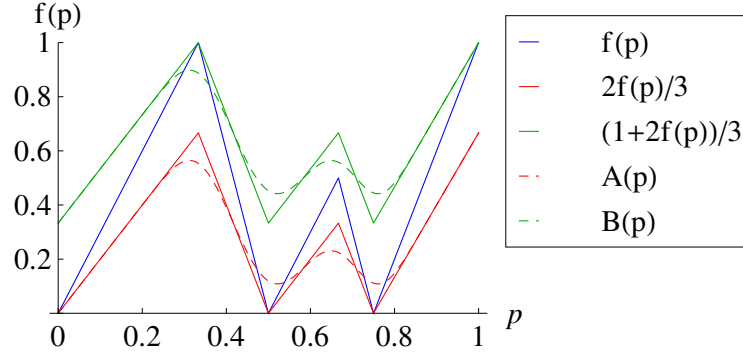


Figure 3.1: Unsuccessful Bounding Functions We attempt to bound the function from above and below. Here we show $A_{100}(p)$ and $B_{100}(p)$, and their counterparts for very large n , for the piecewise linear function depicted. Note that both fail to be good bounding functions when $f(p)$ gets close to zero or one.

Heaviside-type function (that must itself be q -constructible). We focus on $L(p)$ whenever it is obvious how to define the corresponding procedure for $U(p)$.

Defining

$$h_a(p) = \left(\sqrt{p(1-a)} - \sqrt{a(1-p)} \right)^2 \quad (3.11)$$

consider the function

$$T_{m,M,z}(p) := (1 - (1 - h_z(p))^m)^M$$

which is plotted here for $M = 3$, $z = 1/2$ and $m = 400$, in Figure 3.2:

Note that $T_{m,M,z}(z) = 0$ and that $T_{m,M,z}(p)$ is constructible with a finite number of quoins, since $h_z(p)$ is constructible with a single quoin upon which the unitary transformation

$$\begin{bmatrix} \sqrt{1-z} & \sqrt{z} \\ -\sqrt{z} & \sqrt{1-z} \end{bmatrix} \quad (3.12)$$

is performed prior to a computational basis measurement. Bernoulli's inequality[3]

$(1+x)^n \geq 1+nx$ can be used to show that

$$\begin{aligned} \lim_{m \rightarrow \infty} T_{m,M,z}(p) &= 0 \text{ if } h_z(p) < 1/M \\ &= 1 \text{ if } h_z(p) > 1/M \end{aligned}$$

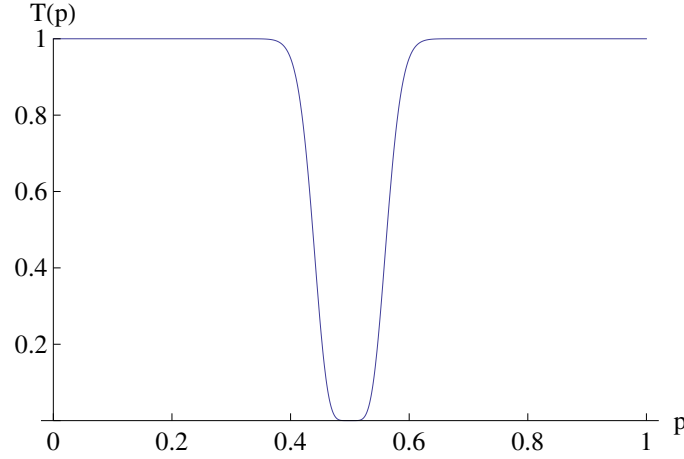


Figure 3.2: Our q -constructible “Heaviside” function Here we show an example of an important function for this task, which approaches zero rapidly close to the chosen point and which can have a controlled width. This function is $T_{400,3,0.5}$

We can use such functions to modify $A_n(p)$, to try and force it to always be a lower bound for $f(p)$. For the same example consider now the product $A_{100}(p)T_{400,5,1/2}(p)T_{400,1,3/4}(p)$:

There are a couple of issues we need to be careful about Fig 3.3:

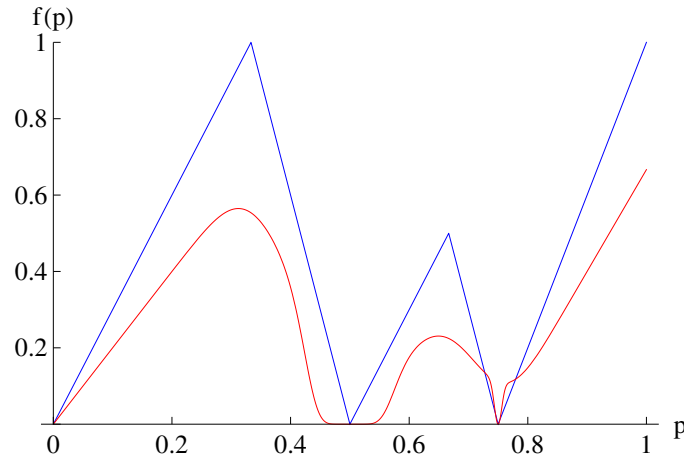


Figure 3.3: Further Difficulties with Bounding Functions The product function $A_{100}(p)T_{400,5,1/2}(p)T_{400,1,3/4}(p)$ is *nearly* a good lower bounding function for $f(p)$. It struggles at 0.5 from being pushed too flat and at 0.75 from not being flat enough.

Issue 1. When we “push down” $A_n(p)$ we increase its distance from $B_n(p)$, which itself is going to be “pushed up” at other (possibly nearby) locations. So we run the risk of increasing the distance between the lower- and upper-bounding

functions to more than $1/2$. You can see $A(p)$ being flattened too much around $p = 0.5$

Issue 2. We want the generalized Heaviside function to go towards zero over a finite width interval around z that is not too narrow - it must include all the problematic points at which $A_n(p)$ becomes larger than $f(p)$. You can see this happening around $p = 0.75$. When this happens we can either increase the interval width or alternatively increase n so that the points where $A_n(p)$ crosses $f(p)$ “slide” inwards.

Since $h_z(z) = 0$ and $h_z(p)$ is convex we readily see that, denoting by μ_i the two solutions to $h_z(p) = 1/M$, any large integer M defines a finite interval $I(M, z) := [\mu_1, \mu_2]$ that (strictly) contains the point z . The interval width decreases as M is increased, but is finite for all $M < \infty$. Inside/outside $I(M, z)$ standard theorems for continuous functions ensure that the convergence (in m) to 0 or 1 respectively is uniform.

A series expansion

$$T_{m,M,z}(p) \approx \frac{(M(p-z)^2)^m}{4z(1-z)} + O((p-z)^{4m})$$

around the point $p = z$ makes clear the reason for the conditions 3 and 4 in Definition 1 - for functions that rise exponentially slowly from any zero z we would not be able to choose parameters M and m such that we can push $A_n(p)$ below $f(p)$ in the vicinity of z .

We now have all ingredients in place to outline a procedure for constructing our lower-bounding function, which will take the form

$$L(p) = A_n(p) \prod_{i|z_i \in Z} T_{m_i, M_i, z_i}(p)$$

for some appropriate choice of parameters n, m_i, M_i . For each zero $z_i \in Z$ we first choose large integers M_i such that we fix non-overlapping intervals $I(M_i, z_i)$ which are smaller than those determined by the associated δ (see Definition 1) for that

z_i . We ensure they also do not overlap with the similar intervals $I(M_i, w_i)$ used in the construction of $U(p)$. However these intervals remain finite because the zeros are isolated so that as n is increased, uniform convergence ensures that all the problematic points where $A_n(p) \geq f(p)$ end up strictly contained within the $I(M_i, z_i)$, so $L(p) \leq f(p)$.

Finally, and critically, we also ensure that

$$\sup_{I(M_i, z_i)} \left(\frac{1}{3} + \frac{2}{3}f(p) \right) < \frac{1}{2} \quad \forall i$$

This is possible, because f is continuous and the interval contains a zero of f so for sufficiently large n there will be an interval containing z for which $U(p) < 1/2$. The importance of this condition is that now, even as our lower bounding function is taken close to 0 within the intervals $I(M_i, z_i)$, our upper-bounding function (which converges uniformly to $(\frac{1}{3} + \frac{2}{3}f(p))$ within $I(M_i, z_i)$) can be brought within a distance $1/2$ from the lower bounding function.

In summary, we now have that $L(p)$ converges uniformly on $[0, 1] - \cup_i I(M_i, z_i)$ to $\frac{1}{3}f(p)$ and on $\cup_i I(M_i, z_i)$ to 0 as the remaining free parameters n, m_i are increased. Similarly we construct

$$U(p) = B_n(p) \prod_{i|w_i \in W} T_{m_i, M_i, w_i}(p) + 1 - \prod_{i|w_i \in W} T_{m_i, M_i, w_i}(p)$$

which converges uniformly on $[0, 1] - \cup_i I(M_i, w_i)$ to $\frac{1}{3} + \frac{2}{3}f(p)$ and on $\cup_i I(M_i, w_i)$ to 1. The conditions of Proposition 1 can clearly be met with some suitable large (but finite) choice of the remaining free parameters.

We are now able to prove Theorem 1.

The proof of this theorem essentially follows that of Keane and O'Brien[1].

Proof. Let $L_k(p)$ and $U_k(p)$ denote lower and upper-bounding functions satisfying

Proposition 1, for the sequence of SPB functions $f_k(p)$, where

$$\begin{aligned} f_1(p) &= f(p) \\ g_k(p) &= \frac{L_k(p)}{1 - U_k(p) + L_k(p)} \\ f_{k+1}(p) &= \frac{4}{3} \left(f_k(p) - \frac{1}{4} g_k(p) \right) \end{aligned}$$

Since $U(p)$ and $L(p)$ are finite events there exists some N after which we can say with certainty whether they occur or not. If neither occur, the output is heads; if both occur, the output is tails; if only $L(p)$ occurs, simulate the functions again.

This procedure produces an output with probability $(1 - U(p) + L(p))$ on each trial and so the probability of this outputting heads is

$$g_i(p) = \frac{L(p)}{1 - U(p) + L(p)} \quad (3.13)$$

$$f_{i+1}(p) = \frac{4}{3} \left(f_i(p) - \frac{1}{4} \cdot \frac{L(p)}{1 - U(p) + L(p)} \right) \quad (3.14)$$

$$> \frac{4}{3} \left(f_i(p) - \frac{1}{2} \cdot L(p) \right) \quad (3.15)$$

$$> \frac{4}{3} \left(f_i(p) - \frac{f_a(p)}{2} \right) \quad (3.16)$$

$$> \frac{2}{3} f_i(p) \quad (3.17)$$

Similarly

$$f_{i+1}(p) = \frac{4}{3} \left(1 - (1 - f_i(p)) - \frac{1}{4} + \frac{1}{4} \cdot \frac{1 - U(p)}{1 - U(p) + L(p)} \right) \quad (3.18)$$

$$1 - f_{i+1}(p) = \frac{4}{3} \left((1 - f_i(p)) - \frac{1}{4} \cdot \frac{1 - U(p)}{1 - U(p) + L(p)} \right) \quad (3.19)$$

$$< \frac{2}{3} (1 - f_i(p)) \quad (3.20)$$

So if $f_i(p)$ is SPB then so is $f_{i+1}(p)$.

Since

$$\lim_{N \rightarrow \infty} \left[f(p) - \sum_{k=0}^N \left(\frac{3}{4} \right)^{k-1} \frac{1}{4} g_k(p) \right] = \lim_{N \rightarrow \infty} \left(\frac{3}{4} \right)^{N-1} f_k(p) = 0$$

we have that $f(p)$ can be convexly decomposed

$$f = \sum_{k=0}^{\infty} \left(\frac{3}{4} \right)^{k-1} \frac{1}{4} g_k(p).$$

Using ancillary randomness to sample an index k with probability $\left(\frac{3}{4} \right)^{k-1} \frac{1}{4}$ we then construct a $g_k(p)$ coin and we have shown that any SPB function can be constructed.

To show that any function constructible with the quantum Bernoulli factory must be SPB is far simpler.

With the exception of the trivial functions $f(p) = 0, 1$, if a function is constructible then there is some number of flips k such that after k flips the probability of outputting 0 and the probability of outputting one are both greater than zero.

The probability of outputting zero must be less than $f(p)$ but be at least some constant multiplied by the least probable string which in turn must be greater than the expression below.

$$P(\text{output zero} \mid k \text{ flips}) = \epsilon \min_{s \in \{0,1\}^k} \prod_{i=1}^n |\langle s_i | U_i | p \rangle|^{2k} \quad (3.21)$$

$$\geq \epsilon \prod_{i=1}^n (|\langle 0 | U_i | p \rangle \langle 1 | U_i | p \rangle|)^{2k} \quad (3.22)$$

$$\geq \epsilon \prod_i^n (p - a_i)^k \quad (3.23)$$

In other words, because we can only use a finite number of unitaries in a finite number of flips, our function must be bounded from below by a finite polynomial.

This yields condition three of Definition 3 and half of condition two. Repeating the process with output one yields condition four and the other half of condition two. Finally we recall that a sum of continuous functions is a continuous function

and since the probability measure on open sets give continuous functions and the event is made up of such sets the probability of the event must be a continuous function. Thus we have shown any function constructible with quoin and a finite number of unitaries must be SPB and we have completed the proof of Theorem 1.

□

3.4 Discussion

3.4.1 On quantum advantage

What conclusions can we draw from this greatly increased class of functions? What is the key to quantum advantage in this scenario. As before, our most intuitive and instructive example will be $f(p) = 2p$. The behaviour of this function at $p = 0.5$ is what causes the classical factory to fail, but its behaviour at $p = 0$ is almost identical and does not cause problems. In many cases, behaviour that causes failure at any other point is handled easily by the classical factory at either endpoint. From this we conclude that the points $p = 0$ and $p = 1$ are in some sense privileged points, which are qualitatively different from other points. The advantage of the quantum factory seems to stem from its power, via unitary operations, to create a finite number of additional privileged points within the functions domain. This privilege comes from the fact that the classical factory can obtain certain, as opposed to probabilistic, information about these points. A result of Heads proves that $p \neq 0$ but there is no classical result that will prove $p \neq 0.5$, in the quantum case there is. The nature of quantum advantage will be probed further in Chapter 4 and the extent to which this advantage is physically realisable will be investigated thoroughly in Chapter 5.

3.4.2 On practicality

We note that while the proof is semi-constructive in that it can always produce as much of the algorithm as one might require, it is not practical. The algorithms in general take infinite expected time, although this is often caused by extreme outliers. The algorithm can also not be easily described as one can only give the procedure to generate it, making computer simulation and experimental realisation difficult. In Chapter 4 we will discuss classes of functions which do have less unwieldy algorithms and in Chapter 5 we will focus on more practical implementations. In addition it can be easily shown that in the case of imperfect equipment, quantum advantage in the class of constructible functions is lost. We will show in Chapter 5 that a large advantage in resources and efficiency can be maintained in the case of errors. We will also show that a difference in the class of constructible functions remains in the case of errors or mixed states when we investigate finite events rather than discernible events.

Chapter 4

Variants on the Quantum Bernoulli Factory

In this chapter we discuss different variants on the Bernoulli factory and how these change when quantum mechanics is introduced. In particular we focus on joint measurement, which does not add anything to the original quantum Bernoulli factory. Here we have several examples of where joint measurement does provide advantage. The first result was discussed in the supplementary material of *Provable quantum advantage in randomness processing* [DJR15]; the rest of this chapter is unpublished work.

4.1 Changing the domain

In the classical case functions which are not simulable are usually so because at a single point they take the value zero or one. In light of this we explored what happened if the domain were changed and this single point removed. As an example the function $f(p) = 2p$ is clearly not simulable on $(0,1)$ but is simulable on $(0,0.1)$. The most interesting question arises when we consider $(0,0.5)$, here the problematic point, $p = 0.5$, itself is excluded but points arbitrarily close to it are included. In the classical case being able to approach a problematic point means that the function will still not be simulable, but in the quantum case it is not so clear cut.

4.1.1 Wastlund approach

When researching the classical Bernoulli factory we came across an alternative proof of the Keane-O'Brien result [KOB94]. We attempted to use both methodologies for our quantum proof but found that each led to a different result. Where Keane-O'Brien approaches the whole function as one entity, the Wastlund proof [W99] focuses first on removing difficult regions. In the classical case these are the end regions but in the quantum case these can be right in the middle of the function. Wastlund's method led us to consider functions such as:

$$f_{\wedge}(p) := \begin{cases} 2p & ; \ p \in (0, 1/2) \\ 2(1-p) & ; \ p \in (1/2, 1) \end{cases}, \quad (4.1)$$

and

$$f_{step} := \begin{cases} \frac{1}{4} & ; \ p \in (0, 1/2) \\ \frac{3}{4} & ; \ p \in (1/2, 1) \end{cases}. \quad (4.2)$$

We found that both these functions were simulable within the quantum Bernoulli factory. Note that the function f_{step} is continuous since 0.5 is not within its domain.

4.1.2 Formal statement of the problem

We note that the problem here can be stated in the same way, only the domain has changed. This could also be viewed as the quantum Bernoulli factory with the addition of a “promise” that p will not take certain values.

Quantum Problem on reduced domain: *A function $f : (0, a_1) \cup (a_1, a_2) \cup \dots \cup (a_n, 1) \rightarrow [0, 1]$ is constructible with the quantum Bernoulli Factory if there exists an event S and a probability measure $P_{p,G}$ such that S is discernible with respect to $P_{p,G}$ and $P_{p,G}(S) = f(p)$ for every $p \in (0, 1)$. Which functions are constructible?*

4.1.3 Result on reduced domain

Theorem 2: A function $f : (0, a_1) \cup (a_1, a_2) \cup \dots \cup (a_n, 1) \rightarrow [0, 1]$ is constructible with the quantum Bernoulli Factory if and only if f is continuous on its domain and there exists a finite list $\{a_1, a_2, \dots, a_{n'}\}$, which contains $\{a_1, a_2, \dots, a_n\}$, and integer k such that

$$a^k(p) \leq f(p) \leq 1 - a^k(p) \quad (4.3)$$

for all $p \in (0, 1)$, where

$$a(p) := p(1 - p) \prod_{1 \leq i \leq n'} h_{a_i}(p)(1 - h_{a_i}(p)) \quad (4.4)$$

This second theorem gives stranger functions than the first. At the set of excluded points $\{a_i\}$ the function can undergo a jump or even have no definite value. Were the functions extended over the whole interval $(0, 1)$ we would find we only require that the functions be piecewise continuous. The list $\{a_1, a_2, \dots, a_{n'}\}$ contains all the points at which $f(p)$ is not defined and any points within the domain where $f(p)$ takes the value zero or one. We prove Theorem 2 by showing that we can pinpoint p to within an interval with high confidence and then approximate $f(p)$; by averaging over many such approximations we can, in fact, achieve an exact sample. There are obvious parallels with promise problems. The unexpected results of this theorem are shown in Figure 4.1.

The proof can be summarised as follows:

- It is shown in proposition 2 that one can “guess” within which of a set of intervals p lies with a small error that depends on p .
- It is shown in proposition 3 that we can choose our intervals such that there is a value q_i which differs from the possible values of $f(p)$ in the interval by some small error that depends on p .
- It is shown in proposition 3 that these two results lead to an approximation

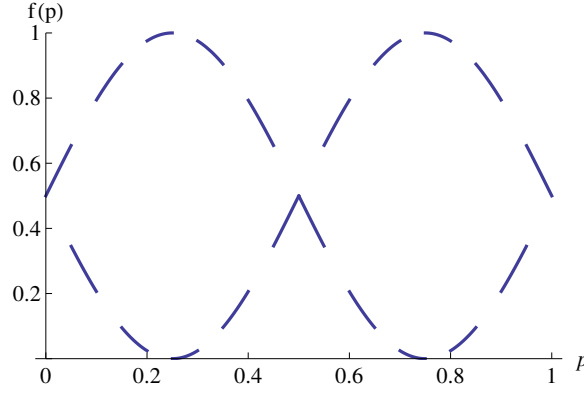


Figure 4.1 One example of the unusual functions one could construct using Theorem 2 The actual discontinuities are excluded. Effectively we gain access to piecewise continuous functions but with their discontinuous points excluded, leaving a function which is continuous on its domain.

of $f(p)$ which differs from the true function by a small error that depends on p

- As with Theorem 1, it is shown that if we can create good approximations we can create an exact sampling algorithm.

Proposition 2: If $(0, a_1) \cup (a_1, a_2) \cup \dots \cup (a_n, 1)$ is covered by countably many open subintervals W_i , $i \in \mathbf{N}$, and k is a positive integer, then there is an discernible function $\alpha : \Omega \rightarrow \mathbf{N}$ such that for every $p \in (0, 1)$,

$$P_p(\alpha^{-1}\{i : p \notin W_i\}) < a^k(p) \quad (4.5)$$

Proposition 2 states that given a set of intervals covering the domain of the function we can pinpoint p to within one of them after a finite number of tosses with the probability that our guess is incorrect being less than $a^k(p)$. This is an essential step in our proof.

Proof

In order to guess which interval we are in we must narrow down a potentially infinite number of intervals to a finite number. To this end we wish to restrict ourselves to a union of closed intervals so that we can invoke compactness.

For each $a \in \{a_1, a_2, \dots, a_n\}$ we flip $h_{a_i}(p)$ -coins until we have observed $nk + 2k + 1$ tails. In addition we flip p -coins until we have observed $nk + 2k + 1$ tails and $nk + 2k + 1$ heads. For each case we call the number of flips before we see the required results m_{a_i} , m_0 for p -coins, and we call the maximum value m_{max} .

Now for some a_i the probability of the required results occurring after m_{a_i} is.

$$P(\geq (nk + 2k + 1) \text{ tails in } m_{a_i} \text{ flips}) < 2^{m_{a_i}} (1 - h_{a_i}(p))^{nk+2k+1} \quad (4.6)$$

We can use this to rule out a region around a_i of size 2ϵ as follows.

$$P(\geq (nk + 2k + 1) \text{ tails} \mid m_{a_i} \& p \in \{a_i - \epsilon, a_i + \epsilon\}) < 2^{m_{a_i}} (1 - h_{a_i}(p))^{2nk+2k+1} \quad (4.7)$$

$$< (1 - h_{a_i}(p))^{2nk+2k} \quad (4.8)$$

$$< \epsilon^{2nk+2k} \quad (4.9)$$

$$< a^k(p) \quad (4.10)$$

The same procedure can also show the probability that $p \in (0, \epsilon]$ or $p \in [1 - \epsilon, 1)$ is also below $a^k(p)$. This means we can say that $p \in [\epsilon, a_1 - \epsilon] \cup [a_1 + \epsilon, a_2 - \epsilon] \cup \dots \cup [a_n + \epsilon, 1 - \epsilon]$ with probability greater than $1 - a^k(p)$. It is possible for some of these regions to be large enough that they merge; this is not a problem.

Now that we are restricted to a union of closed intervals we can, by compactness, choose a finite number of W_i 's which cover this set. We can also find $\delta > 0$ such that $a^k(p) \geq \delta$ throughout this region. From the finite open cover we can choose a closed interval $F_i \subset W_i$ for each i such that the $\{F_i\}$ cover our union of closed intervals. Now we estimate p by flipping classical coins and calculating the frequency of heads, our estimated value is

$$f_N = \frac{X_1 + \dots + X_N}{N}. \quad (4.11)$$

We do this with N flips, N being chosen such that

$$P(f_N \in F_i | p \notin W_i) < \delta \quad (4.12)$$

We now define our discernible function α as $\alpha(f_N \in F_i) = i$. Recall that the $\{F_i\}$ are conditioned on m_{max} so the function is not as simple as it appears.

We have now shown that such a discernible function exists.

□

Proposition 3: If f is any continuous function $(0, a_1) \cup (a_1, a_2) \dots \cup (a_n, 1) \rightarrow (0, 1)$, and l is any positive integer, then there is a function g representing the probability of an discernible event such that for every p in $(0, a_1) \cup (a_1, a_2) \dots \cup (a_n, 1)$,

$$|f(p) - g(p)| < a^l(p) \quad (4.13)$$

Proposition 3 uses Proposition 2 to show that we can find constructible functions which are good approximations to $f(p)$ which allows us to complete our proof of Theorem 2 in a manner similar to that for Theorem 1; we sample a randomly from a set of approximations to $f(p)$ in such a way that on average we sample $f(p)$.

Proof

Since f is continuous, we can choose countably many subintervals $W_1, W_2, W_3, \dots$ covering $(0, a_1) \cup (a_1, a_2) \dots \cup (a_n, 1)$, and numbers $q_1, q_2, q_3, \dots$ so that for every $p \in W_i$

$$|f(p) - q_i| < a^{l+1}(p). \quad (4.14)$$

We have shown in Proposition 2 that we can guess the interval with some small error probability. The above statements show that we can choose our intervals such that they have the q values we need. The idea is that we guess our interval, W_i , and that we have chosen our intervals such that $f(p)$ can be approximated as q_i within them.

In the event that f takes the value zero or one at a point within its domain then we assign a function $q_i(p)$ to an interval containing that point such that for every

$p \in W_i$

$$|f(p) - q_i(p)| < a^{l+1}(p) \quad (4.15)$$

using the techniques from Theorem 1. For ease of notation we will continue the proof assuming this is not necessary

By Proposition 2, there is an discernible function $\alpha : \Omega \rightarrow \mathbf{N}$ such that for every p ,

$$P_p(\alpha^{-1}\{n : p \notin W_n\}) < a^{l+1}(p) \quad (4.16)$$

Once α is known, which we know will be in finite time, we then use an event which occurs with probability q_α .

Thus the total event has probability

$$g(p) = \sum_{i=1}^{\infty} P_p(\alpha^{-1}(i))q_i \quad (4.17)$$

Hence for every p , the error $|f(p) - g(p)|$ is bounded by

$$P_p(p \notin U_\alpha) + \sup\{|f(p) - q_i| : p \in W_i\} = 2a^{l+1}(p), \quad (4.18)$$

the chance of guessing the wrong interval plus the difference between q_i and $f(p)$.

Recalling that

$$a(p) := p(1-p) \prod_{1 \leq i \leq n'} h_{a_i}(p)(1-h_{a_i}(p)) \leq p(1-p) \leq \frac{1}{2}, \quad (4.19)$$

and therefore

$$P_p(p \notin W_\alpha) + \sup\{|f(p) - q_i| : p \in U_i\} = 2a^{l+1}(p) < a^l(p). \quad (4.20)$$

So we have shown the total error is less than the error we are allowed. $\square$

We can now complete the proof Theorem 2 by showing that this ability to

create approximations allows us to create an exact algorithm made from successive approximations.

Proof

Let $f : (0, a_1) \cup (a_1, a_2) \cup \dots \cup (a_n, 1) \rightarrow [0, 1]$ be any continuous function satisfying

$$a^k(p) < f(p) < 1 - a^k(p) \quad (4.21)$$

for some k . If we apply Proposition 3 to f with $l = k + 1$ we find that there exists a function f_1 , representing the probability of an discernible event, such that, for every p ,

$$|f_1(p) - f(p)| < a^{k+1}(p). \quad (4.22)$$

We then have

$$f_1(p) > f(p) - a^{k+1}(p). \quad (4.23)$$

Since $f(p) < 1 - a^k(p)$, we see that

$$f_1(p) > 2f(p) - a^{k+1}(p) - 1 + a^k(p) > 2f(p) - 1 + a^{k+1}(p). \quad (4.24)$$

Similarly,

$$f_1(p) < f(p) + a^{k+1}(p), \quad (4.25)$$

Since $f(p) > a^k(p)$, we see that

$$f_1(p) < 2f(p) + a^{k+1}(p) - a^k(p) < 2f(p) - a^{k+1}(p). \quad (4.26)$$

Combining these gives

$$2f(p) - a^{k+1}(p) < f(p) < 2f(p) - 1 + a^{k+1}(p) \quad (4.27)$$

which can be rearranged as

$$a^{k+1}(p) < 2f(p) - f_1(p) < 1 - a^{k+1}(p). \quad (4.28)$$

If we repeat this argument, replacing $f(p)$ by $2f(p) - f_1(p)$ and k by $k + 1$, we find that there exists a function f_2 which is the probability of an discernible event.

$$a^{k+2}(p) < 4f(p) - 2f_1(p) - f_2(p) < 1 - a^{k+2}(p) \quad (4.29)$$

Continuing in this way we can find functions $f_1, f_2, f_3, \dots$ each representing the probability of an discernible event, such that for every n ,

$$a^{k+n}(p) < 2^n f(p) - 2^{n-1} f_1(p) - \dots - f_n(p) < 1 - a^{k+n}(p) \quad (4.30)$$

If we replace the lower and upper bounds by 0 and 1 and divide by 2^n we get

$$0 < f(p) - \frac{1}{2} f_1(p) - \dots - \frac{1}{2^n} f_n(p) < \frac{1}{2^n} \quad (4.31)$$

Letting $n \rightarrow \infty$, we obtain

$$f(p) = \sum_{n=1}^{\infty} \frac{1}{2^n} f_n(p) \quad (4.32)$$

So if we choose a value n with probability $p_n = \frac{1}{2^n}$ and then sample $f_n(p)$ we have sampled $f(p)$ exactly.

To show that any function constructible with the quantum Bernoulli factory must obey the conditions of Theorem 2 is far simpler.

With the exception of the trivial functions $f(p) = 0, 1$, if a function is constructible then there is some number of flips k such that after k flips the probability of outputting 0 and the probability of outputting 1 are both greater than zero.

The probability of outputting 1 must be less than $f(p)$ but be at least some constant multiplied by the least probable string which in turn must be greater than

the expression below.

$$P(\text{output 1} \mid k \text{ flips}) > \epsilon \prod_{i=1}^n (|\langle 0|U_i|p\rangle \langle 1|U_i|p\rangle|)^{4k} > a^k(p) \quad (4.33)$$

The least probable string after k flips occurs with probability $\min_{a_i} (1 - h_{a_i}(p))^k$

$$\min_{a_i} (1 - h_{a_i}(p))^k > \left(p(1-p) \prod_{1 \leq i \leq n'} h_{a_i}(p)(1 - h_{a_i}(p)) \right)^k = a^k(p) \quad (4.34)$$

Repeating the process with output zero and combining the two yields.

$$a^k(p) \leq f(p) \leq 1 - a^k(p), \quad (4.35)$$

as required.

Finally we recall that a sum of continuous functions is a continuous function and since the probability measure on open sets give continuous functions and the event is made up of such sets the probability of the event must be a continuous function. It should be emphasised that although the extensions of many of these functions are piecewise continuous the functions themselves are always continuous on their domain. This completes the proof of Theorem 2. $\square$

4.1.4 Discussion

This result emphasizes the importance of careful thinking around the Bernoulli factory. In particular the result of the previous chapter certainly leads one to think in terms of *points* which cause problems for the algorithms. In this result we have removed such points from the domain and still the quantum advantage remains. Not only are these points problematic but also the region around them, so long as one can get arbitrarily close to the point it can cause problems. In particular we note that our wedge function is classically impossible on $(0, 1)$ or $(0, 0.5) \cup (0.5, 1)$ but possible on $(0, 0.5 - \epsilon) \cup (0.5 + \epsilon, 1)$ for any finite ϵ .

4.2 Multiple Parameters

The Bernoulli factory does not explore the full space of its inputs by a colossal margin. It takes an input of infinite identical coins but this could be expanded in a multitude of ways such as:

- Two or more types of coin, in a predictable pattern
- Two or more types of coin, in a random pattern
- A series of coins, such as $\{p, p/2, p/4, p/8 \dots\}$
- Coins that use the whole statespace by adding a phase component, $\sqrt{p}|0\rangle + e^{i\theta}|1\rangle$.

We have explored some of the most interesting of these and present the results below.

4.2.1 Two different coins

This problem is a natural extension of the question: "Given two coins of unknown bias, can you determine with certainty whether they have different biases?" [C14,BJV04]

- Classically one cannot, since any two coins could produce any set of outcomes.
- In the Quantum Bernoulli factory one can do this if promised the coins are restricted to a finite set of biases such as $\{0.25, 0.5, 0.75\}$ but not if they are unrestricted on the whole interval
- In contrast to our previous findings however, we find that a joint measurement will allow this determination when it would be otherwise impossible.

In light of these we decided to characterise the set of constructible functions without joint measurements to understand why this difference occurs. We are interested in the scenario where we have access to an infinite supply of two quantum

coins, one with bias $p \in [0, 1]$ and one with bias $q \in [0, 1]$. Note that the coins are separately available, we do not receive a random mixture of the two coins. We are then asked to produce a single bit output with bias $f(p, q)$.

4.2.1.1 Formal statement of the problem

Within the two-coin paradigm our definitions for probability measures must change slightly but most other facets of the proof remain the same.

As before we have a sample space

$$\Omega_{tot} = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n \quad (4.36)$$

to describe our ability to treat coins differently. In this case however the difference is not just the unitary we apply but also the bias of the coin, p or q .

In order to handle this we need to introduce sets b_p and b_q which partition our strings according to their bias. With these sets we can now adapt our original probability measure definition.

$$P_{p,G} \left(\prod_{i=1}^{|G|} V(\underline{v}_i) \right) = \prod_{i \in b_p} \prod_{j=1}^{|\underline{v}_i|} |\langle \underline{v}_{i,j} | U_i | p \rangle|^2 \cdot \prod_{i \in b_q} \prod_{j=1}^{|\underline{v}_i|} |\langle \underline{v}_{i,j} | U_i | q \rangle|^2 \quad (4.37)$$

Quantum Problem with two coins: *A function $f : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli factory iff there exists an event S and a probability measure $P_{p,G}$ such that S is discernible with respect to $P_{p,G}$ and $P_{p,G}(S) = f(p, q)$. Which functions are constructible?*

4.2.1.2 Result for two different coins

Theorem 5: A function $f : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli factory iff it satisfies

1. f is continuous.

2. There exist finite sets, $P = \{p_i\}$ and $Q = \{q_i\}$, where $f(p_i, q) = 0 \quad \forall q$ and $f(p, q_i) = 0 \quad \forall p$, which may be empty.

3. $\forall p_i \in P$ there exists constants $c, \delta > 0$ and integer $k < \infty$ such that

$$c(p - p_i)^{2k} \leq f(p, q) \quad \forall p \in [p_i - \delta, p_i + \delta].$$

4. $\forall q_i \in Q$ there exists constants $c, \delta > 0$ and integer $k < \infty$ such that

$$c(q - q_i)^{2k} \leq f(p, q) \quad \forall q \in [q_i - \delta, q_i + \delta].$$

5. There exists similar sets for when $f(p) = 1$.

6. There exists a set, X , of pairs of values $\{p_j, q_j\}$ such that $f(p_j, q_j) = 0$.

7. $\forall \{p_j, q_j\} \in X$ there exists constants $c, \delta > 0$ and integer $k < \infty$ such that

$$c(p - p_i)^{2k} (q - q_i)^{2k} \leq f(p, q) \quad \forall \{p, q\} \in [p_j - \delta, p_j + \delta][q_j - \delta, q_j + \delta].$$

8. There exists a similar set for when $f(p) = 1$.

These criteria look very similar to our SPB definition from earlier, the only reason we do not label this a two-dimensional extension is that we arrive at another, different, two-dimensional extension in the next subsection.

It may not be immediately apparent what kind of functions this includes so the reader is directed to FIG 4.2

Following similar logic to Theorem 1, we note that if a function is constructible then there is necessarily some number of flips k such that after k flips the probability of outputting zero and the probability of outputting one are both greater than zero. The probability of outputting zero after k flips can be no less than a constant multiplied by the least probable string, since there must be at least one string that outputs zero with some non-zero probability.

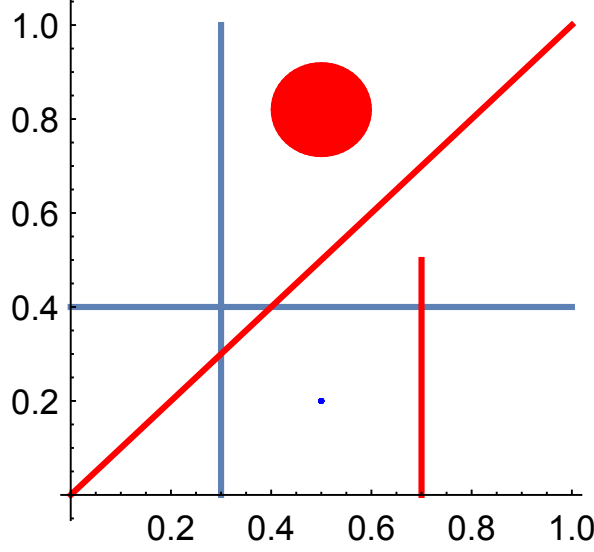


Figure 4.2: Permitted regions for the quantum two coin problem In addition to points, horizontal and vertical lines (shown in blue) where $f(p, q) = 0$ are permitted. Regions, halflines and diagonal lines (shown in red) are not permitted.

$$P(\text{output zero} \mid k \text{ flips}) \geq \epsilon \min_{s \in \{0,1\}^k} \prod_{i \in b_p} (|\langle s_j | U_i | p \rangle|)^{2k} \prod_{i \in b_q} (|\langle s_j | U_i | q \rangle|)^{2k} \quad (4.38)$$

$$\geq \epsilon \prod_{i \in b_p} (|\langle 0 | U_i | p \rangle \langle 1 | U_i | p \rangle|)^{2k} \prod_{i \in b_q} (|\langle 0 | U_i | q \rangle \langle 1 | U_i | q \rangle|)^{2k} \quad (4.39)$$

$$\geq \epsilon \prod_{i \in b_p} (p - p_i)^k \prod_{i \in b_q} (q - q_i)^k \quad (4.40)$$

These sets of p_i and q_i contain, necessarily, the contents of P , Q and X . This shows that we can have values of p , values of q and pairs of values $\{p, q\}$ where our function takes the values zero. More intuitively, our function can go to zero on horizontal lines, vertical lines and points. It is not immediately apparent that halflines are not permitted, so we go into further detail via Proposition 4.

Proposition 4 If $f(p, q)$ is constructible with the two coin quantum Bernoulli factory, then $f(p, q_0)$ must be constructible with the one coin quantum Bernoulli factory for any known value of q_0 .

The proof for this is simple. We know from the classical result that we can always construct an event of constant probability. This means we can construct

events which match any measurement on a q_0 -coin, since q_0 is known. All we then do is simulate the portion of the two-coin algorithm pertaining to q -coins whilst carrying out the p portion normally. This produces an algorithm for $f(p, q_0)$ which can be carried out by the quantum Bernoulli factory. $\square$

Since we know fully the class of functions which can be simulated by the quantum Bernoulli factory this immediately tells us that halflines are impossible since functions of that form cannot be constructed in the one coin factory. Knowing now where values of zero and one are permitted we can easily show, using the methods of Theorem 1, how such a function would be constructed. Again we begin with two bounding functions which almost work

$$\begin{aligned} A_n(p, q) &= \sum_{k=0, l=0}^{n, m} \frac{2}{3} f\left(\frac{k}{n}, \frac{l}{m}\right) \binom{n}{k} p^k (1-p)^{n-k} \binom{m}{l} q^l (1-q)^{m-l} \\ B_n(p, q) &= \sum_{k=0}^n \left(\frac{1}{3} + \frac{2}{3} f\left(\frac{k}{n}, \frac{l}{m}\right) \right) \binom{n}{k} p^k (1-p)^{n-k} \binom{m}{l} q^l (1-q)^{m-l} \end{aligned}$$

We recall how these functions failed around values of zero or one and how we were able to modify them by use of a Heaviside-type function

$$T_{m, M, z}(p) := (1 - (1 - h_z(p))^m)^M$$

We can use this function for our full-lines but not for our single points. Since these points have both a p - and q -coordinate we must make a new version of this function to handle them.

$$O_{m, M, p_0, m, M, q_0}(p, q) := 1 - (1 - T_{m, M, p_0}(p))(1 - T_{m, M, q_0}(q))$$

This now gives us the functions we need to construct our lower bounding function (the upper is constructed in much the same way):

$$L(p, q) = A_n(p, q) \prod_P T_{m,M,p_i}(p) \prod_Q T_{m,M,q_i}(q) \prod_X O_{m,M,p_j,m,M,q_j}(p, q) \quad (4.41)$$

Once again we have shown we can construct upper and lower bounding functions for our target function. Thus following Proposition 3 from our first result which tells us that if we can approximate a function we can construct it, we have proved that we can construct this class of functions. $\square$

We note also an interesting difference between this problem and those considered previously. In the single coin problem joint measurement adds efficiency but does not affect the class of constructible functions. In the two coin problem the addition of joint measurement increases the class of constructible functions. We show this with a brief example. We take a single p -coin and a single q -coin and perform a Bell-basis measurement across them. We find the ϕ^- output gives a function which is zero along the line $p = q$, which according to our result is impossible using only single qubit unitaries, therefore we can assert that here joint measurement adds value, not just efficiency.

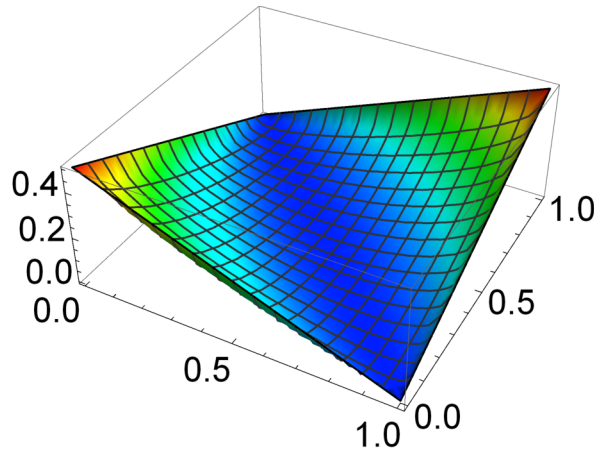


Figure 4.3: A function constructible with joint measurements. We note that along the line $p = q$ the function is zero. This is not possible with only single qubit measurements.

4.2.2 One coin with two parameters

In the original problem we restricted our quantum coins to a small slice of the Bloch Sphere to make comparisons with the classical case more clear. We can also make an addition of a phase variable such that our coins span the entire space and investigate the functions constructible in that case. Paradoxically we find expanding the problem not only produces a far more restrictive set of functions than that of two different coins, but also that the addition of the phase variable hampers the construction of functions that depend only on p . One way to see why this occurs is that a $|+\rangle$ -state which then has a random phase applied to it produces a completely mixed state, as we discuss in chapter 5, mixed states are far less powerful than the pure states within the Bernoulli factory.

4.2.2.1 Formal statement of the problem

Once again we are required to redefine our probability measure but again much of the proof remains the same. With the addition of the phase parameter, ϕ , our probability density becomes

$$P_{p,\phi,G} \left(\prod_{i=1}^{|G|} V(\underline{v}_i) \right) = \prod_{i \in b_p} \prod_{j=1}^{|\underline{v}_i|} |\langle \underline{v}_{i,j} | U_i | p, \phi \rangle|^2, \quad (4.42)$$

where our state is defined as

$$|p, \phi\rangle = \sqrt{1-p}|0\rangle + e^{i\phi}\sqrt{p}|1\rangle \quad (4.43)$$

Quantum Problem with phase: *A function $f : [0, 1] \times [0, 2\pi] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli factory iff there exists an event S and a probability measure $P_{p,G}$ such that S is discernible with respect to $P_{p,\phi,G}$ and $P_{p,G}(S) = f(p)$ Which functions are constructible?*

4.2.2.2 Result for two parameters

Theorem 5: A function $f : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is constructible with the quantum Bernoulli factory iff it satisfies

1. f is continuous.
2. There exists a set, X , of pairs of values $\{p_j, \phi_j\}$ such that $f(p_j, \phi_j) = 0$.
3. $\forall \{p_j, q_j\} \in X$ there exists constants $c, \delta > 0$ and integer $k < \infty$ such that

$$c(\phi - \phi_j)^{2k}(p - p_j)^{2k} \leq f(p, \phi) \quad \forall \{p, \phi\} \in [p_j - \delta, p_j + \delta][\phi_j - \delta, \phi_j + \delta].$$

4. There exists a similar set for when $f(p) = 1$.

For these functions we will also require that our generalised Hadamard matrix be further generalised to include a phase component. We define it as follows:

$$H(a, \theta) = \begin{pmatrix} \sqrt{1-a} & e^{-i\theta} \sqrt{a} \\ \sqrt{a} & -e^{-i\theta} \sqrt{1-a} \end{pmatrix}, \quad (4.44)$$

The first simple observation we make is that this case is very different to the previous case with two coins. Our two coins had two parameters between them p and q and we could ask questions about them separately. Both p and q could be interacted with in the same ways as p in the single coin factory. In this two-parameter-one-coin scenario we must interact with them together. This means that we cannot make measurements that can tell us $p \neq p_0$ or $\phi \neq \phi_0$. Instead we are restricted to pairs of values, when we make a measurement following the Hadamard matrix $H(a, \theta)$ we can learn that the state is not $|a, \theta\rangle$ but we can never learn whether it is the p -value or ϕ -value that is incorrect. We can apply the same argument as we have in each of the previous results that since the function is constructible there must be some number of flips k after which it will produce either output with some non-zero probability. This probability must be greater than a constant multiplied by the least probable string.

$$P(\text{output zero } |k \text{ flips}) \geq \epsilon \min_{s \in \{0,1\}^k} \prod_i (|\langle s_j | H(a_i, \theta_i) | p, \phi \rangle|)^{2k} \quad (4.45)$$

$$\geq \epsilon \prod_i (|\langle 0 | H(a_i, \theta_i) | p, \phi \rangle \langle 1 | H(a_i, \theta_i) | p, \phi \rangle|)^{2k} \quad (4.46)$$

$$\geq \epsilon \prod_i (p - a_i)^k + (\phi - \theta_i)^k \quad (4.47)$$

This expression tells us that our function can only go to zero at specific *points*, in contrast to the previous example which could always go to zero on lines of constant p and q . This means that the class of functions constructible in this case is strictly smaller than that of the previous case and that of the original single parameter, single coin factory. The function $f(p) = 2p$ is possible in both the single coin factory and the two-coin factory but not in the two-parameter factory. In some sense the extra parameter actually reduces the power by adding additional uncertainty whereas the extra coin only adds additional information.

As before, we begin with two bounding functions which almost allow us to approximate our function. In the previous example we could estimate p and q separately; in this case it is simpler to estimate the Bloch vector of the state using measurements in the X , Y and Z bases. We measure in each basis n times and we label the numbers of outcomes with eigenvalue one k_x , k_y and k_z respectively. Our initial attempts at bounding functions are as follows

$$A_n(p, \phi) = \sum_{k_x, k_y, k_z=0}^n \frac{2}{3} f\left(\frac{k_z}{n}, \tan^{-1}\left(\frac{2k_y - n}{2k_x - n}\right)\right) P(k_x, k_y, k_z | p, \phi) \quad (4.48)$$

$$B_n(p, \phi) = \sum_{k_x, k_y, k_z=0}^n \left(\frac{1}{3} + \frac{2}{3} f\left(\frac{k_z}{n}, \tan^{-1}\left(\frac{2k_y - n}{2k_x - n}\right)\right)\right) P(k_x, k_y, k_z | p, \phi) \quad (4.49)$$

Again these functions will, as n increases, bound our target function except at the finite number of points at which it is equal to zero or one. Once again, we make use of a step function in order to correct this. This function must be redefined for

this variant

$$T_{m,M,a_0,\theta_0}(p,\phi) := (1 - (1 - (| \langle 1 | H(a_0, \theta_0) | p, \phi \rangle |))^m)^M \quad (4.50)$$

This now gives us the functions we need to construct our lower bounding function (the upper is constructed in much the same way):

$$L(p,\phi) = A_n(p,\phi) \prod_i T_{m,M,a_i,\theta_i}(p,\phi) \quad (4.51)$$

Once again we have shown we can construct upper and lower bounding functions for our target function. Thus following Proposition 3 from our first result which tells us that if we can approximate a function we can construct it, we have proved that we can construct this class of functions. $\square$

4.3 Automata

Thus far we have concerned ourselves only with the case where the factory has no limitations on its computational power, it can store any amount of information and access that information at any time. Further interesting variants can be found when we restrict ourselves in this manner. These restrictions take us across the family of automata of which the Turing Machine is just one. Possibilities include [RS59,TB73,GS66,A72,S09]:

- A machine that can only be in a finite number of states, also known as a finite automaton
- A machine that can store an infinite amount of information but can only access the most recent addition, also known as a pushdown automaton.
- A machine with a single qubit as its memory, an infinite number of states but a finite amount of accessible information.

- A finite memory machine the rules of which change with the number of flips: a time-dependent finite automaton.
- A machine that must produce an output after N steps, events such as these are finite events.

4.3.1 Finite Events

It was shown in [W99] that the class of finite events have probabilities that correspond to certain classes of polynomials, we extend this result to the finite events quantum Bernoulli factory and demonstrate that this is another instance where access to joint measurement changes the class of functions that can be constructed.

Wastlund’s result for the classical case is as follows:

Classical Result for Finite Events: *A function represents the probability of a finite event if and only if $f(p)$ is a polynomial with integer coefficients which is identically 0, identically 1, or maps $(0, 1)$ to itself.*

Wastlund notes that all these functions can be expressed as “coin-flipping polynomials” of the form

$$f(p) = \sum_{i=0}^d a_{d,i} p^i (1-p)^{d-i}. \quad (4.52)$$

Now we can obviously create our own quantum finite events by simply substituting our $h(a, p)$ probabilities in the place of one or more of the terms. This does not, however, give us the full range of quantum finite events. The simplest explanation for this is that we can adapt where the classical algorithm cannot. Regardless of the result of the first flip, a classical algorithm can do nothing but make an output or flip again. A quantum algorithm can change the unitary applied to the second flip depending on the result of the first just as a classical algorithm can apply a stochastic matrix..

This means that our finite events can be much more complicated, with an event that terminates after N flips potentially involving 2^{N-1} different unitaries. This

means we must define each accepted string differently.

For a finite event which terminates in N flips we can characterise each accepted string, $s \in S$, by an ordered set of at most N values $a_i \in (-1, 1)$. Allowing our a -values to vary over $(-1, 1)$ allows us to avoid an extra set of strings for the desired outputs (eg. HHTH), we can instead encode this information in the sign of the a -value. We need to extend our definition of $h(p)$ for this purpose.

$$h_a(p) := \begin{cases} \left(\sqrt{p(1-a)} - \sqrt{a(1-p)} \right)^2 & ; \quad a \in [0, 1] \\ h_{1+a}(p) & ; \quad a \in [-1, 0] \end{cases}, \quad (4.53)$$

This covers all possible finite events but it also allows some events that are not possible in finite flips, so we must add the requirement that our sets, $\{a_i\}_s$, be *consistent*.

Definition 4: A set of ordered sets $\{\{a_i\}_s\}$

1. All pairs of ordered sets must share a finite initial segment, which may be empty. This condition is trivially satisfied.
2. Following the initial segment that a pair of ordered sets share, the difference between their next two values must be one.

This condition effectively forces the event to be a binary tree. Given that all the events, being coinflips, have only two outcomes this is the most general form it could take.

This allows us to write the probability of any finite event as:

$$f(p) = \sum_{s \in S} \prod_{\{a_i\}_s} h_{a_i}(p), \quad (4.54)$$

where we also require, as the classical result does, that only if $f(p)$ is identically 0, identically 1, or maps $[0, 1]$ to itself.

Similarly any consistent set of ordered sets $\{\{a_i\}_s\}$ describes a function defined as in equation (x) which corresponds to the probability of a finite event.

The increased class of operations as well as the ability to condition based on outcomes means that the quantum finite events can produce a much wider range of functions. What is perhaps more interesting is that the addition of joint measurements once again results in an increase in the class of constructible functions. Further, we show that unlike previous examples joint measurements on more than two qubits also increase the class of constructible functions. This is in contrast not only with our own results so far but also with quantum Turing machines, where there is a gap between single qubit operations and two-qubit operations but further additions add nothing to the power available.

For the two-qubit case we can once again use Bell measurements. A Bell measurement on two quantum p -coins gives a result of ϕ^- with probability

$$P(\phi^-) = \frac{1}{2} - 2p(1 - p). \quad (4.55)$$

This cannot be written in the form described above.

For our surprising three-qubit case we use three-qubit Greenberger-Horne-Zeilinger, GHZ, [G⁺90] states.

$$P(|GHZ^+\rangle) = P\left(\frac{|000\rangle - |111\rangle}{\sqrt{2}}\right) \quad (4.56)$$

$$= \frac{1}{2}((1 - p)^{\frac{3}{2}} - p^{\frac{3}{2}})^2. \quad (4.57)$$

This state also cannot be written in the form for single-qubit states. In addition it cannot be created using a fixed finite number of single and double qubit operations. Both of these functions are plotted in Figure 4.4.

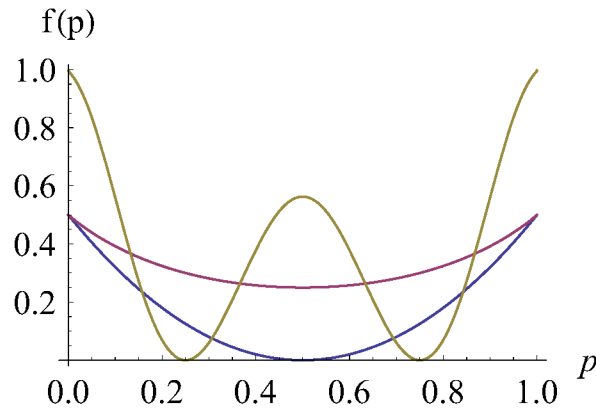


Figure 4.4: A selection of quantum finite events Quantum finite events only possible with at least three-qubit measurements (purple) and at least two qubit measurements (blue). Also included is a simple single-qubit measurement finite event which cannot be constructed by the classical factory, finite or non-finite (green).

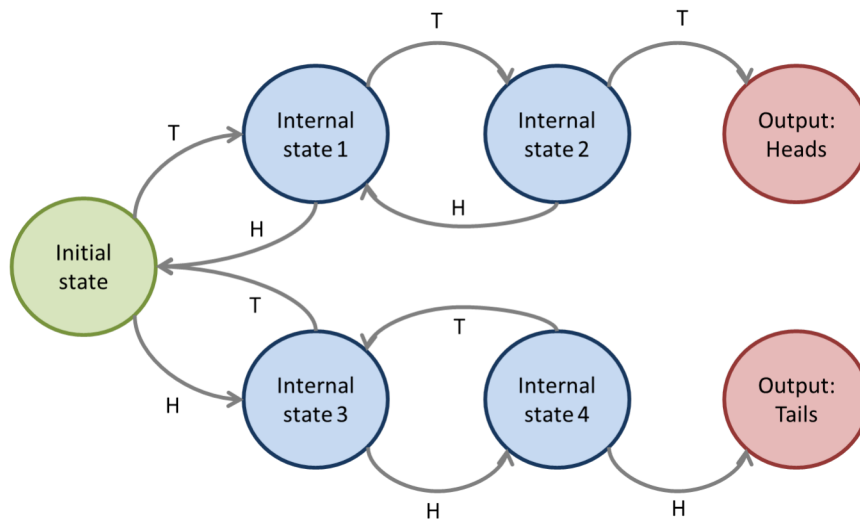


Figure 4.5: An Automaton Finite Automaton, this machine only has a finite number of internal states but still can run indefinitely. A finite *event* would have a similar diagram but with no loops allowed.

4.3.2 Finite and Pushdown Automata

It was shown in [MP05,NP05] that a finite automaton (Fig 4.4) can simulate rational functions of p satisfying certain conditions, and that a pushdown automaton can simulate certain non-rational algebraic functions. We extend these results to the quantum case and find the results to be complicated by joint measurements, which here increase the class of constructible functions.

A rational function is a function which can be defined by a rational fraction: the ratio of one polynomial to another. Mossel and Peres note that for all rational functions $f : (0, 1) \rightarrow (0, 1)$ we can write them as $\frac{d(p)}{e(p)}$ where

$$d(p) = \sum_{i=0}^k d_i p^i (1-p)^{k-i} \quad (4.58)$$

$$e(p) = \sum_{i=0}^k e_i p^i (1-p)^{k-i} \quad (4.59)$$

where for all i the coefficients d_i and e_i are integers such that $0 \leq d_i \leq e_i$. Unfortunately the quantum Bernoulli does not offer so aesthetically neat a solution to the problem of functions constructible with finite automata, but if we can understand how the classical result arises then the quantum result will seem far less alien.

We consider a classical machine that runs for k flips, each flip giving a result of zero or one. We label each string of outcomes with “zero,” “one” or “repeat.” In the case of the first two we simply output that result, in the case of “repeat” we run the machine again. Since it always runs for a fixed finite number of flips, it only requires a fixed finite amount of memory. It is a finite automaton. The probability of each string is, necessarily, of the form $P(s) = p^i (1-p)^{k-i}$ where i is the number of ones. The polynomial $e(p)$ is the probability of all strings which give a definite answer, the polynomial $d(p)$ is the probability of all strings which give an answer of “one.” These are both polynomials with the desired form and, since the second set is a subset of the first, they must satisfy our requirement that $0 \leq d_i \leq e_i$ for all

i. Now that we understand the classical result in terms of labelling strings we can move onto the quantum result and find it far more understandable.

We consider the quantum version of this machine, which also runs for exactly k flips. We can define the set of all possible string outcomes and their associated measurements, which may be adaptive, using the notion of *consistency* set out in the section on finite events. We will use $S = \{0, 1\}^k$, the set of all possible outcome strings of length k , and their associated sets of measurements $\{\{a_i\}_s\}$. Again we will impose that these sets of measurements be consistent, giving us the most general k -flip machine possible.

Now we know that

$$1 = \sum_S \prod_{\{a_i\}_s} h_{a_i}(p) \quad (4.60)$$

So, as with the classical case, we must label each string, s , with either “zero,” “one” or “repeat.” We then define S_x as the set of all strings that give a definite answer, being “zero” or “one.” We also define S_1 as the set of strings that we labelled “one.” We now see that the functions constructible with finite automata must be of the form in Equation 4.61 below. The probability of obtaining a string which outputs “one” over the probability of obtaining a definite output is

$$f(p) = \frac{\sum_{S_1} \prod_{\{a_i\}_s} h_{a_i}(p)}{\sum_{S_x} \prod_{\{a_i\}_s} h_{a_i}(p)}. \quad (4.61)$$

The classical result for pushdown automata is less straightforward, showing that pushdown automaton can construct certain non-rational functions but not characterising the whole set of constructible functions. Similarly we can show that the quantum pushdown automata can construct functions which are not constructible by the quantum finite automata. The example used in Mossel and Peres’s work is that of $\sqrt{p}$ and we can adapt this methodology to produce a finite automaton that produces $2p$. The details of this algorithm are given in Chapter 5 as it is also the focus of many of the algorithms which could be implemented in real experiments.

4.3.3 Coin distinguishing problems

The following work was done in collaboration with Antony Milne at Imperial College London.

It was shown in [AD11] that quantum advantage can be obtained for certain classes of finite automata in coin distinguishing problems [HC70]. We consider eight classes of automata ranging from the relatively weak Classical Finite Automaton to the full power of the Quantum Bernoulli factory in an attempt to answer an open question posed by Aaronson. Unfortunately we do not succeed but the work may prove useful to future attempts at the problem. We consider:

- Whether the automaton's operations are quantum or classical.
- Whether the memory is finite or infinite.
- Whether the automaton's input, the coins, are quantum or classical.

These three binary properties describe $2^3 = 8$ distinct automata. We consider their ability to handle the following problem:

You are given a coin which you are promised has bias p or $p + \epsilon$. Can the automaton distinguish, with certainty, between the two for any finite ϵ ? For this problem we relax the condition that the automaton must give an answer in finite time. Instead the automaton has two “answer states” one corresponding to the fair coin and one corresponding to the unfair coin. We require only that as the number of flips increases the probability that the automaton is in the correct state is at least $2/3$. More formally, if a_i is the probability that the machine is in the correct state after i steps then we require

$$\lim_{t \rightarrow \infty} \frac{a_1 + a_2 + \dots + a_t}{t} \geq \frac{2}{3}. \quad (4.62)$$

Aaronson's open problem asks whether an automaton, with finite memory, classical inputs and quantum operations could solve this problem. We will consider models which are both stronger and weaker than this model in an attempt to solve

the problem. If we find that a stronger model cannot solve the problem, then we know that this model cannot solve it. Similarly if we find a weaker model can solve the problem, then we know that this model must be able to solve it.

We note first that changing the input from classical to quantum always allows the problem to be solved, since measuring in the $\{|+\rangle, |-\rangle\}$ basis will allow the two coins to be distinguished eventually. We next note that the removal of the finite limit on memory is also sufficient by itself, since recording a random walk with all positive states being accepting states is again sufficient to distinguish the coins in the limit.

This leaves us only with the step of making the finite memory into a quantum memory. Aaronson's paper details an algorithm that works for this case if it is allowed to depend on ϵ since in such a case the single qubit can be used as a counter. Unfortunately for the case of an unknown ϵ there is no way to divide up the state space such that it works for all ϵ .

One of many attempts we made to find a viable method for this goes as follows:

- We have a single qubit, initially in state $|0\rangle$ as memory.
- Whenever we flip a coin we apply a small rotation θ to the qubit. Positive rotation for heads; negative for tails.
- For an unbiased coin the state will oscillate around $|0\rangle$. For any bias, however small, it will slowly make its way around the Bloch sphere spending a roughly uniform amount of time in each region of the circle it transcribes.
- Unfortunately this difference is not enough for our algorithm to succeed
- If we halt the algorithm after some finite amount of flips, we can always choose our ϵ to be small enough that we cannot resolve the difference between the two.
- If we allow the algorithm to run forever in the “limit” scenario then the oscillations around $|0\rangle$ by the unbiased coin will become sufficiently large that

both coins will have their states uniformly distributed around the circle. This renders them once again indistinguishable.

It is our impression from time spent on this problem (an impression Aaronson shares [APC]) that the quantum memory finite automaton cannot perform this task under these conditions but the route to proving this is not clear to us.

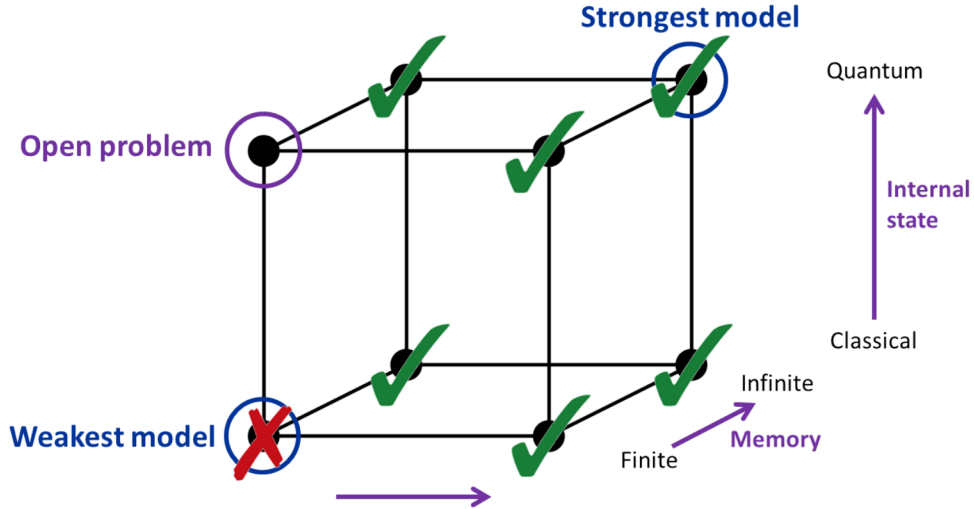


Figure 4.6: A cube depicting the eight automata considered. We note that all stronger models can perform this coin distinguishing operation, but the weaker model cannot. This does not allow us to solve the problem.

4.4 Discussion

We have modified the Bernoulli factory in a wide range of ways to try and understand where the crucial differences between quantum and classical lie. In chapter three we focused on how the quantum factory is able to treat each point in the same way that the classical factory treats zero or one. Here we have widened the scope of the questions we may ask and explored in more detail how much time and memory is required to answer a given question. We note that the power of the quantum Bernoulli factory is greatly diminished by restrictions on memory but advantage over the classical remains in all cases. In the next chapter we will examine how the quantum advantage stands up in more practical scenarios with errors in operations

and measurements. We will then tie it to this chapter by considering how quantum and classical finite events compare when try to emulate a non-finite factory with errors.

Chapter 5

Practicalities of the Quantum Bernoulli Factory

Whilst most of our discussion has been theoretical, our algorithms infinite in expected runtime and our quantum coins without error, it is important to tether ourselves to reality. Here we explore how the quantum Bernoulli factory could be implemented in practical situations.

The work within this chapter is my own unpublished work together with some very brief reviews of the work of two experimental groups. Where the algorithms discussed have been influenced or created by others credit is clearly given.

5.1 A practical example

Terry Rudolph designed the following algorithm to produce a $2p$ -coin. It uses a single joint measurement on two copies of the same p -coin. It can be carried out by a pushdown automaton but not by a finite automaton, as with the classical $\sqrt{p}$ function. As we discussed in the previous chapter, joint measurements do not increase the class of constructible functions for a single coin of one parameter (though we did note that for the case with two different coins they did make a difference).

In fact this method works for the extended function

$$f_{\wedge}(p) := \begin{cases} 2p & ; \quad p \in [0, 1/2] \\ 2(1-p) & ; \quad p \in (1/2, 1] \end{cases}, \quad (5.1)$$

We note that we can expand our target function as

$$f_{\wedge}(p) = 1 - \sqrt{1 - 4p(1-p)} \quad (5.2)$$

$$= \sum_{k=1}^{\infty} \binom{2k}{k} \frac{1}{(2k-1)2^{2k}} (4p(1-p))^k \quad (5.3)$$

$$= \sum_{k=1}^{\infty} a_k (4p(1-p))^k \quad (5.4)$$

$$= \sum_{k=1}^{\infty} a_k g_k(p). \quad (5.5)$$

This decomposition based on the one used for the $\sqrt{p}$ algorithm discussed in Chapter 2.

This last line makes our protocol clear. Since the a_k 's form a probability distribution we can generate a value of k classically. Once we have a value of k we produce the corresponding $g_k(p)$ and if the result is heads, we output heads overall. Unfortunately we notice that the $g_k(p)$ functions are classically impossible and using our earlier quantum results will clearly not lead to a “practical” answer.

We note first that

$$g_k(p) = g_1^k(p). \quad (5.6)$$

This means that if we can produce $g_1(p)$ we can produce all $g_k(p)$. We do this by flipping a $g_1(p)$ -coin k times and outputting heads only if every flip results in heads. To do this we consider a Bell basis measurement performed on two p -quoins:

$$P(|\phi^+\rangle) = P\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right) = \frac{1}{4} \quad (5.7)$$

$$P(|\phi^-\rangle) = P\left(\frac{|00\rangle - |11\rangle}{\sqrt{2}}\right) = \frac{1}{2} - 2p(1-p) \quad (5.8)$$

$$P(|\psi^+\rangle) = P\left(\frac{|01\rangle + |10\rangle}{\sqrt{2}}\right) = \frac{1}{4} \quad (5.9)$$

$$P(|\psi^-\rangle) = P\left(\frac{|01\rangle - |10\rangle}{\sqrt{2}}\right) = 2p(1-p). \quad (5.10)$$

The ψ^+ probability is close to what we want, but off by a factor of two. We can fix this by conditioning on our getting either ψ^+ or ϕ^- . The probability of getting one of these results is one half, so the probability of getting ψ^+ , given that we get one of either ψ^+ or ϕ^- , is $4p(1-p)$. This is our desired function. This means we can produce $g_1(p)$, and therefore $g_k(p)$. Using these, and in conjunction with classically generating a value for k , our function $f_\wedge(p)$ can now be constructed.

One of the reasons this algorithm is so appealing is that it only requires the same quantum measurement over and over again; it does not adapt. This means one can simply set up apparatus to carry this measurement out on two p -coins, collect as many data points as required and then carry out all post-processing and classical randomness afterwards. This streamlines the process greatly. The first few terms of this decomposition of $f_\wedge(p)$ are shown in Figure 5.1.

The same decomposition and measurements can also admit an alternative, but equivalent, variant of the algorithm based on a random walk. This version of the algorithm is based of an algorithm found in [MP05].

One begins at the point marked “Start” and flip a $4p(1-p)$ -coin, created using the methodology described above, at each junction. The first flip places the walker on the ladder. At each step, if the $4p(1-p)$ coin gives tails then the walker moves horizontally across the ladder; if it gives a result of heads then the walker moves randomly up or down the ladder with probability one half for each. If the walker reaches the bottom rung of the ladder then the algorithm terminates. If the walker

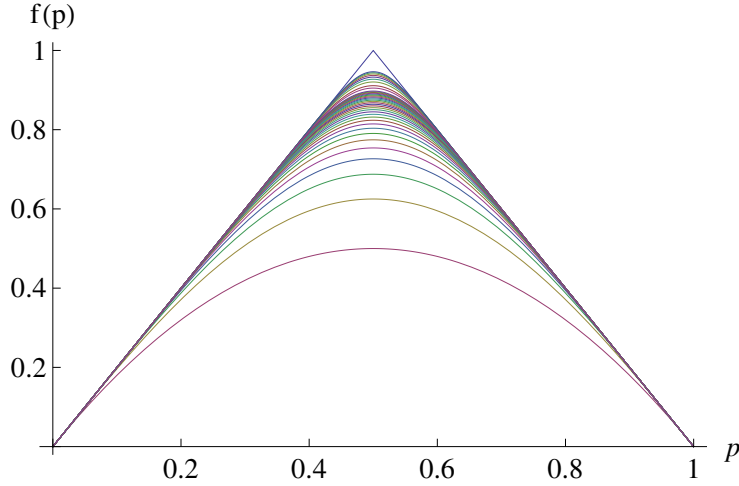


Figure 5.1: The probability amplification function This shows the function $f_{\wedge}(p) : [0, 1] \rightarrow [0, 1]$ and the first 35 terms of its decomposition.

finishes on the left then the algorithm outputs heads; if it finishes on the right then we output tails. This random walk is shown in Figure 5.2.

This algorithm, despite a different order of operations, still requires exactly the same number of quantum measurements.

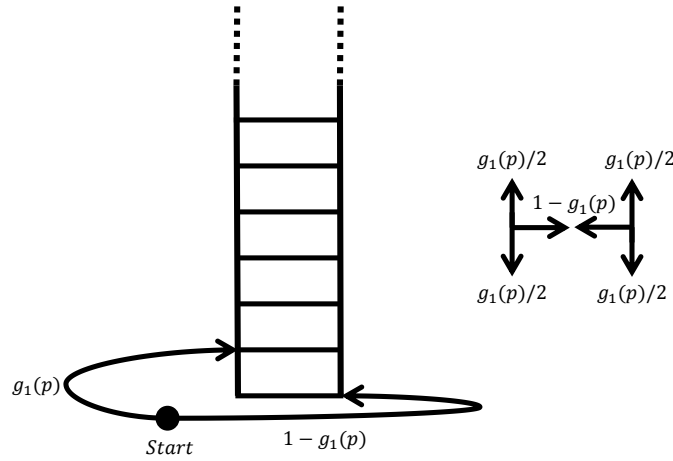


Figure 5.2: The random walk used in the algorithm The algorithm uses a random walk on a semi-infinite ladder with transitions given by Bell-Measurement outcomes on two qubits.

This algorithm has finite expected runtime for all $p \in (0, 0.5)$, although as p approaches 0.5 the runtime does tend to infinity. This behaviour is caused by rare strings which require an extremely large number of flips to determine whether the

event occurs, most strings result in relatively low numbers of flips. Even for $p = 0.5$ we find that, although the expected runtime is infinite, simulations show over 95% of runs terminating within 200 coinflips. The runtime here has properties much like the $\sqrt{p}$ algorithm discussed in Chapter 2 which is unsurprising given the similarities in their construction. The expected number flips required for a given p value are shown in Figure 5.3

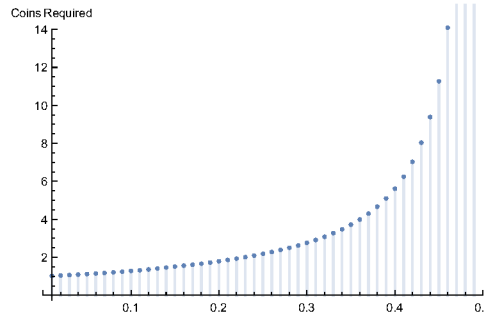


Figure 5.3: Efficiency of $2p$ algorithm This graph shows the expected number of flips required to execute the algorithm for a given value of p .

5.2 Comparisons with the classical case

Classically this function is impossible to construct. Meaningful comparisons can still be made due to the necessary imperfections of any experimental realisation. We will show that many commonly arising errors result in the final function produced being a function that could be constructed classically. We will then show that this does not render the quantum Bernoulli factory impractical since it still offers substantial speed-up over the classical factory for the functions it produces.

To do this we must first have some idea of how many classical coins are required for a given function. To obtain this we follow the methodology of Thomas et al [TB11] and Flegal and Herbei[FH12], who quantify the amount of coins required to

produce various elbow functions. Elbow functions are of the form

$$f_{\wedge}(p) := \begin{cases} 2p & ; & p \in [0, 1/2 - \epsilon/2] \\ 1 - \epsilon & ; & p \in (1/2 - \epsilon/2, 1/2 + \epsilon) \\ 1 - 2p & ; & p \in [1/2 + \epsilon, 1] \end{cases} \quad (5.11)$$

This ϵ sized gap is there to make sure the function is classically constructible. It prevents the function from taking a value of one which would render it impossible according to the Keane-O'Brien Theorem. [KOB94].

Neither of the papers detailing this have algorithms which are provably optimal. In addition the measure we use for efficiency, expected number of flips does have problems since many of the algorithms suffer from, in Thomas's words, "termination by gluttony." Many of these algorithms frequently terminate with reasonable numbers of flips, recall that for $p = 0.5$ the quantum algorithm finishes within 200 flips 95% of the time. In the rare cases in which they do not terminate early the number of flips required grows exponentially. In addition many of these algorithms, like our own from the semi-constructive proofs in chapters three and four, require that polynomials be calculated for each step. These polynomials in these classical algorithms can begin to use massive amounts of memory in the later steps.

Both papers provide estimates of an expected value of around 2^{16} for an ϵ of 0.2. Experiments are expected to produce values of ϵ closer to 0.02 so this value will be even higher. Thomas also provides an estimate comparable to our value of 200 for the practical $2p$ algorithm discussed earlier but note that this is at the expense of expected value. In the case that the algorithm is targeted at often giving quick results it is possible for a classical algorithm to output results within 20 flips 95% of the time. Our algorithm will, for most p values, beat this easily (see Figure 5.3).

Obviously the quantum factory can always run a classical algorithm, so whatever the classical case is optimised for we can at least match it with the quantum factory.

5.3 Errors

Errors in implementations of the quantum Bernoulli factory can broadly be categorised into three types. Obviously there are plenty of other errors that can arise but in almost all cases the end result would be modelled in a way similar to one of the following.

- Imperfect state preparation, perhaps resulting in a partially mixed state.
- Imperfect calibration, causing the value of p to deviate from the desired value.
- Imperfect measurements, such as dark counts in photodetectors.

All these examples will result in a loss of the gap in constructible functions between the quantum and the classical, but the superior efficiency of quantum over classical will be preserved.

5.3.1 Mixed States

Whilst this error can occur in a number of ways we will model it simplistically as a mixture. Our pure quantum p -coin mixed with an ϵ of a classical coin state.

$$\rho = (1 - \epsilon)|p\rangle\langle p| + \epsilon \begin{pmatrix} 1 - p & 0 \\ 0 & p \end{pmatrix} \quad (5.12)$$

It is immediately apparent that this reduces the power of the factory since we can no longer be certain that p does not take certain values. Figure 5.4 shows the results of a $4p(1 - p)$ algorithm running on coins of this form, we note that $f(0.5) \neq 1$.

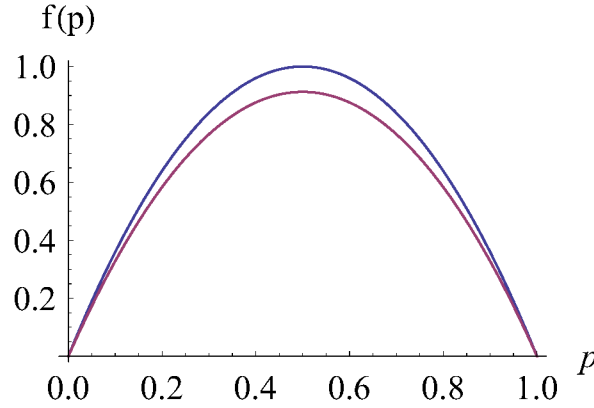


Figure 5.4: The effect of decoherence on the observed function For most regions this produces a function close to the target but at points where the function approaches zero or one the deviation is more pronounced.

5.3.2 Errors in p -values

If our equipment, instead of giving us the p -value we want, gives us a random (normal) distribution around that value.

$$p' \sim \text{Normal}(p, \epsilon) \quad (5.13)$$

For most regions of most functions this will still, on average, produce the correct answer. However changes in gradient can result in a drift upwards or downwards and any points which were previously zero or one will not be afterwards. We use the example of $2p$ here since it shows this behaviour more clearly. The results of this model are shown below in Figure 5.5 for $\epsilon = 0.01$

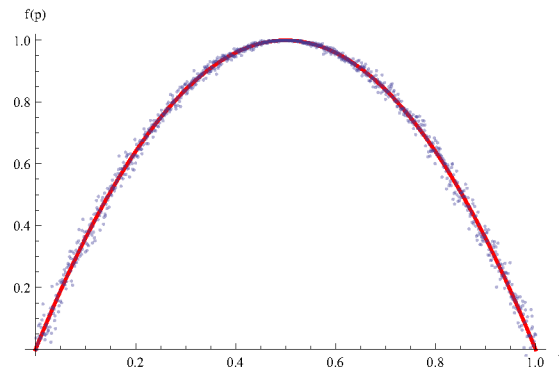


Figure 5.5: The effect of deviations in p -values on the observed function We note that this produces some amount of drift which depends on the derivative of the function.

5.3.3 Dark Counts

For dark counts in a photonic implementation, or other imperfect measurements, we model this as some chance, ϵ , that instead of obtaining the true measurement result we simply obtain a random bit. So our measurement result will instead be, for example,

$$P(0|p) = (1 - \epsilon)|\langle 0|p\rangle|^2 + \frac{\epsilon}{2}. \quad (5.14)$$

For more complicated measurements, like the Bell basis measurement, we can consider each detector individually. This would mean a $(1 - \epsilon)^2$ chance of the true result, and slightly different chances for each other results. The results of this model are shown below in Figure 5.6 for $\epsilon = 0.01$.

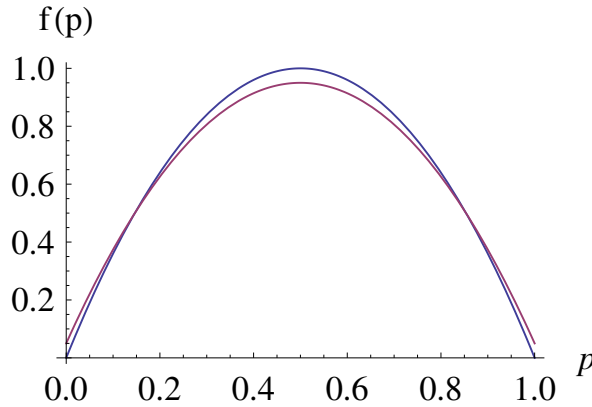


Figure 5.6: The effect of dark counts on the observed function We note that this produces higher deviations the further the function is from a value of one half.

5.3.4 Combined Errors

In reality we will likely have errors similar to all three of these error types in any implementation. Here we model the case where all error types are present. First the value of p is randomly generated from a normal distribution centered around the true value, then some decoherence is applied and finally the measurement is imperfect. The effective order of these might vary with implementation but this is

the order we would expect for most.

We show in Figure 5.7 a comparison between the desired function and the data produced by a system with substantial adjustments from all three categories of error.

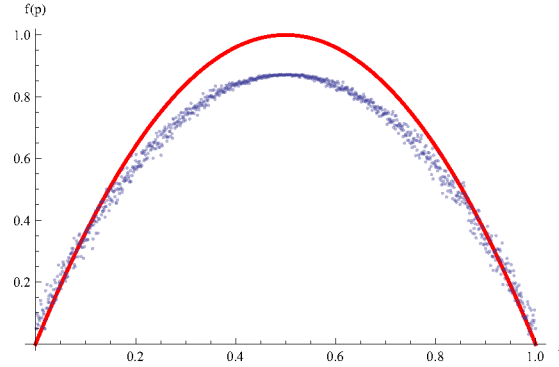


Figure 5.7: Simulated data from a “realistic,” although very error-prone, experiment We see that at $p = 0.5$ the deviation is most severe and that the spread of the datapoints is wider towards the endpoints.

We can use the function produced from this as another way to compare classical and quantum efficiencies. The function is clearly skewed towards the capabilities of the quantum factories but not so much that it is impossible for the classical factory to produce it at all. Since the exact function is unknown we require only that the factories produce function that closely match the data produced.

For the classical case this is achieved using the approximations from Chapter 3,

$$A_n(p) = \sum_{k=0}^n f\left(\frac{k}{n}\right) \binom{n}{k} p^k (1-p)^{n-k}. \quad (5.15)$$

If we set $n = 15$ and $f(p) = 3.5p(1-p)$ we obtain a function that fits the data very well. This method uses fifteen coins.

For the quantum case we find we can obtain the shape very quickly using four $h_{0.5}(p)$ -coins and outputting “heads” if they are all “heads” but unfortunately some scaling is required. Luckily a scaling of $7/8$ is both accurate and easily obtained. Since we can measure our quantum coins in the Y -basis to obtain a fair coin flip. We do this three times. If we obtain result of “tails” from every flip then our algorithm always outputs “tails”, otherwise it uses the four $h_{0.5}(p)$ -coins result. This uses only

seven coins, less than half the amount used by the classical.

The two finite events occur with the following probabilities:

$$f_{\text{classical}}(p) = \sum_{k=0}^{15} 3.5 \left(\frac{k}{15} \left(1 - \frac{k}{15} \right) \right) \cdot \binom{15}{k} p^k (1-p)^{15-k} \quad (5.16)$$

$$f_{\text{quantum}}(p) = 0.875 \left(1 - \frac{1 + 2\sqrt{p(1-p)}}{2} \right)^4. \quad (5.17)$$

They are plotted in Figure 5.8

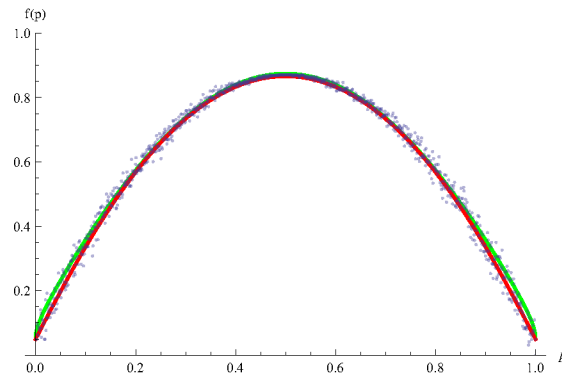


Figure 5.8: Comparison of quantum and classical finite events to the simulated data We see that both classical (red) and quantum (green) factories are able to get good fits to the data with quite a low amount of coins although the quantum uses fewer.

5.3.5 Discussion

We have characterised the gap between quantum and classical as stemming from our ability to ask different questions about the underlying distribution. In all these cases there is additional uncertainty resulting in a different distribution, but this is not the only difference. This additional uncertainty is very much classical uncertainty, representing our lack of knowledge rather than uncertainty which is fundamental to the state. We can ask more complex questions of the quantum state but in these instances we are asking the question of a mixture of quantum states, with each state giving a different answer. As such the certainty we achieve in the quantum pure state case is no longer achievable.

5.4 Implementation

Two approaches to implementing the quantum Bernoulli factory have been discussed, without giving too much experimental detail we will discuss both and the difficulties they face.

5.4.1 Photonic Chips

The Bristol Centre for Quantum Photonics group [P⁺08,OFV09,L⁺10] use adjustable photonic chips to implement Bell basis measurements for various applications [PMO09,P⁺12]. Given that the only measurement used in the method described above for $f_{\wedge}(p)$ and $4p(1-p)$ is a Bell basis measurement and the well-known coherence properties of photons this seemed like an ideal system on which to implement the Bernoulli factory. Since the method requires only one quantum operation, the two qubit measurement, it is possible to simply leave the experiment running overnight. This allows a large amount of data to be collected which can be processed into $2p$ -coins at a later time.

The photonic chips at Bristol use a pulsed laser passed through a non-linear crystal. This generates multiple identical photons via spontaneous parametric down conversion. These are then sent through a waveguide circuit etched onto silicon oxynitride. The final measurements are carried out by a time-correlated single photon counting system.

5.4.2 Superconducting Qubits

We were contacted by a group from Tsinghua university, China, who had read our paper and implemented it on a superconducting system [Y⁺16,DWM04].

The superconducting qubit approach makes use of a different method to generate $f_{\wedge}(p)$ since it does not have access to joint measurements. This part of their method was first brought to our attention by David Speyer, [SPC] albeit in a slightly different form.

- Let us say we have a p -coin and a q -coin, restricted such that $p + q \in (0, 1)$
- We can create a $\frac{p}{1+p}$ -coin from a p -coin by flipping until we flip tails and outputting heads if the number of flips was even.
- We then flip a $\frac{p}{1+p}$ -coin and a $\frac{q}{1+q}$ -coin simultaneously, outputting heads if only the former yields heads and tails if only the latter yields heads. It can be shown that this will give heads with probability $\frac{p}{p+q}$. [SPC]

Given this procedure, a $4p(1-p)$ coin can be created from a $h_{0.5}(p)$ -coin and a p -coin as follows:

- First we take our p -coin and our $h_{0.5}(p)$ -coin, recall that $h_{0.5}(p) = (1 + 2\sqrt{p(1-p)})/2$.
- For each coin we perform the following: flip it twice, if the results are different output heads, if they are not then output tails
- This yields a $2p(1-p)$ -coin and a $2h_{0.5}(p)(1-h_{0.5}(p))$ -coin
- We then use the above algorithm to generate a $4p(1-p)$ -coin as follows:

Let

$$m = 2p(1-p) \tag{5.18}$$

$$n = 2h_{0.5}(p)(1-h_{0.5}(p)) \tag{5.19}$$

$$= \frac{1}{2} - 2p(1-p) \tag{5.20}$$

Recalling that we can generate a $p/(p+q)$ -coin from p -coins and q -coins from the procedure above, we note

$$\frac{m}{m+n} = \frac{2p(1-p)}{2p(1-p) + \frac{1}{2} - 2p(1-p)} \quad (5.21)$$

$$= \frac{2p(1-p)}{1/2} \quad (5.22)$$

$$= 4p(1-p) \quad (5.23)$$

Despite this algorithm seeming more complicated than the previous one we can still separate the measurement and the post-processing. Given a large amount of data for p -coins and $h(0.5)$ -coins the rest can be done afterwards. Improvements can be obtained by conditioning the ratio of flips performed on what has been observed so far. For example, if initial measurements indicate a value for p which is close to one half then a great deal more $h(p)$ -coins will be required than p -coins.

The actual realisation of the factory was carried out in a waveguide transmon qubit which achieves measurement fidelities of around 95%.

5.5 Joint measurements

We now show that while joint measurements do not add any new constructible functions to the quantum Bernoulli factory, they do often increase efficiency. We briefly review how many coins are saved by joint measurement in this instance.

Following the methodology used in [Y⁺16] for creating a $4p(1-p)$ coin with only single qubit operations we find the following:

- We begin with two types of single qubit measurements, requiring a single coin each.
- We then generate $2p(1-p)$ coins from each of the above, this requires two coins deterministically.
- We then require to generate from these coins $p/(1+p)$ -coins. This requires $1/(1-p^2)$ coins on average.

- We then follow the procedure to generate a $p/(p+q)$ coin from a p -coin and q -coin. This requires $1/(1-pq - (1-p)(1-q))$ coins on average

We combine these four expected amounts to arrive at the following formula for number of coins used as a function of p which is somewhat unwieldy

$$\text{Flips}(p) = 2 \left(\frac{1}{1 - 4p^2(1-p)^2} + \frac{1}{1 - \left(\frac{1}{2} - 2p(1-p)\right)^2} \right) \times \quad (5.24)$$

$$\frac{1}{\frac{2p(1-p)}{1+2p(1-p)} \left(1 - \frac{\frac{1}{2} - 2p(1-p)}{\frac{3}{2} - 2p(1-p)}\right) + \frac{\frac{1}{2} - 2p(1-p)}{\frac{3}{2} - 2p(1-p)} \left(1 - \frac{2p(1-p)}{1+2p(1-p)}\right)}. \quad (5.25)$$

Despite the length of this equation due to a few dominant terms it appears relatively simple when plotted. Figure 5.9 indicates that the minimum number of flips is just over 16 at $p = 0.5$. The number of flips approaches infinity for p -values close to zero or one. This method is certainly not provably optimal but we are not aware of a better one.

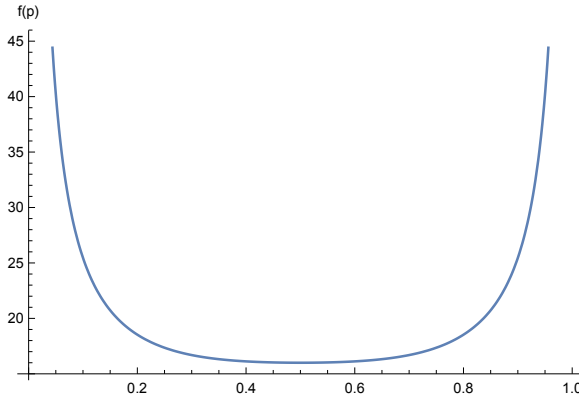


Figure 5.9: Efficiency of single qubits only The number of flips required to generate $4p(1-p)$ with single qubit operations only using the method described in [Y⁺16]

We recall that the joint measurement gives us one of the desired outcomes with probability one half, meaning the expected number of quantum operations required is two. This means the joint measurement improves efficiency in this case by a factor of at least eight for all p .

Chapter 6

Q-Sampling and Quantum Foundations

6.1 Sampling and Q-sampling

Thus far we have only considered the transformation of the quantum p -coins into a state with no coherence, equivalent to

$$\rho = (1 - f(p))|0\rangle\langle 0| + f(p)|1\rangle\langle 1| \quad (6.1)$$

We now instead consider creating a coherent sample, or “q-sample”

$$\phi = \sqrt{1 - f(p)}|0\rangle + \sqrt{f(p)}|1\rangle \quad (6.2)$$

6.1.1 The power of Q-sampling

It is noted in [LR02] that the ability to q-sample certain distributions would lead to powerful quantum algorithms. In particular, if we were granted the power to q-sample any distribution we could generate classically the following, very simple, algorithm for Graph Isomorphism would follow. The Graph Isomorphism problem is that of determining whether two graphs are permutations of each other [M81].

- Note that we can sample from a distribution of random permutations of a given graph classically simply by picking a permutation at random and applying it.
- Now if we instead create a quantum state of this distribution: a state which is a uniform superposition of all permutations of a given graph.
- If we do this for two graphs, G_1 and G_2 , the states produced will be identical if they are isomorphic but orthogonal if they are not.
- We can use a SWAP-circuit [WW12] to determine deterministically whether the two states are orthogonal or identical.

Since many attempts have been made at polynomial time quantum algorithms for Graph Isomorphism [H⁺10,G⁺10,BW11,DW08] and none have succeeded we can conclude that q-sampling is certainly not a trivial problem. In the context of classical computing a quasi-polynomial time algorithm has recently been discovered [B15], cementing the notion that Graph Isomorphism is in some sense “close” to being solvable in polynomial time.

Even in the simplify context of the Bernoulli factory it is not obvious if any of the variants we have discussed lead to a description of the states which can be q-sampled.

6.1.2 What can be q-sampled?

Of the types of events we have considered so far in this work it is the finite events that hold the most promise for q-sampling. Indeed, anything that does not run in a fixed finite number of steps runs afoul of the Quantum Subroutine problem [NO01].

Nishimura’s paper shows that quantum computers with an efficiently computable subroutine oracle can be simulated by quantum computers without such an oracle. However, this proof only works when the subroutine enters an answer state deterministically. A probabilistic subroutine runs into problems because when the subroutine halts, the number of steps it ran in encodes information about the answer,

not to mention the mathematical difficulties of finding operation that implement such a subroutine.

It is easy to see that we have the same problem with our algorithms. Whilst we might be able to map every string onto $|0\rangle$ or $|1\rangle$ that would require an infinite-dimensional unitary acting on infinite qubits, clearly an impossibility. Similarly if we run the algorithm until it halts we destroy our state with the knowledge of how many steps were required. As discussed earlier, one can provide a sample to someone that is a perfect $f(p)$ sample, but then destroy it by telling them the number of steps required to produce it.

For one final time we return to our function $f(p) = 4p(1 - p)$ and give an algorithm for producing a q-sample of it. We begin with two quantum p -coins and the unitary operation defined as:

$$|p\rangle = \sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle \quad (6.3)$$

$$|pp\rangle = |p\rangle \otimes |p\rangle \quad (6.4)$$

$$U = |00\rangle\langle\phi^-| + |10\rangle\langle\psi^+| + |01\rangle\langle\phi^+| + |11\rangle\langle\psi^-|. \quad (6.5)$$

Applying this to our coins and rearranging we arrive at the following:

$$U|pp\rangle = U\left(\sqrt{\frac{1}{2} - 2p(1-p)}|\phi^-\rangle + \sqrt{2p(1-p)}|\psi^+\rangle + \frac{1}{2}(|\phi^+\rangle + |\psi^+\rangle)\right) \quad (6.6)$$

$$= \sqrt{\frac{1}{2} - 2p(1-p)}|00\rangle + \sqrt{2p(1-p)}|10\rangle + \frac{1}{2}(|01\rangle + |11\rangle) \quad (6.7)$$

$$= \frac{1}{\sqrt{2}}((\sqrt{1-4p(1-p)}|0\rangle + \sqrt{4p(1-p)}|1\rangle) \otimes |0\rangle + |+\rangle|1\rangle) \quad (6.8)$$

We note that if we measure the second register we get each result, zero or one, with equal probability. If we get the result zero then the first register is left with

$$|f(p)\rangle = \sqrt{1-4p(1-p)}|0\rangle + \sqrt{4p(1-p)}|1\rangle. \quad (6.9)$$

This is the q-sample that we were aiming for. Since we require a result of zero, which occurs with probability one half, this procedure requires four quantum p -coins on average.

This procedure of reducing the space of n qubits to that of a single qubit can be generalised to produce a set of functions which we *can* q-sample but this set is not exhaustive. If we recall from Chapter 4 our description of finite events as

$$f(p) = \sum_{s \in S} \prod_{\{a_i\}_s} h_{a_i}(p), \quad (6.10)$$

where we also required that the sets $\{a_i\}$ have a set of properties that we called *consistent*. In the general case it is not obvious how to sample a finite event of this form, but it does lead us to a set of functions we can sample.

Consider a finite event where the sets $\{a_i\}$ must all have the same size, N . This restriction does not change the class of functions but simply keeps them in a form which is easier for us to work with. If we consider two strings, s_1 and s_2 , which occur with probabilities P_1 and P_2 we can show that a q-sample with bias $\frac{P_2}{P_1+P_2}$ is fairly straightforward to create. Normally we measure our quantum coins and then apply adaptive measurements conditioned on the outcomes; here we must use quantum gates controlled on previous qubits to maintain coherence. This allows us to create a state of the form

$$|\psi\rangle = \sum_{s \in S} \prod_{\{a_i\}_s} h_{a_i}(p) |s\rangle. \quad (6.11)$$

Now that we are in possession of this state we apply a unitary to it which is a permutation of all strings of length N . It doesn't matter what permutation it is so long as s_1 maps to the string of all zeroes and s_2 maps to the string 1000... (a single one with the remaining bits being zero). Now, as with our $4p(1-p)$ example, we measure all but the first qubit. If we obtain the result of all zeros, which occurs

with probability $P_1 + P_2$ then our first qubit is left in the desired q-sample

$$|p\rangle = \sqrt{\frac{P_1}{P_1 + P_2}}|0\rangle + \sqrt{\frac{P_2}{P_1 + P_2}}|1\rangle. \quad (6.12)$$

The above algorithm is very simple although understanding which functions it includes can be difficult. It always requires an average of $\frac{1}{(P_1+P_2)}$ coins, since $P_1 + P_2$ is our probability of success. The next step is characterise these functions.

Theorem 3: A function $f : (0, 1) \rightarrow [0, 1]$ is q-constructible with the quantum Bernoulli Factory if it can be written as $\frac{f_2(p)}{f_1(p)+f_2(p)}$ where $f_1(p)$ and $f_2(p)$ have the following properties.

- The functions can be described as a string of single qubit measurements on quantum p -coins, which may or may not be adaptive.
- The functions must have the same results (strings s_i) and measurements $\{a_i\}_s$ up to some point, which may be the first measurement, at which point they have the same measurement but opposite results. Mathematically $s_{1,i} + s_{2,i} = 1$ and $|a_{1,i} - a_{2,i}| = 0$
- The strings must have the same length

This is similar to the finite events results but more restrictive; the fact that functions are generally ratios is reminiscent of the finite automata - rational functions result. The best way to understand this is that we can always describe a quantum Bernoulli factory with k steps by its output strings, $S = \{0, 1\}^k$ and sets of consistent measurements $\{\{a_i\}_s\}$.

We can tie our results for finite events, finite automata and q-sampling together with the following table. We consider how we label the output strings of the k -flip machine in Table 6.1.

Table 6.1: Comparison between finite events, finite automata and q-sampling.

Event Class	String labelling
Finite Events	All output strings must be labelled with a definite output “zero” or “one.”
Finite Automata	Some output strings, not necessarily all, are labelled with a definite output “zero” or “one.” The remainder are labelled “repeat” causing the machine to run again. It will run until it obtains a definite output.
Q-Sampling	One string is labelled “zero;” one string is labelled “one.” All remaining strings are labelled “repeat” causing the machine to run again. It will run until the register is in a superposition of definite output states.

6.2 Foundational discussion

It is clear from our q-sampling examples, as well as from the Bernoulli factory itself, that a q-sample is frequently more powerful than a classical sample. What can this tell us about quantum states?

Classical probabilistic states do not exist; their descriptions are really descriptions of our own knowledge. It is tempting to view quantum states in a similar manner. It is certainly true that many quantum behaviours seem much more intuitively acceptable under a “state of knowledge” view. Examples include wavefunction collapse and entanglement. Unfortunately the paper by Pusey, Barrett and Rudolph [PBR12] tells us that this is a dangerous line of reasoning, leading to bizarre conclusions such as the impossibility of independent state preparation [[E⁺13,H11,SF14,W13], superdeterminism or even the complete denial of physical reality [PPC].

We also note that the power we obtain from q-samples does not seem consistent with this statistical interpretation. It makes no sense that we can obtain the result

of measurements of the whole distribution if all that were present were a single sample. This idea is explored somewhat further in the conclusion.

The question of what the difference is between a classical sample and a q-sample is really a question of what quantum coherence actually is. Unfortunately this is a question that has troubled some of the finest minds of the twentieth century [EPR35,B04,B35,S35].

Coherence is the ability of a quantum state to interfere. It is tied to the ideas of identical events and indistinguishable particles, both concepts that rarely have an effect on the day-to-day world humans experience. Much as classical mechanics can be regarded as an approximation to relativity which is close to correct in most practical situations, so we can regard classical probability theory as a generally correct approximation to quantum mechanics. The nature of this approximation is to regard probabilities as scalar quantities rather than vectors. Events have both magnitude (probability) and direction (phase) but the latter only becomes relevant when dealing with events of classes which can interfere: identical events. The most simple example being the Hong-Ou-Mandel dip, caused by two events being identical. The Hong-Ou-Mandel dip is also illustrative in that as the events are made less and less similar the quantum effects become less and less pronounced. This means that, like relativity, quantum mechanics runs counter to our intuition and great care must be taken in applying are conventional notions to it.

Conclusions

The broad goal of this work was to find systems similar to quantum Turing machines where clear results could be proven, in the hope that these might lend insight into the power of quantum computing. The Bernoulli factory succeeds in this regard, being a much more tractable system whilst having plenty of traits in common with the quantum algorithms being explored by mainstream quantum computing research. Through the Bernoulli factory and variants around it I believe we have gained insight into what makes quantum computing powerful.

We will briefly discuss a few parallels between quantum algorithms for quantum Turing machines and the results we have observed in this work.

One of the earliest quantum algorithms, the Deutsch-Jozsa algorithm, has an interesting similarity with our results in that its advantage over classical algorithms disappears when approximate answers are allowed. One of the key differences we observed when creating Bernoulli factory algorithms was that the quantum factories are able to achieve certainty in cases where the classical factories cannot. A quantum factory can be certain that $p \neq \frac{1}{2}$; a quantum Turing machine can be certain as to whether a function is balanced or constant.

Shor's factoring algorithm, in a similar manner to the hypothetical graph isomorphism algorithm discussed earlier, relies on q-sampling to achieve its goal. During the key step of the algorithm a large quantum state is created and then, crucially, this state is queried about a global property of the distribution. Classically many samples must be taken from a distribution and then global properties must be inferred; this is not so in the quantum case.

Grover’s algorithm, in its original form, relies heavily on a promise that there is exactly one solution state. Whilst modifications have been made to get around this restriction, these change the algorithm from one which terminates deterministically in finite time to one which terminates probabilistically. Once again it seems that despite quantum mechanics being inherently *uncertain* it is actually *certainty* about one’s inputs or outputs that allows quantum algorithms to supercede their classical counterparts.

It seems that the difference between the quantum and classical states representing distributions is the questions one can ask. For the classical case, the only thing we can do is take a sample. For the quantum case we can sample from the distribution; sample from a related distribution; or ask about global or periodic properties of the distribution. In a sense a quantum state contains all information about the distribution, but only permits, in general, a single question to be asked. The classical state might be described by the distribution but in reality that description is a description of our knowledge, not of the state. I believe a careful examination of Bernoulli factory and Turing machine algorithms offers further evidence to the already mounting body of that against interpreting the quantum state statistically.

Returning to the specific case of the quantum Bernoulli factory we note, having discussed q-sampling and the power of being able to ask questions about a distribution, that this is exactly what we are doing with the *inputs* to the Bernoulli factory. Our quantum coin state contains all information about the simple $Bernoulli(p)$ distribution and thus our power comes from being able to interrogate this state in more ways than we could interrogate a classical state.

We cannot extract the answers to all possible questions about the distribution. Indeed we note that the seemingly simple question “is p greater than $\frac{1}{2}$?” is not available. We can still this simple distribution ask does “ $p = p_0$?” for any p_0 in the quantum case; questions of this form are impossible classically. It is unsurprising then that as we increase the complexity of the distribution the questions we gain access to via quantum mechanics become further separated from the classical ques-

tions. In the case where we had two quantum coins, we showed that we had access to the question “is the bias of these two coins equal?” This question is very far from being answerable in the classical case.

One recurring theme in these questions, as discussed earlier, is that of one-side certainty. When we ask the questions “does $p = p_0$?” or “is the bias of these two coins equal?” we can only obtain a definite answer in one direction. There are measurement results that prove to us that $p \neq p_0$ and results that prove the biases are unequal but there are no results which can prove the opposite. This mirrors the scientific method where we can only find data to be inconsistent with a hypothesis, but no result can ever prove a theory’s truth.

We put forward the very reasonable conclusion, which may seem like no conclusion at all for those who have worked on quantum algorithms for any amount of time: it’s all about asking the right questions. The key to a quantum algorithm is to produce a complex quantum state which contains the information you are looking for and find a way to extract that information.

References

- AA11** Aaronson, Scott, and Alex Arkhipov. "The computational complexity of linear optics." Proceedings of the forty-third annual ACM symposium on Theory of computing. ACM, 2011.
- AA14** Aaronson, Scott, and Alex Arkhipov. "Bosonsampling is far from uniform." Quantum Information & Computation 14.15-16 (2014): 1383-1423.
- A⁺03** Ambainis, Andris, et al. "The quantum communication complexity of sampling." SIAM Journal on Computing 32.6 (2003): 1570-1585.
- A72** Arbib, Michael A. "Theories of abstract automata." (1972).
- A77** Aumann, Robert J. "The St. Petersburg paradox: A discussion of some recent comments." Journal of Economic Theory 14.2 (1977): 443-445.
- A04** Aaronson, Scott. "Is quantum mechanics an island in theoryspace?." arXiv preprint quant-ph/0401062 (2004).
- A10** Aaronson, Scott. "BQP and the polynomial hierarchy." Proceedings of the forty-second ACM symposium on Theory of computing. ACM, 2010.
- A11** Aaronson, Scott. "The equivalence of sampling and searching." International Computer Science Symposium in Russia. Springer Berlin Heidelberg, 2011.
- A13** Aaronson, Scott. Quantum computing since Democritus. Cambridge University Press, 2013.
- A16** Aaronson, Scott. Complexity Zoo website

- APC** Aaronson, Scott. Personal Communication
- AD11** Aaronson, Scott, and Andrew Drucker. "Advice coins for classical and quantum computation." International Colloquium on Automata, Languages, and Programming. Springer Berlin Heidelberg, 2011.
- ADH97** Adleman, Leonard M., Jonathan Demarrais, and Ming-Deh A. Huang. "Quantum computability." SIAM Journal on Computing 26.5 (1997): 1524-1540.
- AGT92** Asmussen, Sren, Peter W. Glynn, and Hermann Thorisson. "Stationarity detection in the initial transient problem." ACM Transactions on Modeling and Computer Simulation (TOMACS) 2.2 (1992): 130-157.
- AKS04** Agrawal, Manindra, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P." Annals of mathematics (2004): 781-793.
- AR05** Aharonov, Dorit, and Oded Regev. "Lattice problems in NP coNP." Journal of the ACM (JACM) 52.5 (2005): 749-765.
- AW09** Aaronson, Scott, and John Watrous. "Closed timelike curves make quantum and classical computing equivalent." Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. Vol. 465. No. 2102. The Royal Society, 2009.
- B15** Babai, Lszl. "Graph isomorphism in quasipolynomial time." arXiv preprint arXiv:1512.03547 (2015).
- B⁺97** Bennett, Charles H., et al. "Strengths and weaknesses of quantum computing." SIAM journal on Computing 26.5 (1997): 1510-1523.
- B⁺06** Beskos, Alexandros, et al. "Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion)." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68.3 (2006): 333-382.

- B35** Bohr, Niels. "Can quantum-mechanical description of physical reality be considered complete?." *Physical review* 48.8 (1935): 696.
- B04** Bell, John Stewart. *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge university press, 2004.
- B87** Babai, Lszi. "Random oracles separate PSPACE from the polynomial-time hierarchy." *Information Processing Letters* 26.1 (1987): 51-53.
- BG81** Bennett, Charles H., and John Gill. "Relative to a Random Oracle A , $P^A NP^A_{co} = NP^A$ with Probability 1." *SIAM Journal on Computing* 10.1 (1981): 96-113.
- BGS75** Baker, Theodore, John Gill, and Robert Solovay. "Relativizations of the $P=?NP$ question." *SIAM Journal on computing* 4.4 (1975): 431-442.
- BHZ87** Boppana, Ravi B., Johan Hastad, and Stathis Zachos. "Does co-NP have short interactive proofs?." *Information Processing Letters* 25.2 (1987): 127-132.
- BJ10** Bremner, Michael J., Richard Jozsa, and Dan J. Shepherd. "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy." *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. The Royal Society, 2010.
- BJV04** Baigneres, Thomas, Pascal Junod, and Serge Vaudenay. "How far can we go beyond linear cryptanalysis?." *International Conference on the Theory and Application of Cryptology and Information Security*. Springer Berlin Heidelberg, 2004.
- BR05** Beskos, Alexandros, and Gareth O. Roberts. "Exact simulation of diffusions." *The Annals of Applied Probability* 15.4 (2005): 2422-2444.
- BR08** Beskos, Alexandros, Omiros Papaspiliopoulos, and Gareth O. Roberts. "A

factorisation of diffusion measure and finite sample path constructions.” *Methodology and Computing in Applied Probability* 10.1 (2008): 85-104.

BV97 Bernstein, Ethan, and Umesh Vazirani. ”Quantum complexity theory.” *SIAM Journal on Computing* 26.5 (1997): 1411-1473.

BW11 Berry, Scott D., and Jingbo B. Wang. ”Two-particle quantum walks: Entanglement and graph isomorphism testing.” *Physical Review A* 83.4 (2011): 042317.

CGH04 Canetti, Ran, Oded Goldreich, and Shai Halevi. ”The random oracle methodology, revisited.” *Journal of the ACM (JACM)* 51.4 (2004): 557-594.

C36 Church, Alonzo. ”A note on the Entscheidungsproblem.” *The journal of symbolic logic* 1.01 (1936): 40-41.

C14 Chandrasekaran, Karthekeyan, and Richard M. Karp. ”Finding a most biased coin with fewest flips.” *COLT*. Vol. 35. 2014.

C71 Cook, Stephen A. ”The complexity of theorem-proving procedures.” *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 1971.

CNW09 Chiang, Chen-Fu, Daniel Nagaj, and Pawel Wocjan. ”Efficient circuits for quantum walks.” *arXiv preprint arXiv:0903.3465* (2009).

DJ92 Deutsch, David, and Richard Jozsa. ”Rapid solution of problems by quantum computation.” *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 439. No. 1907. The Royal Society, 1992.

DJR15 Dale, Howard, David Jennings, and Terry Rudolph. ”Provable quantum advantage in randomness processing.” *Nature communications* 6 (2015).

- D85** Deutsch, David. "Quantum theory, the Church-Turing principle and the universal quantum computer." *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 400. No. 1818. The Royal Society, 1985.
- DT03** Aharonov, Dorit, and Amnon Ta-Shma. "Adiabatic quantum state generation and statistical zero knowledge." *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003.
- DW08** Douglas, Brendan L., and Jingbo B. Wang. "A classical approach to the graph isomorphism problem using quantum walks." *Journal of Physics A: Mathematical and Theoretical* 41.7 (2008): 075303.
- DWM04** Devoret, Michel H., Andreas Wallraff, and John M. Martinis. "Superconducting qubits: A short review." *arXiv preprint cond-mat/0411174* (2004).
- E⁺13** Emerson, Joseph, et al. "The whole is greater than the sum of the parts: on the possibility of purely statistical interpretations of quantum theory." *arXiv preprint arXiv:1312.1345* (2013).
- EPR35** Einstein, Albert, Boris Podolsky, and Nathan Rosen. "Can quantum-mechanical description of physical reality be considered complete?." *Physical review* 47.10 (1935): 777.
- F⁺98** Fenner, Stephen, et al. "Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy." *arXiv preprint quant-ph/9812056* (1998).
- F82** Feynman, Richard P. "Simulating physics with computers." *International journal of theoretical physics* 21.6 (1982): 467-488.
- F01** Fortnow, Lance. "Comparing notions of full derandomization." In *Proceedings of the Sixteenth Annual IEEE Conference on Computational Complexity*. 2001.

- F09** Fortnow, Lance. "A Simple Proof of Toda's Theorem." *Theory OF Computing* 5.1 (2009): 135-140.
- FH12** Flegal, James M., and Radu Herbei. "Exact sampling for intractable probability distributions via a Bernoulli factory." *Electronic Journal of Statistics* 6 (2012): 10-37.
- FR98** Fortnow, Lance, and John Rogers. "Complexity limitations on quantum computation." *Computational Complexity*, 1998. Proceedings. Thirteenth Annual IEEE Conference on. IEEE, 1998.
- FL81** Fraenkel, Aviezri S., and David Lichtenstein. "Computing a perfect strategy for $n \times n$ chess requires time exponential in n ." *International Colloquium on Automata, Languages, and Programming*. Springer Berlin Heidelberg, 1981.
- FS08** Fortnow, Lance, and Rahul Santhanam. "Infeasibility of instance compression and succinct PCPs for NP." *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008.
- FU00** Filippi, Claudia, and C. J. Umrigar. "Correlated sampling in quantum Monte Carlo: A route to forces." *Physical Review B* 61.24 (2000): R16291.
- G97** Grover, Lov K. "Quantum mechanics helps in searching for a needle in a haystack." *Physical review letters* 79.2 (1997): 325.
- G⁺90** Greenberger, Daniel M., et al. "Bells theorem without inequalities." *Am. J. Phys* 58.12 (1990): 1131-1143.
- G⁺10** Gamble, John King, et al. "Two-particle quantum walks applied to the graph isomorphism problem." *Physical Review A* 81.5 (2010): 052313.
- GS66** Ginsburg, Seymour, and Edwin H. Spanier. "Finite-turn pushdown automata." *SIAM Journal on Control* 4.3 (1966): 429-453.
- H⁺10** Hallgren, Sean, et al. "Limitations of quantum coset states for graph isomorphism." *Journal of the ACM (JACM)* 57.6 (2010): 34.

- H11** Hall, Michael JW. "Generalisations of the recent Pusey-Barrett-Rudolph theorem for statistical models of quantum phenomena." arXiv preprint arXiv:1111.6304 (2011).
- HC70** Hellman, Martin E., and Thomas M. Cover. "Learning with finite memory." *The Annals of Mathematical Statistics* (1970): 765-782.
- K98** Knuth, Donald Ervin. *The art of computer programming: sorting and searching*. Vol. 3. Pearson Education, 1998.
- K08** Koshy, Thomas. *Catalan numbers with applications*. Oxford University Press, 2008.
- KOB94** Keane, M. S., and George L. O'Brien. "A Bernoulli factory." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 4.2 (1994): 213-219.
- KSV02** Kitaev, Alexei Yu, Alexander Shen, and Mikhail N. Vyalyi. *Classical and quantum computation*. Vol. 47. Providence: American Mathematical Society, 2002.
- L73** Levin, Leonid A. "Universal sequential search problems." *Problemy Peredachi Informatsii* 9.3 (1973): 115-116.
- LR02** Grover, Lov, and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions." arXiv preprint quant-ph/0208112 (2002).
- L⁺11** Latuszyski, Krzysztof, et al. "Simulating events of unknown probabilities via reverse time martingales." *Random Structures and Algorithms* 38.4 (2011): 441-452.
- L⁺10** Ladd, Thaddeus D., et al. "Quantum computers." *Nature* 464.7285 (2010): 45-53.

- M81** McKay, Brendan D. Practical graph isomorphism. Tennessee, US: Department of Computer Science, Vanderbilt University, 1981.
- ML72** Meyer, Albert R., and Larry J. Stockmeyer. "The equivalence problem for regular expressions with squaring requires exponential space." SWAT (FOCS). 1972.
- MP05** Mossel, Elchanan, and Yuval Peres. "New coins from old: computing with unknown bias." *Combinatorica* 25.6 (2005): 707-724.
- NO01** Nishimura, Harumichi, and Masanao Ozawa. "Quantum subroutine problem and the robustness of quantum complexity classes." arXiv preprint quant-ph/0107089 (2001).
- NP05** Nacu, erban, and Yuval Peres. "Fast simulation of new coins from old." *The Annals of Applied Probability* 15.1A (2005): 93-115.
- NSS58** Newell, Allen, John Calman Shaw, and Herbert Alexander Simon. "Chess-playing programs and the problem of complexity." *IBM Journal of Research and Development* 2.4 (1958): 320-335.
- OFV09** O'Brien, Jeremy L., Akira Furusawa, and Jelena Vukovi. "Photonic quantum technologies." *Nature Photonics* 3.12 (2009): 687-695.
- P03** Papadimitriou, Christos H. Computational complexity. John Wiley and Sons Ltd., 2003.
- PBR12** Pusey, Matthew F., Jonathan Barrett, and Terry Rudolph. "On the reality of the quantum state." *Nature Physics* 8.6 (2012): 475-478.
- PMO09** Politi, Alberto, Jonathan CF Matthews, and Jeremy L. O'brien. "Shors quantum factoring algorithm on a photonic chip." *Science* 325.5945 (2009): 1221-1221.
- PPC** Pusey, Matthew F. Personal Communication

- P⁺08** Politi, Alberto, et al. "Silica-on-silicon waveguide quantum circuits." *Science* 320.5876 (2008): 646-649.
- P⁺12** Peruzzo, Alberto, et al. "A quantum delayed-choice experiment." *Science* 338.6107 (2012): 634-637.
- RNB12** Reitzner, Daniel, Daniel Nagaj, and Vladimir Buzek. "Quantum walks." arXiv preprint arXiv:1207.7283 (2012).
- RF88** Royden, Halsey Lawrence, and Patrick Fitzpatrick. *Real analysis*. Vol. 198. No. 8. New York: Macmillan, 1988.
- RS59** Rabin, Michael O., and Dana Scott. "Finite automata and their decision problems." *IBM journal of research and development* 3.2 (1959): 114-125.
- S09** Sakarovitch, Jacques. *Elements of automata theory*. Cambridge University Press, 2009.
- S48** Stone, Marshall H. "The generalized Weierstrass approximation theorem." *Mathematics Magazine* 21.5 (1948): 237-254.
- S94** Shor, Peter W. "Algorithms for quantum computation: Discrete logarithms and factoring." *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE, 1994.
- SF14** Schlosshauer, Maximilian, and Arthur Fine. "No-go theorem for the composition of quantum systems." *Physical review letters* 112.7 (2014): 070407.
- T91** Toda, Seinosuke. "PP is as hard as the polynomial-time hierarchy." *SIAM Journal on Computing* 20.5 (1991): 865-877.
- T36** Turing, Alan Mathison. "On computable numbers, with an application to the Entscheidungsproblem." *J. of Math* 58.345-363 (1936): 5.
- TB73** Trakhtenbrot, Boris A., and Ya M. Barzdin. *Finite automata*. American Elsevier Publishing Company, 1973.

- TB11** Thomas, Andrew C., and Jose H. Blanchet. "A practical implementation of the Bernoulli factory." arXiv preprint arXiv:1106.2508 (2011).
- S06** Sipser, Michael. Introduction to the Theory of Computation. Vol. 2. Boston: Thomson Course Technology, 2006.
- S35** Schrdinger, Erwin. "Discussion of probability relations between separated systems." Mathematical Proceedings of the Cambridge Philosophical Society. Vol. 31. No. 04. Cambridge University Press, 1935.
- SPC** Speyer, David. Personal Communication
- V79** Valiant, Leslie G. "The complexity of computing the permanent." Theoretical computer science 8.2 (1979): 189-201.
- V01** Valiant, Leslie G. "Quantum computers that can be simulated classically in polynomial time." Proceedings of the thirty-third annual ACM symposium on Theory of computing. ACM, 2001.
- VN51** Von Neumann, John. "13. Various Techniques Used in Connection With Random Digits." (1951).
- V06** Vukovi, Vladan. "The Theoretical and Practical Application of the Advanced Chess Algorithms." PhD Theses
- W13** Wallden, Petros. "Distinguishing initial state-vectors from each other in histories formulations and the PBR argument." Foundations of Physics 43.12 (2013): 1502-1525.
- W99** Wstlund, Johan. Functions arising by coin flipping. Technical Report, KTH, Stockholm, 1999.
- WW12** Wilmott, Colin M., and Peter R. Wild. "On a generalized quantum SWAP gate." International Journal of Quantum Information 10.03 (2012): 1250034. (2006).

- Y16** Theodore J. Yoder "Building and bounding quantum Bernoulli factories" Unpublished dissertation. Massachusetts Institute of Technology
- Y⁺16** Yuan, Xiao, et al. "Experimental Quantum Randomness Processing Using Superconducting Qubits." Physical Review Letters 117.1 (2016): 010502.