Benchmarking, Verifying and Utilising Near Term Quantum Technology

Daniel James Mills

Doctor of Philosophy
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh

2021

Abstract

Quantum computers can, in theory, impressively reduce the time required to solve many pertinent problems. Such problems are found in applications as diverse as cryptography, machine learning and chemistry, to name a few. However, in practice the set of problems which can be solved depends on the amount and quality of the quantum resources available. With the addition of more qubits, improvements in noise levels, the development of quantum networks, and so on, comes more computing power. Motivated by the desire to measure the power of these devices as their capabilities change, this thesis explores the verification, characterisation and benchmarking techniques that are appropriate at each stage of development. We study the techniques that become available with each advance, and the ways that such techniques can be used to guide further development of quantum devices and their control software. Our focus is on advancements towards the first example of practical certifiable quantum computational supremacy; when a quantum computer demonstrably outperforms all classical computers at a task of practical concern. Doing so allows us to look a little beyond recent demonstrations of quantum computational supremacy for its own sake.

Systems consisting of only a few noisy qubits can be simulated by a classical computer. While this reduces the applicability of quantum technology of this size, we first provide a methodology for using classical simulations to guide progress towards demonstrations of quantum computational supremacy. Using measurements of the noise levels present in the NQIT Q20:20 device, an ion-trap based quantum computer, we use classical simulations to predict and prepare for the performance of larger devices with similar characteristics. We identify the noise sources that are the most impactful, and simulate the effectiveness of approaches to mitigating them.

As quantum technology advances, classically simulating it becomes increasingly resource intensive. However, simulations remain useful as a point of comparison against which to benchmark the performance of quantum devices. For so called ‘random quantum circuits’, such benchmarking techniques have been developed to support claims of demonstrations of quantum computational supremacy. To give better indications of the device’s performance in practice, instances of computations derived for practical applications have been used to benchmark devices. Our second contribution is to introduce a suite of circuits derived from structures that are common to many instances of computations derived for practical applications, contrasting with the aforementioned approach of using a collection of particular instances. This allows us to make broadly applicable predictions of performance, which are indicative of the device’s behaviour when investigating applications of concern. We use this suite to benchmark all layers of the quantum computing stack, exploring the interplay between the compilation strategy, device, and the computation itself. The circuit structures in the suite are sufficiently diverse to provide insights into the noise channels present in several real devices, and into the applications for which each quantum computing stack is best suited. We consider several figures of merit by which to assess performance when implementing these circuits, taking care to minimise the required number of uses of the quantum device.

As our third contribution, we consider benchmarking devices performing Instantaneous Quantum Polynomial time (IQP) computations; a subset of all the computations quantum computers are capable of performing in polynomial time. By using only a commuting gate set, IQP circuits do not require the development of a universal quantum computer, but are still thought impossible to simulate efficiently on a classical computer. Utilising a small quantum network, which allows for the transmission of single qubits, we introduce an approach to benchmarking the performance of devices capable of implementing IQP computations. As the resource consumption of our benchmarking technique grows reasonably as the size of the device grows, it enables us to benchmark IQP capable devices when they are of sufficient size to demonstrate quantum computational supremacy, and indeed to certify demonstrations of quantum computational supremacy. The approach we introduce is constructed by concealing some secret structure within an IQP computation. This structure can be taken advantage of by a quantum computer, but not by a classical one, in order to prove it is capable of accurately implementing IQP circuits. To achieve this we derive an implementation of IQP circuits which keeps the computation, and as a result the structure introduced, hidden from the device being tested. We prove this implementation to be information-theoretically and composably secure.

In the work described above we explore verification, characterisation and benchmarking of quantum technology both as it advances to demonstrations of quantum computational supremacy, and when it is applied to real world problems. Finally, we consider demonstrations of quantum computational supremacy with an instance of these real world problems. We consider quantum machine learning, and generative modelling in particular. Generative modelling is the task of producing new samples from a distribution, given a collection of samples from that distribution. We introduce and define 'quantum learning supremacy', which captures our intuitive notion of a demonstration of quantum computational supremacy in this setting, and allows us to speak formally about generative modelling tasks that can be completed by quantum, but not classical, computers. We introduce the Quantum Circuit Ising Born Machine (QCIBM), which consists of a parametrised quantum circuit and a classical optimisation loop to train the parameters, as a route to demonstrating quantum learning supremacy. We adapt results that exist for IQP circuits in order to argue that the QCIBM might indeed be used to demonstrate quantum learning supremacy. We discuss training procedures for the QCIBM, and Quantum Circuit Born Machines generally, and their implications on demonstrations of quantum learning supremacy.

Lay Summary

In recent decades, computers have become ubiquitous in all walks of life. The computers which are prevalent around us today, which we call 'classical computers' are used for computations as disparate as performing basic arithmetic and predicting the weather. 'Quantum computers' are an emerging technology which, unlike classical computers, make use of the effects of quantum mechanics to perform computation. In principle doing so allows quantum computers to quickly perform computations that would take a classical computer millennia .

However, as with classical computers, the computations that a quantum computer can perform might be limited by the amount of data that it can process, and by corrupting factors such as errors. Indeed the quantum computers that exist now are extremely prone to errors, and are very limited in the amount of data they can process. In this thesis we consider what these small and noisy quantum computers can be used for, and how to assess their performance. Our contributions can be divided as follows:

Chapter 2: We suppose we know how well individual components that make up a quantum computer work, but not how well they would work together. We use classical computers to simulate how well a bigger quantum computer, built up from these small components, would work. We use these simulations to determine how to improve the individual components, and how best to utilise and combine them, so that the performance of the larger device is improved.

Chapter 3: We suppose that the larger quantum computers simulated in Chapter 2 have been built, and explore how to assess their performance directly. Indeed, we consider a few such devices which are available today. We specify computations which measure how well these devices would perform when used for pertinent applications. This uses classical simulation, but as a point of comparison rather than as a predictive tool as in Chapter 2. We determine which of the quantum computers considered are best suited for which application, and which properties of those quantum computers lead to better performance.

Chapter 4: We consider devices that are so large that we cannot use classical simulation to check how well they perform. Instead we present a particular computation that can be run on a quantum computer in order to measure its capabilities. This computation could not be performed by any classical computer, and so can be used to show that quantum computers are more powerful than classical computers. However the computation does not require a large, noiseless quantum computer.

Chapter 5: Finally, as in Chapter 4, we consider quantum computers which can outperform classical computers, but which are still quite small and noisy. We explore the ways in which to use these devices for practical applications; in particular by designing a machine learning algorithm using such quantum computers. We argue that there are some features of a collection of data which could be learnt by this algorithm, but not any algorithm running on a classical computer.

Acknowledgements

"If I have seen further it is by standing on the shoulders of Giants."
— Isaac Newton

This thesis is the product of a great many years of lucky coincidences, flukes of opportunity, and blessed circumstances. If it is by fortuity that this thesis was written, then my greatest fortune of all was to have crossed paths with those individuals and institutions who contributed to this work. It is my great honour to credit them here.

This thesis was completed under the patient, enthusiastic, and insightful supervision of Elham Kashefi. I will be ever grateful to her for introducing me to the topics covered in this work, and for the opportunity and freedom to explore them. Many of the individuals who enriched my understanding of these topics were introduced to me by Elham, and for this I am thankful too.

Indeed I have consistently been able to rely on the encouragement, wisdom and guidance of inspiring teachers throughout my life. Many thanks to Petros Wallden, Anna Pappa, Theo Kapourniotis, and Andru Gheorghiu for your mentorship while at The Univeristy of Edinburgh. Thanks to Chris Heunen and Animesh Datta who, though their careful examination of this thesis, contributed significantly. Thanks to Ross Duncan and Cyril Allouche for sharing their interesting ideas, and for instilling in me a commitment to support the widespread adoption of quantum computing. To Roger Tribe, Oleg Zaboronski and Oliver Hambrey I owe my first introduction to academic research, and memories of your awe inspiring intellect and calm determination continue to motivate me. A particular thanks to Tim Hardingham whose early moral and intellectual guidance provided the momentum which propels me to this day.

I owe much of my understanding of the topics covered in this thesis to long lunch and coffee breaks with my colleges and friend in The University of Edinburgh's quantum information group. I am beholden to Alex Cojocaru, my companion form the outset, for his support, and to Brian Coyle, who under more formal circumstance it would behove me to credit as the editor of this thesis. To Atul, Ellen, Ieva, Iskren, Mahshid, Matty, Mina, Myrto, Niraj and Rawad, I say it was the greatest pleasure to spend part of my time in Edinburgh with you. Thanks to Pierre, Marc, Ulysse, Damian, Elani, Dominik, Leo, Luka, Sean and Tom for your kindness when I visited Paris. Thanks to Bertrand, Thomas, and Simon from Atos for a bountiful supply of expertise in classical simulation of quantum computers. Thanks to Seyon, Silas, Alex, Will, Alec and Travis for sharing your knowledge of the practical utility of quantum computers with me during my first stint at CQC.

This thesis was completed under the auspices of The University of Edinburgh and the School of Informatics. I am eternally grateful to these organisations, and for the support of their staff. Many thanks to Murray Cole, director of the Centre for Doctoral Training in Pervasive Parallelism, for his intellectual and careers guidance, and to all of the staff of the PPar CDT.

Besides this intellectual and practical guidance, this thesis would not have been possible without the ever tolerant support of my friends and family. Notably I would like to thank Paul, Caoimhín, Phil, Jakub, Floyd, Raj, Daniel, Amna, Reese and Vanya for 5 years of great memories, as well as Andrew, Alex and Luke for the great pleasure of our time spent living together. Thanks to my high school friends; with whom I shared many hours of deep philosophical and scientific discussions, before we were of sufficient training to have any right to do so. To those at The University of Warwick, with whom I spent late nights working, conversing and looking expectantly forward, I thank you for sharing my foundational years with me.

Thanks too to Joanne Mills, my closest confidant, ever faithful accomplice, and sister, who without wavering endured the best and worst of me for the whole of my life. It was also essential to the completion of this work, that at its outset I met the love of my life, and wife if not for the pandemic, Martyna Panasiuk, whose reassuring encouragement was indispensable. With a lack of originality, but the greatest of sincerity, I give the final thanks to my mother and father. To a budding scholar perhaps the greatest gift of all is 18 years of tranquil upbringing, gentle encouragement and perpetual support. Of all the luck that was bestowed upon me, it is to the greatest extent that I cannot imagine how I could have reached this point without that.

This collection of names is long, but incomplete. Indeed, throughout my life I've had the fortune to be continually surrounded by persons either or both supportive of me personally, and passionate about scholarship. This fortune is so great that an observer might suppose it is through sheer force of luck, circumstances, and an aligning of the stars that this thesis was written. I am compelled to acknowledge that they'd be right. There is no justice in this fortune being bestowed on so few.

This work was supported by the Engineering and Physical Sciences Research Council (grant EP/L01503X/1), EPSRC Centre for Doctoral Training in Pervasive Parallelism at the University of Edinburgh, School of Informatics.

Publications and Contributions

The publications on which this thesis is based, and the authors who contributed to them, are as follows:

Chapter 2 is based on:

[1] - Iskren Vankov, Daniel Mills, Petros Wallden, and Elham Kashefi. "Methods for classically simulating noisy networked quantum architectures". In: Quantum Science and Technology *5.1 (Nov. 2019), p. 014001.* DOI*:* `10.1088/2058-9565/ab54a4`

Chapter 3 is based on:

[2] - Daniel Mills, Seyon Sivarajah, Travis L. Scholten, and Ross Duncan. "Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack". In: Quantum *5 (Mar. 2021), p. 415.* ISSN*: 2521-327X.* DOI*:* `10.22331/q-2021-03-22-415`

Chapter 4 is based on:

[3] - Daniel Mills, Anna Pappa, Theodoros Kapourniotis, and Elham Kashefi. "Information Theoretically Secure Hypothesis Test for Temporally Unstructured Quantum Computation". In: Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017. *Vol. 266. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2018, pp. 209–221.* DOI*:* `10.4204/EPTCS.266.14`

Chapter 5 is based on:

[4] - Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. "The Born supremacy: quantum advantage and training of an Ising Born machine". In: npj Quantum Information *6.1 (July 2020), p. 60.* ISSN*: 2056-6387.* DOI*:* `10.1038/s41534-020-00288-9`

List of Acronyms

Acronym	Meaning	Reference
BB84	Bennett Brassard 1984. Quantum key distribution scheme.	Section 1.7 & Ref.[5]
BQC	Blind Quantum Computing.	Section 1.6.1
CE	Cross-Entropy.	Definition 1.6.3
CED	Cross-Entropy Difference.	Definition 1.6.4
CHSH	Clauser–Horne–Shimony–Holt. Inequality.	Section 1.6.2 & Ref.[6]
CPTP	Completely Positive Trace Preserving.	Section 1.3.3
CPU	Classical Processing Unit.	
CQC	Cambridge Quantum Computing. Company.	
DDH	Decisional Diffie-Hellman. Assumption.	Section 5.3 & Ref.[7]
DQC	Delegated Quantum Computing.	Section 1.6
EPR	Einstein–Podolsky–Rosen. Quantum state.	Equation (1.5)
E91	Ekert 1991. Quantum key distribution scheme.	Section 1.7 & Ref.[8]
FLOPS	Floating Point Operations Per Second.	
HHL	Harrow-Hassidim-Lloyd. Algorithm.	Chapter 5 & Ref.[9]
HOG	Heavy Output Generation.	Section 1.6.4
IBM	International Business Machines Corporation. Company.	
IQP-MBQC	Protocol for implementing IQP circuits in MBQC.	Protocol 1.5.2
KL	Kullback–Leibler(-divergence).	Equation (1.27)
LINPACK	As in the benchmark set.	Chapter 3 & Ref.[10]
LWE	Learning With Errors.	Section 1.6 & Ref.[11]
MBQC	Measurement Based Quantum Computing.	Section 1.4
MMD	Maximum Mean Discrepancy.	Equation (5.5)
NISQ	Noisy Intermediate Scale Quantum.	Chapter 0 & Ref.[12]
NQIT	Networked Quantum Information Technology. Research collaboration.	Section 2.2 & Ref.[13]

Acronym	Meaning	Reference
NQIT Q20:20	The quantum processor developed by the NQIT hub.	Section 2.2.2 & Section 2.2.3
OT	Optimal Transport.	Equation (5.8)
PQC	Parametrised Quantum Circuit.	Chapter 5 & Ref.[14]
PRF	Pseudo Random Function.	Section 5.3 & Ref.[15]
QAOA	Quantum Approximate Optimisation Algorithm.	Chapter 5 & Ref.[16]
QCBM	Quantum Circuit Born Machine.	Section 5.2
QCIBM	Quantum Circuit Ising Born Machine.	Section 5.2.1
QCVV	Quantum Characterisation Validation and Verification.	Section 1.6
QKD	Quantum Key Distribution.	Section 1.7 & Ref.[5, 8]
QML	Quantum Machine Learning.	Chapter 5
QPU	Quantum Processing Unit.	
QSQ	Quantum Statistical Query.	Section 5.3 & Ref.[17]
QUATH	QUAntum THreshold. Assumption.	Assumption 1
RB	Randomised Benchmarking.	Appendix B.3.2 & Ref.[18, 19]
RSA	Rivest–Shamir–Adleman. Cryptosystem.	Chapter 0 & Ref.[20]
RSP	Remote State Preparation.	Section 1.6.1 & Ref.[21]
SHD	Sinkhorn-Divergence.	Equation (5.7)
UBQC	Universal Blind Quantum Computing.	Section 1.6.1
VQE	Variational Quantum Eigensolver.	Section 3.2.3 & Ref.[22, 23]
VUBQC	Verifiable Universal Blind Quantum Computing.	Section 1.6.2
XQUATH	Cross-Entropy QUAntum THreshold. Assumption.	Assumption 2
2D-DQS	2-Dimensional Dynamical Quantum Simulators.	Protocol 1.5.1 & Ref.[24]

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Daniel James Mills*)

October 26, 2021

To my Mother and Father.

Contents

Chapter 0

Introduction

Outside of the size and speed regimes of our everyday experience, many surprising behaviours emerge. This is true of the behaviour of atoms and subatomic particles, which, during the first half of the 20th century, quantum mechanics was developed to explain. The theory arose from a series of discoveries in the early part of the century, notable amongst which are a solution to the black-body radiation problem by Max Planck in 1900, and an explanation of the photoelectric effect by Albert Einstein in 1905. A flurry of further discoveries followed [25].

Those physical theories that predate quantum mechanics and the theories of relativity, referred to collectively as 'classical mechanics', well describe slow moving macroscopic objects. While these theories fall short of making accurate predictions in small size regimes, they explain most familiar everyday phenomena. In contrast, several predictions of quantum mechanics were, and remain, unfamiliar and counter-intuitive. Phenomena such as quantum tunnelling, which allows a particle to pass though a barrier, or quantum entanglement, which Einstein famously referred to as "spooky action at a distance", defied expectations at the time. The 'measurement problem', which arises from the probabilistic, observer dependent, nature of measurements in quantum theory, eludes explanation even today. Despite the surprising nature of the predictions of quantum mechanics, many have been experimentally verified. This is so much the case that there are now everyday technologies, such as lasers and semi-conductor devices, which rely on uniquely quantum mechanical phenomena.

In the 1980s it was proposed that the quantum mechanical properties of physical systems could be used to reduce the resource costs of performing some computations. This is to say, for example, that the time taken to perform a computation may be reduced if a 'quantum computer' could be used, as compared to the time taken by a 'classical computer'. Initially this idea was inspired by the apparently natural application of quantum computers to the problem of simulating physical systems [26]. The laws of quantum mechanics apply to such systems, and their classical simulation proves extremely difficult. Indeed it was a seminal result in the field which showed that such simulations can be achieved on a quantum computer [27]. Since this initial proposal, it has become apparent that the simulation of physical systems is only one of many applications of quantum computers [28, 29].

A digital quantum computer, referred to in this thesis as simply a quantum computer, can be formalised as acting a sequence of ‘quantum gates’, which are local unitary transformations, on ‘qubits’, which are 2 level quantum systems [30]. This model of computation is sufficient to simulate a ‘Turing machine’, the archetypal model of classical computing, but also to simulate the evolution of physical systems [27, 31, 32] as mentioned above. Importantly, this formalism enabled the discovery of many other domains of interest to which quantum computers may be fruitfully applied. To name but a few, these include: cryptography [5, 8, 33], search and optimisation [16, 34–36], linear algebra [9], and random walks [37].

One particularly important example application of quantum computing is through Shor’s algorithm for prime factorisation [33]. The problem solved by Shor’s algorithm is: given the product of two primes $N = p \times q$, to find p and q. These primes can be found by running Shor’s algorithm on a quantum computer, with the number of time steps required significantly reduced as compared to the best known classical algorithm [38]. The security of the widely used RSA encryption scheme relies on factoring being hard [20], implying that it is not secure in the presence of quantum computers. Because of the ubiquitous use of RSA, Shor’s algorithm has come to epitomise the pertinence of quantum computers.

Quantum Computational Supremacy

At present, the advantage over classical approaches of the aforementioned applications of quantum computers is predominately justified by theoretical performance predictions, rather than practical demonstrations of this advantage. Pragmatically, the goal is to experimentally demonstrate one gains an advantage in solving a set of problems by using a device which utilises quantum mechanics. While what this entails is somewhat subjective, in this thesis we will say that a given device demonstrates *quantum computational supremacy*[1] [39, 40] by disproving the following hypothesis:

For any problem, there is a classical machine performing as well or better at solving the problem than the given device.

As this advantage is measured relative to solving the same set of problems using any available purely classical machine, it is implicitly ensured that a device demonstrating quantum computational supremacy utilises some uniquely quantum phenomena. In this thesis we will almost always measure this advantage by comparing the number of computing steps required to complete a computation.[2]

From this definition, one can extract two conditions that need to be fulfilled in order to demonstrate quantum computational supremacy.

1. Provide a theoretical proof that some quantum algorithm outperforms the best possible classical algorithm for the same task.

[1]Many other terms are regularly used interchangeably with quantum computational supremacy. In particular variations upon “quantum-advantage” or “quantum-superiority” are also popular.

[2]In practice other measures such as the energy consumption may be of concern.

2. Provide experimental evidence of such an advantage via an implementation of this algorithm.

The first condition ensures that a demonstration of quantum computational supremacy is not just a result of asymmetric technological advancement. This is to say, it avoids the situation where quantum computers outperform classical computers at some fixed time, but classical computers could always perform equally as well as quantum computers after the development of new classical technology. This condition will not be of concern from a purely practical point of view, as users will endeavour to solve problems by whichever means is the most efficient at the time, regardless of the technology that may be available in the future. The latter condition ensures that there are no unforeseen hurdles to exploiting the advantage that quantum computers provide. This includes both technological hurdles, and hitherto misunderstood physical laws limiting the precise control of quantum systems.

As mentioned, Shor's algorithm is one example of a quantum algorithm which solves a problem, in this case prime factorisation, in fewer time steps than the best know classical algorithm for solving the same problem. However it fails on both counts as a means to demonstrate quantum computational supremacy. This is respectively because the classical hardness of factoring is poorly understood, and because the resource requirements of Shor's algorithm are significantly beyond that which is available with current realisations of quantum computers [41]. Similarly there remain hurdles to the use of the simulation of physical systems as a means to demonstrate quantum computational supremacy. Firstly, as with Shor's algorithm, the classical hardness of performing such simulations is poorly understood at present. Secondly, and perhaps more philosophically, it appears undesirable to claim a demonstration of quantum computational supremacy was performed by some large molecule simulating its own evolutions. For this reason, it is often taken that a device demonstrating quantum computational supremacy should be programmable or controllable in some way. While we leave 'programmability' somewhat poorly defined, we will take it to mean that the device may be used for a wide selection of computations, as determined by a user.

Noisy Intermediate Scale Quantum Technology

Many possible realisations of a quantum computer are known [42] and include, for example, photonic quantum computers [43, 44], superconducting quantum computers [45], and quantum computers based on trapped ions [46, 47]. Each implementation has its own advantages and disadvantages, and a popular measure of the quality of a realisation of a quantum computer is via a comparison to *DiVincenzo's criteria* [48]. Loosely speaking this amounts to requiring that: 'well-characterised' qubits can be added to the system at not too great a cost; there is available a finite 'universal' set of gates which are collectively sufficient to perform any computation that is possible within the quantum computing model; quantum states can be initialised and measured; and errors are not too impactful, or can be corrected.[3]

[3]DiVincenzo's criteria also includes two criteria that are necessary for quantum communication. These are namely that qubits can be converted between stationary and 'flying' qubits, and that flying qubits can be moved between locations.

At present quantum computing devices meet these criteria to only a limited degree. Indeed, the quantum computers available at present are of insufficient size, measured in number of qubits, to implement many of the most impactful protocols. Adding sufficiently many qubits appears challenging. Besides being few in number (less than 100), qubits in existing technology are also susceptible to 'noise' which can interfere with a computation being performed. This noise perturbs the quantum state, and results from interactions between the quantum device and its environment. This interaction breaks the ideal assumption that a quantum computer can be affected by a programmer, and nothing else.

In general, the *threshold theorem for quantum computation* states that, provided the noise levels in the quantum computation can be reduced below a constant value, [4] error correcting codes [5] can reduce the error arbitrarily [57]. Error correction codes allow for computations to be performed on *logical qubits*, which are often built by redundantly storing information on many *physical qubits*. However, implementing error correction codes of the form demanded by the threshold theorem may require several thousand qubits [54]. This is unfeasible with current technology, possibly raising the cost of Shor's algorithm, and other pertinent protocols, to a few million qubits [58, 59].

However, the difference between the resources required to perform a demonstration of quantum computational supremacy, and those that are required to perform error-corrected quantum computation, appears to be very large [60]. When qubits are noisy, and behave in a poorly controlled manner, they can still, for example, be used to perform certain types of random number generation. These 'sampling problems' are thought not to be accessible to classical computers [60]. Those noisy quantum devices which are not large enough to implement fault-tolerant quantum computations, but which are large enough to demonstrate quantum computational supremacy, are called the Noisy Intermediate Scale Quantum (NISQ) devices [12]. The term NISQ focuses on hardware, but we may use the term 'NISQ-algorithm', or 'near-term application' to refer to protocols that may run on NISQ hardware.

There have been several proposals of sampling problems which could demonstrate quantum computational supremacy using physical architectures with a significantly reduced resource cost as compared to error corrected quantum computers [61–63]. Amongst these proposals are Instantaneous Quantum Poly-time (IQP) circuits [63], which are a subset of all quantum circuits. This subset is namely those circuits that are constructed only from gates that commute with each other. This has several practical advantages, including: a reduced gate set, which may be technically easier to implement; and potentially improved circuit optimisation capabilities as a result of the commuting gate set. Because of these advantages, and others, IQP circuits are of

[4]Presently there appears to be a gap between the error rates needed to achieve quantum computational supremacy and those needed to facilitate fault-tolerance. There are conflicting views on whether this gap is insurmountable [49, 50].

[5]The first error correction codes where developed by Shor [51] and Steane [52] in the 1990s. In the intervening period the problem has attracted much attention, with *surface codes* being amongst the most promising developments [53]. This thesis is concerned with quantum computers that have too few qubits with too high noise levels to perform significant error correction, and so we do not review progress in the area. We direct the reader to several elegantly constructed reviews [53–56].

particular concern in this thesis.

While the experimental realisation of these sampling problems may be within reach, the theoretical proofs that they outperform classical computers depend on conjectures about the computational hardness of certain computations. However, these conjectures are certainly more believable than the rather circumstantial evidence that factoring should be hard; based largely on no fast classical algorithm having been found. Interestingly, these theoretical results extend to give guarantees about the difficulty that a classical computer would have in reproducing the outputs of a quantum computer subject to some limited forms of noise [61, 64–66]. This further aligns these sampling problems to NISQ technology.

Besides their use as a means to demonstrate quantum computational supremacy, it is also of interest to study how NISQ technology can be used for practical applications [67, 68]. It is common, although not always the case, that protocols run on NISQ technology are of the variational kind [67]. The general approach of variational algorithms is to update the parameters of a Parametrised Quantum Circuit (PQC) until the outputs of the circuit minimise some 'cost function'. This cost function may indicate how close the output state is to the ground state of some Hamiltonian, or how close the distribution of outputs is to a desired one. Typically the cost function is calculated by a classical computer, which minimises the number of operations that need to be performed by the quantum computer, and aligns the approach with NISQ technology.

Amongst the first variational quantum algorithm were Variational Quantum Eigensolvers (VQE) [22, 23], which are useful in, for example, quantum chemistry, and the Quantum Approximate Optimisation Algorithm (QAOA) [16] for solving combinatorial optimisation problems. Outside of chemistry and combinatorial optimisation, variational quantum algorithms have been used to: solve numerical problems such as factoring [69]; solve problems in finance such a portfolio optimisation [70]; and explore the prospects for machine learning [14, 71]. In each case the components of the variational algorithm will vary. In particular the choice of the form of the PQC will depend on the problem, and possibly on the quantum device. As such some devices will perform better at some applications.

Quantum Characterisation Verification and Validation

Note that the requirement to provide experimental evidence as part of a demonstration of quantum computational supremacy is a multi-faceted one. While we have discussed the technological restrictions to doing so, verification of such a demonstration provides further theoretical challenges. Explicitly, we would require that the solution to a problem arrived at by a quantum computer could be checked for its correctness by a classical computer. Shor's algorithm lends itself well to this particular problem as the factors p and q can be multiplied together to check that they do indeed constitute factors of N. This gives an approach to verifying the solution to a very particular problem solved on a quantum computer. However, in general it is not clear that the solutions to a computation which can only be arrived at by a quantum computer could be verified by a classical computer. By construction, repeating the computation on a classical computer in order to check by comparison the solution arrived at by a quantum computer would not be possible.

In the case of large universal error-corrected quantum computers, there exist schemes to verify the correct implementation of any quantum computation [72–80]. Instead of checking the solution directly, these schemes work by checking that each step of the computation is implemented faithfully. These rely on the 'Client', who wishes to check for the correct implementation of a computation by the 'Server', either: having access to a small quantum computer [72–76, 80], being able to play two servers off against each other [78, 79], or being able to safely make assumptions about the computational hardness of some particular problems [77]. Unfortunately, these approaches to verification of universal quantum computation are beyond the reach of NISQ devices, requiring many more qubits, and much lower noise levels, than are currently available.

At the other extreme, for very small systems consisting of only a few qubits, Quantum Characterization, Validation, and Verification (QCVV) protocols focus on properties of one or several qubits, and quantify their exposure to noise [81]. For example, procedures such as randomised benchmarking [18, 19] and gate set tomography [82] provide insights into the error rates of gates. This information is highly valuable, but, taken alone, provides limited insights into a device's practical performance. Indeed, as quantum computers evolve from bespoke laboratory experiments comprising a handful of qubits, to more general-purpose, programmable, commercial-grade systems [83, 84], and on towards large devices that might demonstrate quantum computational supremacy, new techniques for characterising them are needed at each step.

Summary of Thesis

As the above discussion highlights, the problems to which quantum computing can be applied depends heavily on how advanced the underlying quantum technology is. Similarly, the QCVV protocols that are accessible, and desirable, vary as technology advances. In this spirit it is the stated goal of this thesis to:

Explore the QCVV protocols that are appropriate as quantum technology develops. In doing so, understand which applications of quantum technology are the most fruitful at each stage of the progress of technology.

The particular domains of interest of each of the chapters of this thesis are compared here and summarised in Table 1. With the exception of Chapter 1, which contains some preliminary material, and should be read first, the chapters of this thesis can be read independently. They are however ordered so that the technological requirements of the schemes proposed within each chapter are roughly increasing throughout this thesis.

Chapter 2 Starting with early stage quantum technology, we consider the utility of the properties uncovered by QCVV protocols run on very small quantum computers. The properties we use include information about the noise levels of individual qubits and gates. While instructive, this information reveals little about the performance of the device as a whole, when implementing computations, or as the size of the device increases. This makes it hard to prioritise further technological advancements, as the impact of an advancement, say in the area of one of DiVincenzo's criteria, on practical performance is unclear.

In Chapter 2 we propose a methodology for incorporating these noise level measurements into classical simulations of larger devices with similar noise properties. By taking measurements of the noise levels present on existing small devices, in this case the NQIT Q20:20 device, we can predict and prepare for the performance of these devices as they grow in size. In particular, we will simulate the behaviour of IQP circuits that are similar in form, but smaller in size, to those that might be used in a demonstration of quantum computational supremacy. By varying the noise levels used within these simulations we can identify dominant noise sources, and suggest the most urgent improvements that should be made. We propose approaches to making the improvements identified, and use our methodology to evaluate these approaches.

Chapter 3 As larger quantum computers are developed, using classical simulation to predict the performance of future developments becomes very resource intensive, and eventually impossible. In this domain we instead consider the use of classical simulation as a point of comparison with quantum technology, rather than as a means of predicting the performance of larger devices, as we did in Chapter 2. In Chapter 3 we introduce a suite of three classes of circuits to be used during such a benchmarking of a device. Assessing the performance of a device at implementing these circuits requires they be run on the quantum device, and that certain properties of the ideal output from the device be calculated classically. The resource requirements of the classical calculation of these properties scale poorly, and this approach to benchmarking would not be possible for arbitrarily large devices. However by demonstrating that the circuits we select have a particular distribution of output probabilities, we ensure the number of samples required from the device scales only polynomially with the circuit size. Further, the classical resource requirements, although scaling exponentially, can be distributed advantageously as a result [62, 85].

Each circuit class is derived from structures that are common to many instances of certain near-term applications of quantum technology, such as those discussed above. By choosing circuits with features common to many instances of applications, rather than random circuits [62, 85] or explicit instances of applications [86–95], this benchmark suite allows us to make broadly applicable predictions of the performance of devices. We use this suite to benchmark all layers of several real quantum computing stacks, exploring the interplay between the compilation strategy, device, and the computation itself. Utilising this information, we identify the applications that each quantum computing stack is best suited for. We also identify the properties of the devices explored, such as the qubit connectivity and noise sources present, which result in improved performance.

Chapter 4 As technology advances to the point of being able to demonstrate quantum computational supremacy, classical simulation, and so the approaches of Chapter 2 and Chapter 3, becomes impossible. However, as discussed, verification schemes for universal quantum computation require resources beyond that of NISQ technology, and so cannot yet be used. Fortunately, schemes for the verification of universal quantum computation are, for our objectives, unnecessarily versatile, and may be simplified for the purposes of demonstrating quantum computational supremacy. Indeed, in the case in which one hopes to demonstrate quantum computational supremacy, only a limited number of computations must be checked, rather than requiring all quantum computations be verified.

In Chapter 4 we explore this middle ground between smaller NISQ technology and devices large enough for the verification of universal quantum computation. In particular we develop a scalable approach to certifying a demonstration of quantum computational supremacy using IQP circuits. The advances we are able to make as a result of studying this reduced model are twofold. We are first able to derive a 'blind' implementation of IQP circuits which does not require that the Server knows the computation to be performed. This implementation can be proven to be compositionally secure, but requires the presence of a network able to transmit single qubits from the Client to the Server. This adapts similar schemes available for universal quantum computing [96], but with a reduced resource cost in our case. Secondly we make use of a class of computations with known output properties in order to certify the accuracy of an implementation. Using the blind implementation we introduce allows us to conceal which computation from this class is being run, and so conceal the output property that will be checked. This prevents the Server from cheating the test by using pre-computed statistics, or a classical device, as we show that to do so would require knowledge of the concealed property. Checking these properties provides a way of benchmarking the device, and also, as a result, a way of verifying a demonstration of quantum computational supremacy.

Chapter 5 Having considered the certification of a demonstration of quantum computational supremacy in Chapter 4, in Chapter 5 we are concerned with extending the notion of quantum computational supremacy to demonstrations via practical application. We consider the case of demonstrating quantum computational supremacy through unsupervised learning, specifically in the case of generative modelling [90, 97]. This is the case in which a quantum model must recognise underlying patterns in a data set, construct a representation of a probability distribution close to that which the data was produced from, and produce samples from that distribution. Such a setting is a natural extension of the discussions in the previous chapters. In particular, as as we have considered certification of sampling problems throughout this thesis, there is a great commonality between generative modelling, where distributions should be learnt and sampled from, and the preceding chapters.

We first introduce quantum learning supremacy to formalise a demonstration of quantum computational supremacy via generative modelling. We then introduce a machine learning model called the Quantum Circuit Ising Born Machine, which is particularly well suited to NISQ technology. The Quantum Circuit Ising Born Machine allows for the output distributions from IQP circuits to be generated, providing grounds to believe the Quantum Circuit Ising Born Machine could learn distributions that could not be learnt by a classical generative model. Finally we discuss means to train the Quantum Circuit Ising Born Machine, and argue why it could be believed that this model, and the associated training procedure, could be used to demonstrate quantum learning supremacy.

	Quantum Technology Required	Contribution of Classical Computing	Subject of Investigation
Chapter 2	🟩 **Few Connected Noisy Qubits** Only indicative noise levels of gates and qubits are required. Only the capacity to implement IQP circuits is needed.	🟥 **Simulation with Noise Model** Simulate larger devices than those that are available. Use noise models to improve predictive power. This is not efficient and **scales poorly** to larger devices.	**Predictions of Performance** Identify dominant noise sources as those that have the greatest negative impact on the simulated performance of larger devices than those that exist. Similarly, evaluate means to manage these errors.
Chapter 3	🟧 **Several Connected Noisy Qubits** Not sufficiently large to demonstrate quantum computational supremacy. Should be able to implement near-term applications.	🟥 **Calculate Ideal Properties** Calculation of some properties of the ideal output of a circuit, against which to compare a real device. This is not efficient and **scales poorly** to larger devices.	**Application-Motivated Benchmark** Predictions and measures of the performance of existing devices when utilised for applications of practical concern.
Chapter 4	🟥 **Many Connected Noisy Qubits** Sufficient to demonstrate quantum computational supremacy. Only the implementation of IQP circuits are required. A quantum network capable of transmitting single qubits is required.	🟩 **Processing of Measurement Results** Minimal processing of classical measurement results from the quantum device. This is in order to guide the computation and benchmark its accuracy. This **scales well** to larger devices.	**Quantum Computational Supremacy** Demonstration of quantum computational supremacy and performance measures of devices capable of demonstrating quantum computational supremacy.
Chapter 5	🟥 **Many Connected Noisy Qubits** Sufficient to demonstrate quantum computational supremacy with practical task. The capacity to implement IQP circuits is a sufficient minimum.	🟩 **Minimise Cost Function** Calculate updates to parameters of PQC in order to improve accuracy of generative model. This **scales well** to larger devices.	**Quantum Learning Supremacy** Demonstration of quantum computational supremacy via practically motivated task.

Table 1: **A summary and comparison of the chapters of this thesis.** Traffic light colouring indicates relative technological complexity: high [🟥], medium [🟧], low [🟩].

Chapter 1

Preliminaries

Here we introduce basic notions from quantum computation and quantum information, as well as background literature which we will require for later chapters. In Section 1.1 we recall fundamental tools which ground later discussions on quantum computing in a solid mathematical framework. To allow us to talk formally about quantum computational supremacy, and the computational power of different models of computing more generally, in Section 1.2 we outline useful definitions and results from computational complexity theory. In Section 1.3 we introduce terminology and results relating to the impact of noise of quantum computation, and to instances where classical computers can recreate the behaviour of quantum ones. We contrast the circuit model of quantum computing introduced in Section 1.1 with Measurement Based Quantum Computing in Section 1.4. The class of quantum circuits called IQP is discussed in Section 1.5. In Section 1.6 we discuss quantum computing in a delegated setting, the security of which can be formalised by abstract cryptography as is introduced in Section 1.7. The dependencies of the remaining chapters of this thesis on the sections of this chapter are detailed in Figure 1.1.

1.1 Quantum Computation and Information

Here we introduce some basic tools and terminology from quantum computing and quantum physics[1]. In particular we introduce the postulates of quantum mechanics, which collectively provide a mathematical framework within which to develop and discuss quantum computation. Models for quantum systems are introduced in Section 1.1.1, while methods for their manipulation are seen in Section 1.1.2. The circuit model allows us to reason about these systems in order to perform quantum computation, and is introduced in Section 1.1.3.

1.1.1 Quantum States, Qubits, and Entanglement

The basic unit of information in quantum computing is the *qubit*. We model a qubit, written using the *ket* notation $|\psi\rangle$, by its *state vector*, which is a linear combination, or

[1]Two popular resources which cover this material in far greater detail than we do here are [25, 28].

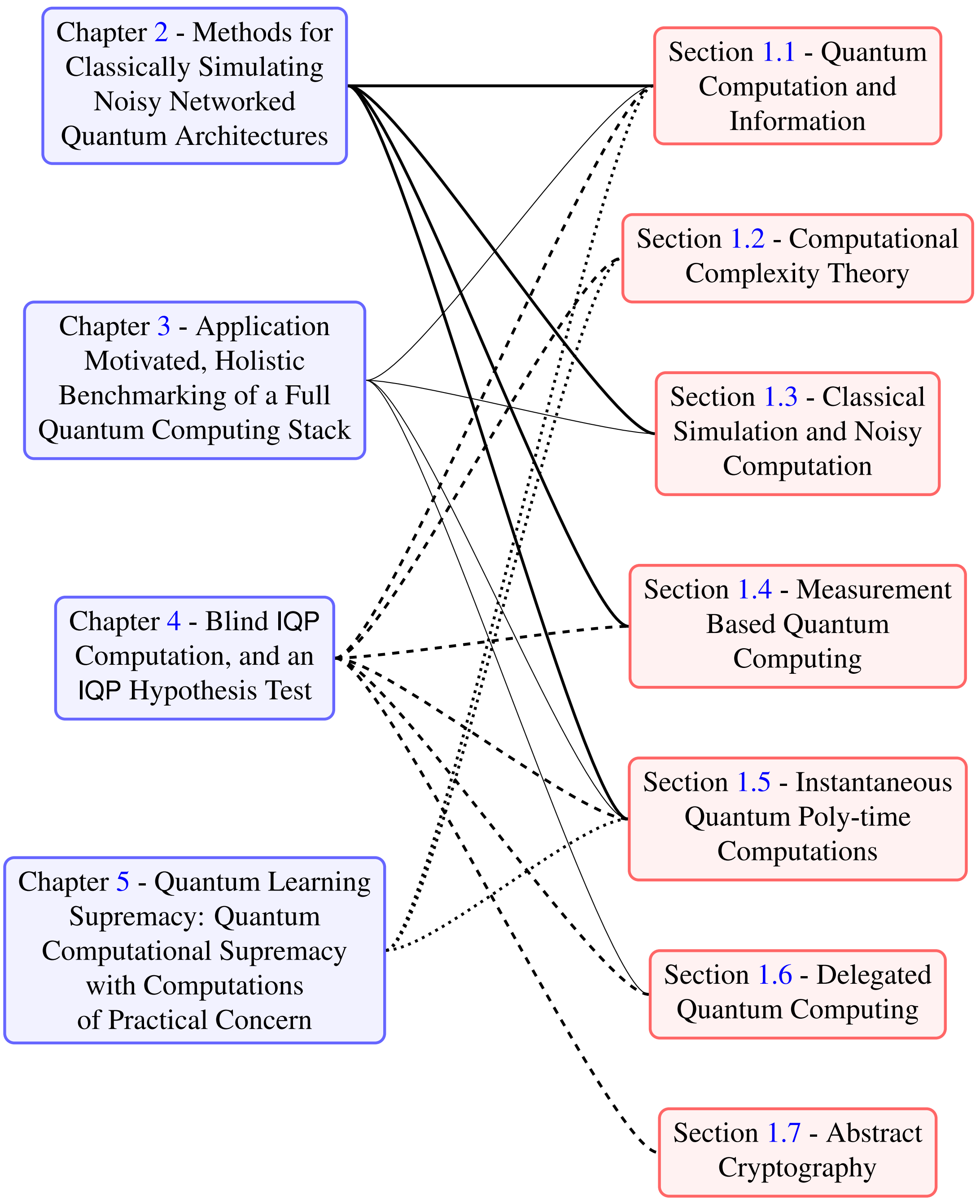

Chapter 2 - Methods for Classically Simulating Noisy Networked Quantum Architectures
Chapter 3 - Application Motivated, Holistic Benchmarking of a Full Quantum Computing Stack
Chapter 4 - Blind IQP Computation, and an IQP Hypothesis Test
Chapter 5 - Quantum Learning Supremacy: Quantum Computational Supremacy with Computations of Practical Concern
Section 1.1 - Quantum Computation and Information
Section 1.2 - Computational Complexity Theory
Section 1.3 - Classical Simulation and Noisy Computation
Section 1.4 - Measurement Based Quantum Computing
Section 1.5 - Instantaneous Quantum Poly-time Computations
Section 1.6 - Delegated Quantum Computing
Section 1.7 - Abstract Cryptography

Figure 1.1: **The dependencies of later chapters on sections of the preliminaries**. A connecting edge indicates that the preliminaries section is a prerequisite for the chapter.

superposition, of the states $|0\rangle$ and $|1\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \text{such that} \quad \alpha, \beta \in \mathbb{C},\ |\alpha|^2 + |\beta|^2 = 1 \tag{1.1}$$

As such the *computational basis* $\{|0\rangle, |1\rangle\}$ defines a two dimensional complex vector space. Denoting the adjoint of a vector $|\psi\rangle$ using the *bra* notation $\langle\psi|$, the inner product between two states $|\psi\rangle, |\phi\rangle$, for which we use the notation $\langle\psi|\phi\rangle$, is the product of vectors $\langle\psi|$, $|\phi\rangle$. This defines a two dimensional Hilbert space denoted $\mathbb{C}^2$, and presents a special case of Postulate 1 of quantum mechanics.[2]

> **Postulate 1** Associated to any physical system is a complex vector space with inner product known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

An alternate basis of this Hilbert space is the *Hadamard basis* of equation (1.2), which is generalised in equation (1.3) by allowing any $\theta \in [0, 2\pi)$.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.2}$$

$$|+_\theta\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\theta}|1\rangle\right) \qquad |-_\theta\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - e^{i\theta}|1\rangle\right) \tag{1.3}$$

The qubit representation in equation (1.1), and the associated conditions on the coefficients, are equivalently well represented by

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right) \quad \text{where} \quad \gamma, \theta, \phi \in [0, 2\pi]\,. \tag{1.4}$$

The factor $e^{i\gamma}$, which we call a *global phase*, can often be ignored, permitting the representation of a single qubit as a point on surface of the *Bloch sphere* of Figure 1.2. The $|0\rangle$ and $|1\rangle$ states are found on the positive and negative Z axis of the Bloch sphere respectively, while the $|+\rangle$ and $|-\rangle$ states are found on the X axis. The $|+_\theta\rangle$ and $|-_\theta\rangle$ states are found on the X-Y plane; the equatorial plane in Figure 1.2.

Larger, multi-qubit systems are constructed from smaller ones using the *tensor product*. The elements of the tensor product, $V \otimes W$, of two vector spaces V and W, are linear combinations of vectors $|v\rangle \otimes |w\rangle$ where $|v\rangle \in V$ and $|w\rangle \in W$. If $|i\rangle$ and $|j\rangle$ form an orthonormal basis of V and W respectively, then vectors of the form $|i\rangle \otimes |j\rangle$ forms an orthonormal basis of $V \otimes W$. This reflects Postulate 2 of quantum mechanics.

> **Postulate 2** The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. If systems 1 to n are prepared in the states $|\psi_1\rangle, \ldots, |\psi_n\rangle$ then the *joint state* of the system is $|\psi_1\rangle \otimes \ldots \otimes |\psi_n\rangle$.

In the case of n qubits, where a single qubit is described by $|x_i\rangle \in \mathbb{C}^2$, the composite system is $|x_1\rangle \otimes \ldots \otimes |x_n\rangle \in \mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2$. The map $|y_1\rangle \otimes \ldots \otimes |y_n\rangle \to |y\rangle$, $y \in \{0,1\}^n$,

[2]Note that the statements and ordering of these postulates may vary between texts.

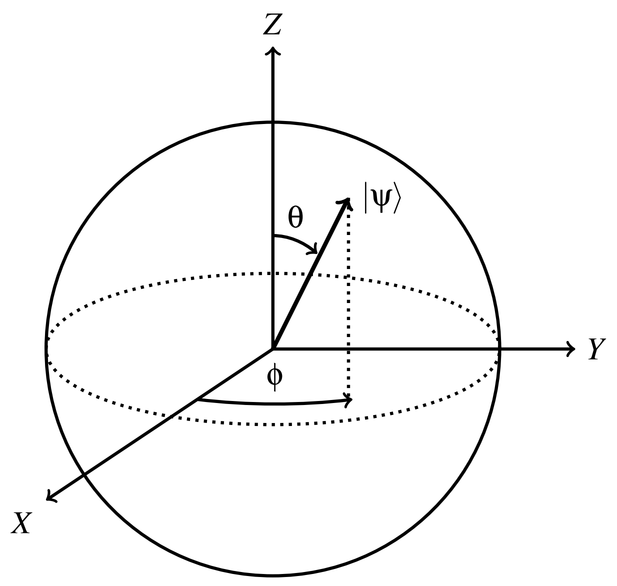


Figure 1.2: **The Bloch sphere.** An intuitive representation of the state vector of a qubit $|\psi\rangle$. The notation used is as in equation (1.4). The state of a qubit is represented by a point on the surface of the sphere.

between the basis vectors of $\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2$ and $\mathbb{C}^{2^n}$ extends by linearity to an isomorphism, and gives us the oftused shorthand $|x_1\rangle \otimes \ldots \otimes |x_n\rangle = |x_1 \ldots x_n\rangle$.

More complicated states, called *entangled states*, are those elements of the composite spaces that cannot be written as a tensor product of elements of the component spaces. A typical example of an entangled state is the following *EPR pair* or *Bell state*.

$$|\Phi_+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.5}$$

Entangled states give rise to *non-locality*, which is behaviour that cannot be explained by a theory that permits influence on an object only by its immediate surroundings.[3]

Elements of tensor product spaces, which we will continue to denote using kets, are called *pure states*, and give a complete description of a quantum system. To capture uncertainty about the state of a physical system, which may, for example, arise from an unknown external influences or probabilistic state preparation, it is necessary to introduce *density matrices*. The density matrix of a pure state $|\psi\rangle$ is

$$\rho = |\psi\rangle\langle\psi|.$$

In this case $\rho^2 = \rho$, which is referred to as being *idempotent*. The converse, that an idempotent density matrix represents a pure state, is also true.

More generally, a *mixed state* is a probability distribution over pure states. For an ensemble of pure states $\{|\psi_i\rangle\}$, each occurring with probability p_i, the corresponding

[3]While this "spooky action at a distance" was unpalatable to Einstein [98], it has since been shown to manifest in experiments [6, 99, 100].

density matrix is

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|. \tag{1.6}$$

While not all density matrices are idempotent, they do all share the property that they are positive semi-definite, Hermitian operators with $\mathrm{Tr}(\rho) = 1$. $\mathrm{Tr}(\rho)$ of a linear operator ρ, called the *trace* of ρ, is the sum of its eigenvalues.

The *maximally mixed state* represents the state of maximal uncertainty about the quantum system. In that case $\rho = \boldsymbol{I}/d$, where d is the dimension of the Hilbert space of the system and $\boldsymbol{I}$ is the $d \times d$ identity matrix. Indeed, if the ensemble $\{|\psi_i\rangle\}$ in equation (1.6) were a basis of a d dimensional Hilbert space, then a maximally mixed state of dimension d is equivalent to a uniform distribution over the basis.

1.1.2 Operators on Quantum States

Qubits can be manipulated by *operators*, which are represented by matrices acting on the state space of the system. The representation of an operator A as a matrix $\boldsymbol{A}$ may be arrived at by considering equation (1.7), where vectors of the form $|k\rangle$ form an orthonormal basis.

$$A = \sum_{ij} |i\rangle\langle j| \boldsymbol{A}_{ij} \quad \text{where} \quad \boldsymbol{A}_{ij} = \langle i|A|j\rangle \tag{1.7}$$

Unitary matrices constitute nondestructive operators on the state space, which is to say they may be reversed. Their action is described by Postulate 3 of quantum mechanics.

> **Postulate 3** In a closed system, the state of the system at time t_1 is related, as seen in equation (1.8), to the state of the system at time t_2 by the unitary operator U which depends only on the times t_1 and t_2.
>
> $$|\psi_{t_2}\rangle = U|\psi_{t_1}\rangle \tag{1.8}$$

$\mathrm{U}(n)$ is the group of all $n \times n$ unitary matrices. When a constant phase can be ignored, we can restrict to the *special unitary group* $\mathrm{SU}(n)$, consisting of $n \times n$ unitary matrices with a determinant of 1.

An alternate but equivalent formalism to that of Postulate 3, which is encountered only in passing in this thesis, is in the 'Schrödinger picture'. In the Schrödinger picture the time evolution of quantum state is given by a Hamiltonian, H. H is a hermitian operator and $|\psi_{t_2}\rangle = e^{iH(t_2-t_1)}|\psi_{t_1}\rangle$ if H is independent of time.

As well as the smooth evolution under a unitary operator, a system may undergo a change due to measurement, as described by Postulate 4. Measurement is not reversible, with some information being lost when measurement is performed, and so we will refer to the operation as destructive.[4]

[4]This use of destructive does not refer to the destruction of the quantum system itself, as in when a photon hits a screen during measurement, but instead to the loss of information.

Postulate 4 A measurement may have m different outcomes. Each is represented by an operator acting on the Hilbert space of the system being measured. Together these operators form the collection $\{M_1, ..., M_m\}$ and must satisfy the *completeness equation*

$$\sum_m M_m^\dagger M_m = 1. \tag{1.9}$$

The probability $p(m)$ of each outcome m relates on the state $|\psi\rangle$ of the system prior to measurement by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle. \tag{1.10}$$

In the case that the outcome of the measurement is m, the state of the system is transformed by

$$|\psi\rangle \to \frac{1}{\sqrt{p(m)}} M_m|\psi\rangle. \tag{1.11}$$

The completeness condition of equation (1.9) ensures the total probability of all measurement outcomes is 1, while equation (1.11) ensures that the outcome of the measurement is a unit vector. In particular, one will notice from equation (1.10) that the condition on the coefficients in equation (1.1) is equivalent to ensuring the probability of measuring either computational basis sums to 1. Global phases, such as that of equation (1.4), do not contribute to the value of the probability calculated in equation (1.10) and may be ignored. As discussed, this facilitates the use of the Bloch sphere of Figure 1.2.

A particularly important class of measurement operators are the *projective operators*. In addition to the conditions on M_m in equation (1.9), projective operators are Hermitian and obey the relation $M_m M_n = \delta_{m,n} M_m$, where $\delta_{m,n}$ is the Kronecker delta. A projective measurement is described by an *observable* A, which is a Hermitian operator in the state space. The *spectral decomposition* of A, seen in equation (1.12), gives A in terms of projection operators, P_a and the corresponding eigenvalues, a. Each P_a projects along the subspace defined by eigenvector $|a\rangle$ of A.

$$A = \sum_a a P_a \tag{1.12}$$

Taking the possible outcomes from the measurement to be the eigenvalues a, Postulate 4 gives the expected value, $\langle A\rangle_\psi$, when measuring the state $|\psi\rangle$ to be

$$\langle A\rangle_\psi = \sum_a a p(a) = \sum_a a\langle\psi|P_a|\psi\rangle = \langle\psi|\sum_a a P_a|\psi\rangle = \langle\psi|A|\psi\rangle.$$

In fact, given the other postulates of quantum mechanics introduced, projective measurements are equivalent to measurements as in Postulate 4, and would work equally well as a postulate [28].

For a density matrix ρ, evolution by U gives the state $U\rho U^{\dagger}$. A measurement by the operators $\{M_i\}$ gives the outcome m with probability

$$p(m) = \mathrm{Tr}\left(M_m \rho M_m^{\dagger}\right)$$

and produces the state

$$\frac{M_m \rho M_m^{\dagger}}{p(m)}.$$

This reveals that a measurement of a maximally mixed state in a basis of the corresponding Hilbert space would measure all possible outcomes with equal probability.

1.1.3 The Quantum Circuit Model

Postulate 3 taught us that unitary operators are used to evolve quantum states. These unitary operators manifest as gates in the gate model of quantum computing. The quantum equivalent of the classical NOT gate is the Pauli-X gate which, when acting on the computational basis states, performs the transformation

$$|0\rangle \rightarrow |1\rangle \quad , \quad |1\rangle \rightarrow |0\rangle.$$

The unitary to which this corresponds is

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Here, and in the remainder of this section, we have used the computational basis and equation (1.7) to derive this representation. In general any basis may be used, and in particular X would be diagonal if the Hadamard basis was used.

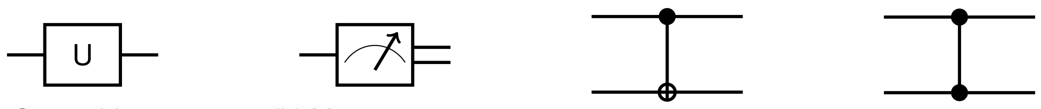
Other commonly used gates are seen in equation (1.13), and are referred to as: the Pauli-Y gate, Y; the Pauli-Z gate, Z; the Hadamard gate, H; and the phase gate, RZ^{θ}.

$$\mathsf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \mathsf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \mathsf{RZ}^{\theta} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \tag{1.13}$$

H can be used to change between the computational and Hadamard basis, with other common gates being $\mathsf{RZ}^{\frac{\pi}{2}} = \sqrt{\mathsf{Z}} = \mathsf{S}$ and $\mathsf{RZ}^{\frac{\pi}{4}} = \sqrt{\mathsf{S}} = \mathsf{T}$. Some useful relationships which we will encounter are $\mathsf{HZH} = \mathsf{X}$ and $\mathsf{HXH} = \mathsf{Z}$. If there is ambiguity as to which qubit the gates are being applied, we will use a subscript, such as X_i, to indicate that it is applied to the i^{th} qubit.

We will refer to such things as ‘a measurement of the Z observable’ which alludes to the use of the projective operators $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, defined by the eigenvectors of Z. These eigenvectors are referred to as the Z *basis*, terminology which extends to all Pauli matrices. This explains the labelling of the axis in the Bloch sphere of Figure 1.2.

We will often draw circuit diagrams to display circuits composed of these gates. In this formalism, qubits are represented by horizontal lines which are acted on by gates



(a) One-qubit gate. (b) Measurement. (c) CX. (d) CZ.

Figure 1.3: **Circuit notation**. This notation is used to represent operations on qubits, represented by a line or wire, with time passing from left to right. Here $\mathsf{U} \in \mathrm{SU}(2)$ while CX and CZ are defined in equation (1.14). Double lines carry classical information.

over time, with time flowing from left to right. For a unitary U acting on a single qubit we will use the notation in Figure 1.3a when drawing circuit diagrams. Measurement is represented as in Figure 1.3b where a double wire carries classical information.

The simplest example of multi-qubit gates is tensor products of operators $\mathsf{U}_1 \otimes \ldots \otimes \mathsf{U}_m$, where $\mathsf{U}_i \in \mathrm{SU}(2^{n_i})$. Each U_i acts on the corresponding subspace of the state space $\mathbb{C}^{2^{n_1}} \otimes \ldots \otimes \mathbb{C}^{2^{n_m}}$, which is to say U_1 acts on the first n_1 qubits, U_2 acts on the next n_2 qubits, and so on. Important examples of two qubit gates are the controlled gates, which apply an operation to a *target* qubit, conditional on the value of a *control* qubit. One example of a controlled gate is CX, or the controlled-X gate,[5] which applies X to the target if the control is in the state $|1\rangle$, and the identity if the control is in the state $|0\rangle$. CZ, or the controlled-Z gate, is the analogue in the case of the Z gate. Both are seen in equation (1.14). Also in equation (1.14) is the SWAP gate, which swaps the position of two qubits.

$$\mathsf{CX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathsf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \mathsf{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.14}$$

One may, for example, write $\mathsf{CX}_{i,j}$ when CX is applied to the control qubit, i, and the target qubit, j. The CZ gate is symmetric, which is to say that whichever of the two qubits is the control or target does not influence the outcome, while in the case of the CX gate this choice is important. In circuit diagrams, these gates are represented as in Figure 1.3c and Figure 1.3d.

Of the gates discussed, the *Pauli group*

$$G_1 = \{\pm\mathsf{I}, \pm i\mathsf{I}, \pm\mathsf{X}, \pm i\mathsf{X}, \pm\mathsf{Z}, \pm i\mathsf{Z}, \pm\mathsf{Y}, \pm i\mathsf{Y}\},$$

and the groups of n-fold tensor products of elements on the Pauli group, G_n, are of particular importance. When dealing with states that are stabilised by elements of a subgroup S of G_n, which is to say states that are fixed by the action of every element of S, it is often more convenient to work with S as opposed to the states themselves. This is of particular importance as S can uniquely define many interesting and highly entangled states, such as the EPR pair of equation (1.5), in this way.

Amongst the uses of this *stabiliser formalism*[6] is as a means of describing the action

[5]This gate also goes by the name CNOT.

[6]The stabiliser formalism is incredibly powerful and we are far from doing it justice here. One of its many applications is in the development of error correction codes referred to as *stabiliser codes* [101].

of gates in the *Clifford group* of operators. These operators *normalise* the Pauli group, which is to say that a Clifford operator U has the property that $UPU^{\dagger} \in G_n$ if $P \in G_n$. *Clifford gates*, which are the gate model representation of elements of the Clifford group, can always be decomposed into combinations of H, CX, and S. Their action on a state stabilised by S can be described by transforming S to give a new subgroup of the G_n. This new subgroup stabilises the state that results from the action of the Clifford gates on the original state. This approach of keeping track of the Pauli operators that stabilise a quantum state as it is acted on by a Clifford circuit is more convenient than tracking the impact on the state directly, which could require the storage of exponentially many amplitudes of a state vector. This is so much the case that we have the following remarkable theorem.

Theorem 1.1.1 (Gottesman–Knill theorem [102])**.** *A computation involving:*

- *state preparations in the computational basis,*
- *the action of Clifford gates,*
- *and measurements of the Pauli observables*

may be reproduced on a classical computer using a number of time steps that grows at most polynomially with the number of qubits.

Although Theorem 1.1.1 covers a great many of the gates discussed, gates from the Clifford group, in combination with the T gate, can be used to approximate any unitary operation. We refer to any set of gates of which a finite sequence can approximate any unitary up to arbitrary precision, as a *universal gate set*. The Solovay-Kitaev theorem [103, 104] allows us to place a reasonable bound on the required length of this sequence, and teaches us that it is efficient to find such a sequence. If we allow for infinitely many gates to be in our universal gate set then we can exactly recreate any unitary. One example of such a gate set is $\{\mathsf{CZ}, \mathsf{U}\}$ where U represents all single qubit gates.

1.2 Computational Complexity Theory

During discussions about quantum computational supremacy, such as those we conduct throughout this work, it will be useful to have a means of formally describing the resources required to solve certain problems. Computational complexity theory is the correct formalism to use for this purpose. As such, to facilitate these discussions we now introduce some commonly used complexity classes, along with some more general notions and notation from the field.[7] For reference, we summarise the complexity classes introduced in this section in Table 1.1.

A succinct comparison of the resource requirements of different computations can be achieved using Bachmann–Landau notation, and in particular, big O notation. This is used to compare the asymptotic resource requirements of computations as the input size n increases. In this notation $O(f(n))$ is used to convey that the requirement a computation has for a resource, such as time or storage space, is some function $g(n)$, and that there are constants c and n_0 such that $|g(n)| \leq cf(n)$ for all $n > n_0$.

[7] More extensive introductions to the topics discussed in this section can be found in [105, 106].

One relevant class of problems is that of *decision problems*, which can be thought of as those problems which admit yes-no answers. More precisely, a decision problem may be defined as a subsets of all finite length binary strings, called a *language*, which should be 'accepted' by a *decision procedure* for that problem. For example, a language L is in the complexity class denoted P if there is a deterministic Turing Machine which runs in polynomial time, and which 'accepts' or assigns 'yes' to all the elements of L, and 'rejects' or assigns 'no' to all other finite length binary strings. Such a Turing Machine is said to have 'decided' L. P is said to contain problems which are 'tractable' or 'efficiently solvable' by a classical computer, and includes such things as linear programming [107] and determining if a number is prime [108]. In general 'efficiently' is taken to mean that the time it takes to solve the problem, given the available conditions and resources, scales polynomially in the size of the input.

A second popular decision class is NP, which contains all languages that can be decided by a polynomial-time non-deterministic Turing Machine. Such a Turing machine may specify more than one possible action to be taken at any step of the computation. These steps create multiple possible 'paths' of computation, with a non-deterministic Turing Machine accepting if at least one path accepts, and rejecting if all paths reject. An alternate characterisation of NP is as the class of decisions problems for which yes instances admit proofs or *witnesses* which can be verified in polynomial time by a deterministic Turing Machine. NP includes problems such as: the travelling salesperson; integer factorisation; SAT, the problem of deciding whether a given Boolean formula has any satisfying assignments; and MaxCut, the problem of determining a bipartition of a graph with the maximum possible number of edges between the resulting vertex subsets. In fact, the travelling salesperson problem, SAT and MaxCut are also NP-hard [109, 110], which is to say that an algorithm for solving any of them can be efficiently translated into an algorithm for solving any problem in NP. When a problem is in NP and is also NP-hard it is called NP-complete, with these notions of hard and complete generalising to other complexity classes. Related to NP is its complement coNP, which contains all languages for which the no instances can be verified in polynomial time by a deterministic Turing Machine.

The power of complexity classes can be boosted by *oracles*, which provide black-box access to solutions of a problem in a single time step. For example, the class of languages which can be decided by a deterministic polynomial-time Turing Machine with access to an oracle function O will be denoted P^O. Indeed, the *Polynomial Hierarchy* uses oracles to generalise P, NP, and coNP.

Definition 1.2.1 (Polynomial Hierarchy (PH) [111])**.** *Let $\mathsf{P} = \Delta_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \Sigma_0^{\mathsf{P}}$ be the 0^{th} level Δ_0 of the* Polynomial Hierarchy*. For $k > 0$, the k^{th} level Δ_k is defined by the three classes $\Delta_k^{\mathsf{P}} := \mathsf{P}^{\Sigma_{k-1}^{\mathsf{P}}}$, $\Pi_k^{\mathsf{P}} := \mathsf{coNP}^{\Sigma_{k-1}^{\mathsf{P}}}$, and $\Sigma_k^{\mathsf{P}} := \mathsf{NP}^{\Sigma_{k-1}^{\mathsf{P}}}$. PH is the union of all of its levels.*

The equality of PH to one of its levels is described as a collapse of PH to that level, but such a collapse is thought to be unlikely [105]. Notably, a collapse to the 0^{th} level would imply $\mathsf{P} = \mathsf{NP}$, contradicting the widely held belief to the contrary [112]. Indeed a conclusion which implies the collapse of PH is often used as evidence that at least one assumption which led to that conclusion is false [61, 63, 66]. Such proofs by contradiction are used by results which argue that it is impossible to reproduce the outputs of quantum computers classically, as we outline in Section 1.5.2.

One vital complexity class which arises throughout this work is BQP, the class of problems solvable by a quantum computer with error probability less than $\frac{1}{3}$ in polynomial time.

Definition 1.2.2 (Bounded-Error Quantum Polynomial-Time (BQP))**.** *A language L belongs to* BQP *if there exists a polynomial p and uniform*[8] *family of quantum circuits $\{C_n\}$, where C_n contains at most $p(n)$ gates, such that for any $x \in \{0,1\}^n$:*

- *if $x \in L$ then $P(C_n(x) = 1) \geq \frac{2}{3}$*
- *if $x \notin L$ then $P(C_n(x) = 1) < \frac{1}{3}$*

Integer factorisation is in BQP [33], although it is not thought to be BQP-hard. As such, that there is no known classical algorithm to factor integers is regarded as weaker evidence for the impossibility of classically simulating quantum computers than is the stability of PH. The analogue to NP in this setting is QMA, which stands for Quantum Merlin-Arthur and refers to the set of problems for which there is a witness for yes instances that can be verified by a BQP machine.

The classical equivalent of BQP is BPP, the class of bounded-error probabilistic polynomial time computations, where quantum circuits are supplemented in the definition of BQP for classical probabilistic Turing Machines. Informally, BPP is the largest class of 'practical' decision problems, which is to say that algorithms for problems in BPP can be run on modern classical computers. As such this class, rather than P, is compared to BQP during discussions on quantum computational supremacy. The class PP of probabilistic polynomial-time computations is similar to BPP, but with weaker error probability demands. Namely it is the class of decision problems solvable by a probabilistic Turing Machine with error probability less than $\frac{1}{2}$.

One class which will arise in this work, due to its connection with the complexity of calculating the probability of outputs from quantum circuits, but which is not a decision class, is #P. This is the class of functions that count the number of accepting paths of a polynomial-time non-deterministic Turing Machine, generalising NP. #P contains such problems as #SAT, which is concerned with counting the number of satisfying assignments of a given Boolean function.

Of particular importance for NISQ technology, and the work of this thesis, are classes of *sampling problems*. In the case of sampling problems the task is to produce samples from a probability distribution, rather than binary outputs as in the case of decision problems. In the case of classical computation, this may be seen as the task of transforming uniformly random bits into non-uniformly random bits. Quantum computing, on the other hand, is probabilistic by the nature of measurement. Several such classes,

[8] In this context *uniform* identifies that there is a deterministic Turing machine running in polynomial-time which, given 1^n as input, outputs a description of C_n. Uniformity prevents significant computational power being hidden in the circuit construction phase. However other uniformity conditions, where the Turing machine producing the circuits have different computational resources available, may also be of use.

Acronym	Meaning
BosonSampling	Boson Samlping.
BPP	Bounded-Error Classical Polynomial-Time.
BQP	Bounded-Error Quantum Polynomial-Time.
IQP	Instantaneous Quantum Polynomial-Time.
NP	Nondeterministic Polynomial-Time.
P	Deterministic Polynomial-Time.
PH	Polynomial Hierarchy.
PP	Probabilistic Polynomial-Time.
QMA	Quantum Merlin Arthur
RCS	Random Circuit Sampling.

Table 1.1: **A summary of complexity classes and acronyms.**

with reduced physical requirements as compared to those of BQP, have been introduced. These same classes are, in spite of the reduced resource requirement, thought not to be contained in BPP.

Two examples of sampling problems which are native to the circuit model are sampling from IQP circuits [113], discussed in Section 1.5, and Random Circuit Sampling (RCS) [62], introduced in Section 1.5.3. One such class of problems from outside of the circuit model is BosonSampling, which we also discuss further in Section 1.5.3, and which requires samples to be taken from linearly scattering individual Bosons [61]. All three are examples of classes of problems which cannot be approximately sampled from by a classical computer if PH does not collapse to its third level, Δ_3 [61, 63, 66]. In general we refer to models of quantum computation with reduced resource requirements as compared to BQP as *sub-universal models*.

1.3 Classical Simulation and Noisy Computation

As exemplified by the Gottesman–Knill theorem of Theorem 1.1.1, it is sometime possible to use classical simulation to efficiently reproduce some property of the output distributions of quantum circuits.[9] In general this may be samples from the distributions, the amplitudes of some outputs, or the full output probability distribution. However, it is thought that, in general, the resource requirement for the classical simulation of universal quantum computation grows exponentially with the number of qubits, and so this becomes increasingly difficult as the sizes of the circuits considered approach those required for demonstrations of quantum computational supremacy. Indeed, a

[9]There are many publicly available classical simulators of quantum computation. Some of the most easily available classical simulators can be accessed through general purpose quantum software packages [114–116], while other highly optimised simulators are also publicly available [117–119].

'brute-force' simulation of a quantum system by multiplying the matrices which appeared in Section 1.1, requires that the full state vector, which grows exponentially in size as the number of quits increases, be stored.

Despite the hardness of classically simulating quantum circuits, the effect of noise channels prevalent in NISQ hardware might be to create distributions that can be simulated. For some circuits and noise types there are theoretical guarantees that this is not the case. We discuss some of the guarantees that exist for the simulation of IQP circuits in Section 1.5.2. The theoretical guarantees that can be given depend on how accurately the noisy distributions approximate the ideal ones, with some notions of approximation discussed in Section 1.3.1.

Further, classical simulation is of particular importance as a means of validating small quantum devices, once again in spite of its difficulty in general. For example, the behaviour of real quantum computers, which may experience errors, can be compared to an ideal implementation simulated using a classical computer. This is of particular importance for our work in Chapter 2 and Chapter 3 where we will explore the impact of noise on the output distributions from real hardware. While in some cases highly optimised brute-force simulators are the best known approach to simulating quantum computation [120–122] improvements are possible, and the review of classical simulation techniques in Section 1.3.2 allows us to select the best such technique for our purposes. In Section 1.3.3 we introduce several noise channels, pertinent to our work of Chapter 2 and Chapter 3, and consider on how they may be modelled and simulated.

1.3.1 Notions of Approximation and Simulation

Measurements of the states that result from quantum circuits produce classical binary strings. We will use $p_C(x) = \langle x|C|0^n\rangle$ to denote the probability that $x \in \{0,1\}^n$ is measured when an n qubit quantum circuit C acts on $|0^n\rangle$. As such, a faithful reproduction of the behaviour of a quantum computer by a classical one would generate samples from the distribution of outputs from a quantum circuit. This is referred to as weak simulation.

Definition 1.3.1 (Weak simulation)**.** *We say that a circuit family $\mathcal{C}$ can be weakly classically simulated if, given a circuit $C \in \mathcal{C}$ acting on n qubits, its output distribution can be sampled from in classical* $\text{poly}(n)$ *time. This is to say that the number of time steps required grows at worst as a polynomial in n.*

A stronger classical simulator would return the output probabilities and marginal probabilities for subsets of qubits. With these values one can reproduce weak simulation by sampling successive bits; using the conditional distributions, conditioned on those bits already seen, to sample the next [123]. As the inverse is not true [124], it is in this sense that *strong simulation* is stronger than weak simulation.

Definition 1.3.2 (Strong simulation)**.** *We say that a circuit family $\mathcal{C}$ can be strongly classically simulated if, given a circuit $C \in \mathcal{C}$ acting on n qubits, any output probability and any marginal probability can be computed to m digits of precision in classical* $\text{poly}(n,m)$ *time.*

It is known that the existence of such strong simulations for some classes of quantum computations would imply the collapse of the PH [123]. In fact, as we will discuss in Section 1.5.2, this is also true for weak simulation.

It is more realistic, even for real quantum computers, to request approximate simulation, as this permits some errors due to noise, such as those discussed in Section 1.3.3. The notion of approximation that one uses affects the noise that can be accommodated. A very strong notion of approximation is up to multiplicative error, which permits little noise, but allows for strong impossibility results to be derived.

Definition 1.3.3 (Weak simulation with multiplicative error)**.** *We say that a circuit family $\mathcal{C}$ can be weakly classically simulated with multiplicative error* $\alpha \geq 1$ *if there is a family of distributions D, where $\mathcal{D}_C \in D$ is a distribution over* $\{0,1\}^n$ *indexed by an n qubit circuit $C \in \mathcal{C}$, such that for all $C \in \mathcal{C}$, $\mathcal{D}_C$ can be sampled in classical* $\operatorname{poly}(n)$ *time, and for all $x \in \{0,1\}^n$ we have:*

$$\frac{1}{\alpha} p_C(x) \leq \mathcal{D}_C(x) \leq \alpha p_C(x).$$

This notion of approximation is, however, unrealistic as it depends on the probability of the sample in the ideal quantum distribution p_C. In particular it demands that outcomes with probability 0 in p_C also have probability 0 in $\mathcal{D}_C$. It is more reasonable, and closer to the true capabilities of noisy quantum computers, to consider closeness in ℓ_1-norm distance, which is independent of the probabilities themselves.

Definition 1.3.4 (Weak simulation with ℓ_1-norm distance error)**.** *A circuit family $\mathcal{C}$ can be weakly classically simulated with ℓ_1-norm distance error* ε *if there is a family of distributions D, where $\mathcal{D}_C \in D$ is a distribution over* $\{0,1\}^n$ *indexed by an n qubit circuit $C \in \mathcal{C}$, such that for all $C \in \mathcal{C}$, $\mathcal{D}_C$ can be sampled in classical* $\operatorname{poly}(n)$ *time and we have:*

$$\ell_1(p_C, \mathcal{D}_C) = \sum_{x \in \{0,1\}^n} |p_C(x) - \mathcal{D}_C(x)| < \varepsilon.$$

Such a metric is sufficiently strong that for several classes of quantum circuits it is known that the existence of an algorithm for weak simulation of all circuits in that class within ℓ_1-norm distance is unlikely [61, 65, 66]. However, this notion of simulation may also be too strong to be representative of realistic noise as constant errors on each gate may result in distributions that are far from the ideal in ℓ_1-norm distance.

1.3.2 Approaches to Classical Simulation

With regards to the possibility of efficient classical simulation of quantum systems, quantum computational supremacy concerns cases where negative results exist. Here we discuss some of the positive results that exist, and some of the approaches that can be taken to improve on the brute-force approach when they do not. One approach to extending the reach of classical simulation is to restrict the class of circuits that a classical simulation algorithm should target. Ideally the structure of these circuits can then be exploited in order to accelerate their classical simulation. This may result in an

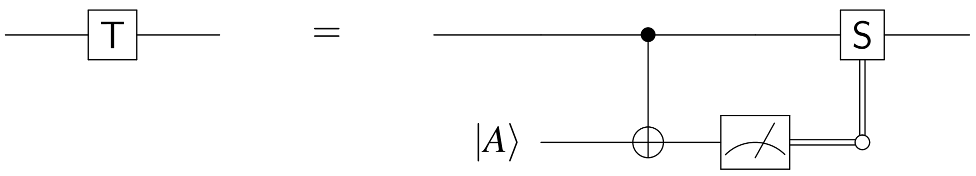


Figure 1.4: **The gadget used to replace a T gate [137].** $|A\rangle$ are the magic states of equation (1.15).

algorithm to efficiently simulate that class of circuits [102, 125–128], or an algorithm with a scaling which is better than that of brute-force simulation [129, 130].

Of particular importance in this regards is the Gottesman-Knill theorem [102], stated in Theorem 1.1.1. That result led to several highly optimised simulators of Clifford circuits and circuits with few non-Clifford gates [131–133]. In particular, as mentioned in Section 1.1, while the Clifford gate set is not universal for quantum computation, adding just the T gate to the set makes it universal for quantum computation. In [134], an algorithm to classically simulate circuits built using the Clifford + T gate set is introduced, and will henceforth be referred to as the *Bravyi-Gosset Simulator*. The Bravyi-Gosset Simulator runs in time that is exponential in the number of T gates but polynomial in the number of qubits and Clifford gates.[10] This allows circuits dominated by Clifford gates to be simulated.

Circuits in the Clifford + T gate set are of concern in Chapter 2, and so we are motivated to utilise the Bravyi-Gosset Simulator there, and to introduce some of the details of the algorithm now. First, all T gates in the circuit are replaced by the gadget of Figure 1.4. The measurement can be replaced by post-selection onto the 0 outcome, which is to say by the projection $|0\rangle\langle 0|$. The *magic states* [136]

$$|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle) \tag{1.15}$$

are replaced by a decomposition into a linear combination of exponentially many stabiliser states. The result is a circuit consisting of only Clifford gates, acting on a linear combination of exponentially (in the number of T gates) many stabiliser states, which can be simulated using the Gottesman-Knill theorem. A randomised algorithm is used to reduce the resources required to estimate single qubit outcome probabilities from the resulting state. As a consequence, this algorithm performs a probabilistic strong simulation of the initial circuit. If an approximate decomposition of $|A\rangle$ is used then probabilities of single outcomes cannot be accurately derived, while such a decomposition is still sufficient to build an algorithm to perform approximate weak simulation.

An alternate approach, when it is necessary to simulate arbitrary quantum circuits, is to tackle the bottlenecks to classical simulation, such as restricted memory, as they arise from the particular classical computing architecture used. *Feynman simulators* compute output bit string amplitudes by adding all Feynman path contributions, and require memory which grows polynomially with the number of qubits, but a number of operations which grows exponentially [85]. This is in contrast with the brute-force approach

[10]The Bravyi-Gosset Simulator has also been generalised to allow for other gate sets [135].

described above, which demanded high levels of storage, but places more reasonable demands on time steps. In fact, there is a 'smooth trade-off' between memory usage and runtime, allowing for some flexibility depending on the resources available [85]. Storage is often the bottleneck, and so Feynman simulators are used to establish the frontier of what is possible on classical computers [60, 138, 139].

1.3.3 Noise in Quantum Computations

The notions of approximation discussed in Section 1.3.1 are particularly relevant for quantum computers exposed to noise. Throughout this work, but in Chapter 2 and Chapter 3 in particular, we are concerned with the impact noise in real quantum devices has on these notions of approximation, and in turn on the applicability of those devices. To facilitate our work in Chapter 2 and Chapter 3 we introduce some pertinent noise channels here. We shall consider two groups of noise channels. A *coherent* noise channel is one which preserves the purity of the input state. This includes, for example, the application of a unitary other than the one intended. More generally, systems on which noise acts are not closed, with interactions with the 'environment' bringing about the noise. Such noise channels do not necessarily preserve the purity of the system, and are referred to as *incoherent*.[11]

Modelling incoherent noise requires an extension of the mathematical tools introduced in Section 1.1. We consider more general operators E which are linear maps $E : \mathcal{D}(\mathcal{H}_1) \to \mathcal{D}(\mathcal{H}_1)$ from density matrices on one Hilbert space to another. To preserve the normalisation of the state, these operators should preserve trace, which is to say $\mathrm{Tr}(E(\rho)) = \mathrm{Tr}(\rho)$. Further, to ensure that $E(\rho)$ is a valid density operator, E should map positive operators to positive operators. This should also be true for $E \otimes I$, when the operator is applied to only part of a system, which we refer to as being *completely positive*. These *completely positive trace preserving maps* (CPTP) constitute all deterministic operations permitted by quantum mechanics.[12]

Any CPTP map can be written, using the *operator-sum representation* [28], as

$$E(\rho) = \sum_i K_i \rho K_i^\dagger$$

where $\{K_i\}$ is a set of linear operators, known as *Kraus operators* [142], which satisfy

$$\sum_i K_i^\dagger K_i = 1.$$

We use this decomposition to introduce the noise channels studied in our later work.

Bit flip: Flips the qubits $|0\rangle$ and $|1\rangle$ with probability $1-p$. In this case $E(\rho) = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger$, where

$$K_0 = \sqrt{p}\,\mathsf{I} \quad K_1 = \sqrt{1-p}\,\mathsf{X}.$$

[11]Coherent errors can interfere constructively in the worst case, and so coherent noise channels are often regarded as more concerning than incoherent ones. Indeed transforming coherent noise channels into incoherent ones proves to be a fruitful approach to reducing their impact [140, 141].

[12]Non-deterministic operations, such as measurement, do not preserve trace.

Dephasing: Applies Z with probability $1-p$. In this case

$$K_0 = \sqrt{p}\,\mathsf{I} \quad K_1 = \sqrt{1-p}\mathsf{Z}.$$

The strength of a dephasing channel is often quantified by T_2, referred to as the device's 'T_2 time'. The probability that a $|+\rangle$ state has not transformed to the $|-\rangle$ state during a time interval t, or vice versa, decays like $e^{\frac{-t}{T_2}}$. This is to say that no error occurs with probability $\mathbb{P}\left(0; \frac{t}{T_2}\right)$, the probability of no events occurring during a Poisson process with mean $\frac{t}{T_2}$

Depolarising: Replaces, with probability p, the initial state with the maximally mixed state, resulting in the overall effect

$$E(\rho) = p\frac{\mathsf{I}}{2} + (1-p)\rho. \tag{1.16}$$

Here we notice that the output probability distribution tends to the uniform one with increasing noise. This is not in the operator-sum notation, but noting that

$$\frac{I}{2} = \frac{\rho + \mathsf{X}\rho\mathsf{X} + \mathsf{Z}\rho\mathsf{Z} + \mathsf{Y}\rho\mathsf{Y}}{4},$$

along with some re-parametrisation, gives

$$E(\rho) = (1-p)\rho + \frac{p}{3}(\mathsf{X}\rho\mathsf{X} + \mathsf{Z}\rho\mathsf{Z} + \mathsf{Y}\rho\mathsf{Y}).$$

Amplitude Damping: Models energy dissipation and the relaxation from an excited state to the lowest energy eigenstate. For a one-qubit system with a probability γ of decaying from $|1\rangle$ to $|0\rangle$ we have

$$K_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \quad K_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}.$$

This error channel may also be referred to as *relaxation*. The strength of an amplitude damping channel of a device is often quantified by T_1, referred to as the device's 'T_1 time'. In that case the probability that a $|1\rangle$ state has not decayed to the $|0\rangle$ state (but not the inverse, as with $|+\rangle$ and $|-\rangle$ in the case of dephasing noise) after a time t falls like $e^{\frac{-t}{T_1}}$.[13]

In all cases above, the Kraus operators, and the CPTP map they define, do not depend on time. In general this property, which is referred to as *Markovianity* and corresponds to assuming that the environments causing separate noise instances act independently, may not hold. However, it is often reasonable and convenient to assume that the noise of a system is Markovian, and it is an assumption that we will make throughout.

In contrast to the modelling of noise channels using density matrices, many classical simulation algorithms, and in particular the Bravyi-Gosset Simulator which we use in

[13]'Decoherence time' is often used to refer to the combination of T_1 and T_2 time, while 'decoherence' may also be used to refer to any noise channel resulting in loss of purity.

Chapter 2, consider only pure states. To address this apparent incompatibility, noise channels are modeled on a classical computer by performing many runs and ensuring that the effect of the noise model is recreated on average. This is achieved by applying a random choice of the Kraus operators which make up a CPTP map, in place of that map. More precisely, during the n^{th} execution of the simulation, the Kraus operator K_{i_n} is applied to the initial state $|\phi\rangle$ to give $\frac{1}{\sqrt{p_{i_n}}}K_{i_n}|\phi\rangle$, where $p_{i_n} = \langle\phi|K_{i_n}^{\dagger}K_{i_n}|\phi\rangle$. Then the average over N executions is

$$\frac{1}{N}\sum_{n=1}^{N}\frac{1}{p_{i_n}}K_{i_n}|\phi\rangle\langle\phi|K_{i_n}^{\dagger}.$$

Letting N_i be the number of times K_i is randomly selected gives

$$\sum_i \frac{N_i}{Np_i}K_i|\phi\rangle\langle\phi|K_i^{\dagger}. \tag{1.17}$$

If K_i is chosen with probability p_i then Np_i converges to N_i, and so equation (1.17) converges to a behaviour equivalent to the desired CPTP map. As a result, such simulations of noise channels using pure states are not only made possible by, but also require many executions.

One notable source of noise, that we do not fit into the above framework, is *readout error*, which results from imperfect measurement. There are many instances, notably in the case of superconducting qubits, where this can be well understood by a classical noise model [143]. In this case, noise is modelled by a transition matrix $\boldsymbol{A} \in \{0,1\}^{2^n \times 2^n}$, where $\boldsymbol{A}_{\widetilde{\boldsymbol{x}},\boldsymbol{x}}$ is the probability of observing $\widetilde{\boldsymbol{x}}$ when the true outcome is $\boldsymbol{x}$. For mathematical convenience or otherwise, the assumption that the noise acts independently on each qubit is often made, which allows for the decomposition of $\boldsymbol{A}$ into a tensor product of n many 2×2 matrices [144].[14] In either case, characterising and inverting $\boldsymbol{A}$ provides a means to mitigate readout error [144–146].

A second such notable source of errors is crosstalk, which broadly encapsulates violations of assumptions of spatial locality and independence of operations. This is to say that operations may inadvertently affect qubits which are not the target of the operation, or that the impact of the operation may depend on the operations applied to other qubits. Crosstalk is challenging to model [147, 148], and we will not attempt to do so in this thesis. This will become particularly important in Chapter 3 where we compare simulations using noise models and the behaviour of real devices. However there has been some success in tackling crosstalk by decoupling qubits to prevent non-local correlated noise [60, 149]. Both readout errors and crosstalk are often significant in near-term devices, as discussed in Chapter 3.

1.4 Measurement Based Quantum Computing

Unlike the circuit model introduced in Section 1.1.3, Measurement Based Quantum Computing (MBQC) [150, 151] is a model of quantum computing without a classical

[14]Intermediate models, which consider limited correlation between qubits, also allow for some efficiency savings as compared to the most general model [144, 145].

analogue. Quantum computation is described in the circuit model by unitary evolution of a quantum state, followed by measurements at the end. Conversely MBQC utilises measurement throughout, with no unitary operations employed after an initial entangled state is prepared. Although measurements are destructive, in Section 1.4.1 we will outline how MBQC can be used to perform universal quantum computation.[15]

For some implementations of quantum computing, such as those based on photonics, MBQC is a natural model. Indeed there are promising proposals of ways that MBQC can be used to perform fault tolerant quantum computing in this way [154]. Further, MBQC clearly divides classical and quantum resources, lending it to a setting where a computationally weak client delegates a computation to a computationally powerful server. This facilitates the use of MBQC to perform blind quantum computing, as discussed in Section 1.6.1. By blindly manipulating the initial entangled state it is possible to perform verification of universal quantum computation, which we introduce in Section 1.6.2. The initial entangled state is manipulated by bridge and break operations, which are discussed in Section 1.4.2.

In Section 1.5.4, we will see how a similar but simplified set of tools can be used to derive an implementation of IQP circuits in MBQC. This implementation is used in Chapter 2 since it explicitly parallelises the computation and reduces the execution time, which proves beneficial for the device explored there. Indeed, by using the tools from this section to implement IQP in MBQC, in Chapter 4 we are able to develop a simple blind implementation, which is vital to other results in that chapter.

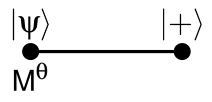
1.4.1 Universality

Typically, MBQC proceeds by initialising qubits in the state $|+\rangle$, entangling them using the CZ operator, and measuring them in the basis $\mathsf{M}^\theta := \{|+_\theta\rangle, |-_\theta\rangle\}$. This is sufficient to perform any single qubit gate, and the CZ gate, themselves sufficient to perform universal quantum computation. We outline how to implement these gates.

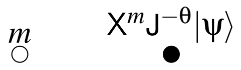
Single Qubit Gates: Consider performing a CZ operation between an initial qubit $|\psi\rangle$ and $|+\rangle$, and measuring the initial qubit in the basis M^θ, as described in Figure 1.5a. This produces the state $\mathsf{X}^m\mathsf{J}^{-\theta}|\psi\rangle$, where m is the measurement outcome and $\mathsf{J}^\phi := \mathsf{HRZ}^\phi$, as indicated in Figure 1.5b. Since any single qubit unitary may be decomposed as $\mathsf{J}^0\mathsf{J}^{\theta_1}\mathsf{J}^{\theta_2}\mathsf{J}^{\theta_3}$, if we could correct for the X^m operation then by composing this technique any single qubit gate is accessible. We shall discuss this correction below.

The CZ Gate: Setting $\theta = 0$ in the single qubit case creates the state $\mathsf{X}^m\mathsf{H}|\psi\rangle$, which we call performing ‘teleportation’ up to a correction $\mathsf{X}^m\mathsf{H}$. Consider enacting CZ between two such teleported states. Notice that the measurement operations, and the CZ operation on the teleported states, act on different subsets of qubits. This means the temporal ordering of these operations can be swapped, in which case all entanglement (i.e. that required for teleportation and for entangling the teleported states) is performed prior to measurement. This is consistent with the

[15]Besides the brief overview of MBQC which we give here, there are several others going into more detail [152, 153].

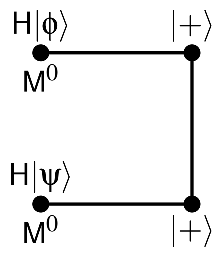


(a) Initial entanglement and measurement pattern.

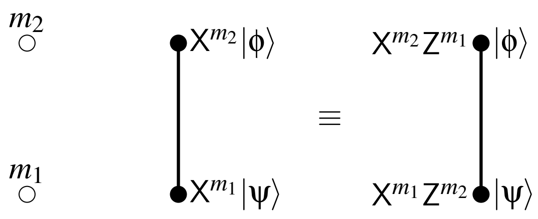


(b) State resulting from measurement. m is the outcome of the measurement described in Figure 1.5a.

Figure 1.5: **Single qubit gate in MBQC**. Filled circuits indicate unmeasured qubits, unfilled circles indicate measured qubits, solid lines indicate CZ operations.



(a) Initial entanglement and measurement pattern. An H gate has been applied using an instance of the single qubit operation of Figure 1.5.



(b) State resulting from measurement. m_1 and m_2 are the results of measurements in Figure 1.6a. To the left of the equivalence we indicate the action of CZ on the teleported state. On the right we describe the effect of commuting the corrections so that they instead act on the entangled state.

Figure 1.6: CZ **gate in MBQC.** Circles and lines are as in Figure 1.5.

philosophy of MBQC. The H correction can be cancelled with a single qubit gate beforehand. Commuting the X correction before the entanglement results in a XZ correction, as indicated by Figure 1.6b. As such we have a scheme to perform a CZ gate, up to a correction, between two qubits using only measurement on an entangled state, which is described in Figure 1.6a.

In both cases, the action of the gate is up to a possible correction of X or Z. Rather than performing the corrections on the unmeasured qubits, this can equivalently be done by altering their own measurement angle. In particular we have the relationships

$$M^\theta X = M^{-\theta} \quad , \quad M^\theta Z = M^{\theta+\pi}$$

which allow us to derive a new measurement angle θ' which depends on the measurement outcomes of neighbouring qubits.

This allows us to define a procedure for performing an MBQC computation which is sufficiently general to be able to perform universal quantum computation. This procedure can be found in Protocol 1.4.1, which first builds a *graph state* before repeatedly performing measurements and measurement angle corrections. Implicit in Protocol 1.4.1 is the order of the measurements, which, along with the measurement angles of each qubit, forms a *measurement pattern*. Indeed qubits whose measurement angles depend on the measure outcomes of others should not be measured before those others. This ordering of the qubits is referred to as *flow* [155], which gives a partial ordering of the qubits and a consistent order of measurements and corrections.[16] Indeed, the

[16]That a measurement pattern has a flow is a sufficient condition for determinism in the MBQC

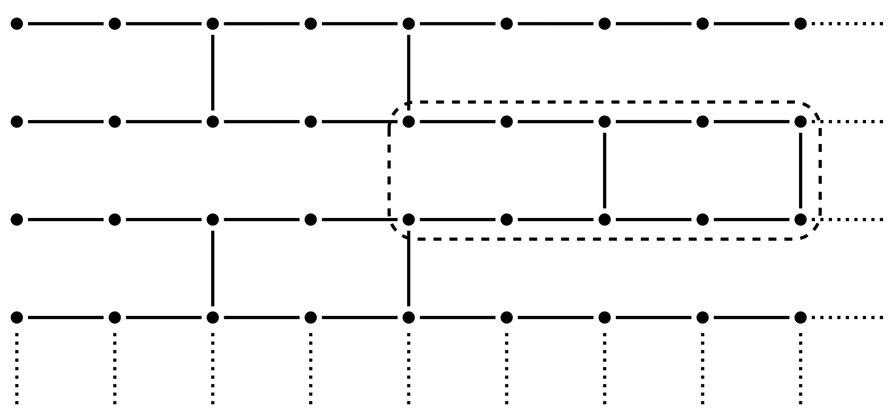


Figure 1.7: **The Brickwork state [96]**. The dotted region is repeated throughout. Performing measurements on this region, at angles $\theta_i \in \left\{0, \frac{\pi}{4}, \dots \frac{7\pi}{4}\right\}$, is sufficient to replicate a universal gate set.

same graph utilising different flows may result in different computations [157]. Alternatively, *non-adaptive* MBQC is a sub-universal model of quantum computation where measurement angles are not adapted according to the outcomes of those before. It is thought that efficiently classically simulating even this model of computation, of which the IQP circuit introduced in Section 1.5 are an example, is not possible [24, 158].

Protocol 1.4.1 A general MBQC computation.

Input: Graph $\boldsymbol{G}$, initial measurement angles $\theta_1, ..., \theta_n$.
Output: Outputs can be classical or quantum.

```
1: Preparation: Prepare qubits $q_1, ..., q_n$ in the state $|+\rangle$.
2: Entanglement: Apply CZ between qubits when their corresponding vertices are
   connected in the graph $G$.
3: for all qubits $\in \{q_1, ..., q_n\}$ do
4:     Measurement: Measure qubit $q_i$ in the basis $M^{\theta_i'}$
5:     Correction: Correct the measurement angles of connected qubits according to
       the measurement outcomes.
6: end for
```

In fact there are fixed graphs $\boldsymbol{G}$ which can be used for all quantum computations. One such *universal graph state* is the *brickwork state* [96], seen in Figure 1.7. In particular, performing measurements on the highlighted pattern of qubits in Figure 1.7, at angles $\theta_i \in \left\{0, \frac{\pi}{4}, \dots \frac{7\pi}{4}\right\}$, is sufficient to replicate a universal gate set. Universal quantum computation can then be achieved with a fixed order of measurements on the brickwork state, namely from top to bottom and then from left to right.

That the graph and order of measurement are the same for any computation is vital for blind quantum computing. This ensures that the computations cannot be distinguished

model. *Generalised flow*, or g-flow, is an extension of flow which gives a necessary condition for determinism [156]

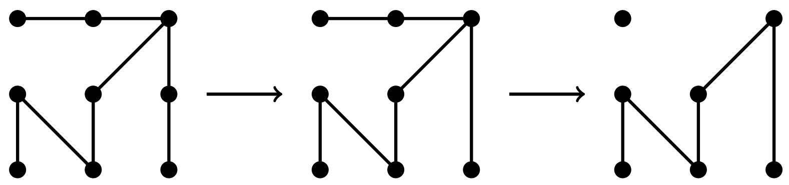


Figure 1.8: **Bridge and break operations**. An example of a sequence of one bridge and one break operation as defined in Definition 1.4.1.

by these properties, which is a technique we will also employ in Chapter 4 to blindly perform IQP computations. What remains is to conceal the measurement angles and measurement outcomes, which we discuss in Section 1.6.1.

While measurements of the brickwork state in the basis M^{θ} for $\theta \in \left\{0, \frac{\pi}{4}, \dots, \frac{7\pi}{4}\right\}$ is sufficient to perform universal computation, it may sometimes be convenient to use other graphs and measurment basis. This may, for example, be because the connectivity and measurement basis made available by a device differ from these. Further, when considering other classes of computation than universal quantum computation it may be preferable to use other basis and graphs if doing so results in some resource saving. Indeed this is the case for IQP circuits, as we discuss in Section 1.5.4.

1.4.2 Break and Bridge Operations

Break and bridge operations [73] allow for some connections within the graph state to be removed. One use of this is to disconnect single qubits from a graph state, which can then be used to check a computation is running correctly, as we discuss in Section 1.6.2. A second use, which is of concern for us in Chapter 4, is to reduce an initial generic graph state to arrive at a more desirable one. Starting from a generic state, and reducing it 'blindly', reduces the information revealed about the computation.

Consider a graph $\widetilde{\boldsymbol{G}} = (\widetilde{V}, \widetilde{E})$, with vertex set $\widetilde{V}$ and edge set $\widetilde{E}$.

Definition 1.4.1 (Bridge and Break Operators)**.** *The* break *operator acts on a vertex $v \in \widetilde{V}$ of degree 2 in a graph $\widetilde{\boldsymbol{G}}$. It removes v from $\widetilde{V}$ and also removes any edges connected to v from $\widetilde{E}$.*

The bridge *operator acts also on a vertex $v \in \widetilde{V}$ of degree 2 in a graph $\widetilde{\boldsymbol{G}}$. It removes v from $\widetilde{V}$, removes any edges connected to v from $\widetilde{E}$ and adds a new edge between the neighbours of v.*

Figure 1.8 gives an example of multiple applications of the bridge and break operators.

Graph states can be built from graphs using a graph state circuit.[17]

Definition 1.4.2 (Graph State Circuit)**.** *Consider a matrix $\boldsymbol{G} \in \{-1, 0, 1\}^{n_a \times n_p}$ and use the function $g(i, j) = k$ to define indices $k = 1, \dots, n_b$ for the elements $\boldsymbol{G}_{ij} = -1$. The matrix corresponds to a graph, which we also denote by $\boldsymbol{G}$, with vertex set*

$$V = \left\{p_1, \dots p_{n_p}, a_1 \dots, a_{n_a}, b_1, \dots, b_{n_b}\right\}$$

[17]The notation we use here is convenient for the remainder of this work, although it may not be the most natural for build graph states more generally. In other work, adjacency matrices are often used.

and edge set including edges between p_j and a_i if $\boldsymbol{G}_{ij} = 1$, and between $b_{g(i,j)}$ and a_i, and, $b_{g(i,j)}$ and p_j, when $\boldsymbol{G}_{ij} = -1$. The graph state circuit *$E_{\boldsymbol{G}}$ on $(n_a + n_p + n_b)$ qubits applies* CZ *operations between qubits p_j and a_i if $\boldsymbol{G}_{ij} = 1$, and between qubits $b_{g(i,j)}$ and a_i, and, $b_{g(i,j)}$ and p_j, when $\boldsymbol{G}_{ij} = -1$.*

Finally, it is known that bridge and break operations on graphs have a corresponding notion in operations on graph states.

Lemma 1.4.1 (From [73])**.** *If it is possible to obtain the graph $\boldsymbol{G}$ from $\widetilde{\boldsymbol{G}}$ through a sequence of bridge and break operations, then for any state $|\phi\rangle$, there is a state $|\psi\rangle$ such that it is possible to obtain $E_{\boldsymbol{G}}|\phi\rangle$ from $E_{\widetilde{\boldsymbol{G}}}|\psi\rangle$ through a sequence of Pauli measurements and local rotations about the Z axis through angles from the set $\left\{0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi\right\}$.*

We will detail, and adapt, the proof of Lemma 1.4.1 in Chapter 4. Importantly for our work in Chapter 4, these bridge and break operations can be delegated, by a client to a server, in such a way that two operations are indistinguishable to a server [73]. This means that a graph state can be constructed by a server, but without revealing the connectivity of the graph to them.

1.5 Instantaneous Quantum Poly-time Computations

In contrast to universal quantum computing, described in Section 1.1, it may be that sub-universal models of quantum computation are easier to implement. This may, for example, be because they: require a restricted gate set, which is more readily implementable; demand limited connectivity between qubits, which is native to available hardware; or are fault-tolerant to some extent. Such models, with accompanying theoretical grounds to believe they would be hard to simulate classically, could expedite demonstrations of quantum computational supremacy. Several such intermediate, sub-universal models of quantum computation, such as the one clean qubit model [159], the Ising model [160], the BosonSampling model [61] and random circuit sampling [62], have been developed with this goal of early implementation in mind. The class of *Instantaneous Quantum Polynomial-time* (IQP) circuits [113] is another such sub-universal model with significant practical advantages [24, 64, 161].

IQP circuits consist of commuting gates, in contrast to the non-commuting gate set needed for universal computations. This property could theoretically be used to parallelise the computation and reduce the requirement for quantum memory, which would otherwise be physically hard to achieve.[18] In spite of this limited gate set, IQP circuits are believed to remain hard to classically simulate [63, 65]; even in the presence of realistic noise [64, 162] and on well motivated architectures [24, 64]. Indeed, current predictions put the number of qubits one expects to require for a demonstration of quantum computational supremacy using IQP circuits within the realm of what is thought to be possible in the near future [163]. Further, circuits consisting of only commuting gates are of significance for super- and semi-conductor qubit implementations, where they are simpler to implement fault-tolerantly than gates drawn from a

[18]Quantum memory is hard to achieve in the sense that it is difficult to store quantum states for long periods of time without them succumbing to noise.

fully universal set [161].

Because of this relationship between IQP circuits and demonstrations of quantum computational supremacy on near-term devices, they are the ideal subject of study when one is concerned with measuring the potential for such demonstrations, as we are in Chapter 2 and Chapter 3. For the same reason, it is of great interest that there exist efficient methods for verifying some IQP computations without classical simulation [24, 75, 113]; an example of which we give in Chapter 4. It can also be reasonably argued, as we do in Chapter 3 and Chapter 5, and as has been done elsewhere [113, 164, 165], that demonstrations of quantum computational supremacy when addressing real world applications would be possible using IQP circuits. To facilitate discussions in these later chapters, we formally introduce the class of IQP circuits in Section 1.5.1, and detail some of the theoretical results mentioned above in Section 1.5.2. We position IQP amongst other sub-universal models of quantum computation in Section 1.5.3, and discuss the physical implementation of IQP circuit in Section 1.5.4.

1.5.1 Definitions

IQP circuits are comprised of commuting gates, with *polynomial* referring to the number of such gates, and *instantaneous* alluding to the theoretical capacity to apply the gates simultaneously as discussed in Section 1.5.4.

Definition 1.5.1 (Instantaneous quantum poly-time (IQP) circuit [63])**.** *An* IQP *circuit on n qubits is a quantum circuit with the following structure: each of the polynomially many gates in the circuit is diagonal in the* X *basis, the input state is $|0\rangle, \dots, |0\rangle$, and the output is the result of a computational basis measurement on a specified set of output qubits.*

Often it is helpful to consider, as we will regularly do, the equivalent case where gates which are diagonal in the Z basis are sandwiched between layers of H gates.

It should be ensured that these circuits can be efficiently represented. One possible representation of an IQP circuit is by an X-program.

Definition 1.5.2 (X-program [113])**.** *An* X*-program is described by a pair $(\boldsymbol{Q}, \theta) \in \{0,1\}^{n_a \times n_p} \times [0, 2\pi)$. Each row $\boldsymbol{q} \in \{0,1\}^{n_p}$ of $\boldsymbol{Q}$, called a* program element*, corresponds to the gate*

$$\exp\left(\mathrm{i}\theta \bigotimes_{j:\boldsymbol{q}_j=1} \mathsf{X}_j \right).$$

Given a gate for each program element, their product defines the action of the X*-program. This is equivalent to an evolution by the Hamiltonian*

$$\sum_{i=1}^{n_a} \bigotimes_{j:\boldsymbol{Q}_{ij}=1} \mathsf{X}_j$$

for a time θ.[19] *Note also that it may sometimes be convenient to allow θ to vary between program elements.*

[19]Note that i, when not used as an index, is the imaginary unit.

An example of one such X-program is

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \; \theta = \frac{\pi}{8} \quad \rightarrow \quad \exp\left(i\frac{\pi}{8}X_1X_3\right)\exp\left(i\frac{\pi}{8}X_2X_3\right). \tag{1.18}$$

Alternate descriptions of IQP circuits, such as defining gates by their diagonal entries and the qubits on which they act, are possible [63]. In that case the gates may act on at most $O(\log n)$ qubits in order to ensure an efficient representation. We choose the X-program representation as it allows us to understand the relationships within large classes of IQP circuits, which we discuss in Section 1.6.3 and utilise in Chapter 4.

Applying an X-program to a computational basis state $|0^{n_p}\rangle$, and measuring the result in the computational basis, constitutes an IQP circuit. Using the random variable X to represent the distribution of output samples, the probability distribution of outcomes $\widetilde{x} \in \{0,1\}^{n_p}$ is

$$\mathbb{P}(X = \widetilde{x}) = |\langle \widetilde{x}| \exp\left(\sum_{i=1}^{n_a} i\theta \bigotimes_{j:\boldsymbol{Q}_{ij}=1} X_j\right)|0^{n_p}\rangle|^2. \tag{1.19}$$

1.5.2 Hardness Results and Their Robustness to Noise

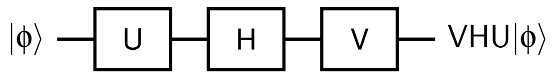
Because of the close connection between IQP circuits and demonstrations of quantum computational supremacy on near-term devices, we will repeatedly utilise them throughout this work. We outline some of the results pertaining to this connection now. We consider weak classical simulation, defined in Section 1.3.1, of IQP circuit families, and how the possibility of weak simulation depends on the accuracy required and the noise present in the system.

The strictest notion of weak simulation, where one demands simulation of the ideal circuit to within a multiplicative error, is known to be impossible in general, assuming the non-collapse of PH.

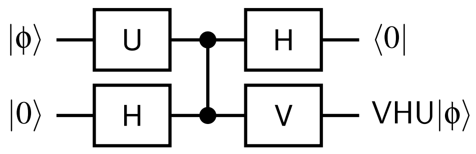
Theorem 1.5.1 (From [63])**.** *If all families of* IQP *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c < \sqrt{2}$ *then* $PH = \Delta_3$.

The proof of Theorem 1.5.1 utilises post-selection, with a post-selected circuit having, in addition to an output register $\mathcal{O}$, a register of lines called the post-selection register $\mathcal{P}$. The output distribution on $\mathcal{O}$ is then taken to be $P(\mathcal{O} = \boldsymbol{x}|\mathcal{P} = \boldsymbol{0})$. In practice this may be implemented by running a computation many times and selecting only those runs for which $\mathcal{P} = \boldsymbol{0}$. As the probability that $\mathcal{P} = \boldsymbol{0}$ may be exponentially small, in reality this may incur an exponential overhead, which provides an insight into the power of post-selection. We will use the prefix post- to indicate the class of problems that can be solved by the post-selected version of circuits from the original class.

Post-selection significantly boosts the power of IQP, with post-IQP = post-BQP [63]. To realise this consider the universal gate set $\{T, CZ, H\}$. Post selecting circuits built from this gate set gives us post-BQP. T and CZ can be constructed from gates of the form $e^{i\frac{\pi}{8}Z}$ and $e^{i\frac{\pi}{8}Z\otimes Z}$, and so it remains to show that H can be implemented using a circuit in post-IQP in order to show that post-IQP = post-BQP. Indeed the Hadamard Gadget of Figure 1.9 is such a circuit.



(a) H intermediate in BQP circuit.



(b) Implementation of intermediate H using a post-IQP circuit. Note that H acts only at the beginning and end of the circuit, as permitted in IQP circuits.

Figure 1.9: **Hadamard Gadget [63].** Figure 1.9b recreates the effect of the action of a H gate acting intermediately between other gates, as seen in Figure 1.9a. $|\phi\rangle$ is an arbitrary input state, and $\langle 0|$ represents post selection onto $|0\rangle$.

Noting post-BQP $=$ PP [166] reveals that PH $\subseteq \mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\text{post-IQP}}$ [167], which should be compared to $\mathsf{P}^{\text{post-BPP}} \subseteq \Delta_3$ [168]. This shows the comparative power of post-IQP, as an oracle, as compared to post-BPP, assuming that there is no collapse of PH. Indeed, the proof of Theorem 1.5.1 proceeds by demonstrating that if such a classical simulation where possible, then post-IQP $=$ post-BPP, and hence that PH would collapse to its third level.

Theorem 1.5.1 is remarkable in its demonstration that quantum computers which are very much weaker than a universal BQP machine are likely to be impossible to efficiently simulate with a classical computer. These results are, however, proven in the setting where one demands a classical simulator produce samples which are within a multiplicative error of the ideal. It better reflects the power of real quantum computers to allow the classical simulator to be wrong up to ℓ_1-norm distance, and in this case too hardness results exist.

Theorem 1.5.2 (From [65]). *Assume either one of two conjectures, relating to the hardness of approximating the Ising partition function and the gap of degree 3 polynomials. If it is possible to weakly classically simulate all families of* IQP *circuits to within an ℓ_1-norm distance of $\frac{1}{192}$ then $PH = \Delta_3$.*

The two conjectures in Theorem 1.5.2 are that it should be #P-hard to approximate the partition function of an Ising model on a randomly weighted complete graph, and to approximate $\text{gap}(f) := |\{x : f(x) = 0\}| - |\{x : f(x) = 0\}|$ of a randomly chosen degree 3 polynomial f over $\mathbb{F}_2$, up to constant multiplicative error on average. While both hold in the worst-case [65, 169, 170] (i.e. for at least one instance of these problems) it is not known if this is the case on average (i.e. for a large fraction of instance) as these conjectures require.

Both the partition function and $\text{gap}(f)$ also emerge as amplitudes in the output probability distributions of families of IQP circuits.[20] It can be shown that if there was a classical algorithm to sample from the output distribution of all IQP circuits to within constant ℓ_1-norm distance, then these samples would be enough for a BPP machine with access to an NP oracle to give an additive error approximation of these ampli-

[20]Such functions emerge as properties of the output distributions of other sub-universal models of quantum computation, such as the one clean qubit model [171].

tudes. Such an algorithm would be in Δ_3, the third level of PH. The existence of an algorithm approximating these amplitudes is guaranteed by Stockmeyer's Counting Theorem [172], and allows for a multiplicative error approximation if the output probabilities of this family of IQP circuits *anti-concentrate*. This is to say that the amplitude of a random outcome in the output distribution is likely to be high, which was proven to be the case for these circuits [65]. In summary this implies that there exists a procedure in Δ_3 which can be used to approximate, to within a multiplicative error, quantities that are conjectured to be #P-hard to approximate in this sense. Since $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$, and because the proposed algorithm for approximating these #P-hard problems is in Δ_3, the existence of such an algorithm would collapse PH.

Similar proof techniques can be used to show a collapse of PH in the case of weak classical simulation of a selection of other sub-universal classes of quantum circuits [61]. In particular, in [24] a family of nearest neighbour, translation invariant, 2 local (which is to say the corresponding Hamiltonian can be decomposed into the sum of hermitian operators acting on only 2 qubits), constant depth IQP circuits called 2-dimensional dynamical quantum simulators (2D-DQS) are defined. The name references the 2D square lattice architecture involved, as motivated by the connectivity of several near-term devices, and that they could be realised with sub-universal quantum simulators. Indeed, because this architecture is similar to that of the NQIT Q20:20 device discussed in Chapter 2, we choose these schemes as a benchmark of that device. Architecture I from [24] is seen in Protocol 1.5.1, the construction is summarised in Figure 1.10, and the relevant hardness theorem is stated in Theorem 1.5.3.

Protocol 1.5.1 A description of an instance of the 2D-DQS problem introduced by [24]. E and V are the edge and vertex set respectively of a $N_x \times N_y$ 2D square lattice.

1: Choose $\tau \in \{0,1\}^{N_x \times N_y}$ uniformly at random.
2: Initialise the product state:

$$|\phi_\tau\rangle = \bigotimes_{i=1}^{N=N_x \times N_y} \left(|0\rangle + e^{i\tau_i \frac{\pi}{4}}|1\rangle\right)$$

3: Allow system to evolve for time $t = 1$ according to the nearest neighbour, translation invariant, *Ising Hamiltonian*

$$H := \sum_{(i,j)\in E} \frac{\pi}{4} Z_i Z_j - \sum_{i \in V} \frac{\pi}{4} Z_i.$$

This is equivalent to applying CZ operations on each edge.
4: Measure all qubits in the X basis.

Theorem 1.5.3 (From [24])**.** *Assume two conjectures, relating to the hardness of approximating the Ising partition function and that the output probabilities of samples from Protocol 1.5.1 anticoncentrate. If it is possible to weakly classically simulate all circuits of the form of Protocol 1.5.1 to within an ℓ_1-norm distance of $\frac{1}{22}$ then $PH = \Delta_3$.*

Unlike in the case of Theorem 1.5.2, anticoncentration is conjectured and numerically

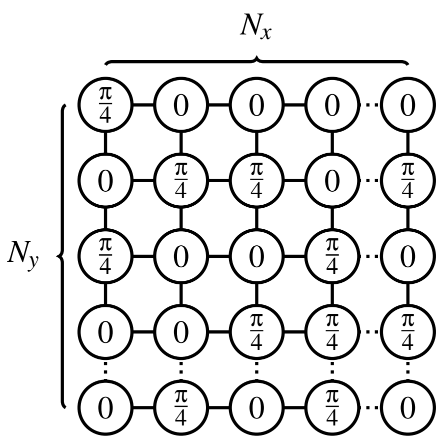


Figure 1.10: **An example of an instance of the 2D-DQS problem for quantum computational supremacy, detailed in Protocol 1.5.1 and introduced in [24].** The value in each qubit describes the state of initialisation while the lines connecting them indicate the application of a CZ gates between those qubits. Each qubit of the resulting state is measured in the Pauli-X basis.

justified in [24], rather than proven. In addition, a procedure for the certification of the correctness of the final state before readout exists for this scheme, as we discuss in Section 1.6.3.

The result of Theorem 1.5.3 goes some way to understanding the persistence of quantum computational supremacy in realistic settings. This is namely by reducing long range interactions between qubits which would otherwise incur significant overheads when mapping to realistic hardware [173]. In a further step towards understanding this setting, in [64] a scheme to generate IQP circuits, which have depth $O(\log n)$ with high probability, is introduced. These circuits can be implemented, with the addition of SWAP gates, on a 2D lattice in depth $O(\sqrt{n}\log n)$ [174] while, of the circuits produced by the scheme, a constant fraction cannot be weakly classically simulated to within constant ℓ_1-norm distance.

However, the noise model remains unrealistic, with models such as constant independent noise applied to each qubit leading to a distribution that is far in ℓ_1-norm distance from the original. Indeed for most IQP circuits, and in particular those from Theorem 1.5.2, if independent depolarising noise with a constant probability p as defined in equation (1.16) is applied to each qubit immediately before measurement then the resulting probability distribution can be approximately sampled from up to constant ℓ_1-norm distance in general [64]. However, a multiplicative error approximation is still impossible [162]. This does not contradict the threshold theorem mentioned in Section 1.3.3, which allows for constant error below some value, as no error correction techniques have been used. Indeed it can also be shown that simple classical error correction techniques can be used to create an equivalent circuit producing a distribution which is a constant ℓ_1-norm distance from the ideal, which cannot be classically simulated [64]. This error correction technique destroys the anti-concentration property on which the classical simulation technique relies, but produces outputs from which outputs from the original IQP circuit can be recovered. That being said, the more general case of constant independent noise being applied to each gate enables the classical simulation of a wide family of circuits [175].

Collectively these results give strong justification for believing that IQP circuits will allow us to exploit the power of quantum computing on near-term devices. As such, the focus of later chapters will be on benchmarking and verifying implementations of these circuits, and identifying applications where they will be of impact.

1.5.3 A Comparison With Other Sub-Universal Classes of Quantum Computation

Besides IQP, the most promising proposals for demonstrating quantum computational supremacy are BosonSampling [61] and random circuit sampling [62]. All three proposals are sampling problems, which are preferred as they better reflect the capabilities of near-term devices than do decision problems. While BosonSampling is little explored in this work, it is a cornerstone of work on sub-universal models of quantum computation and quantum computational supremacy. Indeed recent experiments have demonstrated quantum computational supremacy with a related model [176]. On the other hand, random circuit sampling forms the basis of another recent demonstration of quantum computational supremacy [60], and will be used in our work of Chapter 3.

BosonSampling is the class of problems which formalise sampling from the distribution of detection outcomes from a linear optical network of non-interacting photons. Importantly there are solid grounds to believe that the classical simulation of all BosonSampling output distributions to within a constant additive error is impossible [61]. As with IQP circuits, the necessary proofs are based on conjectures: the first of which relates to the anticoncentration of the permanent[21] of a random matrix, mirroring the anticoncentration result that was proven for IQP circuits; the second of which is the "permanent-of-Gaussians" conjecture, which states that the permanent of a matrix of Gaussian random variables should be hard to approximate up to small multiplicative error on average, mirroring the average case conjectures made in Theorem 1.5.2. Besides this result there have been others showing the robustness [177, 178], and vulnerability [179–181], of BosonSampling to noise. There have been several experimental demonstrations of models related to BosonSampling [176, 178, 182], as well as investigations into the limits of classical simulations of such experiments [129, 130].

Random circuits, like IQP circuits, are native to the circuit model, and closely mirror the capabilities of near-term superconducting computing devices [62, 85]. A random circuit, for a fixed number of qubits n and coupling map G_n, is generated by applying $m = \text{poly}(n)$ uniformly random two-qubit $\text{SU}(4)$ gates between qubits connected by edges of G_n. Here, 'uniformly random' means according to the Haar measure [183]. *Random Circuit Sampling* (RCS) is the task of sampling from the output distribution of random circuits.[22] To perform RCS approximately is to sample from a distribution close to that produced by the random circuit. This has been shown to be hard [66, 186].

[21] Similarly to IQP circuits, the permanent emerges as amplitudes in output distributions of BosonSampling circuits.

[22] In fact, as with IQP circuits but unlike in the case of BosonSampling, the distribution of output samples from random circuits can be proven to anticoncentrate [184, 185].

Theorem 1.5.4 (Informal [66])**.** *There exists a collection of coupling maps G_n, with one for each n, and procedure for generating random circuits respecting each G_n, for which there is no classical randomised algorithm that performs approximate* RCS*, to within inverse polynomial ℓ_1-norm distance error, unless then $PH = \Delta_3$.*

The conditions imposed on which coupling maps and circuit generation procedures are covered by this theorem are quite mild. For example, there are random circuits obeying the connectivity restraints of 2D square lattice coupling maps which illustrate Theorem 1.5.4 [66, 85]. In particular Ref.[85] proposes taking G_n to be a $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ square lattice, and constructing random circuits by acting 2-qubit gates along n^2 edges selected at random (with replacement). This architecture is particularly relevant for NISQ technology.

As with Theorem 1.5.2 and similar results for BosonSampling, Theorem 1.5.4 relies on conjectures. One advantage that both RCS and BosonSampling have over IQP is with regards to conjectures on the hardness of calculating output probabilities. In particular, 'average-to-worst-case reduction' results are known for RCS and BosonSampling. In brief such results ensure that if calculating an output probability of a class of circuits is known to be hard in the worst-case, then it can be shown that this is also true in the average-case. This means that for many classes of RCS and BosonSampling circuits it can be shown that exactly calculating most output probabilities of most of the circuits is hard. This strengthens our belief in conjectures such as the permanent-of-Gaussians conjecture and similar ones made in the case of RCS. Similar conjectures, such as those made in Theorem 1.5.2, stand with less support as no average-case results are known. In all cases these conjectures are not proven as these average-to-worst-case results only prove the hardness of exact calculation of output probabilities, rather than approximate calculation.

1.5.4 IQP in MBQC

As outlined in Section 1.6, MBQC facilitates blind and verifiable implementations of quantum computations. This is of particular concern to us in Chapter 4, where we give a protocol to realise an IQP circuit blindly. An implementation of an IQP computation in MBQC is a prerequisite for that scheme. As such, in the following we give the equivalent MBQC implementation of a given X-program. This approach was originally outlined in [113], while we give a thorough exploration here to introduce definitions and ideas which facilitate later discussion, particularly in Chapter 2 and Chapter 4.

Notice that we can rewrite equation (1.19) as

$$\mathbb{P}(X = \tilde{x}) = |\langle \tilde{x} | \mathsf{H}^{n_p} \left(\prod_{i=1}^{n_a} \exp \left(i\theta \bigotimes_{j:\boldsymbol{Q}_{ij}=1} \mathsf{Z}_j \right) \right) |+^{n_p}\rangle|^2. \qquad (1.20)$$

In combination with Lemma 1.5.1, this allows us to build an MBQC implementation of an X-program by building an implementation of each row of $\boldsymbol{Q}$.

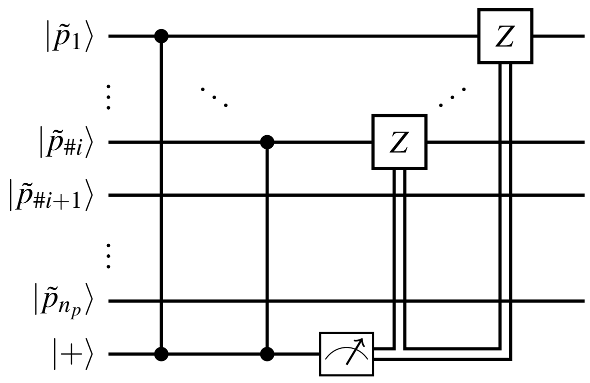

$|\tilde{p}_1\rangle$
$|\tilde{p}_{\#i}\rangle$
$|\tilde{p}_{\#i+1}\rangle$
$|\tilde{p}_{n_p}\rangle$
$|+\rangle$
Z
Z

Figure 1.11: **An MBQC implementation of equation** (1.21). The input qubits $\{p_j\}_{j=1}^{n_p}$ are rearranged so that if $\#i$ is the Hamming weight of row i of the matrix $\boldsymbol{Q}$, then for $k = 1,\dots,\#i$ each $\tilde{p}_k$ corresponds to one p_j such that $\boldsymbol{Q}_{ij} = 1$ and for $k = \#i+1,\dots,n_p$ they correspond to p_j such that $\boldsymbol{Q}_{ij} = 0$. The ancillary qubit measurement is in the basis $\{|0_{2\theta}\rangle, |1_{2\theta}\rangle\}$ of equation (1.24).

Lemma 1.5.1. *The circuit of Figure 1.11 implements*

$$\exp\left(i\theta \bigotimes_{j:\boldsymbol{Q}_{ij}=1} \mathsf{Z}_j \right). \tag{1.21}$$

Proof. We show that the effect of Figure 1.11 and equation (1.21) is the same on all inputs. We can rewrite equation (1.21) as

$$\cos\theta \mathsf{I}_{n_p} + \mathrm{i}\sin\theta \bigotimes_{j:\mathbf{Q}_{ij}=1} \mathsf{Z}_j. \tag{1.22}$$

The action of equation (1.22) on a computational basis state $|p\rangle = |p_1\rangle \dots |p_{n_p}\rangle$, $p_j \in \{0,1\}$ has two possible outcomes.

1. If $\sum_{j:\mathbf{Q}_{ij}=1} p_j$ is even, then there will be a phase change of $\cos\theta + \mathrm{i}\sin\theta$, as the $\bigotimes_{j:\boldsymbol{Q}_{ij}=1} \mathsf{Z}_j$ operator will extract an even number of negatives.
2. If $\sum_{j:\mathbf{Q}_{ij}=1} p_j$ is odd, then the phase change will be $\cos\theta - \mathrm{i}\sin\theta$.

Hence, depending on the parity of $|p\rangle$ in the positions where $\boldsymbol{Q}_{ij} = 1$, the effect is to produce one of the two states:

$$(\cos\theta \pm i\sin\theta)\,|p\rangle = e^{\pm i\theta}|p\rangle \tag{1.23}$$

We show the effect of Figure 1.11 is the same. Consider a permutation of the states $|\tilde{p}_1\rangle,\dots,|\tilde{p}_{\#i}\rangle,\dots,|\tilde{p}_{n_p}\rangle$ in Figure 1.11 such that the first $\#i$ qubits are the ones for which $\boldsymbol{Q}_{ij} = 1$.

The action of the CZ gates is to check the parity of $|1\rangle$'s in the input as each appearance of a $|1\rangle$ will flip the bottom *ancillary* qubit between the states $|+\rangle$ and $|-\rangle$. After the action of all CZ operators, we have the state $|p\rangle|+\rangle$ if there is an even number of

$|\tilde{p}_k\rangle = |1\rangle$ for $k = 1, \dots, \#i$ and $|p\rangle|-\rangle$ if this number is odd. Making a measurement of the ancillary qubit in the basis

$$\{|0_{2\theta}\rangle, |1_{2\theta}\rangle\} = \left\{ \frac{1}{\sqrt{2}} \left(e^{-i\theta}|+\rangle + e^{i\theta}|-\rangle \right), \frac{1}{\sqrt{2}} \left(e^{-i\theta}|+\rangle - e^{i\theta}|-\rangle \right) \right\} \tag{1.24}$$

leaves us with one of the two states $\pm e^{-i\theta}|p\rangle$ in the odd parity case and with the state $e^{i\theta}|p\rangle$ in the even parity case. The negative sign preceding the exponential term in the odd parity case comes from measuring the state $|1_{2\theta}\rangle$ (a measurement outcome of 1) and the positive sign comes from measuring $|0_{2\theta}\rangle$.

In the case of a measurement outcome 1, we then apply Z operators to all unmeasured qubits to ensure that the resulting states are as in equation (1.23) and with the same dependency of the sign on the parity of $|p\rangle$. □

We can now identify an MBQC implementation of IQP circuits.

Lemma 1.5.2. *A graph and measurement pattern can always be designed to simulate an* X*-program efficiently.*

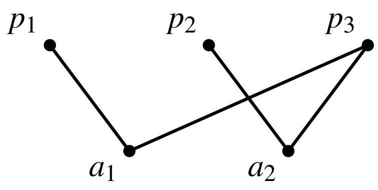
We give an outline of the proof of Lemma 1.5.2. Lemma 1.5.2 is proven formally by Protocol 1.5.2, although we will require some further terminology to appreciate that.

Proof. (*Outline*) Producing the distribution in equation (1.20) can be achieved by inputting the state $|+^{n_p}\rangle$ into a circuit made from composing circuits like the one in Figure 1.11, with one for each term of the product in equation (1.20), and measuring the result in the Hadamard basis. The Z corrections commute with the CZ operations and therefore can be moved to the end of the new, larger circuit. Further the Z corrections, conditional on the measurement outcomes of the ancillary qubits, can be implemented via classical NOT operations after the Hadamard basis measurement of qubits $\{p_j\}_{j=1}^{n_p}$. As such all entanglement precedes all measurement, with no operations acting on unmeasured qubits after the initial entanglement, as is consistent with the philosophy of MBQC. □

Notice that this implementation is non-adaptive, which, assuming the free preparation of graph states, reveals the origin of the term instantaneous in the name of the IQP class. Hence, this approach will be useful for implementations on near-term devices, as we utilise in Chapter 2, since the computation can be parallelised to one round of entanglement and measurement.

The entanglement pattern which is implicit in the proof of Lemma 1.5.2 is that of an *undirected bipartite graph*, which we will refer to as an IQP graph.

Definition 1.5.3 (IQP Graph)**.** *An* undirected bipartite graph*, which we refer to as an* IQP graph*, consists of a bipartition of vertices into two sets P and A of cardinality n_p and n_a respectively. We may represent such a graph by $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$. An edge exists in the graph when $\boldsymbol{Q}_{ij} = 1$, for $i = 1, \dots, n_a$ and $j = 1, \dots, n_p$, with no edge when $\boldsymbol{Q}_{ij} = 1$. We call the set P* primary vertices *and the set A* ancillary vertices.



$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Figure 1.12: **Bipartite graph**. An example of a bipartite graph described by matrix the $\boldsymbol{Q}$ of the X-program of equation (1.18). Here, $n_p = 3$ and $n_a = 2$ while the partition used is $P = [p_1, p_2, p_3]$ and $A = [a_1, a_2]$.

By referring to the bottom qubit of Figure 1.11 as the ancillary qubit and the others as primary qubits we understand why this type of graph is relevant and how the X-program matrix $\boldsymbol{Q}$, interpreted and a bipartite graph, exactly describes the entanglement pattern. Throughout this work, we refer to $\boldsymbol{Q}$ interchangeably as a matrix corresponding to an X-program and a graph. Figure 1.12 gives an example of this relationship.

Throughout this work we refer to the state built in Lemma 1.5.2 as an IQP *state*.

Definition 1.5.4 (IQP State)**.** *Using the notation introduced in Definition 1.4.2, states of the form $E_{\boldsymbol{Q}}|+\rangle^{n_a+n_p}$, where $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$, or* Z *rotations there of, shall be referred to as* IQP states.

With this last piece of terminology, we can now formalise the protocol detailed in the proof of Lemma 1.5.2 in Protocol 1.5.2. We will often refer to IQP computations implemented via Protocol 1.5.2 as instances of IQP-*MBQC*.

1.6 Delegated Quantum Computing

At present, and for the foreseeable future, quantum computing devices are few in number, come with a high maintenance cost, and require specialist facilities and highly trained individuals to operate. This mirrors the situation which applies to classical High Performance Computers (HPC). Therefore it is likely that, as with access to classical HPC, access to quantum computers will be remote, with the devices themselves maintained by resource rich specialists. Indeed, such Delegated Quantum Computing (DQC) is the approach taken today [83, 84], while work on a ‘Quantum Internet’ may permit this approach to be further extended and improved [187].

As input from the *Client*, DQC schemes take a classical description of a computation, and the input to that computation. After possibly several rounds of communication with the *Server*, the result of that computation is returned to the Client.[23] During the design of DQC schemes there are two considerations to be made. Firstly, implementing a computation in this manner should allow the Client to reduce the resources they require below that which they would need to implement the computation alone. Ideally this might mean removing the requirement that the Client perform quantum operations of any kind. Secondly, a DQC scheme may need to account for the Client’s mistrust

[23]Throughout this thesis we will use ‘the Client’ and ‘the Server’ to refer directly to two parties participating in a DQC scheme. Other works may instead use ‘Alice’ and ‘Bob’ respectively.

Protocol 1.5.2 An implementation of IQP computation in MBQC. IQP computations implemented in this way are referred to as an instance of IQP-MBQC.

Input: Graph $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$, measurement angle θ
Output: $\widetilde{x} \in \{0,1\}^{n_p}$

1: **Preparation:** Generate states $|+\rangle = |p_j\rangle$ and $|+\rangle = |a_i\rangle$ for $j \in \{0,...n_p\}$ and $i \in \{0,...n_a\}$.
2: **Entanglement:** Implement $E_{\boldsymbol{Q}}$ on the generated qubits.
3: **Measurement:** Measure primary qubits in the Hadamard basis and ancillary qubits in the basis of equation (1.24) to obtain measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$. Note that the basis used here differs from that of Protocol 1.4.1.
4: **Correction:** Perform the following corrections on the classical measurement outcomes to generate the output $\widetilde{x}$.

$$\widetilde{x}_j = s^p_j + \sum_{i:\boldsymbol{Q}_{ij}=1} s^a_i \pmod 2 \tag{1.25}$$

The correction in equation (1.25) is arrived at by considering which ancillary qubits may pass a Z correction to a given primary qubit, as detailed in Figure 1.11. In particular this is all those which correspond to a program element affecting the particular primary qubit.

of the Server, which could entail providing guarantees that a protocol is secure against malicious behaviour by the Server.

For example, the Client may wish to ensure that the details of the computation being performed are kept from the Server, possibly because they contain some private information. Blind Quantum Computing (BQC) looks to address this concern, and we discuss this in Section 1.6.1. Further, the Client may have cause to believe the Server will incorrectly implement the requested protocol, either because they are malicious, or because they lack the required computing power. Several techniques, referred to under the banner Quantum Characterisation, Verification, and Validation (QCVV), have emerged to address this.[24] At the highest level of assurance, the Client may request the Server be able to prove they have accurately implemented any quantum computation the Client requests, which we call *verification*, as we will discuss in Section 1.6.2. At a lower level of assurance the Client might request the Server prove they can accurately implement a sub-universal class of quantum computations, as we discuss in Section 1.6.3, or just that they are capable of some computation that is outside of the capacity of classical computers, as discussed in Section 1.6.4

The resources required to perform verification of universal quantum computation is typically very high, and certainly inaccessible by NISQ devices. In Chapter 4 we aim to reduce these resource requirements by specialising to a particular IQP computation. The approach derived in Chapter 4 depends heavily on the techniques for blind

[24]Popular resources providing a thorough overview of both BQC and QCVV are [188, 189].

quantum computation introduced in Section 1.6.1, particularly Universal Blind Quantum Computing (UBQC), and is inspired by the verification techniques discussed in Section 1.6.2, particularly Verifiable Universal Blind Quantum Computing (VUBQC). The computation we implement blindly in Chapter 4 is similar to the one used as part of the Shepherd-Bremner IQP Hypothesis Test, introduced in Section 1.6.3.

In Section 1.6.4 we focus in particular on Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking. The figures of merit used as part of these benchmarking schemes are also used in our work of Chapter 3. This is namely because they provide a means for a purely classical client to assess the performance of a quantum device.

1.6.1 Universal Blind Quantum Computing

Intuitively, the Server is 'blind' to the computation being performed if the process of implementing it is indistinguishable from the process of implementing any computation in some large class. A commonly used notion of blindness is to demand that only an upper bound on the size of the computation should be learnt by the Server. The first protocol to achieve this allowed a client with quantum memory, and the ability to perform Pauli operations, to perform universal quantum computing with the assistance of a universal quantum server [190]. The Client and Server exchange quantum states without details of the computation being revealed to the Server. In that case the resources required by the Client, namely the quantum memory, depend on the computation. This dependency is removed by shifting the storage requirements to the Server and permitting the Client to perform arbitrary operations on a constant number of qubits [191].

By utilising MBQC, the requirements on the Client can be further reduced to just the ability to prepare and send single qubits to the Server [96]. As discussed in Section 1.4.1, an MBQC computation on an $N_x \times N_y$ brickwork state is completely defined by the measurement angles θ_i. In the UBQC scheme [96] the Client conceals these by preparing $N_x \times N_y$ states $|\psi_i\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{r_i} e^{i\phi_i}|1\rangle\right)$, where $r_i \in \{0,1\}$ and $\phi_i \in \left\{0, \frac{\pi}{4}, \dots \frac{7\pi}{4}\right\}$ are chosen uniformly at random.[25] The Server is sent these states, but not the values of r_i, and uses them to construct an $N_x \times N_y$ brickwork state. As opposed to requesting a measurement in $\mathsf{M}^{\theta'_i}$, as is done in the MBQC computation outlined in Protocol 1.4.1, the Client requests a measurement in $\mathsf{M}^{\theta'_i - \phi_i}$. Since the random Z rotation at preparation commutes through the entanglement used to build the brickwork state, this measurement undoes the ϕ_i rotation at preparation. By setting $m_i = b_i \oplus r_i$, where b_i is the measurement returned by the Server, the Client recovers the 'true' measurement outcome, while it remains hidden from the Server. This allows the Client to otherwise proceed as if performing a normal MBQC computation on a brickwork state. As such both the measurement angles and the measurement outcomes have been *one-time padded*, while the distribution of messages sent to the Server is maximally mixed. UBQC has been demonstrated experimentally [192], while in Chapter 4 we will use a similar technique, but with reduced resource requirements, to implement an IQP computation blindly. An alternate but similar take on this idea is to have the Client

[25]In other work r and ϕ may be combined as a single variable, while it is convenient here to reason about them separately.

receive and measure the qubits, rather than prepare and send them [193], in which case the measurement angles can be seen to be kept from the Server.

The notion of blindness used when UBQC was introduced [96] limits the Server's information to being completely determined by some function $L(x)$ of the Client's input x. Such a scheme is called *blind while leaking at most* $L(x)$, with UBQC achieving blindness while leaking at most N_x and N_y. This definition is insufficient if, for example, the Server is able to learn more information about x in the future. To address this, an alternate security definition was introduced which relies on Abstract Cryptography, as introduces in Section 1.7, which gives assurances about the compositional security of a BQC protocol [194]. It transpires that UBQC, and the related receive and measure scheme [193], are both secure within this framework [194].

UBQC requires the Client to have the ability to perform quantum operations only in order to ensure that the state in the Server's possession is random, unknown to the Server, and known to the Client. This task, which is referred to as random remote state preparation (RSP) [21], may also be implemented using only classical operations by the Client, if conjectures about the post-quantum security of the learning with errors problem (LWE) are made [195, 196]. This is as opposed to the schemes discussed above where no computational hardness conjectures are made, and so which are *information-theoretically*, or *unconditionally*, secure. In this way, the quantum resources required by the Client to perform BQC can be removed, although the overall number of qubits and computing steps required are increased as compared to UBQC.

1.6.2 Verification of Universal Quantum Computation

A DQC protocol is said to be 'verifiable' if it is possible for the Client to detect malicious behaviour on the part of the Server. That is to say that the probability that the Server can convince the Client of a false statement is small, while the probability they will accept a correct one is high. BQC and verification are often linked, with one presenting a route to achieving the other. Trapification, where a BQC scheme is used to conceal easily computed computations, called *traps*, within a computation to be verified, is one approach to achieving this connection [72–74]. In the VUBQC protocol [73] the break operations discussed in Section 1.4.2 are used to isolate qubits, whose measurement outcomes are known, from the rest of the computation. These can then be used as traps, while their locations are hidden by adapting the UBCQ scheme discussed above. This approach has been demonstrated experimentally [197], and also has parallels in receive and measure verification schemes [72]. Again, the computational resources required by the Client may be reduced to being purely classical by using classical client random RSP [21, 195, 196].

Verification schemes which are not blind are also possible. Post-hoc verification [75, 76], deriving its name from the property that the verification can be performed at any time after the computation, is one such approach. This protocol uses that any BQP computation can be framed as an instance of the 2-local Hamiltonian problem [198, 199]. A k-local Hamiltonian is a hermitian operator $H = \sum_i H_i$, where each H_i is a hermitian operator which acts non-trivially on at most k qubits. For any language $L \in \mathsf{BQP}$ and input x, there exists a 2-local Hamiltonian whose smallest eigenvalue indicates if $x \in L$ or $x \notin L$. Indeed, the lowest energy eigenstate, referred to as the ground state, is a witness that $x \in L$, which can be sent to the Client to verify [200].

Given the circuit of concern, the corresponding Hamiltonian, called a Feynman-Kitaev Hamiltonian, can be found efficiently [198], while its ground state can be prepared by a BQP machine, and verified using single qubit measurements on multiple copies of the state [75, 201]. This scheme is not blind as the circuit required to build the ground state which encodes the solution to the computation is sent to the Server. It is also possible to adapt post-hoc verification to remove the need for quantum operations to be performed by the Client [77]. In this case, the challenge is to ensure the Server can be trusted to perform measurements, in contrast to the case of random RSP where it is ensured that they are trusted to produce random states. This is achieved by committing the Server, at the start of the computation, to performing a particular measurement, with the security of the commitment relying on the conjectured hardness of LWE.

Another approach to achieving classical client verification schemes, with information-theoretic security, is to introduce multiple servers [78, 79]. In this case the Client distributes the computation amongst the Servers, which share entanglement at the start of the computation but do not communicate thereafter. These schemes often make use of *self-testing*, which verifies the state and the share of the Servers, and their behaviour during the scheme. In such procedures, the Client acts as a referee in *non-local* games for which there is a unique optimal strategy that the Servers can carry out. The Servers return responses to questions posed by the Client, who uses these responses to determine if the optimal strategy is being used. The optimal strategy can only be achieved if the Servers share a particular state and perform particular operations, which the Client can verify by determining if the responses to its questions are satisfactory.

In [79], which is a blind verification scheme using two entangled but non-communicating universal servers, the CHSH game [6] is employed. Here the optimal quantum strategy is for the Servers to share and measure multiple copies of the EPR pair from equation (1.5). Verification is achieved by making these games indistinguishable from tests for the accuracy of measurement and state preparation, and from the computation itself. This indistinguishably intertwines the optimal strategy, which can be verified, with the computation. In fact, the power of one of the Servers in this scheme can be reduced to just that of a measurement device [80], which the Client can use to prepare the quantum state required for the computation on the second server.[26] Approaches based on post-hoc verification can also be used to perform non-blind verification using multiple servers and a classical client [76].

1.6.3 Verification of Sub-Universal Quantum Computation

The schemes outlined in Section 1.6.1 and Section 1.6.2 for BQC and verification require the Server has access to operations that are universal for BQP, even if a sub universal class of circuits is being implemented. As such, in the case that the Server is not a universal BQP machine the computational requirements of the protocols would

[26]A similar approach, using the verifiable preparation of a quantum state on one server, is used in [202] with multiple servers.

be too high for them to be used. This motivated the development of several schemes for verifying, both partially and wholly, sub-universal models of quantum computation.

One approach to doing this relies on post-hoc verification techniques to ensure that the ground state of the Feynman-Kitaev Hamiltonian corresponding to an IQP circuit of concern has been produced [75]. One of the prepared states can be used to sample from a distribution which can be verified to be a bounded ℓ_1-norm distance from the output distribution of the IQP circuit. Note that this is not a direct application of the post-hoc techniques discussed in Section 1.6.2 as IQP is not a decision class. While in general the ground state could not be prepared by an IQP circuit, this can be made to be the case for the 2D-DQS circuits described in Theorem 1.5.3. The state prepared by those circuits are the ground states of 2-local Hamiltonians, and so the preparation and sampling can be verified [24]. This notion of verification is stronger than that of the scheme we present in Chapter 4, where we can say something only about the computational power of the device of concern, but the limitations on the noise permitted are higher and may not be realisable in the very near future.

A second approach is to adapt VUBQC so as to reduce the resource requirements to that of sub-universal models. This technique has been used in the one-clean qubit model [203] and the Ising model [204], with trapification utilised as the means of performing verification. In Chapter 4 we take a similar approach, but, rather than trapification, take a proposed scheme for a classical client to benchmark the implementation of IQP computations [113], which we call the Shepherd-Bremner IQP Hypothesis Test, as our starting point. The Shepherd-Bremner IQP Hypothesis Test verifies that a device is capable of performing IQP computations, providing a path to demonstrating quantum computational supremacy, rather than verifying one IQP circuit in particular. This is achieved by concealing a small X-program, which enforces some property on the output, within a larger, hard to simulate, one. This property of the output can be used to aid verification. The security of this hiding relies on computational complexity assumptions that were since shown not to hold. However, in Chapter 4 we adapt VUBQC to recover information-theoretic security guarantees. We do so without requiring all of the operation necessary for VUBQC, although some operation required are outside of the capabilities of IQP circuits. To facilitate the discussion in Chapter 4 we outline the Shepherd-Bremner IQP Hypothesis Test scheme now.

The Shepherd-Bremner IQP Hypothesis Test

The Shepherd-Bremner IQP Hypothesis Test introduces some structure in the program elements, $\boldsymbol{q}_i$, of the X-program which defines an IQP computation. This, in turn, results in some structure in the distribution of the outputs. If the Server is asked to implement an X-program with this structure the Client can use the structure to check the Server's reply. A server capable of IQP computations can reproduce this structure by implementing the X-program. However, if the structure is well hidden, and therefore cannot be used by a weaker server to reproduce the computation, a server not able to perform IQP computations cannot generate outputs obeying the same rules.

The methodology of the Shepherd-Bremner IQP Hypothesis Test can be broken down:

Hard Problem The Client asks the Server to perform an IQP computation that is hard to classically simulate.

Known Structure The Client can check the solution to this computation because they know some secret structure that makes this checking processes efficient.

Hidden Structure The Server must be unable to efficiently uncover this structure.

The aim of the Shepherd-Bremner IQP Hypothesis Test is not to verify the accuracy of a particular IQP computation, as in the case of the schemes introduced in Section 1.6.2. Instead the aim is to measure a device's capacity to implement IQP computations, using a hard instance as a representative example of the IQP class. We refer to schemes of this form as IQP *Hypothesis Tests*. While performing well at an IQP Hypothesis Test for sufficiently large circuits could constitute a demonstration of quantum computational supremacy, an IQP Hypothesis Test should be able to give indications of a device's performance at even larger sizes than those required for such a demonstration. In this way an IQP Hypothesis Test is more powerful than the schemes for certifying demonstrations of quantum computational supremacy outlined in Section 1.6.4.

The particular known structure of the output which is used is its *bias*.

Definition 1.6.1. *If X is a random variable taking values in $\{0,1\}^n$ and $\boldsymbol{s} \in \{0,1\}^n$ then the bias of X in the direction $\boldsymbol{s}$ is $\mathbb{P}\left(X \cdot \boldsymbol{s}^T = 0\right)$ where the product is performed modulo 2. Hence, the bias of a distribution in the direction $\boldsymbol{s}$ is the probability of a sample from the distribution being orthogonal to $\boldsymbol{s}$.*

To calculate the bias of X in direction $\boldsymbol{s} \in \{0,1\}^{n_p}$, consider the matrix $\boldsymbol{Q_s}$ formed from all rows, $\boldsymbol{q}_i$ of $\boldsymbol{Q}$ from the X-program, $(\boldsymbol{Q}, \theta) \in \{0,1\}^{n_a \times n_p} \times [0, 2\pi]$, such that $\boldsymbol{q}_i \cdot \boldsymbol{s}^T = 1$. Consider then the linear code $\mathcal{C}_{\boldsymbol{s}}$ generated by $\boldsymbol{Q_s}$; which is to say, the linear code of which the columns of $\boldsymbol{Q_s}$ form the basis. Defining n_{s} to be the number of rows of $\boldsymbol{Q_s}$, and $\#\boldsymbol{c}$ to be the Hamming weight of a vector $\boldsymbol{c}$, allows us to understand equation (1.26), derived in [113].

$$\mathbb{P}\left(X \cdot \boldsymbol{s}^T = 0\right) = \mathbb{E}_{\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{s}}}\left[\cos^2\left(\theta\left(n_s - 2 \cdot \#\boldsymbol{c}\right)\right)\right] \tag{1.26}$$

We find that the bias of an X-program in the direction $\boldsymbol{s}$ depends only on θ and the linear code defined by the generator matrix $\boldsymbol{Q}_{\mathrm{s}}$. Therefore, a client, with the computational power to calculate the quantity of equation (1.26), has the necessary information to compute the bias and check the samples from the Server adhere to that bias.

In [113] the authors develop a protocol for building an X-program and a vector $\boldsymbol{s}$ performing this type of test. The code $\mathcal{C}_{\boldsymbol{s}}$ used to build the X-program is a quadratic residue code[27] while θ is set to $\frac{\pi}{8}$. In that case the bias value, which is $\cos^2\left(\frac{\pi}{8}\right)$ for their choice of X-program and $\boldsymbol{s}$, can be calculated in polynomial time. In particular, this value is high compared to that if uniformly random bit strings were produced.

The problem of hiding the known structure then reduces to hiding the direction $\boldsymbol{s}$ in which the bias will be checked. As such we now introduce some operations that allow

[27]Details of the procedure for producing a generator matrix of the quadratic residue code, along with an implementation of the procedure, can be found in [205].

us to randomise and pad $\boldsymbol{s}$, while maintaining our knowledge of the bias. In fact it is easier to understand how operations on the X-program matrix $\boldsymbol{Q}$ can be used to randomise $\boldsymbol{s}$, and so we begin with this.

Notice that adding the columns of $\boldsymbol{Q_s}$ to one and other, and duplicating columns of $\boldsymbol{Q_s}$, does not affect the linear code generated. Further, permuting the rows of $\boldsymbol{Q_s}$ does not change the Hamming weight of its vectors. As such the right-hand side of equation (1.26) is equal for all generator matrices in a *matroid.* [28]

Definition 1.6.2 (i-Point Binary Matroid [206])**.** *A i-point binary* matroid *is an equivalence class of matrices with i rows, defined over $\mathbb{F}_2$. Two matrices, $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, are said to be equivalent if, for some permutation matrix $\boldsymbol{R}$, the column echelon reduced form of $\boldsymbol{M}_1$ is the same as the column echelon reduced form of $\boldsymbol{R} \cdot \boldsymbol{M}_2$ (In the case where the column dimensions do not match, we define equivalence by deleting columns containing only* 0*s after column echelon reduction).*

These operations give some freedom to the X-programs which can be used without affecting the expectation in equation (1.26). However, such actions will affect the relationship between the rows and $\boldsymbol{s}$. In order to move to a new matrix within the same matroid, consider the right-multiplication with matrix $\boldsymbol{A}$ on $\boldsymbol{Q}$. Notice that $\boldsymbol{q}_i\boldsymbol{s}^T = (\boldsymbol{q}_i\boldsymbol{A})\left(\boldsymbol{A}^{-1}\boldsymbol{s}^T\right)$ and so rows originally non-orthogonal to $\boldsymbol{s}$ are now non-orthogonal to $\boldsymbol{A}^{-1}\boldsymbol{s}^T$. Hence, loosely speaking, we can identify the matroid which $\boldsymbol{Q_s}$ represents in $\boldsymbol{QA}$ by using $\boldsymbol{A}^{-1}\boldsymbol{s}^T$.[29]

In this way $\boldsymbol{s}$ can be randomised with such an operation $\boldsymbol{A}$. We now understand what to do to the X-program we are considering so that the value of the bias does not change. The matrix might also include additional rows orthogonal to $\boldsymbol{s}$, which do not affect the value of the bias. It is now simply a matter for the Server to implement the randomised X-program and for the Client to check the bias of the output in the new direction, with the final proposal of [113] outlined in Protocol 1.6.1. Importantly the number of samples required from the X-program to be implemented by the Server, denoted by K in Protocol 1.6.1, is constant, ensuring that the scheme can be implemented efficiently.

This discussion allows us to elaborate on the realisation of the methodology detailed above in the case of the Shepherd-Bremner IQP Hypothesis Test:

Hard Problem Given an X-program generated as in Protocol 1.6.1, produce bitstrings orthogonal to $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$ with probability close to $\cos^2\left(\frac{\pi}{8}\right)$. It is conjectured in [113]

[28]There are several cryptomorphic definitions of a matroid [206], of which binary matroids are a special case. Intuitively matroids generalise the notion of linear independence, with independent sets being one popular basis on which to formalise this intuition. There a matroid consists of a set, called the ground set, and a family of subsets, called the independent sets. The equivalence between the definition via independent sets and that of Definition 1.6.2 comes from identifying the ground set as the set of columns of a matrix, and the independent sets as those columns that are independent as vectors. The definition via matrices used in this work is convenient as it makes clear connections to binary linear codes which we use later.

[29]This ease with which operation on $\boldsymbol{Q}$ and the effect on outputs can be related is powerful. Indeed, as introduced in Section 1.5.2, by considering $\boldsymbol{A}$ to be a generator matrix of an error correcting code, simple noise channels can be corrected [64].

Protocol 1.6.1 An outline of the Shepherd-Bremner IQP Hypothesis Test of [113]. The complete construction can be found at [205].

Public: $\theta = \frac{\pi}{8}$
Client input: n_a prime such that $n_a + 1$ is a multiple of 8, $K \in \mathbb{Z}$.
Client output: $o \in \{0,1\}$

Client

```
 1: Set $n_p = \frac{n_a+1}{2} + 1$
 2: Take the quadratic residue code generator matrix $\boldsymbol{Q_r} \in \{0,1\}^{n_a \times (n_p - 1)}$
 3: Let $\boldsymbol{Q_s} \in \{0,1\}^{n_a \times n_p}$ be $\boldsymbol{Q_r}$ with a column of ones appended to the last column.
        ▷ Notice that the all 1 vector is a code word of the quadratic residue code,
    and that all rows are non-orthogonal to the vector $(0,\dots,0,1)$.
 4: Append $n_a$ rows to the bottom of $\boldsymbol{Q}_s$, which are random subject to having 0 as their
    last entry.
        ▷ Notice that all the appended rows are orthogonal to the vector $(0,\dots,0,1)$.
 5: Randomly permute the rows.
 6: Row reduce the matrix. Suppose that $\boldsymbol{Q} = \boldsymbol{Q}_s\boldsymbol{A}$ is the row reduced form of $\boldsymbol{Q}_s$
 7: Send $\boldsymbol{Q}$ to the Server and request $K$ output string.
```

Server:

```
 8: Perform the IQP computation $\boldsymbol{Q}$ using implementation of choice.
 9: Return $K$ samples $\widetilde{x^i}$ to the Client.
```

Client:

```
10: Let $\boldsymbol{s} \in \{0,1\}^{n_p}$ be the vector with entries all equal to zero with the exception of
    the last which is set to one.
11: Test the orthogonality of the output $\widetilde{x^i}$ against $\boldsymbol{A}^{-1}\boldsymbol{s}^T$, setting $o^i = 0$ if it is not
    orthogonal and $o^i = 1$ if it is orthogonal.
12: Return $o = 1$ if the fraction of $i$ with $o^i = 1$ is close to $\cos^2\left(\frac{\pi}{8}\right)$, and $o = 0$ other-
    wise.
```

that doing so, without knowing the relevant $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$, is hard to do on a classical computer.

Known Structure The Client will check if this hard problem has been solved by calculating the probability that the returned bitstrings are orthogonal to $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$, which the Client themselves decided upon.

Hidden Structure It is conjectured in [113] that recovering $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$ from the given X-program is hard to do using a classical computer.

We turn then to the basis for the conjectures made above. That is to say, we wish to understand if the scheme outlined in Protocol 1.6.1 could be fooled by a classical device. Note that to do so, it would be sufficeint to break either of the two conjectures outlined. In particular, if the Server could recover $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$ from the X-program received, then it would be sufficient to return classical strings which are orthogonal with the appropriate probability. As stated when describing the hidden structure, it is conjectured in [113] that to recover $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$ is not possible, with this conjecture supported by analogies to finding cliques in graphs, which is an NP-complete problem. It is this conjecture that we are able to remove in Chapter 4.

Note that the conjecture made in the case of the hard problem implicitly assumes that the class of X-programs generated by Protocol 1.6.1 are hard to classically simulate. If they were not, it would be possible to sample from their output distributions using a classical computer, and to easily pass the test. Proofs for such hardness results are rare, and in this particular case in [113] this hardness is conjectures based on the hardness of simulating circuits from the IQP class more broadly, as discussed in Section 1.5.2. However, even if it was not possible to sample from these distributions classically, this would not be enough to ensure a classical client could not spoof this particular test by generating bitstrings with a high bias in some other way. As such, this particular conjecture is supported in [113] by giving a 'best attempt' at classical simulation without knowledge of $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$ which achieves maximum bias value 0.75. This classical scheme works by showing that $\boldsymbol{y} \cdot \boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$, where $\boldsymbol{y}$ is the sum of the rows of the X-program received by the Server that are orthogonal to either of two randomly generated binary vectors, is equal to the product of random codewords of the quadratic residue code. These codewords are orthogonal if either codeword has even parity which, by the nature of the quadratic residue code, occurs with probability 0.5. Hence the probability that either has even parity, and so the expectation of $\boldsymbol{y} \cdot \boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$, is 0.75. In Chapter 4 we are able to strengthen this conjecture.

As discussed in Section 1.5.2, there are strong grounds to believe a direct simulation of the role of the Server in Protocol 1.6.1 using a classical device would be hard. Indeed in [113] it is conjectured that the distribution produced by the X-program of Protocol 1.6.1 should be hard to sample from in a reasonable amount of time without an IQP capable computing device for $n_a \approx 250$ [205]. However a non-simulation attack on the scheme has been identified which allows for a purely classical server to recover, with high probability, the hidden string in time polynomial in the number of qubits [207]. This attack adapts the classical approach of [113] by fixing one of the random codewords so that, with probability 0.5, all of the vectors $\boldsymbol{y}$ in that scheme are orthogonal to $\boldsymbol{A}^{-1}\boldsymbol{s}^{\mathsf{T}}$. This probability can be boosted arbitrarily close to 1, and the vectors $\boldsymbol{y}$ then provide a set of linear equations which can be used to recover $\boldsymbol{A}^{-1}\boldsymbol{s}^{\intercal}$.

Vital to the scheme outlined in Protocol 1.6.1 is an element of blindness, the security of which is found wanting by the attack of [207]. In particular the conjecture that uncovering $\boldsymbol{A}^{-1}\boldsymbol{s}^{\intercal}$ should be hard was unfounded. To avoid the attack of [207], in Chapter 4 we propose an alternate hiding scheme which requires the Client has small quantum computing power, but which provides information-theoretical security against uncovering the hidden string.

1.6.4 Classical Certification of Demonstrations of Quantum Computational Supremacy

When the number of qubits is few and the Client is not capable of any quantum operations, the above schemes become inappropriate. Those schemes that require quantum networks are inaccessible, while Random RSP and measurement commitment schemes require too many qubits to be used to compensate for this. By running additional circuits, and making assumptions of the form of the noise, it is possible to bound the ℓ_1-norm distance between an ideal and real output probability distribution without requiring quantum operations by the Client, or increases in the number of qubits [24, 208]. While such schemes can allow even for spatially and temporally correlated noise [208], other necessary noise assumptions, such as the absence of coherent errors, may be unfounded on NISQ technology.

In the very near-term we require both computations that are well suited to the architecture being considered, and certification schemes that do not add any more qubits or circuits than are required by the computation. One such combination of computation and certification scheme is RCS and either Heavy Output Generation Benchmarking or Cross-Entropy Benchmarking [62, 85]. Both Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking use guarantees about the shape of the distribution of output probabilities from RCS circuits in order to measure the performance of a real, possibly noisy, implementation. In particular they use these guarantees to check that the outputs produced most regularly by an implementation are indeed the outputs that are the most likely in the ideal distribution. As it is thought that RCS is hard for classical computers, Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking are used to certify demonstrations of quantum computational supremacy [60].

Neither Cross-Entropy Benchmarking nor Heavy Output Generation Benchmarking can be used to bound the ℓ_1-norm distance. This prevents us from using them to draw strong conclusions about demonstrations of quantum computational supremacy, such as those discussed in Section 1.5.2 [66], which rely on bounds to the ℓ_1-norm distance. However, there are grounds to believe that circumventing classical simulation of RCS by directly 'spoofing' these benchmarks should be hard. Here spoofing roughly means producing outputs that have large probabilities in the ideal output distribution of the circuit with a greater frequency than those with a small probability, without implementing the circuit directly. In the case that the noise is depolarising, scoring highly at Cross-Entropy Benchmarking when using a classical computer to simulate RCS leads to a contradiction of the Strong Exponential Time Hypothesis [60]. Without assumptions about the noise, it seems that spoofing should be no easier than making a non-trivial estimate of the output amplitudes [85, 209], i.e. making a better guess than a uniform distribution.[30]

Both Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking can be used to approximate the fidelity of an implementation of RCS [60].[31] In the demonstration of quantum computational supremacy given in [60], Cross-Entropy Benchmarking is used to calculate the average circuit fidelity of a large circuit by decoupling two halves of the device,[32] approximating the fidelity of the circuit built from gates in the larger circuit which act only on each half respectively, and multiplying together the results of both.[33] When demonstrations of quantum computational supremacy are of concern, calculating the cross-entropy difference of the larger circuit would otherwise be too computationally costly, as producing output samples for this distribution is believed intractable. The fidelity of the implementation in [60] was only mildly better than that which could be achieved by uniformly randomly outputting samples. As we discuss further below, this provides a route to demonstrating quantum computational supremacy that does not require we directly bound the ℓ_1-norm distance.

Later, in Chapter 3, we will use these benchmarks as figures of merit when measuring the suitability of a selection of quantum computing stacks for certain applications. Hence, in this section we explore their properties in so far as this facilitates their use and justify our choosing them. For both Cross-Entropy Benchmarking and Heavy Output Generation Benchmarking, we detail: their definition, the continuous range of values they can take, their dependence on noise, and the procedure for calculating their value from samples produced by an implementation.

We will suppose a quantum computer runs an n qubit circuit C, on the input $|0\rangle^n$. Repeated runs produce a set of classical bitstrings $x_1,...,x_k$ (with k being the total number of runs). Figures of merit compare $p_C\left(x_j\right) = |\langle x_j|C|0^n\rangle|^2$, the ideal output probabilities of each x_j in C, and $\mathcal{D}_C\left(x_j\right)$, the probability that x_i is produced by a real implementation, which may be noisy. The ℓ_1-norm distance can be seen trough this lens, and indeed will be used as such throughout this thesis. Similar to the ℓ_1-norm distance is the Kullback-Leibler divergence (KL-divergence), defined as in equation (1.27). The KL-divergence upper bounds the ℓ_1-norm distance via Pinsker's inequality, and like the ℓ_1-norm distance will be referenced throughout this work.

$$\mathrm{KL}\left(p_C, \mathcal{D}_C\right) = \sum_{x \in \{0,1\}^n} p_C \log\left(\frac{p_C\left(x\right)}{\mathcal{D}_C\left(x\right)}\right) \tag{1.27}$$

In practice, access to $\mathcal{D}_C\left(x_j\right)$ is impossible. Since the number of possible bitstrings x_j

[30] The impossibility of making a non-trivial estimate is a strong but plausible assumption. However, for some circuit depths it may still be that spoofing Cross-Entropy Benchmarking is easier than performing full state vector simulations [210].

[31] The standard deviation of the estimator if Heavy Output Generation Benchmarking is used to approximate circuit fidelity is larger than that for Cross-Entropy Benchmarking [60].

[32] In the work of [60] both decoupled, partially coupled, and fully coupled circuits are investigated to ensure the accuracy of this method of combining fidelities.

[33] This approach is feasible when it can be justified, through numerical simulations and experimental implementations, that the average circuit fidelities do combine in this fashion. This is so when the errors on each output are uncorrelated with the amplitude of that output in the ideal probability distribution.

generally grows exponentially with the number of qubits, calculating $\mathcal{D}_C\left(x_j\right)$ for every j would require an exponentially increasing number of samples. Therefore, instead of using the KL-divergence, or indeed the ℓ_1-norm distance, it is importatnt to focus on figures of merit which can be approximated with a number of samples from $\mathcal{D}_C$ which is polynomial in the size of the circuit. This is the case for Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking. Note that if the number of qubits is small and constant, it is possible to reasonably accurately approximate measures such as the ℓ_1-norm distance, and we will make use of this.

Unlike with access to $\mathcal{D}_C\left(x_j\right)$, simulations can be used to calculate $p_C\left(x_j\right)$ directly, and both Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking make use of this. Importantly making use of $p_C\left(x_j\right)$, rather that the full state vector, allows for the utilisation of *Feynman simulators*. This alleviates the memory storage problem [85, 120], and allows for classical simulations of, and as a result benchmarking of, larger systems than would otherwise be possible [60, 62].

Heavy Output Generation Benchmarking

Heavy Output Generation [85] (HOG) is the problem which demands that, given a quantum circuit C as input, strings $x_1,\ldots,x_k$ be generated which are predominantly those that are the most likely in the output distribution of C. That is to say, outputs with the highest probability in the ideal distribution should be produced most regularly. If the ideal distribution is sufficiently far from uniform, this problem provides a means to distinguish between samples from the ideal distribution and a trivial attempt to mimic such a sampling procedure by producing uniformly random strings. Although a simple problem to state, this task is conjectured to be hard for a classical computer [85].

Definitions and Related Results An output $z \in \{0,1\}^n$ is *heavy* for a quantum circuit C, if $p_C(z)$ is greater than the median of the set $\{p_C(x) : x \in \{0,1\}^n\}$. Intuitively the *heavy output probability of* $\mathcal{D}_C$ is the frequency with which the heavy outputs of p_C are produced when sampling from $\mathcal{D}_C$. More precisely, let $\delta_C(x) = 1$ if x is heavy for C, and 0 otherwise. Then the heavy output probability of $\mathcal{D}_C$ is defined as:

$$\mathrm{HOG}\left(\mathcal{D}_C, p_C\right) = \sum_{x\in\{0,1\}^n} \mathcal{D}_C(x)\,\delta_C(x)\,.$$

An important note is that while the ℓ_1-norm distance bounds the heavy output probability, the reverse is not true [66]. This has important implications on the relationship between the heavy output probability and results on quantum computational supremacy which measure closeness in ℓ_1-norm distance.

$\mathrm{HOG}\left(\mathcal{D}_C, p_C\right)$ varies in value between 0, when outputs which are heavy in p_C are never produced by $\mathcal{D}_C$, and 1, when only the heavy outputs have non-zero probability in $\mathcal{D}_C$. Consider now $\mathrm{HOG}\left(p_C, p_C\right)$, which corresponds to the total probability of the heavy outputs of p_C, or equally the heavy output probability if p_C was implemented ideally. In the case $\mathrm{HOG}\left(p_C, p_C\right) \approx 1/2$, outputs which are heavy have similar probabilities of occurring as outputs that are not heavy. Such distributions are well approximated by the uniform distribution, where all outputs are equally likely. Hence, for $\mathrm{HOG}(\mathcal{D}_C, p_C)$ to help us distinguish between an ideal implementation of C and a trivial attempt to mimic it by generating random bitstrings, $\mathrm{HOG}(p_C, p_C)$ should be greater than $1/2$. In fact, $\mathrm{HOG}(p_C, p_C)$ is expected to be $(1+\log 2)/2 \approx 0.846574$ [85] for circuit classes whose distribution of measurement probabilities, p, is of the exponential form $\Pr(p) = Ne^{-Np}$, where $N = 2^n$.[34] This is to say that heavy outputs of an exponential distribution have a cumulative probability of ≈ 0.846574 of occurring. When the output distributions of a class of circuits is shown to take this form it is meaningful to define the Heavy Output Generation problem.

Problem 1 (Heavy Output Generation [85]). *Given a measure μ over a class of circuits, sample from a family of distributions $\{\mathcal{D}_C\}$ satisfying:*

$$\mathbb{E}_{C\leftarrow\mu}\left[\mathrm{HOG}(\mathcal{D}_C, p_C)\right] \geq \frac{2}{3}. \tag{1.28}$$

A popular measure of the performance of a programmable quantum computing device, called *quantum volume*, is based on measuring the largest circuits for which Problem 1 can be solved [212]. In particular, quantum volume considers the heavy output probability of a class of circuits, which we refer to as *quantum volume circuits*, and which have exponentially distributed output probabilities. Those circuits grow linearly in depth as the number of qubits, n, grows. The impact of noise is greater for circuits C_n using more qubits, since the number of gates is increased. This means that it is harder to sample from distributions $\{\mathcal{D}_{C_n}\}$ solving the HOG problem of Problem 1 for larger n. The quantum volume of a device is 2^N where N is the largest n for which distributions $\{\mathcal{D}_{C_n}\}$ which solve the HOG problem can be sampled from by the quantum device. This reveals that 'volume' here refers to the size of the Hilbert space that can be accurately sampled from. This gives a single value by which to assess the quality of a device. The philosophy of quantum volume was since extended to volumetric benchmarks [213, 214], which explore the trade-off between the depth and width of circuits more generally. Indeed this is the approach we take in Chapter 3.

The motivation for the introduction of quantum volume, and the inequality of equation (1.28) is the classical hardness of solving the HOG problem of Problem 1 for random circuits, under the QUATH assumption of Assumption 1.

Assumption 1 (QUATH [85]). *The QUAntum THreshold assumption (QUATH) is that there is no polynomial time classical algorithm that takes as input the description of an n qubit random circuit $C \leftarrow \mu$ and which guesses whether $|\langle 0^n|C|0^n\rangle|^2$ is greater or less than the median value in $\{p_C(x) : x \in \{0,1\}^n\}$ with success probability at least $1/2 + \Omega(1/2^n)$ over the choices of C.*

As opposed to the statement that HOG is hard, QUATH does not reference sampling, and concerns only the difficulty of providing non-trivial approximations of output amplitudes. This assumption is strong compared to those used before, such as the non-collapse of PH, but it can be be evidenced by observing the difficulties of calculating output probability amplitudes [85].

[34]This is also commonly referred to as the Porter-Thomas distribution [211]. This is discussed at length in Appendix B.1, where distributions of outputs from the circuits of Chapter 3 are studied.

Empirical Estimation From Samples We approximate $\mathrm{HOG}(\mathcal{D}_C, p_C)$ using only a polynomial number of samples from the real distribution $\mathcal{D}_C$, by calculating the ideal probabilities $p_C(x)$. Note that while this is sample efficient, calculating $p_c(x)$ requires a number of operations which grows exponentially with the number of qubits. To approximate $\mathrm{HOG}(\mathcal{D}_C, p_C)$ we calculate the following expression, where $x_1, \ldots, x_k$ are samples drawn from $\mathcal{D}_C$.

$$\frac{1}{k}\sum_{i=1,\ldots,k}\delta_C(x_i)$$

By the law of large numbers, this converges to $\mathrm{HOG}(\mathcal{D}_C, p_C)$ in the limit of increasing sample size.

Ideal and Noisy Implementations HOG is solved efficiently by a quantum computer, simply by implementing the circuit C. In the case of extreme noise, and the convergence of the real distribution $\mathcal{D}_C$ to the uniform distribution $\mathcal{U}$, $\mathrm{HOG}(\mathcal{D}_C, p_C) = 1/2$. This is compared to the case where the output probabilities are exponentially distributed, where $\mathcal{D}_U = p_U$, when we would expect to have $\mathrm{HOG}(\mathcal{D}_C, p_C) = (1 + \log 2)/2$. The continuum of values in between provides a valuable figure of merit for a quantum computing stack. We refer to the calculation of this figure of merit as *Heavy Output Generation Benchmarking*.

Cross-Entropy Benchmarking

The results of Cross-Entropy Benchmarking [62] relate to the average probability, in the ideal distribution, p_C, of the outputs which are sampled from the real distribution, $\mathcal{D}_C$. For distributions which are far from uniform, and with a spread of probabilities of outcomes, this measure can be used to distinguish an ideal from a real implementation. Ideal implementations regularly produce the higher probability outputs, obtaining a high benchmark value, while even a small shift in the distribution lowers the value.

Definitions and Related Results Intuitively, the entropy, $H(\mathcal{D})$, of a distribution, $\mathcal{D}$, as defined in equation (1.29), measures the expectation of ones 'surprise' at observing samples from $\mathcal{D}$. This is measured by $f_{\mathcal{D}}(x) = -\log(\mathcal{D}(x))$, which accordingly decreases with increasing probability of the outcome occurring.[35]

$$H(\mathcal{D}) = \sum_{x\in\{0,1\}^n}\mathcal{D}(x)\log\left(\frac{1}{\mathcal{D}(x)}\right) \tag{1.29}$$

The cross-entropy measures ones surprise when sampling from $\mathcal{D}$ when expecting $\mathcal{D}'$, or the additional information required to describe $\mathcal{D}$ given a description of $\mathcal{D}'$.

[35] An alternate definition sets $f_{\mathcal{D}}(x) = 2^{-n} - \mathcal{D}(x)$, in which case the related quantity is referred to as *linear* cross-entropy [60]. The function $f_{\mathcal{D}}(x)$ may be any which decreases with increasing outcome probability. The choice depends on the relationship between the fidelity of the resulting state, and the standard deviation of the estimator of the associated definition of cross-entropy difference. In this case the connection to the average probability of the outputs sampled is clearer.

Definition 1.6.3 (Cross-Entropy)**.** *The* cross-entropy *between two probability distributions $\mathcal{D}$ and $\mathcal{D}'$ is*

$$\mathrm{CE}\left(\mathcal{D},\mathcal{D}'\right)=\sum_{x\in\{0,1\}^n}\mathcal{D}(x)\log\left(\frac{1}{\mathcal{D}'(x)}\right).$$

The cross-entropy is notably similar in form to the KL-divergence, defined in equation (1.27). However, while the KL-divergence upper bounds the ℓ_1-norm distance via Pinsker's inequality, no such bound on the ℓ_1-norm distance by the cross-entropy exists [66]. As with the heavy output probability, the cross-entropy can be bounded by the ℓ_1-norm distance.

The cross-entropy difference is $\mathrm{CE}(\mathcal{U},\mathcal{D}')-\mathrm{CE}(\mathcal{D},\mathcal{D}')$, where $\mathcal{U}$ is the uniform distribution.

Definition 1.6.4 (Cross-Entropy Difference)**.** *The* cross-entropy *difference between two probability distributions $\mathcal{D}$ and $\mathcal{D}'$ is*

$$\mathrm{CED}\left(\mathcal{D},\mathcal{D}'\right)=\sum_{x\in\{0,1\}^n}\left(\frac{1}{2^n}-\mathcal{D}(x)\right)\log\left(\frac{1}{\mathcal{D}'(x)}\right).$$

Therefore, the cross-entropy difference can be thought of intuitively as answering "is the distribution $\mathcal{D}'$ best predicted by $\mathcal{D}$ or by the uniform distribution?".

As with the connection between QUATH of Assumption 1 and Heavy Output Generation Benchmarking, the hardness of spoofing Cross-Entropy Benchmarking for some classes of random circuits can be reduced to the hardness of making non-trivial estimations of output probabilities.

Assumption 2 (XQUATH [209])**.** *The Cross-Entropy QUAntum THreshold assumption (XQUATH) is that there is no polynomial time classical algorithm that takes as input the description of an n qubit random circuit $C\leftarrow\mu$ and produces an estimate p of $p_C(0)=|\langle 0^n|C|0^n\rangle|^2$ such that*

$$\mathbb{E}\left[\left(p-p_C(0)\right)^2\right]=\mathbb{E}\left[\left(p_C(0)-2^{-n}\right)^2\right]-\Omega\left(2^{-3n}\right)$$

where the expectation is taken over C and the internal randomness of the algorithm.

Empirical Estimation From Samples By the law of large numbers, the following expression converges to $\mathrm{CE}(\mathcal{D}_C,p_C)$, where $x_1,\ldots,x_k$ are samples drawn from $\mathcal{D}_C$.

$$\frac{1}{k}\sum_{i=1,\ldots,k}\log\left(\frac{1}{p_C(x_i)}\right)\tag{1.30}$$

This can be used by a classical computer to approximate the value for $\mathrm{CED}(\mathcal{D}_C,p_C)$. While only a polynomial number of samples x_i are required, the calculation of $p_C(x_i)$ takes exponential time.

Ideal and Noisy Implementations The cross-entropy, $\mathrm{CE}(\mathcal{D}_C, p_C)$, when $\mathcal{D}_C$ results from an ideal implementation, reduces to the entropy of p_C. In the case where the probabilities $p_C(x)$ are approximately independent and identically distributed according to the exponential distribution, we have that $H(p_C) = \log 2^n + \gamma - 1$ [62], where γ is Euler's constant.

In the case where the probabilities $\mathcal{D}(x)$ are uncorrelated with those of $p_C(x)$ we arrive at the following prediction of the cross-entropy [62].

$$\mathbb{E}_C[\mathrm{CE}(\mathcal{D}_C, p_C)] = \log 2^n + \gamma$$

$\mathcal{D}_C(x)$ and $p_C(x)$ are uncorrelated if, for example, $\mathcal{D}_C$ is the uniform distribution, or, in the case of demonstrations of quantum computational supremacy, if $\mathcal{D}_C$ is the output of an efficient classical algorithm [62].

These results allow us to identify the extreme values of the cross-entropy difference.

$\mathcal{D}_C = p_C$**:** When the unitary is implemented perfectly $\mathrm{CED}(\mathcal{D}_C, p_C) = 1$.

$\mathcal{D}_C = \mathcal{U}$**:** When samples are generated uniformly at random $\mathrm{CED}(\mathcal{D}_C, p_C) = 0$.

The cross-entropy difference gives a value between 0 and 1 which measures the accuracy of the implementation of a circuit, the calculation of which is called *Cross-Entropy Benchmarking*.

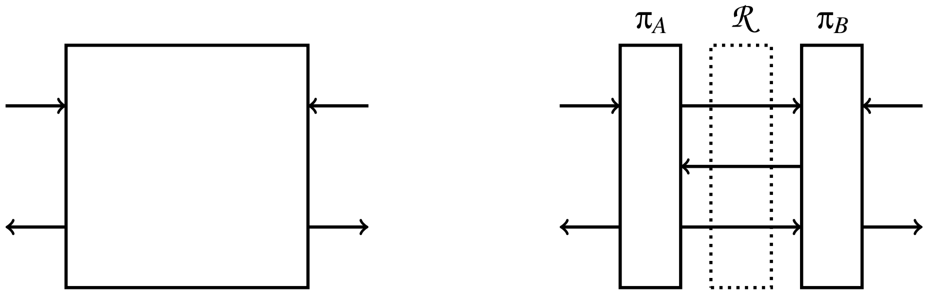
1.7 Abstract Cryptography

The approach originally used to prove the security of early quantum key distribution (QKD) protocols was to show the mutual information content, between the information gained by the eavesdropper and the key produced, is 'small'. The BB84 [5] and E91 protocols [8] were shown to satisfy this condition. However, this approach was shown to be insufficient [215] as it neglected to consider the impact of incorporating protocols into larger ones, or their parallel and repeated use. Indeed, even device independent QKD, which guarantee security based only on the output statistics of components used, is insecure if untrusted devices are used more than once [216].

Further, it is of concern to this work that, as discussed in Section 1.6.1, the definition of blindness originally utilised is insufficient when a BQC scheme is to be composed with, or utilised within, other protocols. Stronger notions of the security of BQC protocols have been introduced [194] which use tools from *Abstract Cryptography*[36] [221, 222]. The Abstract Cryptography framework can also be used to prove the composable security of QKD [222] while we will use it in Chapter 4 to prove the composable security of a blind delegated implementation of IQP computations.

The intuition on which this technique is based is that the *ideal functionality* of a protocol, which will complete the task in mind perfectly, but without considering the details

[36] Abstract Cryptography forms part of a tradition of considering the composable security of both classical and quantum protocols [217–220]. These different approaches all have in common that they consider the 'distance' between the implementation of a protocol and some ideal behaviour.



(a) Ideal Resource. (b) Real Protocol.

Figure 1.13: **A comparison of the real protocol and ideal resources**. The protocols π_A and π_B communicate via the resource $\mathcal{R}$. The ideal resource, unlike the real protocol, does not consider the details of the computation.

of the protocol, should be defined first. The *real protocol*, which must take these details into consideration, can then be introduced and compared. Showing the two to be indistinguishable means they may be reasoned about interchangeably.[37] For example, the ideal functionality of a BQC protocol allows the Server to choose the output to the Client, which would be the result of the computation if they behave honestly, but limits the output to the Server to an upper bound on the size of the computation. The Client's input is a classical description of the computation and the input to the computation. Some real protocols implementing this functionality are discussed in Section 1.6.1.

These functionalities and protocols are achieved using 'resources', and themselves define resources for use in other protocols. By way of an example, QKD may be used as a resource when implementing a one-time pad, while it itself requires resources such as an authenticated classical channel and an insecure quantum channel [222].[38]

Definition 1.7.1 (I-resource)**.** *An* I-resource *is an abstract system with interfaces specified by a set I. Each interface $i \in I$ is accessible to a user i and provides them with the ability to present inputs and read outputs. Resources are equipped with a parallel composition operator, $||$, that maps two resources to another.*

A resource implementing the ideal functionality is called the ideal resource. The real protocol is also a resource and consists of a set of other resources composed together using protocols. We hope the real protocol approximates the ideal resource. We are concerned with two-party DQC, introduced in Section 1.6, and so it is sufficient to denote the communication and computation protocols of each player by π_A and π_B for the Client and Server respectively, as Figure 1.13 exemplifies. There, the communication channel is a resource with an interface for each party. The Server is the *adversary*, although there may be several in general.

[37] This real-world-ideal-world paradigm may alternatively be formalised in the language of category theory. In this language objects correspond to resources, as defined in Definition 1.7.1, and morphisms correspond to converters, as defined in Definition 1.7.2. An appropriate definition of attacks, and security against them, allows one to show that protocols secure against attacks form a symmetric monoidal category [223].

[38] The definitions that follow are based on those from [222] and we direct the reader to that work for a thorough exposition of the Abstract Cryptography framework.

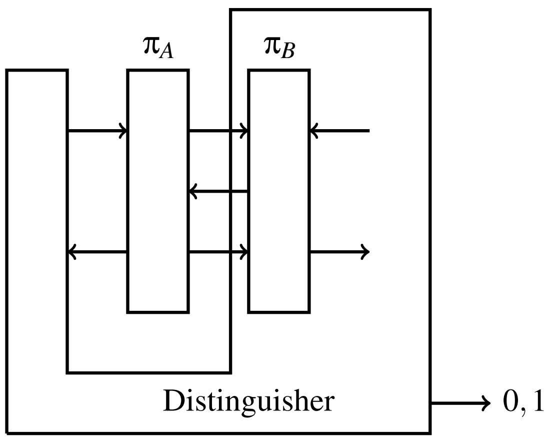


Figure 1.14: **Distinguisher interacting with the real protocol.** The distinguisher has control over both parties inputs and outputs, and implements the adversary's protocol, π_B. The binary output indicates which of the ideal resource or real protocol the distinguisher believes itself to possess.

The approach of Abstract Cryptography is to consider the *distinguisher*, which captures all the parts of the protocol outside of the behaviour of the honest party. The distinguisher picks the inputs of the honest player, collects their outputs, and plays the role of the adversary. The distinguisher is given access to either the real protocol or ideal resource and must decide which of the two it possesses. If they cannot, the real protocol is compositionally secure.

Figure 1.14 illustrates the role of the distinguisher, and reveals that the interface of the real protocol differs from those of the ideal resource. This difference may aid in distinguishing them. For example, BQC protocols may involve the exchange of qubits, which could be abstracted away by the ideal resource. To address this, a converter is used to alter the interface with the ideal resource to mimic that of the real protocol. It is also the case that the protocols π_A and π_B in Figure 1.13 constitute converters.

Definition 1.7.2 (Converter)**.** *A* Converter *is an abstract system with an inside and outside interface. The inside interface connects to an interface of a resource and the outside becomes the new interface of the constructed resource.*

$\alpha_i\mathcal{R}$ *denotes the new resource, with the converter* α_i *connected to the interface i of the resource* $\mathcal{R}$*. Use the notation* $\alpha\mathcal{R}$ *for a set of converters* $\alpha = \{\alpha_i\}_i$ *when it is clear to which interface they are connected. Serial and parallel composition of is defined by*

$$(\alpha\beta)_i\mathcal{R} := \alpha_i(\beta_i\mathcal{R})$$
$$(\alpha||\beta)_i(\mathcal{R}||\mathcal{S}) := (\alpha_i\mathcal{R})\,||\,(\beta_i\mathcal{S})\,.$$

Specifically, a simulator is introduced, which acts as an interface between the ideal resource and the adversary. The simulator should produce outputs indistinguishable from the communication of the real protocol.

Definition 1.7.3 (Simulator)**.** *A* simulator, σ, *is a converter connected to the adversary's interface to the ideal system. It is defined by a set of operations* $(\sigma_1, ..., \sigma_t)$*, one for each step of the protocol. The simulator may call the ideal resource at any time.*

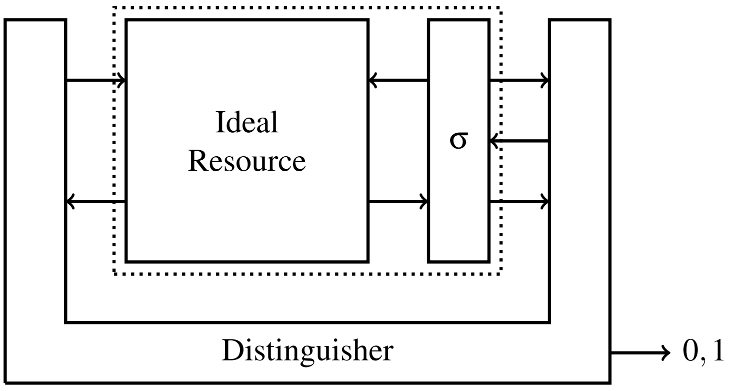


Figure 1.15: **Simulator σ addresses the difference between the interfaces of the real protocol and the ideal resource.** The dotted region should be interfaced with in an equivalent way to the real protocol of Figure 1.14.

The simulator, visualised in Figure 1.15, only learns that which is outputted by the ideal resource. As such it does not provide any additional information to the adversary, as the adversary could reproduce the simulators behaviour on its own. If the ideal resource along with the simulator is indistinguishable from the real protocol, the real protocol is considered secure as it does not reveal anything more than the ideal resource reveals.

A second converter of interest is the filter. This enforces honest behaviour by the adversary by preventing deviation and interaction with the communications.

Definition 1.7.4 (Filter). *A* filter *is a converter which, when placed over the adversary's interface, prevents access to controls necessary to act maliciously and to anything other than the standard inputs and outputs. A pair, $(\mathcal{R}, \#)$, of a resource $\mathcal{R}$ and a filter # together define a* filtered resource *which may be written $\mathcal{R}_{\#}$.*

If the filtered ideal resource and filtered real protocol are indistinguishable then the real protocol is correct in the case the adversary behaves honestly. To model this, we always assume that one of the adversary's inputs is instructions on how to deviate from honest behaviour. Blocking this interface then enforces honest behaviour.

To distinguish two resources is to adopt a measure of the differences between them. That measure should assign 0 to the difference between a resource and itself, but also assign 0 in the case of different but indistinguishable resources. As such a pseudo-metric $d(\mathcal{R}, \mathcal{S})$, where $\mathcal{R}$ and $\mathcal{S}$ are resources, is the correct measure. This pseudo-metric should be non-increasing under composition with resources and converters. This is because a converter should not be able to make it easier to distinguish between two resources, as it would otherwise mean the converter has added some information that helps with distinguishing them. For such a pseudo-metric we can define compositional security.

Definition 1.7.5. *Let $\pi_{AB} = (\pi_A, \pi_B)$ be a protocol and $\mathcal{R}_{\#} = (\mathcal{R}, \#)$ and $\mathcal{S}_{\Diamond} = (\mathcal{S}, \Diamond)$ denote two filtered resources. We say that π_{AB} constructs $\mathcal{S}_{\Diamond}$ from $\mathcal{R}_{\#}$ within $\varepsilon \in [0,1]$ if the two following conditions hold.*

Correctness: *We have*

$$d\left(\pi_{AB}\mathcal{R}\#_E, \mathcal{S}\Diamond_E\right) \leq \varepsilon$$

Security: *There exists a simulator σ_E such that*

$$d\left(\pi_{AB}\mathcal{R}, \mathcal{S}\sigma_E\right) \leq \varepsilon$$

The correctness condition of Definition 1.7.5 gives, when the adversary behaves honestly, a bound ε on the likelihood that the resources could be distinguished from each other. The security condition gives such a bound in the case that the adversary behaves maliciously.

We can now be more specific and define the pseudo-metric we will use.[39]

Definition 1.7.6. *The* distinguishing advantage *that a computationally unbounded distinguisher, which can guess with probability $p_{\text{distinguish}}$ whether it is interacting with the resource $\mathcal{R}$ or $\mathcal{S}$, is*

$$d\left(\mathcal{R}, \mathcal{S}\right) := 2p_{distinguish}\left(\mathcal{R}, \mathcal{S}\right) - 1.$$

Definition 1.7.6 presumes that the distinguisher uses the distinguishing strategy which maximises the distinguishing advantage amongst all the strategies available. In particular the distinguisher can always distinguish between the resources with probability $p_{distinguish} = \frac{1}{2}$ by simply guessing at random which of the two resources they are interacting with, giving the bound $0 \leq d\left(\mathcal{R}, \mathcal{S}\right) \leq 1$. This also demonstrates that when the resources can be correctly identified with probability $p_{distinguish} = \frac{1}{2}$ the distinguisher has no knowledge of the resource they are interacting with. In that case the distinguishing advantage is $d\left(\mathcal{R}, \mathcal{S}\right) = 0$, which we write as $\mathcal{R} \equiv \mathcal{S}$.

[39] One may choose to define a weaker distinguisher than that of Definition 1.7.6 by, for example, bounding the computational power of the distinguisher. Here we are concerned with only information theoretic security and so we do not consider this.

Chapter 2

Methods for Classically Simulating Noisy Networked Quantum Architectures

The review of existing approaches to QCVV conducted in Section 1.6 reveals a trade-off between the assurances that can be given by QCVV algorithms, and the power of the quantum computing device tested. With large fault-tolerant quantum computers it is possible to verify the correctness of any BQP computation, as discussed in Section 1.6.2, while for NISQ devices it may only be possible to perform verification in specific cases, as discussed in Section 1.6.3 and Section 1.6.4. Before the development of even NISQ devices, classical simulation is possible, and can be invaluable. However, great care should be taken before extending performance assurances that can be given for classically simulated devices, to larger devices. In this chapter we provide and exemplify a methodology for performing meaningful extrapolations of this sort.

While classical simulation cannot reproduce large quantum computations, the technique can reproduce the behaviour of small instances. We can then compare these simulations with experiments to confirm that the behaviour matches our predictions. By scaling our simulations beyond what is experimentally possible we can predict and prepare for the device's behaviour as the technology scales.[1] Our aims are therefore twofold. Firstly, to use classical simulations to predict the performance of devices larger than those currently available, but at a scale where classical simulation is possible. Secondly, to ground our simulation in theoretical results that allow us to predict the performance of devices of which extensive classical simulation is not possible.

For us to make reasonable predictions, our simulations must mimic the limitations of physical implementations. Arguably, chief among these limitations is noise, which we discussed in Section 1.3.3. Here we explore the impact of noise on the shape of distributions produced by quantum computers, but not if the noisy distributions are hard to reproduce classically. Indeed, as discussed in Section 1.5.2, in some cases it is known

[1]In addition, by pushing classical simulations to their limit it is possible to understand what is classically possible, giving a lower bound on the scale at which we would expect to observe quantum computational supremacy for a given computation [120, 129, 130, 139].

that a small amount of noise can prevent demonstrations of quantum computational supremacy. Even in this case, classical simulation can be valuable. By varying noise levels in the simulations we can determine which types of imperfections lead to the greatest deviation from the perfect output. We can then suggest experimental groups prioritise improvements on those imperfections.

In quantum mechanics 'the total is greater than the sum of its parts' so testing small components of a quantum system is not sufficient to make precise predictions about its behaviour at larger scales. This applies to testing small problem instances too. That being said, by simulating systems of size as close as possible to the classical limit we may more assuredly extrapolate that the device functions as modelled in the quantum computational supremacy regime. This is firstly because by testing, for example, 20-qubit devices, we are more confident that the phenomena we identify will manifest in larger systems than if we had tested single or two qubit modules. Secondly, since the regime of quantum computational supremacy is by definition just beyond the realm of classical simulation, it is reasonable to assume phenomena in the realm where classical simulation is possible, but close to the quantum computational supremacy realm, exist in some form in the quantum computational supremacy domain. Indeed, the transition from a regime where classically simulation is possible, to the realm of quantum computational supremacy, is one of scale, rather than the results of a paradigm shift in the underlying model of the physical system. For our purposes, this corresponds to saying that we do not expect noise sources which are significant when classical simulation is possible to become insignificant for only slightly larger, but classically intractable, computations. We proceed in this chapter under the assumption that there is no such loss of significance.[2]

While we find classical simulation to be an invaluable tool, other complementary benchmarking techniques have been explored. For example, randomised benchmarking [19], which measures average error rates, and tests for quantum computational supremacy both utilise the quantum technology directly. We regard the tools of this chapter as intermediate between, and complementary to, these approaches. While it is possible to consider more qubits using classical simulation in the way described above than can be considered using randomised benchmarking, it is not possible to consider as many as in the case of certifying demonstrations of quantum computational supremacy. Conversely, classical simulation will likely rely on the results of randomised benchmarking to build noise models, and allows for more predictive power and control than is accessible via certification of quantum computational supremacy. This will become apparent in Chapter 3 where we imagine having access to a NISQ device within reach of being capable of demonstrating quantum computational supremacy. In that case the insights we can gain are reduced as classical simulation becomes incredibly resource intensive, and universal verification remains out of reach. It is however possible to directly probe the performance of a device performing quantum computational supremacy, which is only indirectly possible here.

[2]As there have been few demonstrations of quantum computational supremacy, assumptions such as these are hard to justify. However, in the demonstration of quantum computational supremacy presented in Ref.[60] they were able to demonstrate the related property that the noise of the device used was well predicted by a function of the measured noise on each half of the device.

In this work we give a methodology to follow when using classical simulation to both predict the performance of small devices as they are scaled in size, and guide experiments pursuing a demonstration of quantum computational supremacy. We exemplify the methodology by considering IQP problems, discussed in Section 1.5, implemented on the NQIT Q20:20 quantum device, which we introduce in Section 2.2.2 and Section 2.2.3. Strong theoretical results exist for IQP problems, providing grounds for our predictions, while the NQIT Q20:20 quantum device is designed with scalability in mind. We show that the current size and noise-levels of the NQIT Q20:20 device make a demonstration of quantum computational supremacy in the way considered here unlikely. We further show that dephasing errors are the main source of degradation and so recommend experimental labs prioritise reducing this type of error. We suggest, and simulate, an error correction code, which corrects for these errors. Our results indicate that this approach improves performance considerably and makes a demonstration of quantum computational supremacy by implementing IQP instances on the NQIT Q20:20 device more likely.

Section 2.1 contains the aforementioned methodology, which is then illustrated with examples in the following sections. In particular, in Section 2.2 we illustrate the technique, discussed in Section 2.1.1, for choosing the problems, architecture and simulator for our purposes. In Section 2.3 we illustrate the principles for numerical experiment design presented in Section 2.1.2 and: present simulations which can be used to benchmark the NQIT Q20:20 device, vary the noise levels in order to identify the main sources of error, and suggest steps to reduce these errors. We conclude in Section 2.4.

2.1 Methodology

Here we detail the methodology followed, addressing two areas. First, in Section 2.1.1, we give principles to follow when choosing a computational problem, experimental system, and classical simulator for the purpose of exploring quantum computational supremacy in near-term devices. Second, in Section 2.1.2, we give a methodology for designing numerical experiments, specifically when trying to assess the plausibility of a quantum computational supremacy demonstration. We have two desired outcomes:

Outcome 1 - *Benchmark Device*: By choosing parameters such as noise and problem size to be comparable with an actual experiment, we use the simulation to certify the experiment/device and to predict its performance as the technology scales.

Outcome 2 - *Feedback to Experimentalists*: By altering the parameters we determine which imperfections have the greatest negative impact and provide advice about which are the most urgent and beneficial hardware improvements.

2.1.1 Problem, Architecture and Simulator Selection

Here we give the method utilised in selecting the problem, experimental setup, and classical simulator used in achieving the above outcomes. We represent this methodology schematically in Figure 2.1.

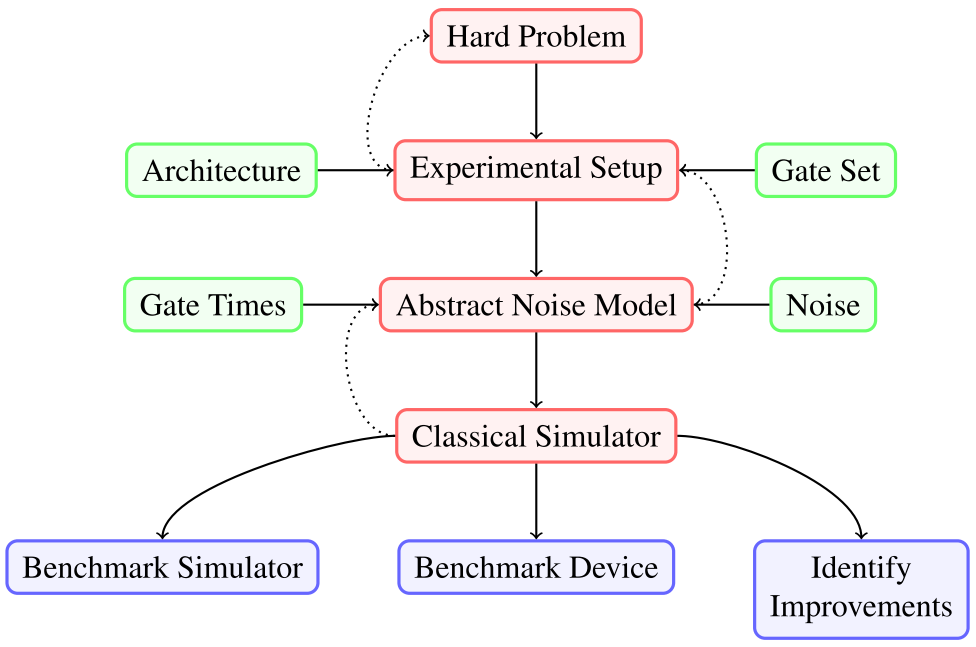


Figure 2.1: **The methodology proposed in this chapter**. The consideration of each step is preceded by its ancestor in the diagram, with feedback (dotted arrows) between steps, and contributing factors indicated from the sides. Outcomes are detailed at the base of the figure.

Step 1 - *Hard Problem*: Select a set of problems which: we know, or conjecture, to be classically hard; despite their hardness, need not be BQP-complete (i.e. do not exhibit the full power of quantum computation) and are easier to implement than a universal quantum computation; and show indications of the advantage in the quantum case persisting in the presence of noise.

It's reasonable to assume that for some time the problems used to demonstrate quantum computational supremacy will fit the above description, as they have so far [60, 176].

Step 2 - *Experimental Setup*: Select an experimental set-up which there exists reason to believing could be built in the near-term. Examine architecture restrictions including the quantum computation model (circuit, measurement-based, etc), the connectivity of the qubits, and the operations which are natural to the setting.

Step 3 - *Abstract Noise Model*: Decide on a noise model to use, which should depend on the experimental implementation studied and on experimental measurements of the noise. For the quantum computation being considered, translate the noise into abstract operations appropriate for simulation.

Step 4 - *Classical Simulator*: Select a classical simulator that is best suited for the problem under consideration. This is not, in general, a brute-force simulation by matrix multiplication, and the specific choice can be such that it performs better for the problem, or instances there of, being considered.

While we consider each step in turn, we encourage feed-back between them. From the conclusions drawn at each step we 'tailor-make' the construction of others.

2.1.2 Numerical Experiment Design

Our analysis consists of three parts for each numerical experiment. In the first we test the suitability of the classical simulator we plan to use, while in the second we use the simulator and take into account realistic or projected noise. While the first part benchmarks the simulator, the second allows us to achieve Outcome 1 listed in the introduction to this section. The third part of the experiment involves altering the parameters to achieve Outcome 2.

Part 1 - *Simulator Benchmarking*: Typically, the best classical simulators are probabilistic with errors which scale with the size of the computation. Therefore one must test the simulator chosen works as expected, specifically for the problem considered. Do this by running smaller instances of the problem and comparing the resulting distributions to a less efficient brute-force simulation. In particular:

- Generate random small instances of the problem.[3]
- Complete a brute-force simulation of the generated problem.
- Adapt our chosen simulator to solve those instances, and solve many times.
- Compare the brute-force and aggregated simulator outcomes.[4]

In this way we establish the simulator's accuracy. We proceed to Part 2 only if the simulator is indeed shown to be accurate in the case of the problem considered.

Here, as in Part 2 and Part 3, it is vital to clearly outline the random process by which problem instances are generated. Firstly, the randomness with which problem instances are generated can impact the applicability of results on quantum computational supremacy which often depend on this element of the circuit generation process. Secondly, even when quantum computational supremacy results do not apply, we wish to ensure the problem instances used for benchmarking are representative of practical performance, as opposed to, for example, accidentally corresponding to a subset of all circuits where the device performs particularly well. In the case of the problem classes we select, namely IQP-MBQC and 2D-DQS, we give the circuit generation procedures explicitly in Appendix A.2.1 and Protocol 1.5.1 respectively. In the constraints sections of Section 2.3.1 and Section 2.3.2, we make clear if and how the resulting distribution of circuits relates to the quantum computational supremacy results, and how it ensures that the benchmarks are representative of practical performance.

Part 2 - *Device Benchmarking*: To address Outcome 1, impose constraints reflecting the implementation, possibly at scales larger than those which have been implemented. Where possible, compare these simulations with experiments to determine the accuracy of any predictions made. Use the following steps:

- Generate random instances of the problem, restricted to the architecture.
- Generate many random instances of noise to generate many noisy circuits.
- Solve each noisy circuit and the original perfect circuit many times.

[3]Here the problem that we simulate need not be hard as we are simply benchmarking the simulator, and not the prospect for quantum computational supremacy. The hard problem we consider should, however, be a subset of the general class we simulate here.

[4]The simulator we use has a non-deterministic outcome so we take the average or 'aggregated simulator outcome' as a means to compare.

- Compare the aggregated simulations in the perfect case and the average of the aggregated noisy simulations.
- Use suitable parameters and compare with actual experimental realisations.

In this way one can estimate the noise's influence. Part 3 aims to understand how the device can be improved to reduce the noise influence. As such it would be beneficial in general to continue to Part 3. However, one may wish not to proceed to Part 3 if it becomes apparent that the device is completely unsuited to the problem being considered, or sufficiently well suited without reducing the influence of the noise.

Part 3 - *Guiding Future Experiments*: Impose constraints coming from the realistic setting to the simulation and compare results with exploratory simulations with varying noise levels. This comparison is done to obtain an indication of the speed at which the noise corrupts the computation. Use this as a tool to provide feedback to experimental groups about which aspects of their devices they should prioritise improving. In so doing, we address Outcome 2.

- Proceed as in Part 2 but with a varied noise model.
- Compare these results with simulations using the original noise model to understand the impact of the new noise model.
- If some change to the noise model is shown to result in a large improvement of the quality of the computation:
  1. Propose experimentalists prioritise reducing this type of noise.
  2. Consider theoretical methods to mitigate this specific type of error and test the performance in simulations. For example, introducing partial error correction to deal with the single most important source of error.

While each part builds on from its predecessor, and so should follow it in the order of experiments, we may stop at some part if, for the reasons outlined above, it becomes apparent that proceeding would not be advantageous.

In this work we will not compare our results to those of experimentalists, as we describe above. We recognise this as an important step and hope to do so in future work. Here we focus on using classical simulation to predict the impact of noise.

2.2 Exemplifying the Problem, Architecture, and Simulator Selection Methodology

Following the methodology for selecting a problem, architecture and simulator, discussed in Section 2.1.1: in Section 2.2.1 we present the class of problems considered; in Section 2.2.2 and Section 2.2.3, the physical system investigated; and in Section 2.2.4, the classical simulation technique used.

2.2.1 Step 1 : Instantaneous Quantum Polytime

Here we consider IQP circuits, as introduced in Section 1.5, for which results on the hardness of weak simulation up to additive error exist. We do so as we regard such results as an indication that a class of problems is promising for an early demonstration of quantum computational supremacy. This is because it seems plausible that noise will have a similar impact on average case problems, which we simulate, and worst case problems, for which hardness results exist. Thus we can draw conclusions about the impact of noise on the hard cases from its impact on average cases.

In our work, we do not explore the impact of noise on theoretical results on demonstrations of quantum computational supremacy, as was done in the work introduced in Section 1.5.2, but suggest that numerical exploration be done in parallel with theoretical analysis. This would guide us in understanding which realistic experimental setting is best to demonstrate quantum computational supremacy with IQP problems.

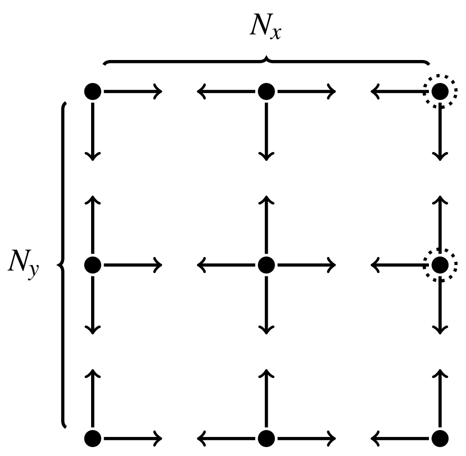
We will make use of the realisation of the IQP class in MBQC, as discussed in Section 1.5.4. This is particularly useful since it explicitly parallelises the computation, minimising the required circuit depth. We will also extensively explore 2D-DQS circuits defined in Protocol 1.5.1. We recall that this problem in particular seems a good candidate for our purposes, as described in the hard problem selection methodology of Step 1 in Section 2.1.1, since it is hard to simulate classically in general, and is experimentally realisable in the near-term. A further advantage of this scheme is that the authors of [24] provide an explicit means for a client with a simple measurement device to verify the protocol. This is an important feature for extending the analysis beyond the limits were classical simulation is possible.

2.2.2 Step 2 : NQIT Q20:20 Architecture

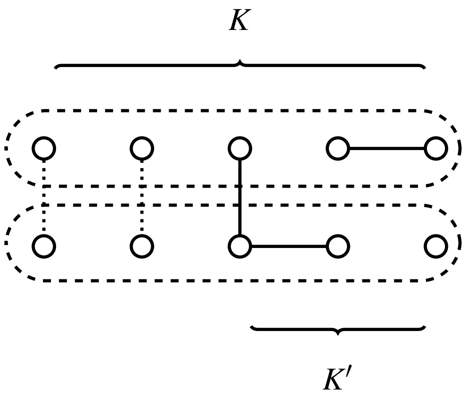
The second choice to make is the physical system that we consider (Step 2 of Section 2.1.1). We chose the Q20:20 device being developed by the Networked Quantum Information Technologies Hub (NQIT) [13, 224]. In fact we will model this device as closely as possible so it will also determine our choice of the noise model, as discussed in Step 3 of Section 2.2.3.

Networked architectures like NQIT Q20:20, which combine matter degrees of freedom in modules which are entangled via photonic degrees of freedom, have two important advantages. Firstly, once the implementation of connections between modules is perfected, this architecture can easily scale without significant extra challenges. The second advantage is that this architecture can be combined easily with communication tasks. Many applications of quantum computation are likely to involve multiple parties, a setting to which networked architectures are best suited.

Upon completion, the NQIT Q20:20 device would consists of $N = 20$ ion traps [46, 47] with $K = 20$ ions (physical qubits) in each. Traps are arranged on a 2D grid with only nearest-neighbour interactions allowed, giving a maximum number of connections $D = 4$. Different ion-traps are connected via high-fidelity entanglement between dedicated *linking qubits*. This high-fidelity entanglement is realised through *entanglement distillation* [225, 226] and consumes some of the physical qubits of each ion-trap, leaving $K' < K$ available qubits, before considering the cost of potential error correction. Two-qubit gates between ion-traps can be applied by *teleporting* the qubits into the same trap. Single and two-qubit gates within a single ion-trap take place in special



(a) **Connectivity between ion traps.**. The dotted circles align with the ion traps that are expanded in Figure 2.2b. Note that, in the language of Section 2.2.2, $N = N_x N_y$.



(b) **Expanded view of individual ion traps.** Dotted lines between ions in different ion traps indicate lower fidelity entanglement which is used to distil a higher fidelity entanglement, indicated by the solid line.

Figure 2.2: **Architecture of the NQIT Q20:20 device [224].**

gate zones. A summary of this information can be seen in Figure 2.2.

These details are based on information obtained early in the NQIT project [224]. Since the project is still underway, the system parameters N, K, K', D, and others, may change [226] and so we let them vary in our simulation toolbox. Like the architecture itself, the operations that are possible on the NQIT Q20:20 device may vary. We elect to use the following plausible set. While not certain to be the correct choice in the long-term, it will at least result in compilation to circuits with a comparable gate count and execution time to the final choice; both key factors in determining the effect of noise.

Preparation and measurement: It is possible to prepare qubits in the Hadamard basis and measure qubits in the computational basis.

Single qubit operations: The possible single qubit operations consist of H and rotations by arbitrary angles, about arbitrary axes in the $X - Z$ plane. For practical reasons the axes will likely be restricted to integer multiples of fractions of π. Here we will choose $\frac{\pi}{4}$ giving us access to T gates.

Two qubit operations: Here the controlled Z gate, CZ, is permitted.

Operations between traps: It is possible to create a bell pair $|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ between traps.

2.2.3 Step 3 : NQIT Q20:20 Noise

Following Step 3 of Section 2.1.1, we give a brief summary of all types of noise, the degree to which they impact computations in the case of NQIT Q20:20, and how we will model them. We divide the noise into time-based, which we model as occurring randomly in time on each physical qubit independently, and operation-based, which we model as occurring when an operator is applied, and is only applied to the qubits on which the gate acts.

The values listed below are acquired through measurements of the NQIT Q20:20 device [224]. This is a subset of the errors described in Section 1.3.3, and a particular modelling of these errors. However, the structure is general and other versions of the NQIT Q20:20 device or other quantum devices are likely to have similar specifications. Therefore the toolbox developed should be adaptable to other quantum computation devices. Appendix A.1.2 contains a systematic description of the noise.

Time-based Noise

Depolarising Caused by scattering of amplitudes of the electron's wave-function between different energy levels of the ion. Modelled by a random single-qubit Pauli on each qubit at a rate of $\approx 9 \times 10^{-4} s^{-1}$.

Dephasing Entanglement reduction that destroys data not stored in the computational basis. Modelled by Z gate on each qubit at a rate of $\approx (7.2 \pm 1.4) \times 10^{-3} s^{-1}$.[5]

To simulate these noise channels we need the execution times of different operations:

- Preparation - $1 - 1.5$ms
- Measurement - $2 - 2.5$ms
- Single or two-qubit operation within a trap - 0.5ms [6]
- Linking between traps - $1 - 2$s [7]

Operation-based Noise

Preparation Error in preparing a state. Modelled by bit flip error, simulated by Pauli X at rate of $\approx 2 \times 10^{-4}$.

Measurement As with preparation, measurement is also noisy. Rate of $\approx 5 \times 10^{-4}$ to measure any qubit incorrectly, which corresponds to bit flip and so an X gate.

Single-qubit gates Random Pauli operator applied with probability $\approx (1.5 \pm 0.45) \times 10^{-6}$ after each single-qubit gate. This models depolarising noise.

Two-qubit gates Depolarising error, modelled by independent single-qubit random Pauli errors, on both qubits, each with probability $\approx (5.5 \pm 3.5) \times 10^{-4}$ and a further two-qubit error $Z \otimes Z$ with probability $\approx 6 \times 10^{-5}$.

Linking operations This error is determined by the fidelity of the entanglement between traps, and so depending on the amount of entanglement distillation used [226]. If 10-qubits are used for distillation, then the error in the linking operation between traps is approximately the same as the errors incurred by two-qubit gates between qubits in the same trap [224]. Moreover, using more qubits for

[5]Using the language introduced in Section 1.3.3, the T_2 time is $\approx (1/(7.2 \pm 1.4)) \times 10^3 s$.
[6]This set of operations includes moving the qubits to the *gate zone*.
[7]This timing information is for the case of 10 distillation qubits.

distillation would not improve the computation since the same ion-trap qubit gates will still have higher errors.

2.2.4 Step 4 : Clifford + T simulator of Bravyi and Gosset

Here we will use the improved Clifford + T simulator of [134], introduced in Section 1.3.2 and referred to as the Bravyi-Gosset Simulator in this work, which produces the probability of measuring a single outcome. This simulator has runtime exponential in the number of T gates[8] but polynomial in the number of qubits and Clifford gates. This allows efficient simulation of circuits with a logarithmic number of T gates. Furthermore, because of the small exponent, it enables the classical simulation of larger instances than regular brute-force simulators. The size of the circuits we are able to simulate therefore depends on a limit on the number of T gates, as imposed by the computational cost of simulating them. In the case of the 2D-DQS problem, this limit on the number of T gates will also limit the number of qubits. This is because, as shown in Protocol 1.5.1, the circuits used require roughly half the number of T gates as qubits. Importantly, this restriction on the number of T gates does not alter the structure of the circuits, ensuring that our findings are representative across the whole circuit class. In the case of IQP-MBQC circuits this limitation on the number of T gates corresponds to limiting the number of rows of the corresponding X-program, and so the number of ancillary qubits. This does not, however, limit the number of primary qubits, which we vary depending on the size of the device being modelled. As such, this restriction again does not limit the structure of the circuits; in principle allowing, for example, for arbitrary connections between primary and ancillary qubits.

Using the Bravyi-Gosset Simulator, the authors of [134] simulate about 40 qubits and 50 T gates in what they quote as "several hours". Here we require the simulation of several thousand circuits and so we simulate fewer T gates in order to complete our simulations in a reasonable time. The 2D-DQS problem is highly entangled, beyond stabiliser simulation and conveniently represented in the Clifford + T gate set without costly (in gate count) gate decomposition [227, 228]. This makes the Bravyi-Gosset Simulator perfect for our purposes, and others mentioned in Section 1.3.2 less useful.[9]

2.3 Exemplifying the Numerical Experiment Design Methodology

We present the results of two sets of numerical experiments, in accordance with the numerical experiment design methodology introduced in Section 2.1.2, utilising discussions, in Section 2.2, regarding the problem, architecture and simulator to be used.

[8]The exact expression has $2^{\beta t}$, where $\beta = \frac{1}{2}$ and t is the number of T gates.

[9]The Bravyi-Gosset Simulator is well suited to the types of IQP circuits which we simulate in this chapter, but there is no reason to believe it is particularly well suited to simulating IQP circuits in general. To the authors' knowledge there is no classical simulation algorithm which consumes fewer resources when simulating IQP circuits than when simulating comparable BQP circuits. The development of such a scheme would certainly benefit the approach taken here.

The first considers the 2D-DQS problem, the restricted class of IQP computations presented in Protocol 1.5.1, and is used to demonstrate the potential of classical simulators as a tool to guide experimental research. In Section 2.3.2 and Section 2.3.3, where we present results for this problem, we simplify NQIT Q20:20 architectural constraints to focus on the impact of noise.

We embrace the full complexity of the NQIT Q20:20 architecture in a second numerical experiment presented in Section 2.3.2. We restrict IQP-MBQC instances, seen in Section 1.5.4, to the NQIT Q20:20 architecture. The hardness of the IQP problem could, in principle, be destroyed by these restrictions. Here we focus on the effect of architectural constraints on simulations, while the proof of hardness and detailed noise analysis is left for future works.

For reference throughout this section, we summarise the settings explored by each figure in Table 2.1. Similarly, while we will reference the simulation details, architectural constraints and figures of merit used in each of the experiments, we note some traits which are common in all of our experiments.

Simulation To introduce some terminology, each *numerical experiment* consists of several *trials* which are simulations of several different but related circuits. Often a trial will consist of many *runs*, themselves involving several simulations of the same circuit. For example, an experiment might have many trials, each containing a run simulating a probability amplitude for an output of a perfect circuit and several runs each simulating the same output probability amplitude, but with different noisy versions of that circuit.

Indeed, each trial will compare a *perfect run* and possibly several *noisy runs*, which we will identify in each numerical experiment. In particular, in the case of numerical experiments benchmarking the simulator itself the perfect run will be conducted using a brute-force simulator while the noisy run will utilise the simulator of our choice. In this case, the noisy simulation is noisy in the sense that the outcomes of the chosen simulator are probabilistic. In the cases where the device is being benchmarked, the perfect run will not consider the architectural noise model, while each noisy run will.

Constraints Within each numerical experiment we must identify the constraints on the family of circuits we are considering in order to ensure that it is consistent with the philosophy of this chapter. In particular, we must ensure that the perfect runs have the necessary theoretical support, for which we will fall back on the IQP hardness results detailed in Section 1.5.2.

A general restriction which is pervasive in our work concerns the degree to which operations can be parallelised in the circuits we consider. While, in theory, IQP circuits are parallel by construction, qubits are physical systems and, in the circuit model, one may be required to apply multiple gates on the same systems, which may not be possible experimentally. To increase parallelisation of the computation, in our numerical experiments we consider instances of IQP-MBQC, where all measurements can be made simultaneously, allowing us to neglect the impact of time based noise during measurement. If we used a less parallel realisation of IQP circuits, it would be prone to the same type and size of noise as a general universal quantum computation and may

	Subject of Study	Architectural Restrictions	Noise Levels	Circuits Run
Figure 2.3	Simulator Benchmarking	None	Noise results from probabilistic nature of simulator	Random unrestricted IQP-MBQC
Figure 2.4	Device noise simulation	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS
Table 2.2	Faithful device simulation	Full NQIT Q20:20 as in Figure 2.2	NQIT Q20:20 as in Section 2.2.3	Random IQP-MBQC restricted as in Figure 2.5
Figure 2.6	Impact of operation-based versus time-based noise	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS
Figure 2.7	Impact of depolarising versus dephasing noise	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS
Figure 2.8	Impact of error correction	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS
Figure 2.9, Figure 2.10	Impact of continuous noise reduction on fixed output of many circuits	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS
Figure 2.11, Figure 2.12	Impact of continuous noise reduction on many outputs of fixed circuit	2D lattice abstraction of NQIT Q20:20 as in Figure 2.2a	NQIT Q20:20 as in Section 2.2.3	Random 2D-DQS

Table 2.1: **Summary of the subjects of interest of key figures presented in Section 2.3**. Figure pairs indicate they present the same results in alternate ways. Results and implications thereof are found in the relevant text.

not be a better candidate for demonstrating quantum computational supremacy than a universal quantum computation.

Similarly, as discussed in Section 2.2.2, while the NQIT Q20:20 device is universal, to apply a 2-qubit gate on qubits which belong to ion-traps that are far apart on the 2D lattice, would require many SWAP gates, each consuming linking qubits. This can result in a large overhead [173] and so a high noise level. Thus, we aim to minimise the number of such gates when deriving our restrictions and we will see that very few SWAP gates are required for our choices of problems.

Figures of Merit To compare perfect runs, which will be justified in their use by the discussion on constraints, with noisy runs, we must consider what figures of merit we will use to judge the quality of those noisy runs. When quantum computational supremacy is not of concern, for example when benchmarking the classical simulator we use, as is demanded by Part 1 of the numerical experiment design methodology of Section 2.1.2, and as we do in Section 2.3.1, the figure of merit will relate to its reliability in producing accurate outcomes.

Statistical test for model closeness In this case, the output of the simulations are single values of probability amplitudes. A statistical test will be necessary to compare the probability amplitudes from perfect runs to those of the noisy runs. We will use the coefficient of determination to measure the quality of the noisy runs as a model for the perfect runs. This is detailed further in Section 2.3.1.

In the case of simulator benchmarking we compare the probability amplitudes from a brute-force simulation to those of the probabilistic simulator, which can be seen as a model of the brute-force simulator. We use the same statistical test in Section 2.3.2 when we simulate restricted instances of the detailed NQIT Q20:20 architecture as we are less concerned by exploring quantum computational supremacy when the theoretical foundations have been weakened by this restriction. There we will focus on the application of our work to restricted architectures, and study the implications for more general architectures, but find the quantum computational supremacy motivated figures of merit discussed below to be inappropriate in that case.

By comparison, when considering the prospect of a device demonstrating quantum computational supremacy, the figure of merit will relate to the anticipated usefulness of a larger scale real world implementation of the circuits we are simulating in such a demonstration. Such a consideration is demanded by Part 2, device benchmarking, and Part 3, guiding future experiments, of the numerical experiment design methodology of Section 2.1.2, and is performed in Section 2.3.2 and Section 2.3.3.

In the case of the simulations of noisy circuits in Section 2.3.2 and Section 2.3.3, while we do not formally consider their hardness, our measure will be the closeness of the simulated probability amplitudes to the perfect simulations, for which the hardness results of Section 1.5.2 apply. The theoretical results regarding the hardness of noisy distributions typically concern their ℓ_1-norm distance from the perfect noise free distribution. In our case we do not have access to this information because, as discussed in Section 2.2.4 where the simulator we use is introduced, we access only the amplitudes of a single output, rather than fully characterising the distribution. As such we will often use proxy measures of the ℓ_1-norm distance between perfect and noisy distributions.

While there are classical simulations which would give us access to a full characterisation of the distribution, here we forgo this option. This is because our aim is to explore the impact of noise at the boundary between what can be simulated classically and what cannot. To do so we have chosen to use a simulator which allows us access to a higher number of qubits than can be implemented experimentally on the NQIT Q20:20 architecture, and than could be implemented using simulators which characterise the full probability distribution.

With this in mind, we note the following figure of merit which will be used in Section 2.3.2 and Section 2.3.3.

Accuracy and far from uniformity of noisy runs: We will consider a numerical experiment to have demonstrated that the current noise values are likely to bring implementations within the reach of classical simulation if trials show either; the noisy probability amplitudes to be within a standard deviation of the uniform distribution, or greater than one standard deviation from the perfect amplitude.

These far from uniform points are of great importance for several theoretical reasons. Their existence is shown to be indicative of quantum computational supremacy [62, 229] while their accuracy is also shown to be vital. For example, studies of the heavy outputs of random circuits, as discussed in Section 1.6.4, show that a device could demonstrate quantum computational supremacy by preserving those probabilities with higher than median value. In addition, measures such as multiplicative error, on which many quantum computational supremacy statements are based, as discussed in Section 1.5.2, and cross-entropy difference, as discussed in Section 1.6.4, are particularly sensitive to the effect noise has on outcomes with small probabilities.

Contradicting this accuracy and far from uniformity statement can therefore be seen as an indication, but not proof, of the ability to demonstrate quantum computational supremacy in the setting being considered. In particular, note that contradicting this condition tells us that there are more outcomes with probability both far from the uniform value, which can be easily classically simulated by generating outputs at random, and close to the ideal value, which the relevant theoretical results indicate should be hard to achieve. As such, we will consider a demonstration of quantum computational supremacy to be more likely if the noisy distribution more often contradicts the statement. Note that contradicting this condition is not a proof of a demonstration of quantum computational supremacy, but provides a lower bound on what must be achieved for a demonstration of quantum computational supremacy to be deemed likely by us.

This accuracy and far from uniformity measure also implies that values close to uniform ones in the ideal distribution remain so in the noisy distribution. This follows as such values would otherwise be ‘far from the perfect amplitude’. However, in many cases noise has the effect of bringing probability values close to the uniform distribution and so little information about the effect of noise can be obtained from these outputs as they will be little changed. The noise types listed for the NQIT Q20:20 device in Section 2.2.3 are modelled by the action of random Pauli gates. The result is a flattening of the output distribution, and a convergence to the uniform one. While this is not the case for noise channels such as amplitude damping, these errors would also be captured by this figure of merit as it would have the effect of decreasing the probability of likely outputs towards the uniform distribution value. Further, it would be impossible to distinguish close to uniform values which have been achieved through accurate reproduction of the ideal distribution and those which have been achieved through a naive approximation by a uniform distribution. While it is true that these points are of value to the form of the distribution as discussed in Section 1.5.2, as we cannot make this distinction we do not include them in our analysis.[10] By isolating outcomes which have far from the uniform probability in the ideal distribution we obtain the additional advantage of being able to limit the outputs which we must study in our experiments, allowing us to run larger circuits as a trade off.

It is valuable that contradicting this condition implies there are points which are further than one standard deviation from the uniform distribution. It ensures points which are far from uniform in the ideal distribution reliably remain this way in the noisy one. This is to say that a large proportion of the points are not equal to the relevant uniform distribution probability, which could be easily classically simulated. Further statistical tests on the distribution of outputs would be needed to calculate the proportion of the outputs which are within one standard deviation of the mean, and so away from the uniform probability. However conjecturing a normal distribution allows for the majority of points to be captured. One may wish to demand that a larger proportion of the outputs are far from the uniform probability by increasing to 2 or higher standard deviations. Capturing the majority of points, as one standard deviation does, is sufficient for our purposes.

That the mean noisy output probabilities be within one standard deviation of the ideal is not strictly necessary as quantum computational supremacy statements typically allow for a constant deviation of the whole distribution. However, as we calculate the probabilities of only single outputs, and so are unable to calculate the deviation of the whole distribution, demanding mean output probabilities are within one standard deviation of the ideal seems like a reasonable substitute for the deviation of the whole distribution. In fact, as we will see, in Section 2.3.3 there is a relationship between this measure and more direct proxies for the ℓ_1-norm distance, which provides further justification for the use of this figure of merit. Indeed we are justified in saying that the mean noisy output probability being one standard deviation away from the ideal suggests the noisy distribution could be classically simulated as it suggests the majority of noisy values are not equal to the ideal, to which theoretical results certainly apply.

However, as we focus on single amplitudes, it may be that this is a strong metric. While it is shown to be hard for a classical device to sample from the output distribution of arbitrary IQP circuits, which are subject to constant independent depolarising noise on each qubit, up to a small multiplicative error in each probability [162], this is often possible to do to within ℓ_1-norm distance [64].

The previous two figures of merit have the advantage that they are the best utilisation of

[10]Due to the anticoncentration property of IQP distributions, this might result in considerable filtering of our simulations.

the simulator that we have chosen to use. In particular, they extract a significant amount of information from the single probability values which we have access to. That being said, as we mentioned before, theoretical results often refer to global properties of the probability distributions. The following figure of merit addresses this disparity.

Close in ℓ_1-norm distance: When the circuit considered in a numerical experiment is considered to not be unlikely to demonstrate quantum computational supremacy, as defined in the above condition, we will consider the closeness of the noisy and perfect runs using proxies for the ℓ_1-norm distance.

The paticular proxies for the ℓ_1-norm distance which we make use of are introduced as they arise in Section 2.3.3. Because of the relationship between this figure of merit and the theoretical results for IQP in Section 1.5.2 and, in particular, 2D-DQS in Protocol 1.5.1, this figure of merit can more reasonably be expected to be a predictor of demonstrations of quantum computational supremacy than in the previous case. Once again we will often refer to the relative likelihood of a demonstration of quantum computational supremacy between noise settings as measured by the degree of improvement in the ℓ_1-norm distance. In this case we have the additional benefit of being able to study the closeness of the measured value of the ℓ_1-norm distance to the value specified in the relevant theoretical results of Section 2.2. However, as we do not provide formal bounds on the ℓ_1-norm distance by the proxies we introduce, these proxies do not allow us to make formal claims about demonstrations of quantum computational supremacy.

By encapsulating results related to those far from uniform outcome probability values and proxies for the ℓ_1-norm distance we cover a diverse set of theoretical results. We believe there is great value in this diversified approach and as such we will combine both the accuracy and far from uniformity condition and the close in ℓ_1-norm distance condition throughout our work.

For each numerical experiment we will use considerations of the simulation method, the constraints of the architecture, and the appropriate figures of merit to draw conclusions pertaining to the goals of this chapter.

2.3.1 Part 1 : Simulator Benchmarking

As we outlined in Section 2.1.2, Part 1 of the numerical experiment builds confidence in our simulator by comparing the outputs to a brute-force simulation. Here we detail the numerical experiment used to do so.

Constraints Here we do not consider the specifics of the architectural noise as we are measuring the impact of using a probabilistic simulator as compared to a brute-force one. It is sufficient to benchmark the probabilistic simulator by comparing the outputs to those of a brute-force simulation of unrestricted IQP-MBQC instances of Section 1.5.4. We do not restrict to a particular architecture here but the generality we utilise ensures the functioning of the simulator for restricted instances explored later. Note that we are not concerned with results on quantum computational supremacy here, but instead with ensuring the distribution of circuits covers those of practical concern later, as described.

Simulation As described in Appendix A.2.1, during each trial we will generate a random unrestricted instance of IQP-MBQC, and simulate the circuit to obtain the probability of measuring the $|0^n\rangle$ state. The randomly generated circuits will have between 5 and 12 qubits, and between 5 and 15 T gates. In the case of the perfect run, the solution will be obtained by using the brute-force simulator, while in the case of a noisy run it will be solved by taking the mean of several simulations using the probabilistic simulator of Section 2.2.4. Together these two runs constitute a trial. The resulting values for the runs in each trial are then compared to calculate the coefficient of determination as described in the figures of merit section.

As discussed, while the brute-force simulation is deterministic, the simulator of Section 2.2.4 which we are testing against it is probabilistic. As such, each noisy run will consist of calculating the given probability distribution many times, and averaging. The mean and standard deviation are plotted in Figure 2.3.

Here it is sufficient to consider only the probability of measuring the state $|0^n\rangle$ as no additional error is added by measuring other states. As measuring other basis states requires only the appropriate X gates, which can be applied deterministically by the simulator of Section 2.2.4, unlike T gates which are applied probabilistically, no additional error will result from considering only the $|0^n\rangle$ state.

Figures of Merit The measure we will use to compare the perfect and noisy runs is the coefficient of determination, which can be said to measure the correlation between the outputs of a model and those from its target. Given outputs m_i from a model, and the corresponding target outputs d_i, with mean $\bar{d}$, the coefficient of determination is calculated using (2.1). In (2.1), $r = \sum_i (d_i - m_i)^2$ is the residual sum of squares and $v = \sum_i \left(d_i - \bar{d}\right)^2$ is the total sum of squares.

$$R^2 = 1 - \frac{r}{v} \tag{2.1}$$

Here the model is the simulator of Section 2.2.4 and the target is the brute-force simulation. The data, d_i and m_i, are the values for the amplitudes of the $|0^n\rangle$ state produced by the brute-force and probabilistic simulator, respectively, during the i^{th} trial.

Conclusion Results in Figure 2.3 show that the average of the simulator outputs exhibits strong correlation with the true values from a brute-force simulation, giving a coefficient of determination $R^2 = 0.9619$. As such we can have confidence in our choice of simulator for the problems we will tackle in the following sections.

2.3.2 Part 2 : Device Benchmarking

Continuing to follow the method of Section 2.1.2, Part 2 of each numerical experiment is to impose the constraints that come from the experimental system used. In the fol-

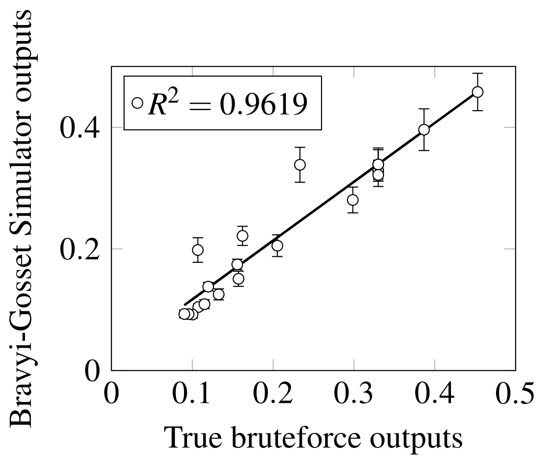


Figure 2.3: **Comparison between brute-force outputs and probabilistic Bravyi-Gosset Simulator outputs when calculating the probability of measuring the $|0\rangle^n$ state for 20 random X-programs.** Each point indicates the mean probability of measuring the $|0\rangle^n$ state for one fixed X-program according to the simulator, with the error bars indicating one standard deviation in the probabilistic simulator's output. The number of qubits is in the range [5,12] and the T gate count is in the range [5,15]. Details of this simulation can be found in Appendix A.2.1. Strong correlation is observed with $R^2 = 0.9619$. Here, unlike in later plots, the axes are not scaled as the probabilities are of a reasonable magnitude due to the smaller circuit sizes.

lowing we restrict, with differing degrees of strictness, problems previously mentioned, to the NQIT Q20:20 architecture.

NQIT Q20:20 Noise Restricted 2D-DQS

We consider the 2D-DQS problem as introduced in Protocol 1.5.1 and constrain it according to the noise of NQIT Q20:20 as listed in Section 2.2.3. For simplicity, we use a modified version of the NQIT Q20:20 architectural restraints of Section 2.2.2; namely we assume a 2D square lattice connectivity between qubits, as detailed below.

Constraints By making the simplifying assumption that we use a single logical qubit per ion-trap[11] we can map every grid vertex of the 2D-DQS circuit onto a single NQIT Q20:20 device ion trap. Figure 1.10 and Figure 2.2 then reveal that the 2D-DQS problem can be easily overlaid onto the NQIT Q20:20 architecture, which also permits the necessary measurements, state preparations, and single and 2-qubit gates.

As the adapted NQIT Q20:20 architectural restraints, detailed above, adhere to those required for the 2D-DQS problem seen in Protocol 1.5.1, the worst case additive error hardness result of the 2D-DQS problem, as seen in Theorem 1.5.3, applies. While we have agreed that this setting constitutes one that is worthy of investigation, as the

[11]Using more qubits per ion-trap could be possible, but then the connectivity of qubits would not be identical to that of the problem considered. Since in this example we focus on the issue of noise, we make this assumption and let non-trivial architectural constraints be considered in the next numerical experiment.

noise levels are independent for each qubit and not dependent on the problem size, the additive error permitted by Theorem 1.5.3 is likely exceeded. Hence, we would expect that in the noisy case the distribution becomes far from the perfect one and for the quantum computational supremacy to diminish.

Simulation We consider 4×5 grids, modelling 20 ion traps in total, and use them to perform the 2D-DQS computation of Protocol 1.5.1. Note that this is more ion traps than are currently available experimentally and so this experiment explores the performance of the technology as it scales. Protocol 1.5.1 requires, on average, half as many T gates as qubits; in this case 10 and 20 respectively. Details of the numerical specifics of the experiments can be found in Appendix A.2.2. Here it suffices to say that we use four steps to generate the entangled 2D cluster. The number of steps plays a role in the amount of noise as it determines the duration of the computation.

We perform 20 trials, each concerning one perfect circuit and a random output string. For each trial there are 20 noisy runs, each with their own random noisy version of the trial's circuit. This random noisy version of the perfect circuit is generated by considering the noise type and strength of the experiment as described in Appendix A.2.2. We simulate all 21 circuits 20 times, calculating the mean probability of measuring the corresponding bit string in each case. We will then take the mean and standard deviation of the noisy runs.

While, as noted in [134], simulation of up to 40 qubits and 50 T gates is possible using this simulator, as is also noted in that work, doing so takes several hours. In our case we simulate 20 trials, each with 21 runs and 20 simulations per run and so we restrict the number of qubits and T gates to a more manageable amount. Later in this work we go further and perform many thousands of simulations in each numerical experiment, justifying our restriction. Notice that this this is a relatively small number of output strings to explore, especially considering that the distributions considered may have support on up to 2^{20} output values. However, as we discuss in the following 'Figure of Merit' subsection, the outputs are selected to belong to a particularly important subset of all outputs. In particular, by using the accuracy and far from uniformity condition we restrict to a subset of outputs whose accuracy is indicative of the accuracy across the whole distribution, as discussed in the introduction to Section 2.3.

Figure of Merit For this numerical experiment we will utilise the 'accuracy and far from uniformity of noisy runs' condition from the introduction to Section 2.3. In particular, we will consider a perfect run to be far from uniform when it is either greater than twice the uniform value, or less than half. The outputs are randomly generated but post-selected to be of this form. In this way we will identify if the noise level reveals that, as we expect, the potential for a demonstration of quantum computational supremacy should be dismissed, rather than if one could be achieved.

Conclusion The results are shown in Figure 2.4 where we have plotted the value for the perfect run, and the mean value for the noisy runs. As expected, including noise at the levels of the NQIT Q20:20 device leads to an outcome probability that is between the ideal and the totally random output. However in most cases the noise that

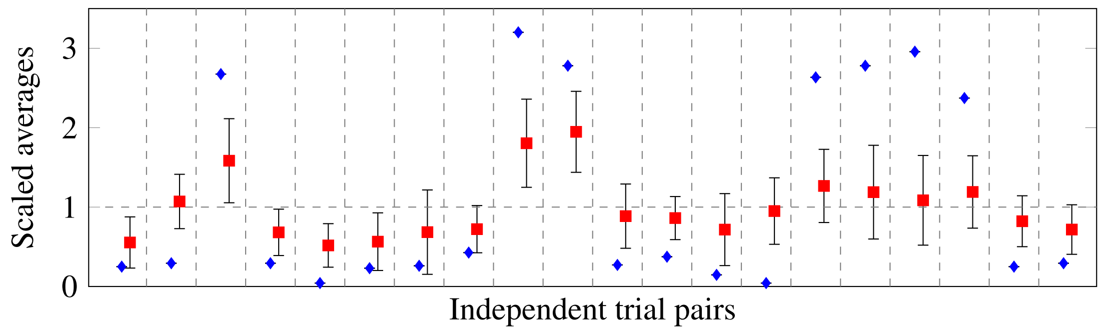


Figure 2.4: **Comparison between ideal and noisy circuit results for a** 4×5 **ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has a different initial 2D-DQS circuit, and different output string. Every consecutive pair is one trial and contains the perfect run (blue diamond), and the mean of the noisy runs (red square). The error bars indicate one standard deviation of the noisy runs. The means and standard deviations for each trial have been normalised by the uniform distribution (dotted horizontal line).

we include leads to a result within one standard deviation of the uniform distribution, or greater than one standard deviation from the perfect run. Referring to our figures of merit, we regard this to be a sign that the 2D-DQS computation run on the simplified NQIT Q20:20 architecture explored here is unsatisfactory for demonstrating quantum computational supremacy with NQIT Q20:20 noise at its current levels.

In Section 2.3.3 we use the simulator as a tool to investigate which of the aspects of our noise model are the main sources of this failure. To form a complete picture, and to benchmark the device's performance when implementing these problems, we must compare our numerical experiments with actual experiments. This work concerns only numerical experiments, while in the future we plan to collaborate with experimental groups to provide these benchmarks.

NQIT Q20:20 Noise and Architecture Restricted IQP-MBQC

The second numerical experiment we perform takes the unrestricted IQP-MBQC of Section 1.5.4 and imposes constraints equivalent to the architecture of NQIT Q20:20. We consider the case where each ion-trap has multiple physical qubits, as discussed in Section 2.2.2. We restrict to IQP instances involving gates acting on qubits belonging to neighbouring ion-traps so as to further lower the circuit depth.

Constraints In principle different gates of an X-program may act on any subset of qubits, or in the MBQC model, the ancillary qubits may be entangled with any subset of the primary qubits. This is not realistically achieved in the NQIT Q20:20 setting, where qubits belonging in different ion-traps cannot be connected arbitrarily with qubits of other ion-traps. Since NQIT Q20:20 admits universal quantum computation, one could achieve arbitrary connectivity by using SWAP gates between the qubits. However, by using SWAP gates the advantage of smaller waiting times offered by IQP is destroyed. We thus impose conditions on the connectivity, limiting the class of problems we use.

We assume each ion-trap has $K = 20$ physical qubits, of which 10 are dedicated to entanglement distillation, leaving $K' = 10$ for use in computation. As discussed in Section 2.2.3, this allows us to fix a constant two-qubits gate noise, whether the gate involves qubits in the same or neighbouring ion-traps. This does not apply to the waiting time, which is greater in the case of gates between ion-traps.

We will choose the minimum links between different ion-traps (while maintaining full connectivity within each trap). This means a 1 dimensional configuration of ion-traps.[12] This, in itself, might not be a big restriction, since even considering two-qubit gates that act on nearest neighbour qubits only, as discussed in Section 1.5.2, is still believed to be a hard problem. However, this configuration, while it is not 1 dimensional as far as the qubits are concerned, is still likely to admit a classical efficient simulation based on tensor networks. In contrast, in the first numerical experiment, there *is* a complexity-theoretic proof of hardness. However, since our purpose in this section is to illustrate how to implement architecture constraints, the issue of classical hardness in comparison to the best classical methods, is not crucial. Indeed it is likely that reasonable predictions can be made about the impact of noise on the 2 dimensional architecture of NQIT Q20:20, outputs from which are less likely to be reproducible on a classical computer, using results from these 1 dimensional simulations. It is in this way that the distributions of problem instances explored provides insights into the practical performance of the device.

In IQP-MBQC, applying gates between primary qubits corresponds to entangling them with the same ancillary qubit. In the case that the primary qubits belong to different ion-traps, the gate is applied using teleportation, with the help of entanglement links distilled between neighbouring ion-traps. Protocol 2.3.1 shows how to achieve this using only one entanglement link between the two ion-traps. Distilling entanglement between multiple traps takes a longer time, which is why we restricted our attention to X-programs that involve gates with qubits in at most two ion-traps.

In this setting, each ion-trap is connected by entanglement links to two neighbouring ion-traps. Each ion-trap has one ancillary qubit (a in Protocol 2.3.1) and one qubit reserved to receive the ancillary qubit coming from its neighbour (c in Protocol 2.3.1). This leaves 8 primary qubits. This entanglement structure can be achieved in two time-steps. First, all ion-traps at odd positions use their entanglement links to teleport the qubit required using Protocol 2.3.1. This is repeated for all even positions. With these restrictions X-programs can be mapped to the NQIT Q20:20 architecture. An example of an MBQC graph for such restricted instances is given in Figure 2.5.

Simulations A full description of the simulation procedure can be seen in Appendix A.2.1. In summary, we let each ancillary qubit act on a random subset of the primary qubits in its own ion-trap before, after being teleported, acting on a random subset of the qubits in the next ion-trap. We performed 20 trials, each involving a randomly generated circuit of the form described above, along with a random output string. Each

[12] We could consider the 2 dimensional case too, as in the first numerical experiment, but our choice is the simplest and within reach of our classical simulator. A 2 dimensional configuration would require a larger number of traps, which is outside of our simulation capabilities.

Protocol 2.3.1 This algorithm constructs part of the resource state for a given ancillary qubit a in trap 1 according to its corresponding row p of the X-program $\boldsymbol{Q}$. Q_1 is the set of all qubits in cell 1 with $a, l_1 \in Q_1$. Analogously, $c, l_2 \in Q_2$. c is the qubit that will eventually be used for measurement after a's value is teleported there.

```
function EntangleTwoTraps($p$, $a$, $c$, $l_1$, $l_2$, $Q_1$, $Q_2$)
    for all $q \in Q_1 : p(q) = 1$ do
        CZ $(a, q)$
    end for
    CZ $(a, l_1)$
    Distil a Bell pair between $l_1$ and $l_2$
    Bell measurement on $(a, l_1)$ which teleports $a$ to $l_2$
    SWAP $(c, l_2)$
    for all $q \in Q_2 : p(q) = 1$ do
        CZ $(c, q)$
    end for
end function
```

trial has one noisy and one perfect run. A perfect run involves simulating the perfect circuit several times and calculating the mean probability of measuring the selected output string. A noisy run is equivalent but with a random noisy instance of the circuit.

In this case, at their largest, we simulate significantly more qubits than in the previous and following sections. The largest circuit we simulate has 12×8 qubits but still only 10 T gates on average. This is because we have limited the probability that a T gate will be required, which corresponds, as discussed in Appendix A.2.1, to limiting the probability of creating connections between the ancillary and primary qubits. As the computation time grows exponentially with the number of T gates, and polynomially in the number of qubits, we can afford this increase in the qubit count. Again, the number of qubits simulated is greater than the number available experimentally.

Figure of Merit In this case, as we expect that the architectural restrictions used will make a demonstration of quantum computational supremacy using this scheme unlikely, we will not consider the figures of merit as in Section 2.3.2. Instead we again consider the coefficient of determination as in Section 2.3.1 to establish the impact of noise models more broadly. Here the model outputs m_i are the probability amplitudes from the noisy run, while the target outputs d_i are those from the perfect run.

Conclusion We compared the two means of each run to calculate the coefficient of determination. In the case of the maximum system (12 ion-traps, with 8 primary qubits each) we noticed that, with the existing level of noise, the results corrupt fully the output leading to $R^2 \approx 0$. We then ran similar experiments for smaller instances. Lowering the number of qubits, we observed that the R^2 value was increasing but still remained extremely low with NQIT Q20:20 noise level. Decreasing the size yielded the results seen in Table 2.2.

These R^2 values, far below one, indicate that even for small system sizes, the noise

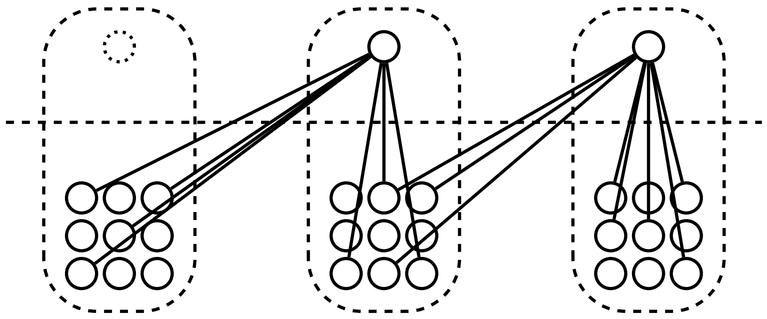


Figure 2.5: **An example of a restricted** IQP**-MBQC pattern for 3 traps, where primary qubits are on the bottom and ancillary qubits are on the top.** Ancillary qubits are still physically in the cells with the primary ones, although they are separated by a horizontal dotted line here for clarity. We have one ancillary qubit for every two neighbouring cells, with considerations made for boundary cases. Once an ancillary qubit is entangled in its native trap it is moved. There is one less ancillary qubit than the number of traps so that each is entangled to two traps. The dotted ancillary qubit indicates a location which has been vacated when the ancillary qubits move between traps. The reader may wish to return to Figure 2.2 where, like here, the dashed bubbles indicate individual ion traps with a single qubit in each acting between them.

$a \times b$	12×8	9×8	4×8	4×2
R^2	0.0086	0.0237	0.0333	0.5561

Table 2.2: **Coefficient of determination between ideal and noisy** IQP**-MBQC instances, restricted to the noise and architecture of NQIT Q20:20.** Here $a \times b$ means a ion-traps with b primary qubits per trap

is too high and there is little correlation between the perfect and noisy runs. For this reason, and because theoretical results about quantum computational supremacy in this case are not as strong, in the subsequent section where we examine the effects of varying noise, we restricted attention to the numerical experiment of Section 2.3.1 only and do not proceed to Part 3 of the numerical experiments in this case.

2.3.3 Part 3 : Guiding Future Experiments Using NQIT Q20:20 Noise Restricted 2D-DQS

To identify the main sources of error in the numerical experiment of Section 2.3.2 we run experiments with varying noise levels. In this section, the protocol we implement will be the 2D-DQS of Protocol 1.5.1 as detailed in Appendix A.2.2. We group the different noise types of Section 2.2.3 together and identify which contributes most to the corruption of the perfect output. We then "fine-grain" further by considering the different types of noise within that group. Once we have identified the main source of error, we will explore how the potential for a demonstration of quantum computational supremacy is affected by reducing this noise, both by known error correction techniques, and hypothetical proposals.

In these numerical experiments we will use the same constraints and simulation design as in the first 2D-DQS simulations of Section 2.3.2. The difference here is the noise model used. In particular, we will be comparing random single output probabilities. We will also use the same 'accuracy and far from uniformity of noisy runs' figure of merit as in Section 2.3.2 in order to identify when a demonstration of quantum computational supremacy is unlikely. As we identify cases where such a demonstration is not unlikely, we will explore proxy measures for the ℓ_1-norm distance and relate these measures back to the theoretical results, in Section 1.5.2, on the conditions for a demonstration of quantum computational supremacy using the 2D-DQS protocol.

Operation-Based Verses Time-Based Noise

At the coarsest level of detail, we divide the noise sources into what we call 'time-based noise' , and 'operation-based noise' . Time-based noise consists of depolarising and dephasing noise, both of which act at all points in the circuit with strength dependent on the time for which they are applied. Operation-based noise includes noise during state preparation, measurement, single and two qubit gates (including the noise during distillation) and act only when an operation is enacted. In each run we eliminate either the time-based noise or operation-based noise, while keeping the other at the same level as in the NQIT Q20:20 device. Results for the behaviour of outputs with far from uniform probability in the ideal output distribution can be seen in Figure 2.6.

We can see that the largest contribution to the corruption of the output appears to be from the time-based noise. With reference to our figures of merit, including only time-based noise almost always brings the output probability of the bit string in noisy runs to within one standard deviation of the uniform value, or greater than one standard deviation away from the perfect run amplitude value. As such we conclude that it is a significant obstacle to demonstrating quantum computational supremacy. On the

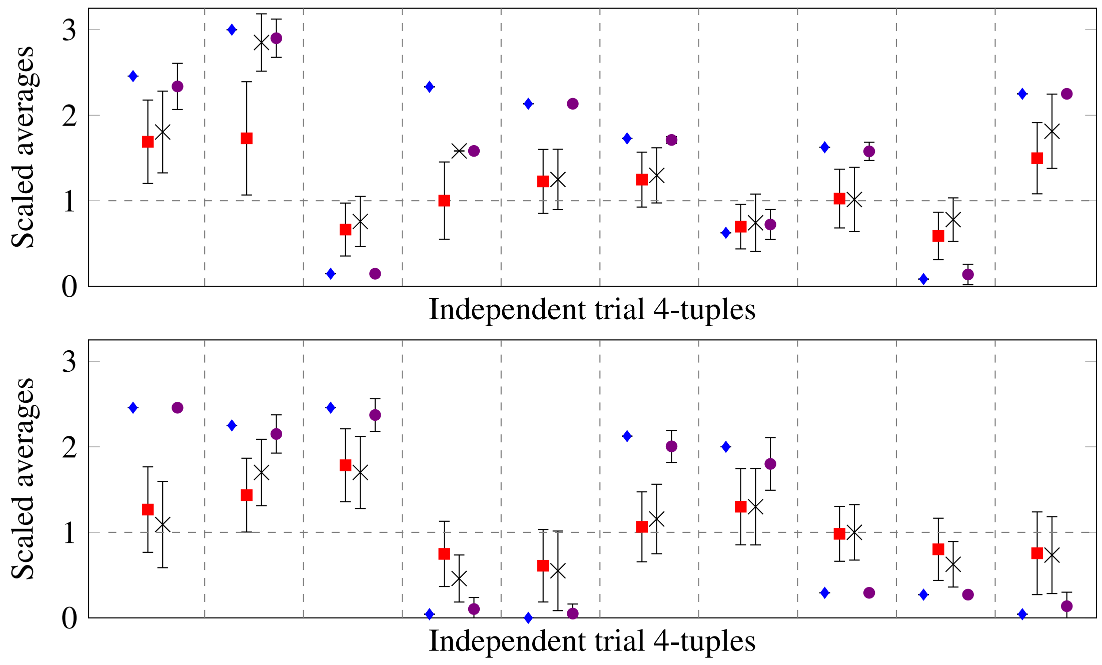


Figure 2.6: **Results including either only gate based noise or only time based noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has a different initial 2D-DQS circuit, and different output string. Every independent trial is described by a 4-tuple of a perfect run (blue diamond), the mean of 20 noisy runs (red square), the mean of 20 only time-based rates noisy runs (grey cross) and the mean of 20 only gate rates noisy runs (violet circle). The error bars show one standard deviation. The means and standard deviations have been normalised by the respective uniform distribution (dotted horizontal line).

other hand, as the randomly selected bit string amplitude, when only gate based noise is considered, is in all but one case within one standard deviation of the perfect run, and further than one standard deviation from the uniform distribution value, we do not immediately conclude that it is a significant obstacle.

Below are values for a proxy for the ℓ_1-norm distance between the ideal and noisy distributions for the noise levels discussed above, calculated as follows. Here a trial consists of an ideal run, measuring the probability of a random output of a random 2D-DQS circuit of the form discussed in Protocol 1.5.1, and 20 noisy runs for each noise type, considering noisy versions of the ideal circuit. The average difference between the noisy and ideal runs within each trial are themselves averaged to give a proxy for the ℓ_1-norm distance, once scaled by the uniform distribution. Note that this proxy does not provide a bound on the value of the ℓ_1-norm distance between the noisy and ideal distributions, but instead gives an approximation of this value. Each run is itself the average of 20 simulations of the same circuit. A similar pattern is seen in this data as was identified in the study of single outputs; namely that the largest contribution to the deviation of the noisy distribution from the ideal is a result of the time based noise.

full noise levels	only time base noise	only gate based noise
0.286316488	0.276119941	0.033008605

As discussed in Section 2.3 our analysis of both far from uniform outputs and the ℓ_1-norm distance lead us to regard a system with reduced time-based noise as relatively more likely to demonstrate quantum computational supremacy than a system with reduced gate-based noise. Removing time based noise results in a value below the $\frac{1}{22}$ specified in Theorem 1.5.3 suggesting that a demonstration of quantum computational supremacy may be possible here. We hope to identify the main source of error more precisely, and as such we continue to explore the reduction of time-based noise.

Depolarising Versus Dephasing Noise

We now look more closely at the time-based noise and consider separately the contribution from dephasing noise and from depolarising noise. The results for outputs with far from uniform probability in the ideal output distribution are seen in Figure 2.7.

In this case, the amplitudes produced by runs considering only dephasing noise are always either within one standard deviation of the uniform distribution, or greater than one standard deviation from the perfect run. By comparison the runs considering only depolarising errors are always within one standard deviation of the perfect run, and greater than one standard deviation from the uniform distribution output.

Below are values for the same proxy for the ℓ_1-norm distance between the ideal and noisy distributions discussed above, but for the noise levels considered in this section. A similar pattern is seen in this data as was identified when considering the accuracy and far from uniformity figure of merit; namely that the largest contribution to the deviation of the noisy distribution from the ideal is a result of the dephasing noise.

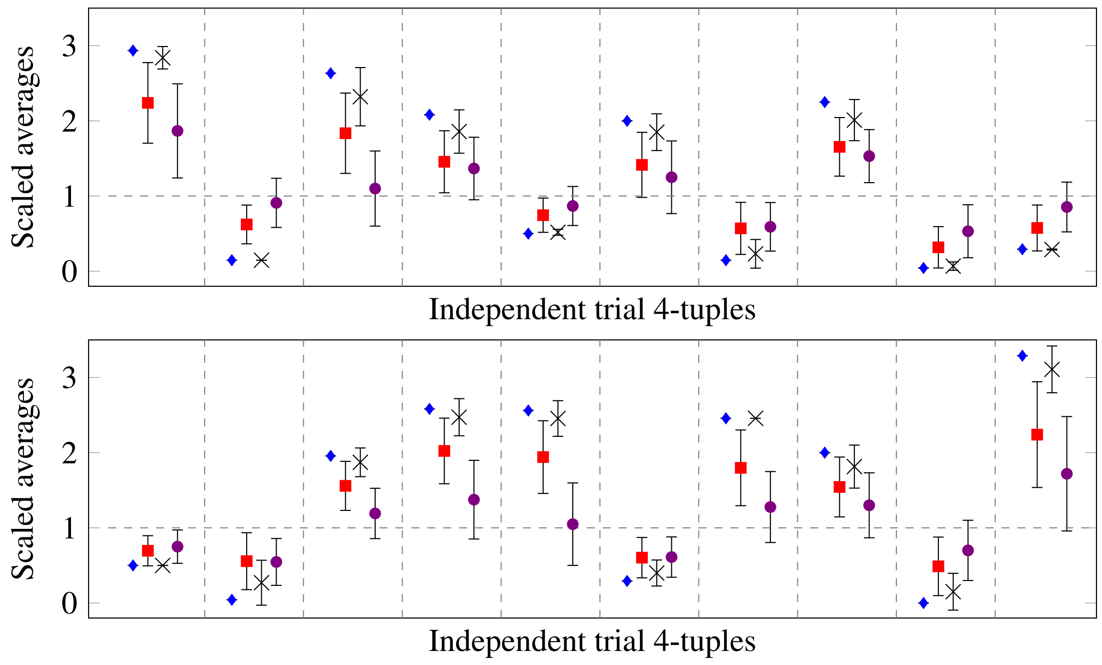


Figure 2.7: **Results including either only dephasing or only depolarising noise rates for a** 4×5 **ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has a different initial 2D-DQS circuit, and different output string. Every independent trial is described by a 4-tuple of a perfect run (blue diamond), the mean of 20 noisy runs (red square), the mean of 20 only depolarising rates noisy runs (grey cross) and the mean of 20 only dephasing rates noisy runs (violet circle). The error bars show one standard deviation. The means and standard deviations have been normalised by the respective uniform distribution (dotted horizontal line).

full noise levels	only depolarising noise	only dephasing noise
0.433746955	0.111777366	0.4555678

As as a result of the analysis of these two figures of merit, we identify dephasing error as a larger obstacle to a demonstration of quantum computational supremacy than depolarising noise.

The Impact of Noise Reduction by Error Correction

Having identified the main obstacle to a demonstration of quantum computational supremacy to be dephasing errors, we examine the effect that reducing this type of noise would have. Concretely, one could introduce a phase-flip code[13] [28]. Recall that in the numerical experiments of Section 2.3.1, we only used a single qubit from each ion-trap. This means that we could use three qubits from the ion-trap to implement one round of phase-flip code, which would reduce the dephasing noise. By using such a simple phase-flip code we obtained an effective improved dephasing rate of $\approx 2.3 \times 10^{-4}$ per second as compared to the one of the NQIT Q20:20 noise-level $\approx 7.2 \times 10^{-3}$ per second. The results for outputs with far from uniform probabilities are found in Figure 2.8.

In this case, roughly half of the runs considering the error corrected dephasing pass our test that the probabilities should be at least within one standard deviation of the perfect run, and greater than one standard deviation of the uniform distribution. This demonstrates partial improvement while being inconclusive as a demonstration of the potential for quantum computational supremacy. In this case an analysis of the ℓ_1-norm distance is particularly valuable.

The readers will find the data required for such an analysis below. In this case, as in the case of the previous figure of merit, a large improvement can be achieved by utilising a simple repetition code. However this improvement might not be as significant as one might expect having seen the results of Figure 2.8 with the ℓ_1-norm distance still being significantly far from the $\frac{1}{22}$ value required by Theorem 1.5.3. Note that the particular value of $\frac{1}{22}$ appearing in Theorem 1.5.3 applies to asymptotically large circuits, but remains of interest in the case of the small experiments conducted here. In particular, we would expect the ℓ_1-norm distance between noisy and ideal distributions to grow as the computations grow larger, and so grow more vulnerable to noise. As such we take as a guide that the noisy distributions of small circuits should be within at least an ℓ_1-norm distance of $\frac{1}{22}$ of the ideal. Indeed even without dephasing noise the ℓ_1-norm distance its too high to expect a demonstration of quantum computational supremacy. While potentially disappointing, this reveals the utility of our approach as we are able to identify dominant noise sources and the effect of approaches to implementing them without the need to construct real experiments. Indeed we are able to predict that both improved error correction codes and error correction applied to other noise channels are required for a demonstration of quantum computational supremacy.

[13]This idea was suggested earlier by Niel de Beaudrap when their initial analysis of the noise model [224] showed dephasing to be the major source of error.

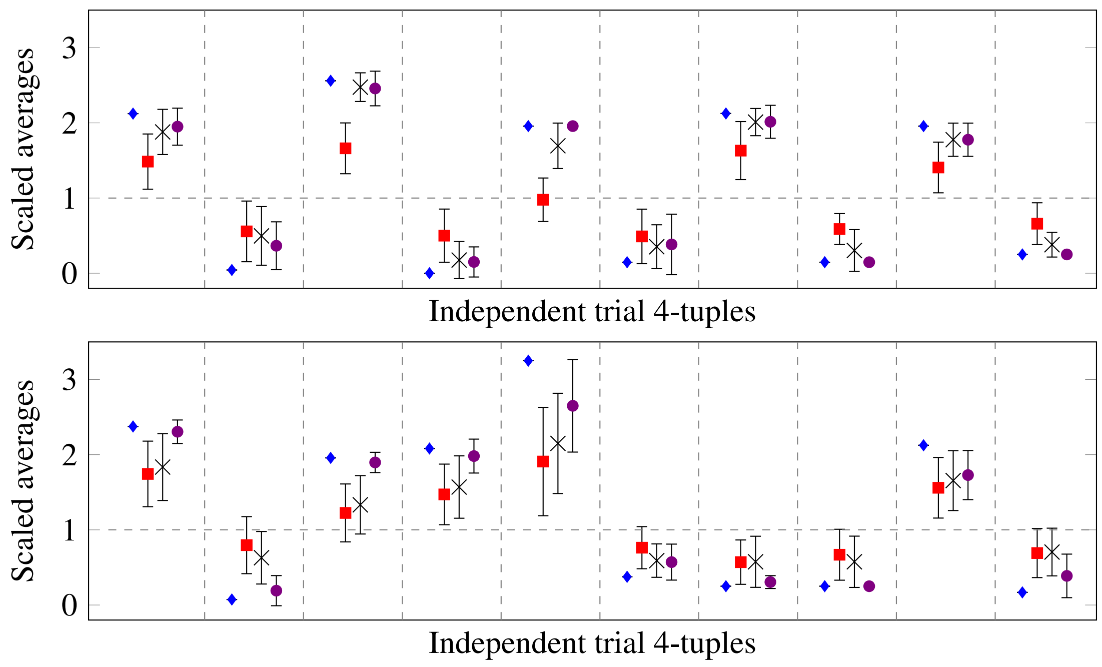


Figure 2.8: **Results including reduced dephasing noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has a different initial 2D-DQS circuit, and different output string. Every independent trial is described by a 4-tuple of a perfect run (blue diamond), the mean of 20 noisy runs (red square), the mean of 20 dephasing rates reduced by repetition code noisy runs (grey cross) and the mean of 20 no dephasing rates noisy runs (violet circle). The error bars show one standard deviation while. The means and standard deviations have been normalised by the respective uniform distribution (dotted horizontal line).

full noise levels	with repetition code	without dephasing noise
0.321564704	0.270893212	0.0656717

This is a partial improvement relative to the uncorrected results. Hence we find a demonstration of quantum computational supremacy using this error correction scheme to be more likely than in the uncorrected case. However, further improvements are required for such a demonstration.

The Impact of Continuous Noise Reduction

More generally than testing a single error correction code, we can understand how the likelihood of a demonstration of quantum computational supremacy is affected with a continuously varying noise parameter. Here we will consider dephasing errors, which we have identified as the most damaging form of error. This continuous variation corresponds to, for example, reductions in the gate application time, improvements in the compilation methods or the improved storage of quantum states. The results of this experiment are shown in Figure 2.9.

While Figure 2.9 appears to demonstrate the continuous improvement which can be achieved by reducing the dephasing error, it seems that it cannot be said that the amplitudes are regularly within one standard deviation of the perfect run until the dephasing rate is reduced to 0. We do however see that, with regards to our accurate and far from uniform condition, a demonstration of quantum computational supremacy does become continuously more likely as the dephasing error rate is reduced.

This fact is reinforced by Figure 2.10 which shows the average difference between the perfect and noisy runs for each of the values of dephasing error rate. We can use this as a proxy measure for the ℓ_1-norm distance, as discussed in the experimental design methodology introduced in Section 2.3, and as was done earlier in Section 2.3.3. In this case we can say that an experiment has a reasonable chance of demonstrating quantum computational supremacy if we can be convinced that the ℓ_1-norm distance between the noisy and perfect implementations is bounded by $\frac{1}{22}$ which is demanded by the hardness result for the 2D-DQS algorithm as seen in Theorem 1.5.3. As we do not have access to the full characterisation of the probability distributions, here we will approximate the ℓ_1-norm distance by taking the average difference and proposing that it is representative of the full distribution by scaling it by the uniform distribution.

We see that even in the case of 0 dephasing error, the ℓ_1-norm distance is not brought within the $\frac{1}{22}$ value. Instead the average difference in that case is approximately 0.155 which is significantly higher. However, by our figure of merit, a demonstration of quantum computational supremacy is made continuously more likely by this fall in dephasing error, showing the advantage in endeavouring to achieve such a fall.

An alternate proxy measure for the ℓ_1-norm distance is to explore the differences between the noisy and perfect amplitudes for a selection of different output bit strings of the same circuit. In Figure 2.11, every trial considers the same 2D-DQS circuit, but measures the probability amplitude of a different output bit string.

This plot can again be examined further by studying the average difference between

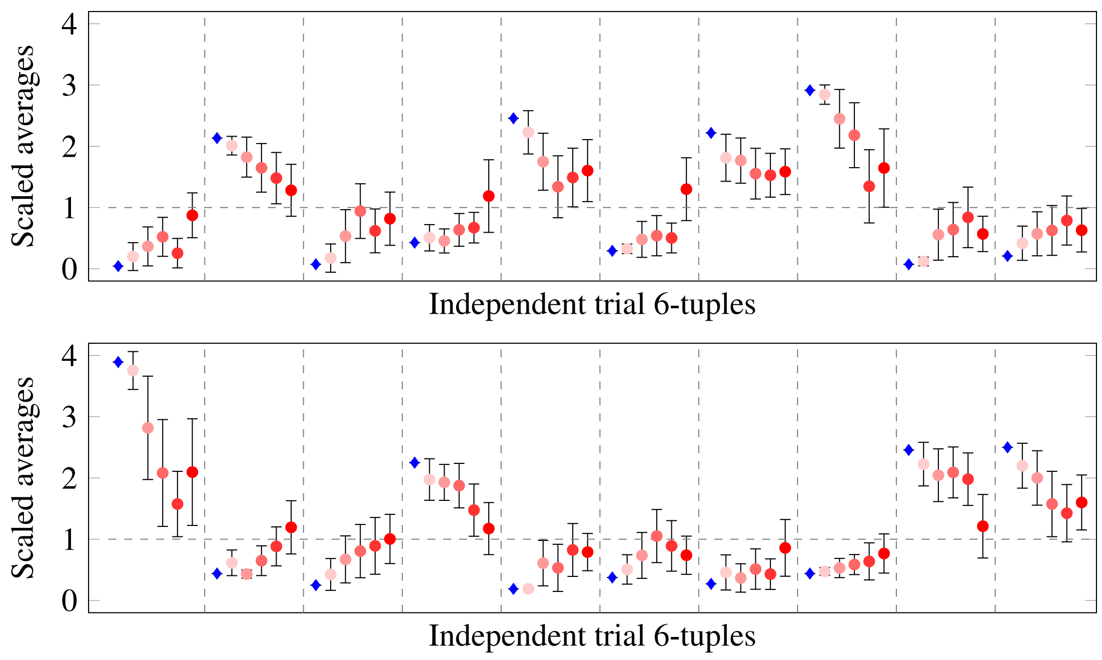


Figure 2.9: **Results including reduced dephasing noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has a different initial 2D-DQS circuit, and different output string. Every independent trial is described by a 6-tuple, from left to right, of a perfect run (blue diamond), the mean of 20 noisy runs with no dephasing errors, the mean of 20 noisy runs with $\frac{1}{4}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with $\frac{1}{2}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with $\frac{3}{4}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with the NQIT Q20:20 dephasing rate. The error bars show one standard deviation. The means and standard deviations have been normalised by the respective uniform distribution (dotted horizontal line).

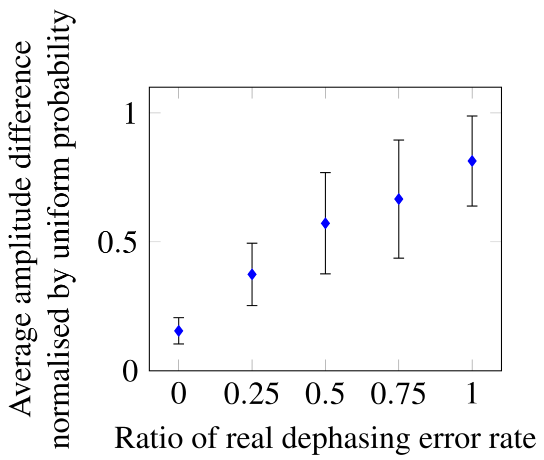


Figure 2.10: **Results for varying dephasing noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the difference between the probability amplitudes in noisy and perfect runs when measuring a randomly chosen output string of a random 2D-DQS circuit. The error bars show one standard deviation. The means and standard deviations have been normalised by the uniform distribution.

the noisy and perfect runs. This proxy measure for the ℓ_1-norm distance is plotted in Figure 2.12 but once again the value of 0.135 is more than twice the $\frac{1}{22}$ which is demanded by the hardness result for the 2D-DQS algorithm as seen in Theorem 1.5.3.

In conclusion, while it seems that reducing, and indeed removing, dephasing error alone will not be enough to bring a demonstration of quantum computational supremacy using this scheme within reach, we have seen that utilising a simple 3 qubit correction code would result in a significant improvement on the noise levels. We recommend that this error correction technique is used in conjunction with other techniques, correcting for other error types.

We expect, however, that as the system size grows the ℓ_1-norm distance between the perfect and noisy circuits will grow as the noise modelled is constant for each gate and qubit. This would push a demonstration of quantum computational supremacy further away. Indeed it is known [64], as discussed in Section 1.5.2, that samples can be efficiently drawn by a classical computer from a distribution produced by an IQP circuits subject to independent depolarising noise on each qubit at the end of the circuit. In that case, however, error correction can be used to recover classical impossibility, if one allows for more complex connectivity, or several rounds of SWAP gates. While we have restricted the connectivity and circuit depth in our case, there may be gains to be made by removing these restrictions.

2.4 Conclusion

We have examined classical simulation of small instances of realistic quantum computational supremacy computations. The motivation is not to obtain solutions to the problems considered, but to faithfully model the physical system and computation de-

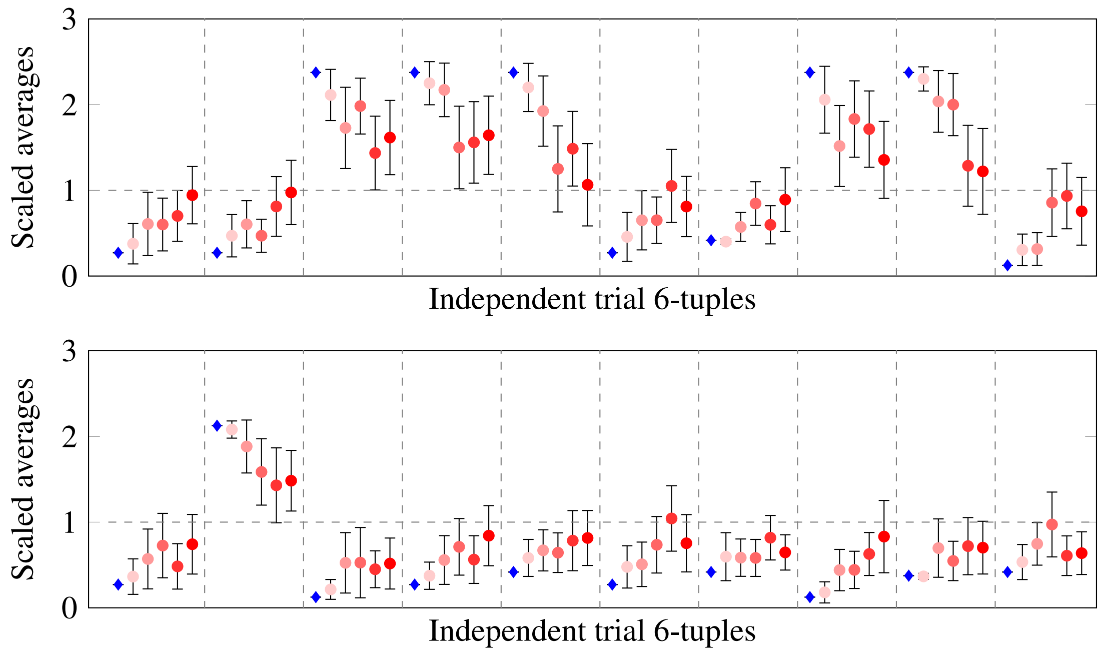


Figure 2.11: **Results including reduced dephasing noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the probability of measuring a randomly chosen output string, where each trial has the same initial 2D-DQS circuit, and different output string. Every independent trial is described by a 6-tuple, from left to right, of a perfect run (blue diamond), the mean of 20 noisy runs with no dephasing errors, the mean of 20 noisy runs with $\frac{1}{4}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with $\frac{1}{2}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with $\frac{3}{4}$ of the NQIT Q20:20 dephasing rate, the mean of 20 noisy runs with the NQIT Q20:20 dephasing rate. The error bars show one standard deviation while. The means and standard deviations have been normalised by the respective uniform distribution (dotted horizontal line).

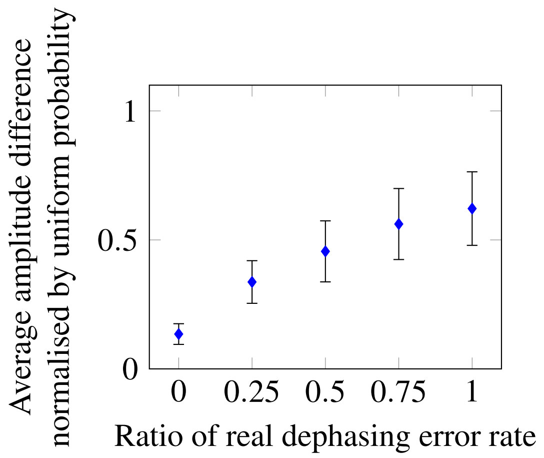


Figure 2.12: **Results for varying dephasing noise rates for a 4×5 ion trap grid.** The results referenced by this plot are the difference between the probability amplitudes in noisy and perfect runs when measuring a randomly chosen output string from a single 2D-DQS circuit. The error bars show one standard deviation. The means and standard deviations have been normalised by the uniform distribution.

vice. Having achieved a faithful modelling of the system, classical simulations can be used as a tool in two ways. Firstly, we can use them to benchmark a given device by confirming that the effect of the modelled noise scales correctly. If instances increase in size and continue to match outcomes of real experiments, we extrapolate that the same is true for the, non classically simulatable, quantum computational supremacy regime. Secondly, we can examine the impact of varying the noise and other constraints and imperfections. By doing so one can identify which limitations contribute most to the degradation of the results, compared to the perfect case. We can then provide feedback to experimentalists as to which aspects of their system they should prioritise in improving, in order to achieve the best results in the specific problem considered. Importantly the required improvements can be identified without partaking in the resource intensive process of building larger devices.

We gave a methodology for using classical simulations in the way described above, and exemplified this methodology with two examples. In both cases, we considered IQP problems, one of the prominent candidates for demonstrating quantum computational supremacy. The constraints we imposed were those from the NQIT Q20:20 device [13, 224], while the classical simulator used was the Bravyi-Gosset Simulator developed in [134].

The first example used was a subclass of IQP instances, called the 2D-DQS problem, defined in [24] and outlined in Section 1.5.2. The main focus when exploring this example is the effect of noise. While current NQIT Q20:20 levels of noise are too high, by using our technique we identified that dephasing noise is the most significant source of errors. This led us to a potential solution to improve such computations, namely to use a small phase-flip code to protect from precisely this type of errors, which we showed provided improvements. We also showed that a continuous improvement in the likelihood of a demonstration of quantum computational supremacy can be achieved by a continuous improvement in the dephasing noise levels. However, we also showed that correcting dephasing error alone would not be sufficient to demonstrate quantum computational supremacy using the 2D-DQS protocol on NQIT Q20:20 hardware.

In the second example, we considered an IQP-MBQC instance with constraints coming, this time, from architectural limitations. This example was to illustrate how to model different architectures in our framework. We noticed that the current level of noise of NQIT Q20:20 was even more destructive than in the first example.

We give several directions for future research within the parameters of the motivations of this chapter, both specific to the examples considered and more general involving the methodology developed. In Section 2.3 we provide a tool for benchmarking NQIT Q20:20, but to do such benchmarking, one needs to run these examples on the NQIT Q20:20 device and compare with the modelling we obtained. This is naturally the first next step complementing our work. A second direction is to derive theoretical prediction for the effect of noise on our examples, for our problems and with our constraints. This continues the work of [64] as discussed in Section 1.5.2 and reveals what is required to achieve a demonstration of quantum computational supremacy.

Moreover, the use of these simulations as a tool for guiding future experiments should be made more systematic. In Section 2.3.3 we varied the noise starting from coarser grouping of the noise-sources and going to a '*finer-graining*' in order to identify the major source of errors. We recommend modelling the reduction of a mixture of different noise sources as we have shown that removing only one, namely dephasing, would not be sufficient. This could also be enhanced with other techniques, which may also vary the architecture. Optimisation and machine learning techniques may be used to minimise both errors and experimental resources.[14]

Finally, the choices made in Section 2.2 will be heavily influenced by the development of classical simulation techniques and quantum technology. The methodology of Section 2.1 was derived to be sufficiently general that they can be applied once such advances have been made, and as models of devices and their noise change, and we encourage this pursuit.

The domain of relevance of the work of this chapter is that in which the power of the quantum devices of concern is less than that of available classical resources. We have exploited the predictive power of classical computers in that domain to guide the development of these smaller devices. Once the power of quantum devices approaches or surpasses that of classical computers, the utility of classical devices as predictors of the behaviour of quantum ones is limited.[15] However classical devices are still of use here, for example in assisting with the certification of demonstrations of quantum computational supremacy, as introduced in Section 1.6.4. In Chapter 3 we further extend the utility of classical computers once such quantum devices are developed. In particular we develop benchmarks to inform users as to the devices which are best suited for given applications.

[14]Indeed, since the completion of the work of this chapter, such approaches have been taken to assist with the design of superconducting circuit [230]

[15]It may be necessary that smaller quantum devices begin to be utilised to predict the behaviour of larger ones [231]

Chapter 3

Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack

In Chapter 2 we explored classical simulation as a tool for the development of quantum technology. As shown there, classical simulation is enlightening as a means of understanding the behaviour of small quantum computers, and as a means of predicting the behaviour of larger ones. However, as these larger devices are developed, it becomes necessary to measure their performance directly. These direct measures of performance can be used to compare devices,[1] assess the impact of changes to the quantum technology, and hence establish routes to making improvements. For these purposes, the performance of quantum computing devices is poorly described by simple, local noise models, such as those used in Chapter 2. For example, quantities such as gate error rates, or T_1 and T_2 times, measure the form and magnitude of noise present in the system, but are only *proxy* measures of the device's practical performance (i.e. when implementing computations).

In this chapter we will introduce a 'benchmark suite' to directly assess the performance of quantum computing devices. The implications of the results of our benchmark suite are sufficiently wide ranging for it to be adopted as a standard. In order to conduct the most comprehensive study of the practical performance of real devices, we take the view that the best benchmarks will encompass all possible influences on performance. As such we utilise "holistic benchmarks" which are those that test the quantum computational capabilities of the complete system. This captures the performance of the system as an integrated unit, rather than the performance of individual components. This philosophy leads us to benchmark the "full-stack" – qubits, compilation strategy, classical control hardware, etc. – collectively. Indeed it can be misleading to consider qubits in isolation of their control software, with performance depending heavily on how a circuit is optimised and distributed before it is run.

To reveal how different combinations of the quantum computing stack's components

[1]This should be compared to the benchmarking of classical computers, in which case the LINPACK benchmarks are used to build the TOP500 ranking of supercomputers [10].

change system performance, full-stack benchmarking should make explicit the variable components of the stack, and systematically vary those components to isolate how a particular one affects system-level performance.[2] Focusing on systems made available by IBM Quantum [83], we investigate two components of the stack[3]:

The compilation strategy used to map an abstract circuit onto one that is executable on a quantum computer.

The device used to run the compiled circuit and return the results.

Problem instances requiring compilation, which are often more representative of real world problems, typically show differing performance from those that do not [95]. As such, the design of new compilers for quantum circuits is an active area of research [114, 115, 234, 235]. The proliferation of compilers necessitates understanding how the inclusion of particular compilation strategies in the quantum computing stack affects performance. In particular, noise-aware compilation strategies, which use knowledge of the physical properties of the system's qubits to improve results, make assumptions about the influence of noise processes on overall system performance. Full-stack benchmarking is necessary to verify those assumptions.

The benchmarks defined here have two parts:

A circuit class describing the type of circuit to be run.

A figure of merit quantifying how well the system performed when running circuits.

Because quantum computing systems are used for particular applications, the circuit classes should test the performance of a system in those arenas. At least two notions have been put forth as to how to define such classes. The first proposes benchmarks based on often-used quantum algorithmic primitives [213], the examples given being primitives of Grover iterations [34] and Trotterized Hamiltonian simulation [27]. An alternative is to pick a particular instance of an application and check for the accuracy of the results returned by the system when running that instance. Naturally, to measure the performance of near-term systems the applications and instances must be well suited for these systems. Such benchmarks have been defined in the context of quantum simulation [86–88], quantum machine learning [89–91], and discrete optimisation [92–95]. The downside of this approach is that performance as measured by one instance of an application may not be predictive of performance for the application generically. Further, while the wide selection of circuits presented in the aforementioned literature covers an array of applications, deriving benchmarks independently of each other may result in a lack of coverage, or unnecessary repeated coverage, of circuit classes. This would be unsatifactory for a standard benchmark suite.

The "application-motivated" circuit classes defined here draw inspiration from both approaches by focusing on computational primitives of near-term quantum computing

[2]The improvements that can be achieved by a well constructed quantum computing stack is exemplified in [232]. There a doubling of quantum volume is achieved by careful optimisation at each layer, with only small improvements to the device itself.

[3]While the particular systems used here have other components, such as pulse synthesizers [233], because control of these components are not exposed to us at the time of this work's completion we do not look at the impact of those pieces on full-stack performance.

applications. This approach has the advantage that a system which does well on an application-motivated benchmark should do well in running the application the benchmark was derived from. Further, by considering application primitives, the result of the benchmarks apply to the application generally, and not only to a particular instance of the application. In summary the circuits used are similar enough to instances of applications to be predictive of their performance, while avoiding the risk that devices might be engineered towards very particular instances.[4]

Three such application-motivated circuit classes are introduced here. Drawing inspiration from the volumetric benchmarking approach, the classes cover varying depth regimes, and are controllable in depth.[5] The definitions and motivations for each class are given in Section 3.2 but, in brief, the classes – as labelled by their depth regimes – and the applications that motivate them, are:

Shallow: Inspired by 'hardware-efficient ansatze' [236, 237] (parametrised quantum circuits chosen to account for restrictions imposed by the device, such as qubit connectivity) which may be useful for near-term quantum machine learning and chemistry applications [4, 164, 165]. The width of shallow circuits grow with a minimal increase in depth, allowing the impact of including many qubits to be explored.

Square: Inspired by the circuits used to calculate a system's quantum volume [212]. These circuits utilise gates sampled uniformly at random from all $\mathrm{SU}(4)$ gates, making them a test of general-purpose, programmable quantum computers.

Deep: Inspired by product formula circuits, including state preparation circuits used in the Variational Quantum Eigensolver (VQE) algorithm [22, 23, 238–240]. The depth of these circuits grows quickly with width, giving a thorough coverage of depth regimes.

Importantly, as discussed in Section 3.2, these circuits are complementary; they cover a wide selection of applications and circuit types. By covering many applications with few circuit classes, we are able to concisely present far-reaching results, and minimise the number of computations to be performed in deriving those results. Note that we will not demand that a quantum computing stack performs well when implementing all of these classes of circuits, instead hoping to identify the applications where a quantum computing stack might be most fruitfully applied. Indeed, we may expect that devices are tuned to particular applications.

How well a stack executes a circuit is assessed here via continuous figures of merit, rather than binary ones which may only verify correctness. This is because the outcomes from noisy devices will likely not be correct, while information about closeness to the correct answer is still highly valuable. Note also that the techniques for the verification of universal quantum computation, as introduced in Section 1.6, require many

[4]Although we do not do so here, it is of interest to quantify the similarity of the benchmarking circuits to the applications they represent, and their 'coverage' there of. We discuss this further in Section 3.5.

[5]The circuit depth is the fewest number of time steps it would take to run a circuit if each gate took one time step, and gates with common qubits could not be run in the same time step.

qubits, or qubit communication, or both; none of which are accessible using present-day noisy devices. This also discounts the use of the scheme we introduce in Chapter 4 which requires qubit communication. Indeed, to reflect the current state-of-the-art, where devices are both few and poorly connected to one another, we will focus on examples of how classical computers can be used to perform benchmarks.

The three figures of merit used in this chapter are: heavy output probability [85], cross-entropy difference [62], and ℓ_1-norm distance. As discussed in Section 1.6.4, scaling this approach to tens or hundreds of qubits will be challenging in general. By considering circuits with few qubits we allow ourselves the ability to calculate these figures of merit, and to gain an insight into the behaviour of larger devices [60, 241]. However, significant improvements in the time needed to perform benchmarks can be made if the circuits and figures of merit are developed jointly. Indeed, the calculation of the heavy output probability and cross-entropy difference requires only a polynomial number of samples from a quantum device for some circuit classes. Importantly, we show that this is the case for deep circuits and square circuits. Therefore, deep circuits provide the first instance of chemistry-motivated circuits, which are likely not possible to classically simulate in general, but which can be benchmarked using polynomially many samples from the output distribution from a device implementing them.

We refer to a set of benchmarks as a *benchmarking suite*, each benchmark being defined by unique combinations of circuit class and figure of merit. Using a benchmarking suite enables the derivation of broad insights about the behaviour and performance of a quantum computing system across a wide variety of possible applications. The varying demands of each benchmark on the quantum computing resources allows for the exploration of the best routes to extract the most utility from near-term quantum computers. In sum, in order to predict the systems' performance in practice, our benchmarking approach is application-motivated, holistic and of the full quantum computing stack. For the reasons stated above, we regard all of these properties to be necessary of a standard benchmark suite.

In Section 3.1 we will reflect on why the figures of merit chosen are the correct ones for our purpose. In Section 3.2 we detail the circuit classes used, including algorithms for generating them. Section 3.3 introduces the software stack, as well as the devices we will be benchmarking while in Section 3.4 we give the results of our benchmarks, along with some analysis. We conclude in Section 3.5.

3.1 Figures of Merit

The figures of merit used in this work are the heavy output probability, the cross-entropy difference and the ℓ_1-norm distance. The formal definitions of Heavy Output Generation Benchmarking, used to calculate the heavy output probability, and Cross-Entropy Benchmarking, used to calculate the cross-entropy difference, are given in Section 1.6.4. That of the ℓ_1-norm distance is given in Section 1.3.1. Here we focus on nuances to their use which are specific to our work. In particular in Section 3.1.1 we discuss our motivations for using these figures of merit, while in Section 3.1.2 we detail how we calculate and evaluate them.

3.1.1 Motivations

The circuits we investigate in Section 3.4 are small enough that $\mathcal{D}_C$, the distribution of outcome probabilities from the real device implementing a circuit C, can be well characterised by a reasonable absolute number of samples. This is to say that while approximating $\mathcal{D}(x)$ for every output x requires a number of samples which grows exponentially with the number of qubits, the number of qubits investigated here is sufficiently small that the required computational resources remain reasonable. This allows us to make use of the ℓ_1-norm distance in this chapter.[6] The ℓ_1-norm distance is a particularly powerful metric, and relates to several theoretical results, such as those discussed in Section 1.5.2, motivating its use.

For larger numbers of qubits than are used in the results of Section 3.4, Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking are preferred over calculating the ℓ_1-norm distance. This is because they, unlike calculating the ℓ_1-norm distance, require only a polynomial number of samples from the real device. It is therefore beneficial to explore these figures of merit, both to familiarise ourselves with them, and to explore their relationship with the stronger ℓ_1-norm distance.[7]

In addition Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking provide valuable insights of their own. The connections between Heavy Output Generation Benchmarking and quantum computational supremacy allow us to extract valuable insights into the ability of a quantum computing stack to demonstrate quantum computational supremacy. Further, Heavy Output Generation Benchmarking provides a minimal, intuitively interpreted, single value with which to compare quantum computing stacks. This is of great use in a standard benchmarking suite.

Similarly the comparison to the uniform distribution which the cross-entropy difference provides is valuable as, if an honest attempt is being made to recreate a distribution, at worst the uniform distribution could be produced. Further while it is not known if Cross-Entropy Benchmarking on its own can be used to distinguish error channels, in combination with the techniques introduced here, it can provide insight into this information, as we discuss in Section 3.4. Importantly, since $\mathrm{HOG}\left(\mathcal{D}_C, p_C\right)$ and $\mathrm{CE}\left(\mathcal{D}_C, p_C\right)$ are expectations of different functions of ideal output probabilities, $\delta\left(p_C\right)$ and $-\log\left(p_C\right)$ respectively, over the experimental output distribution, they capture different features of the outputs, and it is advantageous to consider both [66].

[6] In the case that sufficient samples can be obtained to characterise $\mathcal{D}_C$ well, several other figures of merit are accessible, such as the KL-divergence defined in equation (1.27). We limit our investigations to the three discussed to remain thorough but concise, but invite further analysis of the data generated by our experiments [242].

[7] The ℓ_1-norm distance is stronger in the sense that it provides a lower bound for both the heavy output generation probability and the cross-entropy difference. While Cross-Entropy Benchmarking and Heavy Output Generation Benchmarking cannot be used to bound the ℓ_1-norm distance, interestingly our empirical results show a slight negative correlation between ℓ_1-norm distance and the heavy output generation probability found experimentally, as is discussed in Appendix B.4.

3.1.2 Evaluating Stack Performance

Given samples $s = \{x_1, \dots, x_m\}$ from $\mathcal{D}_C$, let s^x be the number of times x appears in s. Define $\widetilde{\mathcal{D}_C}$ by $\widetilde{\mathcal{D}_C}(x) = s^x/m$. The approximation we will use for $\ell_1(\mathcal{D}_C, p_C)$ is $\ell_1\left(\widetilde{\mathcal{D}_C}, p_C\right)$, and so an ideal implementation of a unitary would result in $\ell_1(\mathcal{D}_C, p_C) \approx 0$.[8] As such, in the results of Section 3.4 we will say that a quantum computing stack performs better the closer the value, averaged over a class of circuits, is to 0. However, noise will likely make it incredibly difficult for even fault tolerant quantum computers to achieve a ℓ_1-norm distance of 0. Hence bounds, such as that discussed in Section 1.5.2, are often put on the value instead. For the circuit classes for which such results are relevant, as we discuss in Section 3.2, we will use these results to define a value below which a performance can be considered as good.

In the results of Section 3.4, we say a quantum computing stack has performed well if, on average over the class of circuits, $\text{HOG}(\mathcal{D}_C, p_C)$ is between $(1 + \log 2)/2$ and $2/3$, with $(1 + \log 2)/2$ being best of all. Recall from the discussion around Problem 1 that there are cases where sampling from a family of distributions $\{\mathcal{D}_C\}$ with $\text{HOG}(\mathcal{D}_C, p_c) \geq 2/3$ is thought to be hard to do classically. This is true in particular when the ideal output distributions of the circuits C are distributed exponentially, in which case heavy outputs have cumulative probability $(1 + \log 2)/2$. Such circuits are relevant here, justifying the range of values of $\text{HOG}(\mathcal{D}_C, p_C)$ which we consider to demonstrate a quantum computing stack performing well. A poorer performance is indicated by average values between $2/3$ and $1/2$, with $1/2$ being worst of all. Indeed a value of $1/2$ could be achieved by generating all possible output with equal probability. For the circuit classes in Section 3.2 which have exponentially distributed output probabilities, we will often consider the largest n for which distributions $\{\mathcal{D}_{C_n}\}$ solve Problem 1, which is to say the largest n for which heavy outputs are produced with probability greater than $2/3$. This approach is inspired by quantum volume and is useful as an indicator of the largest Hilbert space accessible to a quantum computing stack [212]. For circuit classes with output probabilities that are not exponentially distributed, we will explicitly calculate the ideal heavy output probability as a point of comparison.

In the case of Cross-Entropy Benchmarking, we will say that better performance is indicated by a value, averaged over the class of circuits, closer to 1, and worse by a value closer to 0. Notice that values above 1, or below 0, are possible for individual circuits if, for example, the ideal output distribution happens to be very skewed towards heavy outputs, or if unlikely outcomes occur very often, respectively. Unlike in the case of Heavy Output Generation Benchmarking, we do not give a value which a score above would indicate good performance.[9] Further, to avoid requiring the inverse of 0 in the approximation of equation (1.30), we chose to use

$$\max\left\{p_C(x), 2^{-n^2}\right\}.$$

[8]Note that unlike in Chapter 2, where we could access output probabilities, here we interact with real devices, and so only samples are available.

[9]Where such a value has been defined, it is perhaps surprisingly close to 0, and so the uniform distribution [60].

in place of $p_C(x)$. This choice of an inverse exponential in the number of qubits is inspired by, although not directly derived from, the average case supremacy results related to random circuits [66, 186].

3.2 Circuit Classes

This section presents the formal definitions of the circuits used in this work, while also identifying the motivations for their use in benchmarking. These motivations include both the class of applications they represent and the properties of the quantum computing stacks that they probe. Collectively, this selection of circuit classes encompasses an array of potential applications of quantum computing, covering circuits of varied depth, connectivity, and gate types.

3.2.1 Shallow Circuits: IQP

As discussed in Section 1.5, IQP circuits [113] consist of commuting gates and, as well as being simpler to implement than universal quantum circuits, are believed, even in the presence of noise, to be impossible to simulate efficiently using classical computers [63–65]. This has allowed for the fruitful application of noisy quantum technology in areas such as machine learning [4, 164, 165], as discussed in Chapter 5, and interactive two-player games, as discussed in Chapter 4. These applications, and the connection between IQP and a demonstration of quantum computational supremacy on near-term hardware, makes their implementation a pertinent benchmark of the performance of these devices.

In Protocol 3.2.1 we introduce the *shallow circuits* class for benchmarking, which is a subclass of IQP circuits. Indeed the close connection, through Theorem 1.5.2, of quantum computational supremacy and shallow circuits[10] provides a measure of a quantum computing stack's quality. Namely, this is by analysing the closeness of the distributions it produces to the ideal ones, as measured by the ℓ_1-norm distance, and comparing this value to $1/192$. The shallow circuits class is exemplified in Figure 3.1.

As discussed below, there is a constant bound on the depth of uncompiled shallow circuits, meaning that their compiled depth increases slowly with width. As such these circuits probe the performance of a quantum computing stack in fine-grained detail by measuring the impact of including more qubits (quasi-) independently of increasing circuit depth. This is useful for understanding the performance of a device being utilised for applications whose qubit requirement grows more quickly than the circuit depth.

The uncompiled depth of shallow circuits may be arrived at by observing that finding a valid order in which to apply the CZ gates in the circuit is equivalent to finding an edge colouring of the graph G_n. By Vizing's theorem, it is known that a colouring of

[10]Theorem 1.5.2 is a worst case hardness result, and may not apply to shallow circuits. However we regard the performance of shallow circuits as indicative of the performance of those circuits to which Theorem 1.5.2 does apply. Indeed, similar hardness results to Theorem 1.5.2 exist for other families of sparse, constant depth IQP circuits [24].

Protocol 3.2.1 The pattern for building shallow circuits. An example output can be seen in Figure 3.1.

Input: Number of qubits, $n \in \mathbb{Z}$
Worst case depth: 7
Output: Circuit, C_n

```
1: Initialise $n$ qubits, labelled $q_1,...,q_n$, in the state $|0\rangle$.
2:
3: for all $i \in \{1,...,n\}$ do
4:     Enact H on $q_i$
5: end for
6:
7: Let $G_n$ be the graph indicating between which qubits a CZ gate can act. Let $G_n$
   in this case be a random binomial graph, $G_n$, with $n$ vertices and edge probability
   0.5, post selecting on those that are connected and have degree less than 4.
8:
9: for all edges $\{i, j\}$ in $G_n$ do
10:     Enact CZ between $q_i$ and $q_j$
11: end for
12:
13: for all $i \in \{1,...,n\}$ do
14:     Generate $\alpha_i \in [0, 2\pi]$ uniformly at random.
15:     Enact RZ$^{\alpha_i}$ on $q_i$ .
16: end for
17:
18: for all $i \in \{1,...,n\}$ do
19:     Enact H on $q_i$
20: end for
21:
22: Measure $q_1,...,q_n$ in the computational basis
```

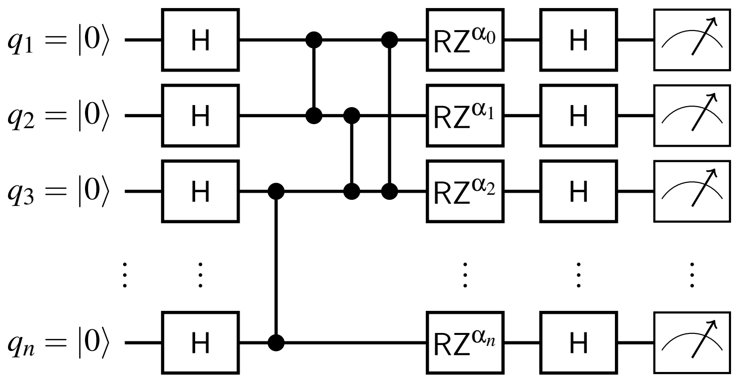


Figure 3.1: **An example of shallow circuits, as generated by Protocol 3.2.1.**

undirected graphs using at most one more colour than the maximum vertex degree exists. Constructive proofs of Vizing's theorem therefore demonstrate that a 4-colouring can be found in polynomial time [243].[11]

Further, because Protocol 3.2.1 limits the connectivity allowed between the qubits, the increase in circuit depth due to compilation onto limited-connectivity architectures is minimised, while avoiding a choice of connectivity favouring one device in particular. By bounding connectivity, but allowing all connections in principle, we avoid biasing against architectures that allow all-to-all connectivity, which would still perform well.

Shallow circuits may be compared to other sparse IQP circuits [64], IQP circuits on 2D latices such as the 2D-DQS circuits of Protocol 1.5.1, and random 3-regular graphs used for benchmarking [95]. In our case we aim to avoid favouring particular architectures, and so avoid 2D lattices. In addition we avoid highly structured 3-regular graphs, in favour of allowing reduced vertex connectivity.[12] This avoids preferring very particular applications of low depth IQP circuits; instead exploring a variety of applications simultaneously. In particular, in Section 1.6.3 it was outlined that a post-hoc verification scheme can be used to test the ℓ_1-norm distance of the ideal from the real output distributions of 2D-DQS circuits. However the connectivity is too architecture-specific for our purposes, with the verification scheme requiring limits to the measurement noise which we cannot guarantee.

In the case of shallow circuits, the output probabilities are not exponentially distributed. As expanded on in Section 1.6.4, this property allows us to simplify calculations required when performing both Cross-Entropy Benchmarking and Heavy Output Generation Benchmarking. In particular the theoretical value of heavy output probability for circuits with exponentially distributed output probabilities cannot be used here. Instead, we use the empirical value of the ideal heavy output probability, in the place of a theoretically derived one, as a point of comparison with the behaviour of the quantum computing stack being benchmarked. This approach requires the calculation of all output probabilities and the summation of the probabilities of those that are heavy. This can be done for the small circuits investigated here, but does not allow for the benchmarking of as many qubits as would be accessible if a theoretical value was known.

In summary, inspired by near-term applications of IQP circuits in machine learning [4, 165], we introduce shallow circuits in Protocol 3.2.1. Performance when implementing these circuits is indicative of the performance when implementing those applications,

[11]It is possible that the chromatic index of a particular graph is lower than 4, either because: the maximum degree in less than 3, as could be the case with these random graphs; or because a 3-colouring exists, as is the lower bound on the chromatic index of an undirected graph of maximum vertex degree 3. However a 4-colouring certainly provide an upper bound and so it suffices for our discussion as we are concerned with upper bounding the depth of the circuit.

[12]As n grows the probability that the maximum degree is at most 3 will tend to zero, and so Protocol 3.2.1 may become infeasible to implement. While for the number of qubits considered in this paper Protocol 3.2.1 works well, for larger n it may become necessary to alter the graph generation step, possibly to the use of 3-regular graphs, for which the fast random generation algorithms have been extensively researched and developed. However, regular graphs are too few in number for graphs with low numbers of qubits to provide a good benchmark set of many possible circuits, and so we avoid them here.

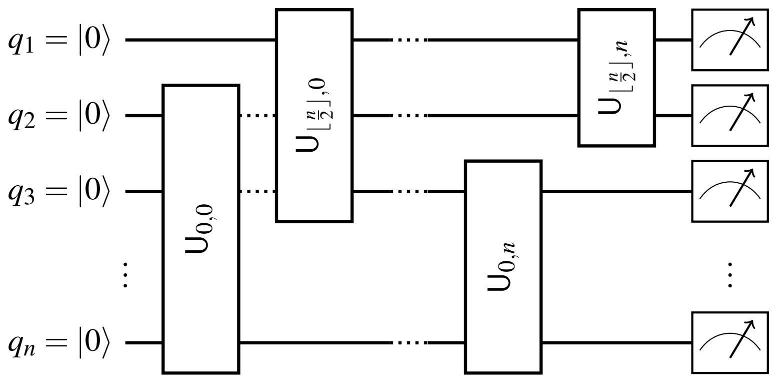


Figure 3.2: **An example of square circuits, as generated by Protocol 3.2.2.** Note that the $U_{i,t}$ gates act between 2 randomly selected qubits.

but also, more generally, of applications requiring circuits which grow slowly in depth. As a result, these circuits also probe the performance of a quantum computing stack in fine-grained detail by measuring the impact of including more qubits (quasi-) independently of increasing circuit depth.

3.2.2 Square Circuits: Random Circuit Sampling

Square circuits are generated according to Protocol 3.2.2 and consist of n *layers* of random two-qubit gates acting between a bipartition of the qubits. The square circuits class is exemplified in Figure 3.2. While circuits required for applications are typically not random, by utilising uniformly random two-qubit unitaries, square circuits provides a benchmark at all layers of the quantum computing stack. In particular it tests the ability of the device to implement a universal gate set, the diversity and quality of the gates available, and the compilation strategy's ability to decompose these gates to the native architecture. Further, as quantum circuits can always be approximated up to arbitrary precision using two-qubit unitary gates [28], square circuits can help us understand the performance of quantum computing stacks when implementing computations requiring a universal gate set. In addition to the advantages of using random two-qubit unitary gates listed above, the particular distribution from which they are samples, as specified on line 8 of Protocol 3.2.2, proves sufficient to allow us to apply HOG, as defined in Problem 1. In particular , the distribution p_C is sufficiently far from uniform in the required sense, which we demonstrate in Appendix B.1.1. Sampling 2-qubit unitaries according to the Haar measure has the additional advantage that the resulting circuits are similar in structure to those used for RCS, as introduced in Section 1.5.3, providing insights into the performance of a quantum computing stack when addressing that application.

As mentioned in Section 1.5.3, sampling from the output distributions of random circuits on 2D latices using a classical computer is thought to be hard. While this architecture is relevant for devices built using superconducting technology [60], we wish to avoid biasing in favour of this technology in particular. By allowing two-qubit gates to act between any pair of qubits in the uncompiled circuit, square circuits avoid favouring any device in particular [60, 62, 85]. In addition, assuming all-to-all connectivity

Protocol 3.2.2 The pattern for building square circuits. An example output can be seen in Figure 3.2.

Input: Number of qubits, $n \in \mathbb{Z}$
Worst case depth: n
Output: Circuit, C_n

```
1: Initialise n qubits, labelled $q_1, ..., q_n$, in the state $|0\rangle$
2:
3: for each layer t up to depth n do
4:     ▷ The contents of this for loop constitutes a layer. The choice of the number of layers used here is discussed in Appendix B.1.1.
5:
6:     Divide the qubits into $\lfloor \frac{n}{2} \rfloor$ pairs $\{q_{i,1}, q_{i,2}\}$ at random.
7:     for all $i \in \mathbb{Z}$, $0 \leq i \leq \lfloor \frac{n}{2} \rfloor$ do
8:         Generate $\mathsf{U}_{i,t} \in \mathrm{SU}(4)$ uniformly at random according to the Haar measure.
9:         Enact $\mathsf{U}_{i,t}$ on qubits $q_{i,1}$ and $q_{i,2}$.
10:    end for
11: end for
12:
13: Measure all qubits in the computational basis.
```

passes the burden of mapping the circuit onto the device to the compilation strategy, which is in line with our wish to benchmark the full quantum computing stack. That said, any architecture whose coupling map closely mirrors the uncompiled circuit will be advantaged, as even a naïve compilation strategy will perform well in that case.

In [212] almost identical circuits to those of Protocol 3.2.2 are used to perform quantum volume benchmarking, but with all-to-all connectivity supplemented for nearest neighbour connectivity on a line, and the addition of permutation layers. As this disadvantages devices with a completely connected coupling map,[13] a property which would typically be an advantage, we choose not to make this restriction here. Notice, however, that a naïve compilation of square circuits onto an architecture with nearest neighbour connectivity on a line would create a permutation layer to move qubits together where necessary, recreating the quantum volume circuits. Indeed, compiling square circuits to superconducting devices (where connectivity is low) will generally result in a circuit similar to those used in the quantum volume benchmark, as many SWAP operations are required regardless.

In summary, by allowing for the most general gate set and connectivity, square circuits provide a means to thoroughly interrogate all layers of the quantum computing stack. This class of circuits is inspired by quantum volume circuits [212] but are somewhat generalised to avoid device bias. Because of this generality, and the linear increase in

[13]Note that some compilation strategies may identify that the SWAP gates in the permutation layer may be removed for devices with all-to-all connectivity. We avoid this dependence on the compilation strategy by fixing the connectivity in the uncompiled circuit to be all-to-all.

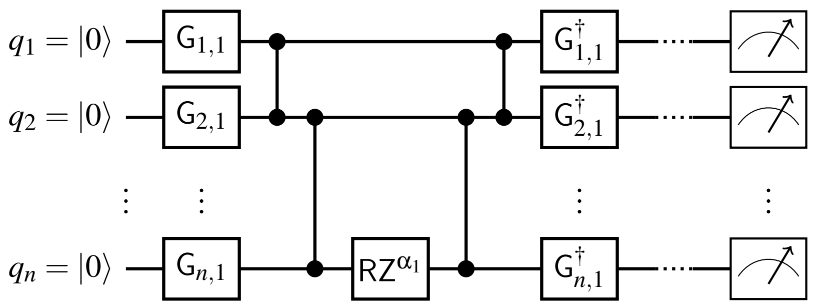


Figure 3.3: **An example of deep circuits, as generated by Protocol 3.2.3.** This gives the structure of one Pauli gadget, with this structure repeated throughout the circuit. Here G represents a single qubit gate, as described on line 11 of Protocol 3.2.3.

depth with number of qubits, square circuits are complementary to the other classes introduced in this work.

3.2.3 Deep Circuits: Pauli Gadgets

Pauli gadgets [244] are quantum circuits implementing an operation corresponding to exponentiating a Pauli tensor. Sequences of Pauli gadgets acting on qubits form *product formula* circuits, commonly used in Hamiltonian simulation [238]. These circuits are the basis of trial state preparation in many variational algorithms, which are amongst the most promising applications of noisy quantum computers. A notable example of this in quantum chemistry is the physically-motivated Unitary Coupled Cluster family of trial states used in the VQE [239, 240]. As near-term quantum computers hold promise as useful tools for studying quantum chemistry, we propose that the quality of an implementation of these gadgets is a useful benchmark, and use them to define the *deep* circuit class.

Deep circuits are built as in Protocol 3.2.3 and are constructed from several layers of Pauli Gadgets, each acting on a random subset of n qubits. The deep circuits class is exemplified in Figure 3.3. In the worst case each Pauli Gadget will demand $4n+1$ gates: $2n$ Pauli gates, $2(n-1)$ CX gates, and one RZ^α gate. The number of layers is chosen to ensure an exponential distribution of the output probabilities from deep circuits, as we establish in Appendix B.1.2. This allows ourselves the capacity to use Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking. Doing so constitutes a novel extension of those approaches to application motivated benchmarking, and the unique ability for us to benchmark application-motivated circuits, using polynomially many samples from a device.

The philosophy justifying the introduction of deep circuits is similar to that used in [84] when the *random phase gadgets* circuit class is introduced. Random phase gadget circuits are similar to quantum volume circuits, but with the Haar random two-qubit unitary gates replaced by random two-qubit phase gadgets. While these circuits are also motivated by applications in variational algorithms, they have the same drawback as quantum volume circuits, as discussed in Section 3.2.2, and are further in form from the application of concern than deep circuits. Note also that deep circuits differ from running the VQE end-to-end. By focusing on the state preparation portion of a VQE

Protocol 3.2.3 The pattern for building deep circuits. An example output is given in Figure 3.3.

Input: Number of qubits, $n \in \mathbb{Z}$
Worst case depth: $(4n-1)(3n+1)$
Output: Circuit, C

```
function PhaseGadget($\alpha$, $\{\tilde{q}_1, ..., \tilde{q}_p\}$)
    if $p = 1$ then
        Enact RZ $(\alpha)$ on $\tilde{q}_1$
    else
        Enact CX between $\tilde{q}_1$ and $\tilde{q}_2$
        PhaseGadget($\alpha$, $\{\tilde{q}_2, ..., \tilde{q}_p\}$)
        Enact CX between $\tilde{q}_1$ and $\tilde{q}_2$
    end if
end function

function Pauli($\{\tilde{q}_1, ..., \tilde{q}_p\}$, $s$)
    if $s_1 = \mathsf{X}$ then
        Enact RX $\left(\frac{\pi}{2}\right)$ on $\tilde{q}_1$
    else if $s_1 = \mathsf{Y}$ then
        Enact H on the $\tilde{q}_p$
    end if
    Pauli($\{\tilde{q}_2, ..., \tilde{q}_p\}$, $s$)
end function

function PauliGadget($\alpha$, qubits, $s$)
    Pauli(qubits, $s$)
    PhaseGadget($\alpha$, qubits)
    Enact the inverse of Pauli(qubits, $s$)
end function
```

Protocol continues below...

Protocol 3.2.3 Continued

```
25: Initialise $n$ qubits, labelled $q_1,...,q_n$, in the state $|0\rangle$.
26:
27: for each layer t up to depth $3n+1$ do
28:         ▷ The contents of this for loop constitutes a layer. The choice of the number
        of layers used here is discussed in Appendix B.1.2.
29:
30:     Select a random string $s \in \{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^n$
31:     Generate random angle $\alpha \in [0,2\pi]$
32:     PauliGadget($\alpha$, $\{q_i : s_i \neq \mathsf{I}\}$, $s$)
33:
34: end for
35:
36: Measure all qubits in the computational basis
```

circuit, we might deduce performance of the quantum computing stack when running the VQE on a number of molecules[14]. The intuition here is that if the state preparation sub-component is accurate, then the error in the expectation values of measured observables will be due to errors in implementing those observables, or the readout process itself.

In summary, we have introduced deep circuits to probe the performance of a quantum computing stack when implementing common quantum chemistry applications. The deep circuits circuit class is complimentary to the other classes featured in this work by having a depth which grows quickly with the number of qubits.

3.3 Quantum Computing Stack

Each component of a quantum computing stack exerts an influence on overall performance, and identifying the distinct impact of a particular component is often hard. To disentangle these factors, we must clearly identify the components used during benchmarking. Here we detail the components used to build the quantum computing stacks explored in Section 3.4. This diverse selection of components allows us to investigate a variety of ways of building a quantum computing stack.

3.3.1 Software Development Kits

We use a combination of tools available via pytket [235, 245] and Qiskit [114, 246]. pytket is a Python module which provides an environment for constructing and implementing quantum circuits, as well as for interfacing with CQC's t|ket⟩, a retargetable compiler for near-term quantum devices featuring hardware-agnostic optimisation. Qiskit is a open-source quantum computing software development framework

[14]Here we do not explore the relationship between the performance of a quantum computing stack when implementing deep circuits and when implementing VQE but regard it as important for future work.

for programming, simulating, and interacting with quantum processors, which also provides a compiler. Details of the versions of the software used are seen in Table B.1 of Appendix B.2.

We use three parts of Qiskit in this work. First is the *transpiler architecture*, which enables users to define a custom compilation strategy by executing a series of *passes* on the input circuit, as discussed in Section 3.3.2. The second part of Qiskit we use is the library of predefined passes. Finally, a *provider* is used to access hardware made available over the cloud by IBM Quantum. The provider enables users to send circuits to hardware, retrieve results, and query the hardware for its properties.[15]

Similarly, we use pytket to generate and manipulate circuits in several ways. Firstly we use the t|ket⟩ compiler to construct compilation strategies which optimise the input circuit for the target hardware, utilising predefined passes available in t|ket⟩. Secondly we use pytket to define abstract circuits and to convert t|ket⟩'s native representation of the circuit into a Qiskit `QuantumCircuit` object which is then dispatched to IBM Quantum's systems for execution.

3.3.2 Compilers

Compilers provide tools to construct executable quantum circuits from abstract circuit models. This is done by defining *passes* which may manipulate a representation of a quantum circuit, often by taking account of limited connectivity architectures, or minimising quantities such as gate depth, but need not perform any manipulation.[16] These passes are composed to form *compilation strategies* which should output executable quantum circuits. Quantum compiling is an active area of research [141, 247–250], and there are many pieces of software available for quantum compiling [114, 115, 234, 235]. For the purposes of this work, the problem of quantum compilation is divided into three tasks.

Placement: Determine onto which *physical* qubits of a given device the *virtual* qubits in the circuit's representation should be initially mapped. This can influence the performance of routing [251] and the impact of noise through noise aware placement [250].

Routing: Modify a circuit to conform to the qubit layout of a specific architecture, for example, by inserting SWAP gates to allow non-adjacent qubits to interact [173]. Circuits are often designed without the qubit layout in mind, so this step is important [95].

Optimisation: Work to minimise some property of a circuit. This may be gate count or depth, which are targeted as proxies for noise.

Both pytket and Qiskit have multiple placement, optimisation, and routing passes. We compare the performance of 5 compilation strategies built from these passes. Two

[15]These properties include the graph connectivity, single- and two-qubit error rates, and qubit T_1 and T_2 times. Some of the *noise-aware* compilation strategies we use require knowledge of these properties.

[16]An example of this is a pass which counts the gates in the circuit.

of them, noise-unaware pytket and noise-unaware Qiskit, compile the circuit *without* knowledge of the device's noise properties. Another two, noise-aware pytket and noise-aware Qiskit, do take noise properties into account. As a baseline, we consider a simple compilation strategy from pytket using only routing, without optimisation or noise-awareness; we refer to this pass as only pytket routing. We detail these schemes in Appendix B.2. The main difference between the noise-aware schemes is that, by the design of pytket and Qiskit, noise-aware pytket prioritises the minimisation of gate errors during placement,[17] whereas noise-aware Qiskit prioritises readout and CX errors [250].

3.3.3 Devices

We benchmark some of the devices made available over the cloud by IBM Quantum. For reasons of accessibility, we do not utilise the Q20:20 device discussed in Chapter 2. The devices we use are referred to by the unique names ibmqx2, ibmq_16_melbourne, ibmq_singapore and ibmq_ourense. Each device has a set of *native gates* which all gates in a given circuit must be decomposed to. For all the devices considered here, the native gates are: an identity operation, I; 3 "*u*-gates", as defined in equation (3.1); and a controlled-NOT (CX) gate.

$$\begin{aligned}
\mathsf{U}_3(\theta,\phi,\lambda) &= \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} \\
\mathsf{U}_2(\phi,\lambda) &= \mathsf{U}_3\left(\frac{\pi}{2},\phi,\lambda\right) \\
\mathsf{U}_1(\lambda) &= \mathsf{U}_3(0,0,\lambda)
\end{aligned} \tag{3.1}$$

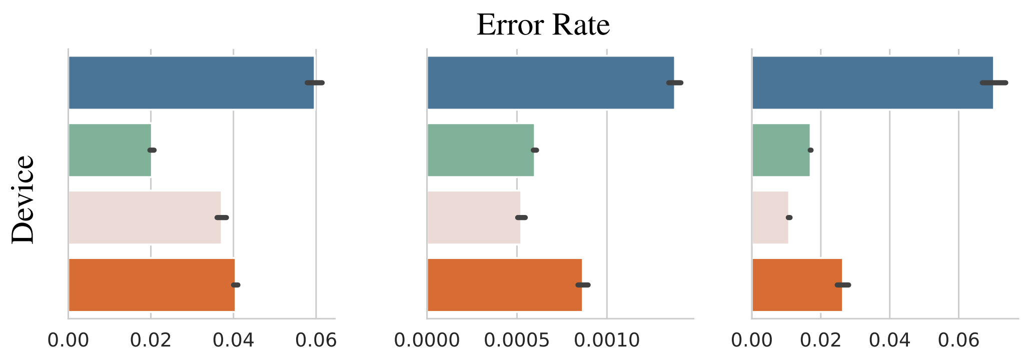
Two of the device properties used by the noise-aware compilation strategies are their *connectivity* and *calibration data*. Information about the connectivity of a device is contained in its *coupling map* which, in the cases of the devices studied here, are shown in Appendix B.3.1 and summarised in Table 3.1. A coupling map is a graph with a vertex for each qubit, and edges between vertices when there exists coupling between the corresponding qubits.

We expect the coupling map of a device will influence the device's performance according to our benchmarks. Coupling maps with a high average *degree* (mean number of edges incident on each vertex) have more qubits directly connected to one another, reducing the requirement for SWAP operations that would increase the circuit depth. Coupling maps with low *radius* (minimax distance over all pairs of qubits) have qubits which are closer to one another, again reducing the need for SWAP gates. Coupling maps with a high number of *vertices* (i.e., qubits), enables higher-width circuits to be run. Finally, coupling maps where the number of vertices equals the *minimum cycle length* (smallest number of edges per cycle over all cycles) have the advantage that any routing operation has more paths available to it, potentially allowing for some parallelisation.

[17]This is true of pytket 0.3.0; as a result of this work, later versions of pytket take into account readout error.

Device	Vertices	Average Degree	Radius	Minimum Cycle Length
ibmqx2	5	2.4	1	3
ibmq_16_melbourne	15	$2\frac{2}{3}$	4	4
ibmq_ourense	5	1.6	2	N/A
ibmq_singapore	20	2.3	4	6

Table 3.1: **Selected graph properties of the coupling maps of devices studied in this work**. See Appendix B.3.1 for full details of the coupling maps of the devices explored here.



(a) Readout Error. (b) Error per U_2 gate. (c) Error per CX gate.

Figure 3.4: **Average error rates across devices used in this work**. Bars show the mean error rates across the whole device, while error bars give the standard deviation. Devices shown here are: ibmqx2 [🟩], ibmq_ourense [⬜], ibmq_singapore [🟧], ibmq_16_melbourne [🟦]. Further details can be found in Appendix B.3.2

Device calibration data includes information about single- and two-qubit error rates, readout error, and qubit frequency, T_1, and T_2 times. The noise-aware compilation strategies we investigate use the gate error rates and readout error. Full details of noise levels can be found in Appendix B.3.2 with average values given in Figure 3.4. This information is updated twice daily, with the data in Figure 3.4 averaged over the period 2020-01-29 to 2020-02-10 during which time our experiments were conducted. The results of Section 3.4 depend heavily on the noise levels of the device at the time at which the computation is implemented. This is doubly true in the case of the noise-aware optimisation schemes as a circuit optimised at one time may not perform as well over time as the noise levels of the devices change. To reduce this effect we endeavoured to compile and run circuits within as short a time interval as possible.

3.4 Experimental Results

This section presents experimental results of running the circuit families defined in Section 3.2 on various quantum computing systems. We present a sub-sample of all our results in the figures which follow. We study these results in 3 contexts:

Section 3.4.1: Full Stack Benchmarking – Incorporating and thoroughly investigating the compilation strategy helps develop an understanding of how circuit compilation influences the performance of the quantum computing stack. For noise-aware compilation strategies, our results show that the assumptions made by the strategy about the importance of different kinds of noise impacts performance.

Section 3.4.2: Application Motivated Benchmarks – By including three quite different circuit classes in our benchmark suite, we explore how a quantum computing stack may perform when implementing a wide array of applications.

Section 3.4.3: Insights from Classical Simulation – We explore how benchmarks assist in developing new noise models. By identifying when benchmark values for real implementations and those we expect from simulations using noise models differ, noise channels which should be added to the noise models to achieve greater agreement with real devices can be identified. This is of particular importance as noise-aware compilation strategies often utilise noise properties.

For each circuit class and fixed number of qubits, 200 circuits were generated according to the algorithms of Section 3.2. Each circuit is compiled by a given compilation strategy onto a particular device. The compiled circuits were then run on the device, using 8192 repetitions (samples) from each compiled circuit, which generates 8192 bitstrings. The compiled circuits are also classically simulated using a noise model built from the device calibration information at the time of the device run. See [242] for access to the full experimental data set.

The resulting bitstrings are then processed according to the figures of merit given in Section 3.1. The distribution of the figures of merit are compared by their mean and distribution via a box-and-whisker plot. In particular boxes show quartiles of the dataset, whiskers extend to 1.5 times the interquartile range past the low and high quartiles, and white circles give the mean. Uncompiled circuits were also perfectly simulated without noise in order to calculate the ideal heavy output probability. These points are referred to as *Noise-Free* in the figures below. Note that, as discussed in Section 1.6.4, we expect the mean of the ideal, Noise-Free, heavy output probability to converge to $(1+\log 2)/2 \approx 0.846574$ as the number of qubits grows. However, as we can see in the following results, there is some fluctuation around this value for smaller circuits.

3.4.1 Full Stack Benchmarking

Impact of the Compilation Strategy

We study the compilation strategy and the device on which the compiled circuit is run. Using a fixed device and comparing multiple strategies allows us to determine which strategy tends to perform well. Further, aggregating performance over all compilation strategies provides an estimate of the performance of a "generic" strategy. Similarly, fixing the strategy, and comparing its performance on multiple devices, shows how (possibly erroneous) assumptions made by the strategy about the devices impact performance.

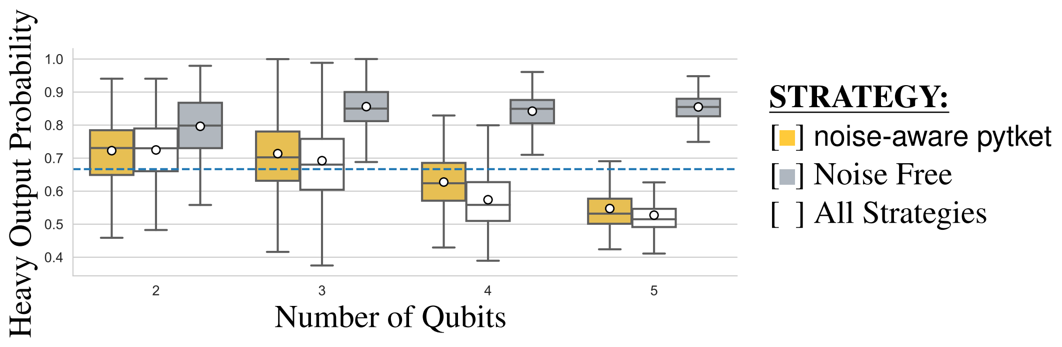


Figure 3.5: **Comparison of fixed compilation strategy to average of all strategies using, the heavy output probability metric.** Here we have run square circuits using the real ibmq_16_melbourne device. Values above $2/3$ (dotted blue line), and closer to Noise Free, indicate better performance.

Figure 3.5 displays experimental results when implementing square circuits on ibmq_16_melbourne, using heavy output probability as the figure of merit. The noise-aware pytket compilation strategy performs somewhat better, on average, than a generic strategy. Because the aggregated information ("All Strategies" in Figure 3.5) includes aggregation over noise-aware pytket, these results indicate that other compilation strategies perform worse, since the performance of the aggregate is generally lower than that of noise-aware pytket.[18] This reveals the potential for compilation strategy driven improvements in performance.

Aggregation over compilation strategies provides a way of identifying devices which perform well, by "washing out" the effect of the compilation strategy on performance. Figure 3.6 shows that by considering performance with a fixed compilation strategy ibmq_singapore would be considered to perform similarly, if not slightly better than ibmq_ourense, as measured by ℓ_1-norm distance. However, aggregating over all strategies (Figure 3.7) shows ibmq_ourense to perform better, suggesting that ibmq_ourense might be a better device for a 'generic' compilation strategy to compile to.

An instance-by-instance comparison of different compilation strategies also reveals their limitations and advantages. For example, Figure 3.8 shows noise-aware pytket works best at reproducing the ideal distribution of heavy output probabilities of square circuits on ibmq_16_melbourne. When compared to the strong performance of only pytket routing, this suggests these results are due in part to the routing scheme.

Similarly, Figure 3.9 shows that noise-aware pytket is amongst the worst-performing compilation strategies for lower numbers of qubits, while it is amongst the best-performing for higher numbers. This could be a result of the way in which noise-aware pytket prioritises noise in its routing scheme, with gate errors taking precedence.[19]

[18]Note that due to the fact we aggregate over all 5 compilation strategies, the distribution of heavy output probabilities amongst the "All Strategies" category contains five times as many points as compared to those for noise-aware pytket.

[19]For larger numbers of qubits and deeper circuits, gate errors become more impactful on the total noise, and materialise as giving noise-aware pytket an advantage for larger numbers of qubits.

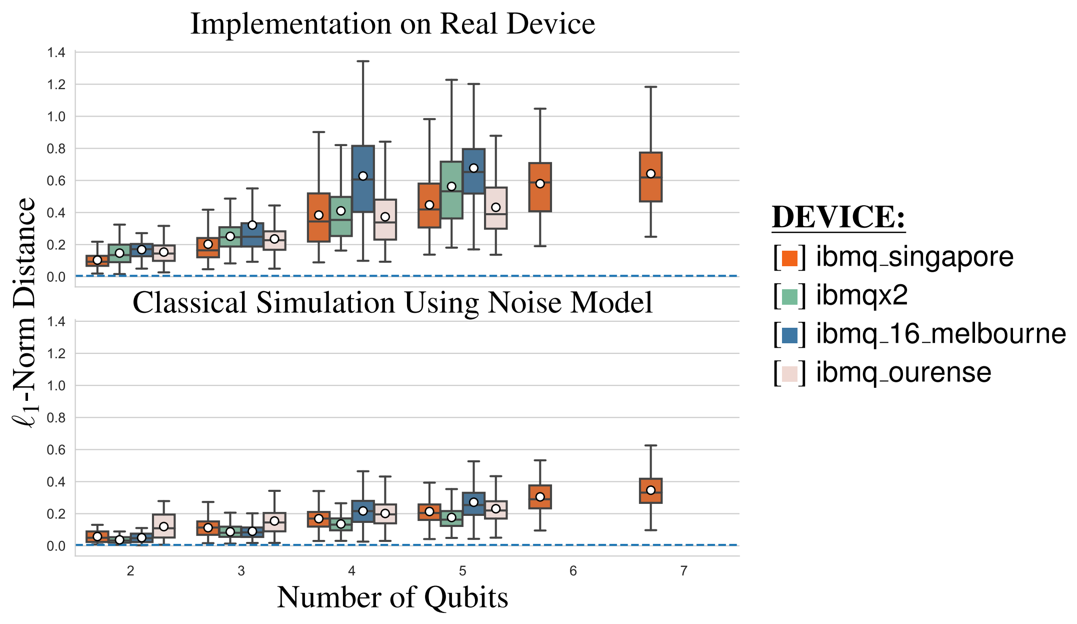


Figure 3.6: **Comparison of devices using the ℓ_1-norm distance metric, when running shallow circuits compiled using noise-aware pytket**. Values close to 0 indicate better performance. Values below $1/192$ (dotted blue line) can be regarded as performing very well. Both simulations using Qiskit noise models, and implementations on real devices, are included.

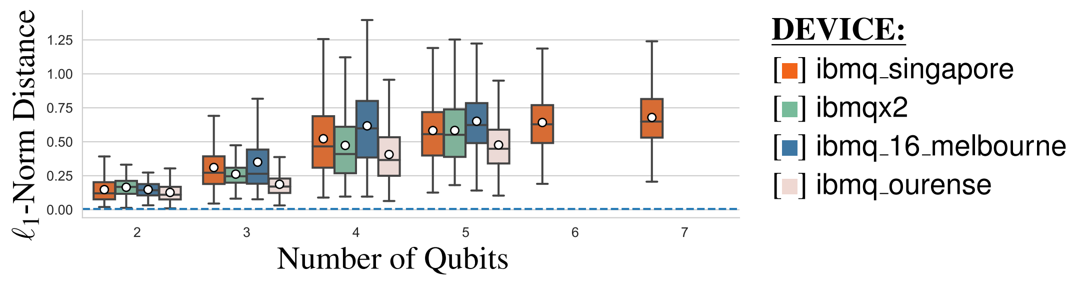


Figure 3.7: **Comparison of real devices, using the ℓ_1-norm distance metric, when running shallow circuits compiled using all compilation strategies**. Here we compile onto each device using all compilation strategies, including all compiled circuits in this plot. Values close to 0 indicate better performance. Values below $1/192$ (dotted blue line) can be regarded as performing very well.

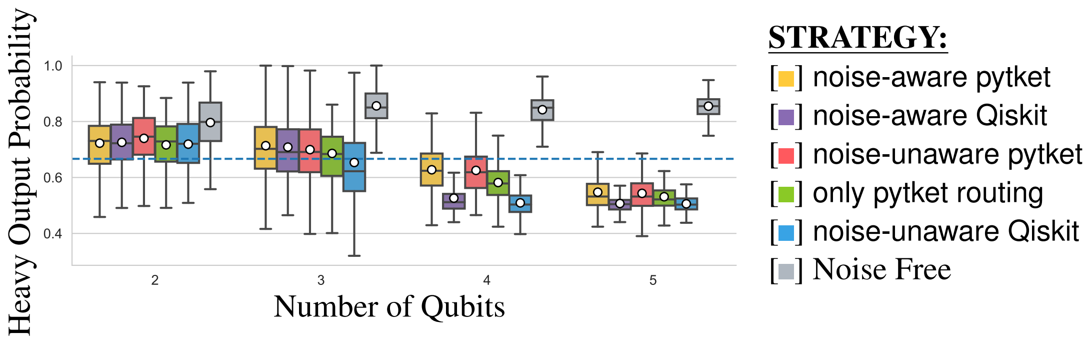


Figure 3.8: **Comparison of compilation strategies, using the heavy output probability metric, when running square circuits on the real ibmq_16_melbourne device**. Values above $2/3$ (dotted blue line), and closer to Noise Free, indicate better performance.

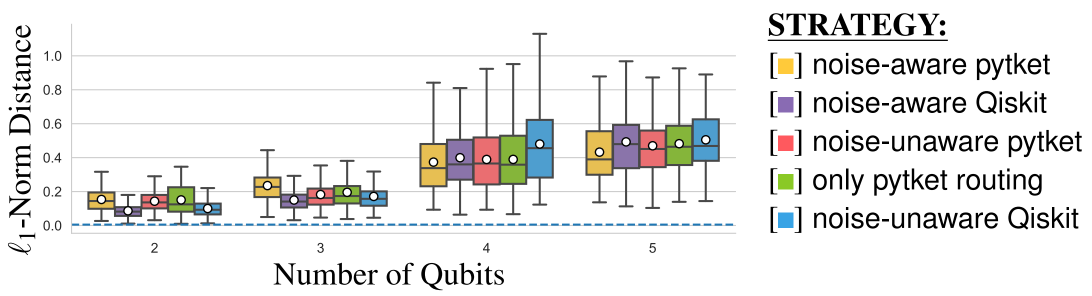


Figure 3.9: **Comparison of compilation strategies, using the ℓ_1-norm distance metric, when running shallow circuits on the real ibmq_ourense device**. Values close to 0 indicate better performance. Values below $1/192$ (dotted blue line) can be regarded as performing very well.

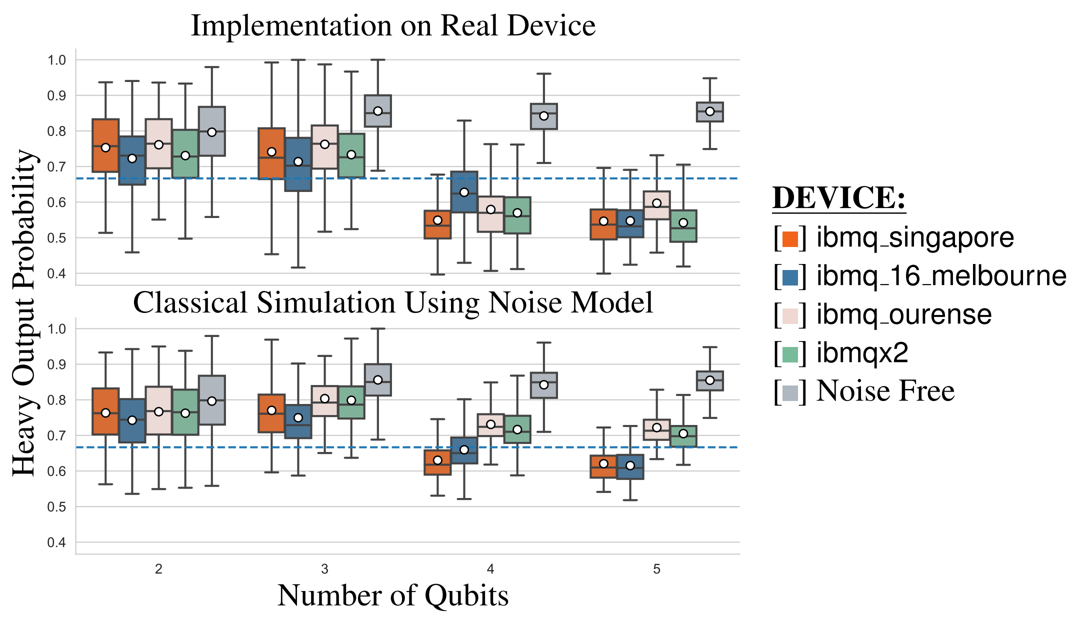


Figure 3.10: **Comparison of devices using the heavy output probability metric, when running square circuits compiled using noise-aware pytket**. Values above $2/3$ (dotted blue line), and closer to Noise Free, indicate better performance. Both simulations using Qiskit noise models, and implementations on real devices, are included.

Noise Level, Connectivity Trade Off

More highly-connected architectures typically allow for shallower implementations of a given circuit as compared to less-connected ones. However, the noise levels may be higher due to crosstalk [252], resulting in a trade-off between connectivity and the total amount of noise incurred when running a computation. Noise affects the accuracy of the computation, so this trade-off has practical implications for the performance of a device. Reducing the connectivity between superconducting qubits has been used to reduce noise levels [252]. In superconducting qubits, this can also be counteracted using tunable couplers [60], but this is not utilised in the devices studied here.[20]

Figure 3.10 shows that devices with lower noise levels (ibmq_singapore and ibmq_ourense) typically outperform devices with higher noise levels (ibmqx2 and ibmq_16_melbourne) despite the latter's higher connectivity. An interesting exception to this is for 4 qubits, where ibmq_16_melbourne performs best, likely because of the 4-qubit cycles in its connectivity graph, as mentioned in Table 3.1. This reduces the SWAP operations necessary for implementing the circuit, reducing the overall circuit depth. This reveals the increase in performance that can be expected when the connectivity of the device and the problem instance are similar [95]. Similar results are revealed by Cross-Entropy Benchmarking, as shown in Figure 3.11.

In general, we expect that circuits whose structure can naturally be mapped to the

[20]While we focus on the connectivity of superconducting architectures here, more generally the comparison between the limited connectivity of superconducting devices, and the completely connected coupling maps of ion trap devices is of interest [89–91].

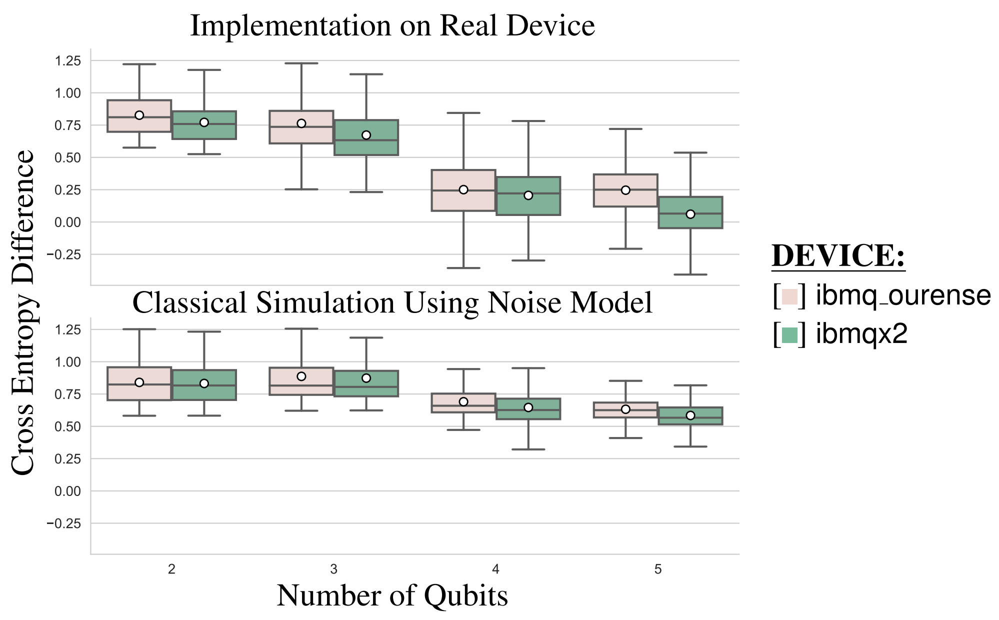


Figure 3.11: **Comparison of devices using the cross entropy difference metric, when running square circuits compiled using noise-aware pytket**. Values close to 1 indicate better performance. Both simulations using Qiskit noise models, and implementations on real devices, are included.

connectivity of the device will generally perform well, whereas those which cannot, will not. In general though, lower-noise devices will tend to perform best.

3.4.2 Application Motivated Benchmarks

The same quantum computing stack will perform differently when running different applications, as the structure of the circuits they require will generally be different. Differences in performance are seen in the context of our application-motivated benchmarks. For example, consider Figure 3.12, which shows performance when implementing sparsely connected circuits, and Figure 3.13, which shows performance when implementing chemistry-motivated circuits. In the case of Figure 3.12, the ibmqx2 device outperforms ibmq_singapore, while in the case of Figure 3.13 the reverse is true.

Quantum Chemistry

Figure 3.13 suggests ibmq_ourense is best for quantum chemistry applications, because it performs well when running deep circuits.[21] In particular Figure 3.13 indicates that the average circuit fidelity is highest for implementations on ibmq_ourense.

In Figure 3.13, all devices converge to the minimum value of cross-entropy difference at 4 qubits. To extend an investigation of this sort to more qubits would require lower

[21]This comes with the caveat, as mentioned in Section 3.2.3, that the connection between the quality of an implementation of these computational primitives, as measured by this benchmark, and accurate ground state energy calculations in VQE has not been demonstrated experimentally.

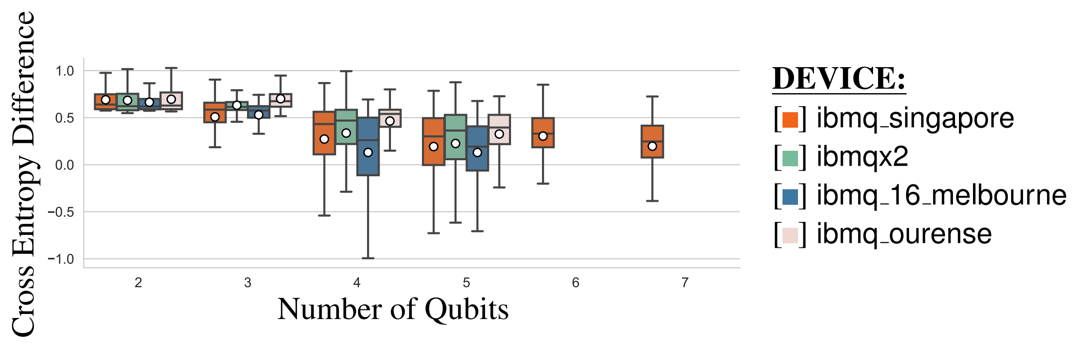


Figure 3.12: **Comparison of real devices, using the cross entropy difference metric, when running shallow circuits compiled using noise-aware Qiskit**. Values close to 1 indicate better performance.

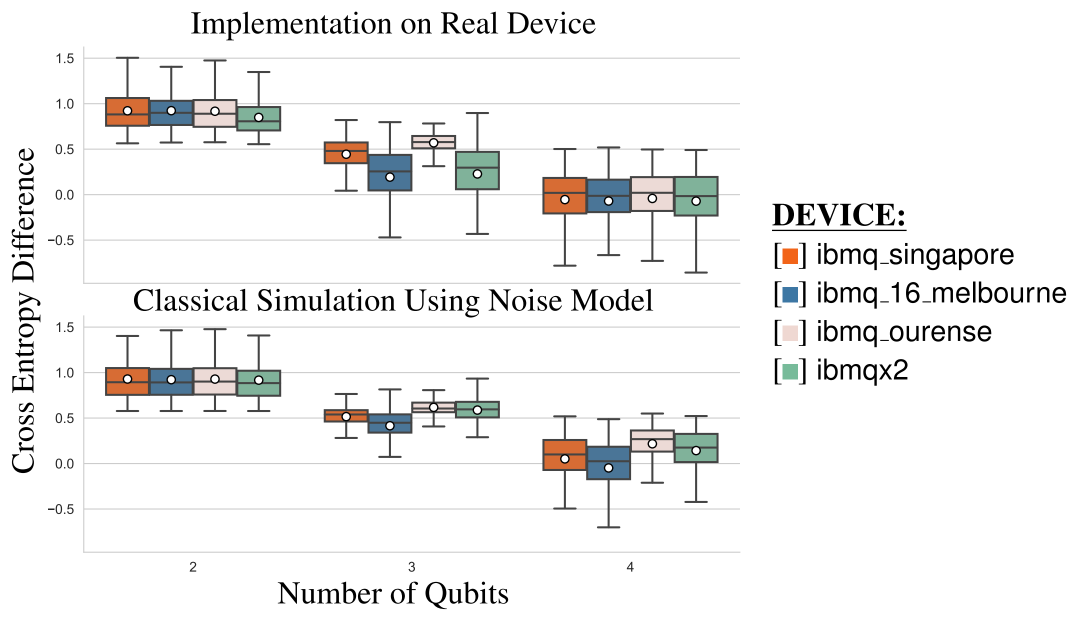


Figure 3.13: **Comparison of devices using the cross entropy difference metric, when running deep circuits compiled using noise-aware Qiskit**. Values close to 1 indicate better performance. Both simulations using Qiskit noise models, and implementations on real devices, are included.

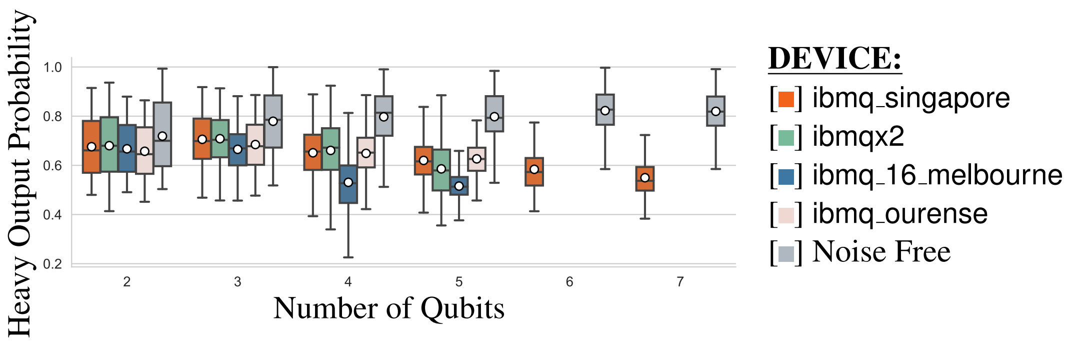


Figure 3.14: **Comparison of real devices, using the heavy output probability metric, when running shallow circuits compiled using noise-aware pytket**. Values close to Noise Free indicate better performance.

noise levels or chemistry motivated circuits which generate exponentially distributed output probabilities at lower depth.

Shallow Circuits as a Benchmark

Figure 3.14 demonstrates that shallow circuits allow us to benchmark the behaviour of a quantum computing stack for applications involving circuits with many qubits but low circuit depth [3, 4, 113].

The results show ibmq_singapore outperforms the comparably sized ibmq_16_melbourne and has similar performance to ibmq_ourense for smaller numbers of qubits. ibmq_singapore outperforms ibmq_ourense by having more qubits available. This superior performance of ibmq_singapore is in comparison to the results of Figure 3.10, where ibmq_ourense was shown to perform well. This justifies our suggestion that shallow circuits should be included in benchmarking suites. Doing so enables exploring higher-width circuits, and in this setting devices that perform poorly when implementing square circuits or deep circuits may perform well.

3.4.3 Insights from Classical Simulation

Noise in a non-fault-tolerant quantum computer results in discrepancies between results obtained from running on real hardware and those that would be obtained from an ideal quantum computer. Noise models are utilised to help identify why these discrepancies occur, as discussed in Chapter 2. However, a perfect model of the noise – which could reproduce the results of real hardware (up to statistical error) – could require many parameters to completely specify it. Therefore, most noise models consider only a small handful of physically- motivated noise sources. Consequently discrepancies between the results of noisy simulation and running experiments on real hardware always remain.

Historically, closing this gap required developing noise models of increasing sophistication. Doing so typically requires a great deal of physics expertise to identify new noise channels. Further, new experiments would have to be designed in order to estimate the parameters in the noise model. Here, we show how application-motivated benchmarks can be helpful in identifying whether new noise channels should be incorporated into a noise model. By isolating the circuit types and coupling maps for which the discrepancies are greatest, we gain intuition about the possible causes of the mismatch.

For the devices explored here, the noise models are built using Qiskit. They are derived from a device's properties, and include one- and two-qubit gate errors[22] and single-qubit readout errors. We find these noise models are inadequate to explain some of the discrepancies observed in the data.

Noise Does Not Just Flatten Distributions

One discrepancy between experiments and noisy simulations is the spread of the data. For example, Figure 3.10 shows that only in the experimental case do the whiskers of the plot fall below the value 0.5, indicating the heavy outputs are less likely than they would be in the uniform distribution. Some noise type, in particular one which shifts the probability density, rather than uniformly flattening it, is not considered, or is under appreciated, by the noise models used. Identifying that noise channel is left to future work, though we speculate it may be related to a kind of amplitude damping.

Noise Models Under-Represent Some Noise Channels

The classical simulations in Figure 3.10 suggest ibmqx2 should perform similarly to ibmq_ourense in most cases. In fact, it quite consistently performs worse. This is isolated in Figure 3.11, with the same phenomenon being observed in Figure 3.6 and Figure 3.13, showing the behavior is consistent across all circuit types and figures of merit. This difference between simulated and experimental results is pronounced in the case of Figure 3.13, where deep circuits are used, suggesting the noise models may be underestimating the error from time-dependent noises such as depolarising and dephasing, or from two-qubit gates which are more prevalent in deep circuits.

Another such example of a two-qubit noise channel, which is explicitly not accounted for in our noise models, is crosstalk. The results in Figure 3.11 are consistent with the expectation that crosstalk should have the greatest impact on more highly connected devices [252]. As such crosstalk may be the origin of the discrepancy. Of note is the fact this benchmark wasn't explicitly designed to capture the effects of crosstalk, and yet those effects manifest themselves in its results. We anticipate that including crosstalk-aware passes in compilation strategies [253] would reduce the discrepancy.

3.5 Conclusion

The performance of quantum computing devices is highly dependent on several factors. Amongst them are the noise levels of the device, the software used to construct and manipulate the circuits implemented, and the applications for which the device is used. The impacts of these factors on the performance of a quantum computing stack

[22]These are modelled to consist of a depolarising error followed by an amplitude damping errors.

are intertwined, making the task of predicting its holistic performance from knowledge of the performance of each component impossible. In order to understand and measure the performance of quantum computing stacks, benchmarks must take this into consideration.

In this work we have addressed this problem by introducing a methodology for performing application-motivated, holistic benchmarking of the full quantum computing stack. To do so we provide a benchmark suite utilising differing circuit classes and figures of merit to access a variety of properties of the device. This includes the use of three circuit classes: deep circuits and shallow circuits, which are novel to this work; and square circuits, which resemble random circuits used in other benchmarking experiments [212]. In addition we make use of a diverse selection of figures of merit to measure the performance of the quantum computing stacks considered, namely: Heavy Output Generation Benchmarking, Cross-Entropy Benchmarking, and the ℓ_1-norm distance.

In particular, in the form of deep circuits we present an alternative to previous approaches to application-motivated benchmarking. This is by considering circuits inspired by one of the primitives utilised in VQE, namely Pauli gadgets employed for state preparation, rather than VQE itself. Further, we have found that the performances of quantum computing stacks are indistinguishable when using square circuits and Heavy Output Generation Benchmarking for a large number of qubits. However shallow circuits extend the number of qubits for which detail can be observed.

We demonstrate this benchmark suite by employing it on ibmqx2, ibmq_16_melbourne, ibmq_ourense, and ibmq_singapore. In doing so we justified our hypothesis that the accuracy of a computation depends on several levels of the quantum computing stack, and that each layer should not be considered in isolation. For example, identifying that the increased connectivity of a device does not compensate for the increased noise, as we do in Section 3.4.1, shows the impact of this layer of the stack, and justifies investigating devices with a variety of coupling maps and noise levels. By showing the differing performance between five compilation strategies, we are able to identify, in Section 3.4.1, the dependence of the best compilation strategy to use on the device and the dimension of the circuit. This illustrates the dependence of the performance of the quantum computing stack on the compilation layer, and the interdependence between the compilation strategy, device and application on the overall performance of the quantum computing stack. In particular, noise-aware compilation strategies often perform well, when the noise model used by the strategy is accurate, as discussed in Section 3.4.3.

In Section 3.4.2, the wide selection of circuits within the proposed benchmark suite reveals that the same device, evaluated according to a fixed figure of merit, will perform differently when running different applications, whose circuits are compiled by the same compilation strategy. Indeed the comparative performance of (compilation strategy, device) pairs is shown to vary between our circuit classes. This justifies our inclusion of circuit classes which collectively cover a wide selection of applications in the benchmark suite proposed here, and our full quantum computing stack approach.

We foresee the benchmarks conducted in this work providing a means to select the best quantum computing stack, of those explored here, for a particular task, and vice versa. As such we also anticipate that a variety of new quantum computing stacks could be benchmarked in the way described in this work, empowering the user with knowledge about the performance of current quantum technologies for particular tasks. Indeed, these benchmarks may, in time, come to complement noise models and calibration information as a means to disseminate information about a device's performance. This parallels the use of the LINPACK benchmarks [10] alongside FLOPS to compare diverse classical computers. Recently, quantum volume, as defined in [212], has started to be adopted as one such metric [84, 232, 254, 255], and we hope the benchmark suite developed here will be incorporated similarly.

The work presented here could be extended in several directions. The first is to examine the impact of incorporating these benchmarks into a compilation strategy. While noise-aware compilation strategies currently use properties of qubits to decide how to compile a circuit, it would be interesting to explore if instead optimising for these benchmarks would change the compilation. For example, performing these benchmarks on two subgraphs of the device's coupling map may help to decide which qubits to use at the placement step of compilation. Such an approach is well suited to deterministic compilers, however with a sufficiently large sample of benchmark circuits, the compiler's performance on the benchmark suite would be indicative of its performance in practice, even if the compiler is probabilistic. Second, the philosophy of application-motivated benchmarking could be extended to circuits which are more easily classically simulable. Because of their reliance on classical simulation, the benchmarks introduced here may be used up to, but not after, the point of demonstrating quantum computational supremacy. To address this, while adhering to the volumetric benchmarking philosophy utilised in this chapter, work following that of this chapter introduced an approach to benchmarking using circuits built from structured circuits, followed by their inverse [214]. Third, as the results of Section 3.4.1 show that changing the hardware can dramatically influence performance, we envision a need to systematically study how properties of hardware, such as noise levels or connectivity, influence a given device's performance. In future work, we hope to incorporate more devices to advance our ability to perform such a systematic inquiry. Fourth, our benchmarks may facilitate an understanding of how new, or hard-to-characterise, noise affects the practical performance of quantum computers, as implied by the classical simulations of Section 3.4.3. Fifth, it would be valuable to quantitatively measure the extent to which the circuit classes introduced cover the circuits which correspond to instances of applications. This would provide some confidence that the possibility a quantum computing stack could be engineered to perform well at the benchmarks introduced, but might perform less well when implementing an instance of the application they represent, could be avoided. At present an appropriate measure is not known to the authors, but notions of expressibility [236] might inspire the definition of such a measure. Finally, there is a need to explicitly study the correlation between the results of an application-motivated benchmark and the performance of a quantum computing stack at running the application which motivated it. This would experimentally justify that benchmarking application subroutines provides reliable predictors of performance when running the application itself.

Chapter 4

Blind IQP Computation, and an IQP Hypothesis Test

In Chapter 2 and Chapter 3 we considered the verification, characterisation and benchmarking of quantum devices of a size at which classical simulation is possible. This allowed us to present useful measures and predictions of the performance of these devices, which can be used to further develop and improve quantum technology. In this chapter we consider NISQ devices of a size that, in principle, makes them impossible to simulate on a classical device. We introduce a means for a server of this form to prove its capacity to perform a demonstration of quantum computational supremacy to an untrusting and computationally weak client. The resource consumption of our scheme scales polynomially with the system size, which contrasts with the approaches taken in previous chapters. As such, this also allows for the benchmarking of systems significantly larger than that needed to demonstrate quantum computational supremacy.

Tools for the verification of quantum computation, such as those discussed in Section 1.6.2, may be used to certify demonstrations of quantum computational supremacy. Alternatively problems in $\mathsf{NP} \cap \mathsf{BQP}$, but for which no algorithm in P is known, such as factoring, may be used for this purpose. However, the cost of implementing these methods on a quantum computer is high, and so they are likely to be inaccessible by NISQ devices [58]. Instead we use only technology which it could reasonably be believed will become available in the near-term. In this spirit, we assume that the Server is capable of implementing at least IQP circuits, but not necessarily BQP circuits. Further, we assume that the Client has access to a limited quantum network, which may indeed be available in the near-term [187], on top of polynomial classical computing resources.

We adapt the Shepherd-Bremner IQP Hypothesis Test, outlined in Section 1.6.3, which, under some (since disproved) assumptions, can only be passed by a server that is capable of implementing IQP circuits. Our adaptation ensures the security of the protocol against any malicious server wishing to impersonate the capability to implementing IQP circuits. We require the Client is endowed with the ability to prepare and send single qubits, on top of the polynomial classical resources demanded by the Shepherd-Bremner IQP Hypothesis Test. Importantly, this increase in the Client's resources does not include an increase in the classical computational power. This compares favourably with Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking used in Chapter 3, which have a resource consumption that grows exponentially with circuit size. The requirements our scheme demands from the Server are also mildly increased upon those of the Shepherd-Bremner IQP Hypothesis Test. These requirements now include the capacity to perform a two round adaptive MBQC computation, rather than the one round non-adaptive computation that is typical for IQP computations.

The key to developing our IQP Hypothesis Test will be by incorporating blindness, as discussed in Section 1.6.1, in place of the since broken hiding scheme used in the Shepherd-Bremner IQP Hypothesis Test. We do so by utilising the implementation of IQP computations in MBQC, as introduced in Section 1.4, to develop an information-theoretically secure blind delegated protocol for IQP computations that keeps the details of the computation hidden from the device performing it. Our approach makes use of tools introduced to perform DQC using MBQC as discussed in Section 1.6.1 and Section 1.6.2. However, the number of different states we require the Client to produce is reduced as compared to VUBQC and UBQC.

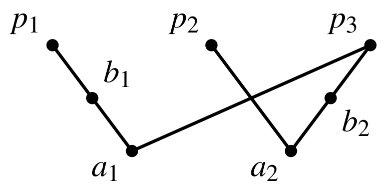
Our information-theoretically secure blind delegated implementation of IQP circuits is introduced in Section 4.1, and we use the Abstract Cryptography framework to show that it is compositionally secure. The information-theoretic security of our blind scheme prevents attacks on computational complexity assumptions such as those which befell the Shepherd-Bremner IQP Hypothesis Test. This use of blindness allows us to utilise known properties of the output, without the need for exponential classical compute resources, as in Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking. In Section 4.2 we develop our IQP Hypothesis Test, which a limited quantum client can run on an untrusted server, by utilising this functionality to implement a similar protocol to the Shepherd-Bremner IQP Hypothesis Test.

4.1 Blind Delegated IQP Computation

As mentioned when introducing verification in Section 1.6.2, ofttimes blindness is a key component of verification schemes. We pursue this approach here and begin by building a method for delegating an IQP computation without revealing the X-program which defines it. The intuition behind the method used to perform this hiding is that the Client will ask the Server to produce a quite general graph state, and then move from that one to the state that is required for the computation. This movement will be conducted using the bridge and break techniques of Section 1.4.2. If this is done in a blind way then the Server only has some knowledge of the general starting state from which any number of other quantum states may have been built. This may be broken down into three key problems, which we address separately:

General Graph Selection: In Section 4.1.1 we introduce a graph which may be transformed into any one of many IQP graphs, as they are defined in Definition 1.5.3. We discuss the process of transforming this graph to a particular IQP graph.

Graph State Transformation: In Section 4.1.2 we discuss how these graph theoretic operations are transformed into operations on graph states. The result of this



$$\widetilde{\boldsymbol{Q}} = \begin{pmatrix} -1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

Figure 4.1: **An extended** IQP **graph.** This extended IQP graph, as defined in Definition 4.1.1, is described by the matrix $\widetilde{\boldsymbol{Q}}$ with $(n_a, n_p, n_b) = (2,3,2)$, $P = [p_1, p_2, p_3]$, $A = [a_1, a_2]$, and $B = [b_1, b_2]$. It can be created by replacing $\boldsymbol{Q}_{1,1}$ and $\boldsymbol{Q}_{2,3}$ in the IQP graph of Figure 1.12 with -1. The function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to \mathbb{Z}_{n_b}$ is defined as $g_{\widetilde{\boldsymbol{Q}}}(1,1) = 1$ and $g_{\widetilde{\boldsymbol{Q}}}(2,3) = 2$.

will be a method for transforming a general quantum state into an IQP state, as introduced in Definition 1.5.4.

Blind Graph Transformation: In Section 4.1.3 we present a proof that this transformation between graph states can be performed blindly by the Server when the Client is endowed with the ability to prepare and send single qubits.

4.1.1 General Graph Selection

The *extended IQP graph* addresses the problem of general graph selection.

Definition 4.1.1 (Extended IQP Graph)**.** *An* extended IQP graph *is represented by* $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$. *The vertex set contains* $A = \{a_1, \ldots, a_{n_a}\}$ *and* $P = \{p_1, \ldots, p_{n_p}\}$. *As for* IQP *graphs in Definition 1.5.3,* $\widetilde{\boldsymbol{Q}}_{ij} = 1$ *indicates an edge between vertices* a_i *and* p_j*, while* $\widetilde{\boldsymbol{Q}}_{ij} = 0$ *indicates no edge.*

We interpret $\widetilde{\boldsymbol{Q}}_{ij} = -1$ *as the existence of an intermediary vertex* b_k *between vertices* p_j *and* a_i*, and denote with* n_b *the number of -1s in* $\widetilde{\boldsymbol{Q}}$*. As such the vertex set also includes* $B = \{b_1, \ldots, b_{n_b}\}$*, which we henceforth refer to as the* bridge and break vertices*, and the edge set includes edges between* b_k *and* a_i *as well as* b_k *and* p_j *when* $\widetilde{\boldsymbol{Q}}_{ij} = -1$*. To keep track of these connections, we define the surjective function* $g_{\widetilde{\boldsymbol{Q}}}$*, where* $g_{\widetilde{\boldsymbol{Q}}}(i,j) = k$ *when there is an intermediate vertex* b_k *connected to* a_i *and* p_j*.*

An extended IQP graph $\widetilde{\boldsymbol{Q}}$ can be built from an IQP graph $\boldsymbol{Q}$ by replacing any number of the entries of $\boldsymbol{Q}$ with -1. Throughout the remainder of this work we will use the tilde notation to represent an extended IQP graph $\widetilde{\boldsymbol{Q}}$ build from an IQP graph $\boldsymbol{Q}$ in this way. By way of an example, the extended IQP graph of Figure 4.1 is created by replacing $\boldsymbol{Q}_{1,1}$ and $\boldsymbol{Q}_{2,3}$ in the IQP graph of Figure 1.12 with -1. Equally, by applying bridge operators to both b_1 and b_2 in $\widetilde{\boldsymbol{Q}}$ of Figure 4.1 we arrive at $\boldsymbol{Q}$ of Figure 1.12. In general it is always possible to recover the IQP graph $\boldsymbol{Q}$ from the extended IQP graph $\widetilde{\boldsymbol{Q}}$ in this way.

4.1.2 Graph State Transformation

We can now state Lemma 4.1.1 which teaches us how to translate bridge and break operations from graph theoretical ideas into practical operations on quantum states.

Lemma 4.1.1. *Consider a quantum state $E_{\boldsymbol{Q}}|\phi\rangle$, where we have used the graph state circuit notation of Definition 1.4.2, and where $|\phi\rangle$ is an arbitrary quantum state. If $\widetilde{\boldsymbol{Q}}$ is an extended* IQP *graph built from $\boldsymbol{Q}$ then there exists a state $E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle$, which can be transformed into the state $E_{\boldsymbol{Q}}|\phi\rangle$ through a sequence of Pauli-*Y *basis measurements on qubits and local rotations around the* Z *axis of the unmeasured qubits through angles $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$.*

Lemma 4.1.1 is a restatement of Lemma 1.4.1 in the setting appropriate for this work, and its proof is also similar. We provide the proof in this new setting in Appendix C.1.

Note that the state $|\psi\rangle$ is as yet unspecified in Lemma 4.1.1, ensuring only the existence of such a state. However the proof in Appendix C.1 is constructive, providing us with a process for generating the state $|\psi\rangle$. The process described in Appendix C.1 will be used throughout this chapter. In summary $|\psi\rangle$ is equivalent to $|\phi\rangle$ but for the addition of bridge and break qubits of the form $|b_k\rangle = \mathsf{Y}^{r_k^b}\sqrt{\mathsf{Y}}^{d_k^b}|0\rangle$ when $g_{\widetilde{\boldsymbol{Q}}}(i,j) = k$. Here $r_k^b \in \{0,1\}$ can be assigned without immediate consequence, although it will be vital later. d_k^b is set to 0 in the case of break operations, and 1 in the case of bridge operations. Measurements of the qubits corresponding to bridge and break vertices of $E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle$ in the Pauli-Y basis result in the state

$$\prod_{k=1}^{n_b}\left(\mathsf{S}_{p_j}^{(-1)^{1-s_k^b+r_k^b}} \otimes \mathsf{S}_{a_i}^{(-1)^{1-s_k^b+r_k^b}}\right)^{d_k^b}\left(\mathsf{Z}_{p_j}^{r_k^b} \otimes \mathsf{Z}_{a_j}^{r_k^b}\right)^{1-d_k^b} E_{\boldsymbol{Q}}|\phi\rangle$$

where the quantity s_k^b is the outcome of this measurement on qubit b_k.

It is possible to perform an IQP computation using this method. Although the quantum state generated using this method would equal $E_{\boldsymbol{Q}} \bigotimes_1^{n_a+n_p}|+\rangle$ up to some S corrections, these corrections may be accounted for by making corrections to the primary and ancillary measurement bases. Protocol 4.1.1 uses the methods discussed to build an IQP state and perform IQP computations.

4.1.3 Blind Graph Transformation

We can now present our scheme for blind graph transformation. To do so, we establish that the procedure for building IQP graph states discussed in Section 4.1.2 can be used to blindly create an IQP state at the Server side. Our wish is to construct the Ideal Resource of Figure 4.2 which takes as input from the Client an IQP computation, $(\boldsymbol{Q},\theta)$, and in return gives a classical output $\widetilde{x}$. If the Server is honest, then $\widetilde{x}$ comes from the distribution corresponding to $(\boldsymbol{Q},\theta)$. If the Server is dishonest, then they can input some quantum operation $\mathcal{E}$ and some quantum state ρ_B and force the output to the Client into the classical state $\mathcal{E}(\boldsymbol{Q},\theta,\rho_B)$. We would like for the Server to only receive an extended IQP graph $\widetilde{\boldsymbol{Q}}$ which can be built from $\boldsymbol{Q}$, the distribution $\mathcal{Q}$ over

Protocol 4.1.1 IQP computation, starting from an extended IQP graph. This adapts the approach of Protocol 1.5.2 to account for this new graph state preparation step.

Input: $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$, $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$, $\theta \in [0,2\pi]$
Output: $\widetilde{x} \in \{0,1\}^{n_p}$

1: Generate the states $|+\rangle = |p_j\rangle$ and $|+\rangle = |a_i\rangle$ for $j \in \left\{0,...n_p\right\}$ and $i \in \{0,...n_a\}$
2: Create $d^b \in \{0,1\}^{n_b}$ in the following way: For $i = 1,\dots,n_a$ and $j = 1,\dots,n_p$, if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 0$, then $d^b_k = 0$ else if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 1$ then $d^b_k = 1$. Keep track of the relation between k and (i,j) via the surjective function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to \mathbb{Z}_{n_b}$. This assignment of d^b_k corresponds to adding a break ($d^b_k = 0$) qubit when a bridge or break qubit exists ($\widetilde{\boldsymbol{Q}}_{ij} = -1$) but no edge is required in the final state ($\boldsymbol{Q}_{ij} = 0$), and adding a bridge qubit ($d^b_k = 1$) when a bridge or break qubit exists and edge is required in the final state ($\boldsymbol{Q}_{ij} = 1$).
3: Generate $r^b \in \{0,1\}^{n_b}$ at random and produce the states $|b_k\rangle = \mathsf{Y}^{r^b_k}\left(\sqrt{\mathsf{Y}}\right)^{d^b_k}|0\rangle$ for $k \in \{1,...,n_b\}$.
4: Implement $E_{\widetilde{\boldsymbol{Q}}}$.
5: Measure qubits $b_1,...,b_{n_b}$ in the basis $\left\{|+^{\mathsf{Y}}\rangle, |-^{\mathsf{Y}}\rangle\right\} := \{|0\rangle + i|1\rangle, |0\rangle - i|1\rangle\}$. Take the outcome of measuring qubit b_k to be $s^b_k \in \{0,1\}$ if the measurment projects the output to the state $|0\rangle + i(-1)^{s^b_k}|1\rangle$.
6: Calculate $\boldsymbol{\Pi}^z, \boldsymbol{\Pi}^s \in \{0,1\}^{n_p}$ and $\boldsymbol{A}^z, \boldsymbol{A}^s \in \{0,1\}^{n_a}$ using

$$\Pi^z_j = \sum_{i,k:g(i,j)=k} r^b_k \left(1 - d^b_k\right)$$
$$\Pi^s_j = \sum_{i,k:g(i,j)=k} (-1)^{1-s^b_k+r^b_k} d^b_k$$
$$A^z_i = \sum_{j,k:g(i,j)=k} r^b_k \left(1 - d^b_k\right)$$
$$A^s_i = \sum_{j,k:g(i,j)=k} (-1)^{1-s^b_k+r^b_k} d^b_k$$

7: Calculate $\boldsymbol{A} \in \{0,1,2,3\}^{n_a}$ and $\boldsymbol{\Pi} \in \{0,1,2,3\}^{n_p}$ for the ancillary and primary qubits respectively, where $A_i = A^s_i + 2A^z_i \pmod 4$ and $\Pi_j = \Pi^s_j + 2\Pi^z_j \pmod 4$.

Protocol continues below...

Protocol 4.1.1 Continued

8: Measure qubits in the basis of equation (4.1), for the ancillary and primary qubits respectively, producing measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$.

$$\mathsf{S}^{-A_i}\{|0_{2\theta}\rangle, |1_{2\theta}\rangle\} \quad , \quad \mathsf{S}^{-\Pi_j}\{|+\rangle, |-\rangle\} \tag{4.1}$$

Here we have used the notation $|0_{2\theta}\rangle, |1_{2\theta}\rangle$ introduced in equation (1.24).

9: Generate and output $\tilde{x} \in \{0,1\}^{n_p}$ using equation (1.25).

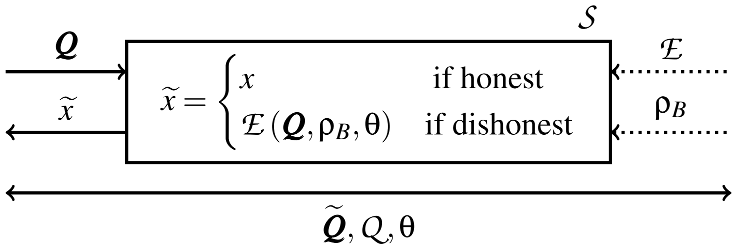

$\mathcal{S}$
$\boldsymbol{Q}$
$\tilde{x}$
$\tilde{x} = \begin{cases} x & \text{if honest} \\ \mathcal{E}(\boldsymbol{Q}, \rho_B, \theta) & \text{if dishonest} \end{cases}$
$\mathcal{E}$
ρ_B
$\tilde{\boldsymbol{Q}}, \mathcal{Q}, \theta$

Figure 4.2: **The ideal blind delegated** IQP **computation resource.**

the possible $\boldsymbol{Q}$ from which $\tilde{\boldsymbol{Q}}$ could be built (i.e. the distribution from which $\boldsymbol{Q}$ was picked), and θ. Let us assume that $\tilde{\boldsymbol{Q}}$, $\mathcal{Q}$ and θ are public knowledge.

The proposed real communication protocol is described in detail by Protocol 4.1.2 and graphically shown in Figure 4.3. One difference between Protocol 4.1.2 and Protocol 4.1.1 is that in the former case the operations have been distributed between the Server and the Client. In particular the generation of all single qubit states is assigned to the Client, and all entanglement and measurement is assigned to the Server. This allows the Client to introduce blindness to Protocol 4.1.1 by randomly rotating the primary and ancillary qubits. This introduction of randomness is the other main difference between Protocol 4.1.2 and Protocol 4.1.1. This is a similar approach to other blind DQC schemes implemented in MBQC, as discussed in Section 1.6.1, but differs from those by using only eigenstates of single qubits Pauli operators.

Consider when the ideal resource and real protocols are filtered, in which case the distinguisher will have no inputs or outputs on the Server's side. Correctness, as defined in Definition 1.7.5, then follows if the output on the Client's side is from the requested IQP distribution. This is evidenced by observing that the transformations between Protocol 4.1.1 and Protocol 4.1.2 leave the output unchanged. Indeed the random rotations are corrected by rotating the measurement bases of those qubits, therefore ensuring that the original IQP computation is being performed.

During the execution of the real protocol of Protocol 4.1.2, the Server twice sends classical bit strings to the Client which correspond to the measurement outcomes of the sent qubits. These are used by the Client to correct later measurements angles, which are dictated to the Server by the Client. If the Server wants to deviate from the protocol, i.e. in the unfiltered case, they will use some quantum map $\mathcal{E}$ on the information received so far, together with the state ρ_B in their own register. At the final step of the protocol the Server may output some quantum state ρ'_B. We would

Protocol 4.1.2 Blind delegated IQP computation. This adapts the approach of Protocol 4.1.1 by randomly rotating the primary and ancillary qubits, making the necessary corrections to the measurement angles, and distributing the operations between the Server and the Client.

Public: $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$, $\theta \in [0,2\pi]$, $\mathcal{Q}$ (the distribution from which $\boldsymbol{Q}$ is picked)
Client input: $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$
Client output: $\widetilde{x} \in \{0,1\}^{n_p}$

Client:

```
1: Randomly generate $r^p, d^p \in \{0,1\}^{n_p}$ and $r^a, d^a \in \{0,1\}^{n_a}$ where $n_p$ and $n_a$ are the numbers of primary and ancillary qubits respectively.
2: Generate the states $|p_j\rangle = \mathsf{Z}^{r^p_j}\mathsf{S}^{d^p_j}|+\rangle$ and $|a_i\rangle = \mathsf{Z}^{r^a_i}\mathsf{S}^{d^a_i}|+\rangle$ for $j \in \{1,\dots,n_p\}$ and $i \in \{1,\dots,n_a\}$
3: Create $d^b \in \{0,1\}^{n_b}$ in the following way: For $i = 1,\dots,n_a$ and $j = 1,\dots,n_p$, if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 0$, then $d^b_k = 0$ else if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 1$ then $d^b_k = 1$. Keep track of the relation between $k$ and $(i,j)$ via the surjective function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to \mathbb{Z}_{n_b}$.
4: Generate $r^b \in \{0,1\}^{n_b}$ at random and produce the states $|b_k\rangle = \mathsf{Y}^{r^b_k}\left(\sqrt{\mathsf{Y}}\right)^{d^b_k}|0\rangle$ for $k \in \{1,...,n_b\}$.
5: State $\rho$ comprising of all of the Client's produced states is sent to the Server.
```

Server:

```
6: Implement $E_{\widetilde{\boldsymbol{Q}}}$.
7: Measure qubits $b_1,...,b_{n_b}$ in the basis $\{|+^{\mathsf{Y}}\rangle, |-^{\mathsf{Y}}\rangle\} := \{|0\rangle + i|1\rangle, |0\rangle - i|1\rangle\}$. Take the outcome of measuring qubit $b_k$ to be $s^b_k \in \{0,1\}$ if the measurment projects the output to the state $|0\rangle + i(-1)^{s^b_k}|1\rangle$.
8: Send the outcome $s^b \in \{0,1\}^{n_b}$ to the Client.
```

Protocol continues below...

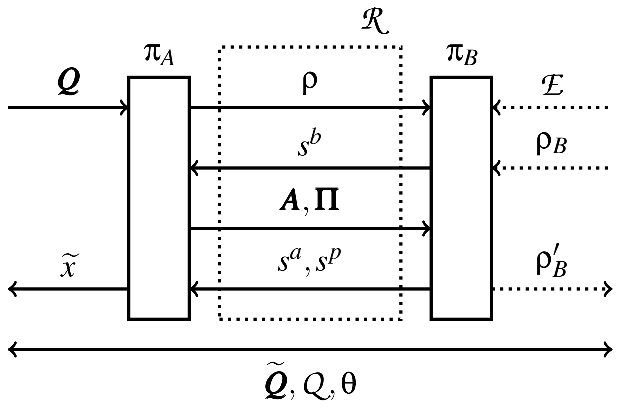


Figure 4.3: **The real resource of Protocol 4.1.2.**

Protocol 4.1.2 Continued

Client:

9: Calculate $\boldsymbol{\Pi}^z, \boldsymbol{\Pi}^s \in \{0,1\}^{n_p}$ and $\boldsymbol{A}^z, \boldsymbol{A}^s \in \{0,1\}^{n_a}$ using: (4.2), (4.3), (4.4) and (4.5).

$$\Pi^z_j = \sum_{i,k:g(i,j)=k} r^b_k \left(1 - d^b_k\right) - r^p_j \tag{4.2}$$

$$\Pi^s_j = \sum_{i,k:g(i,j)=k} (-1)^{1-s^b_k + r^b_k} d^b_k - d^p_j \tag{4.3}$$

$$A^z_i = \sum_{j,k:g(i,j)=k} r^b_k \left(1 - d^b_k\right) - r^a_i \tag{4.4}$$

$$A^s_i = \sum_{j,k:g(i,j)=k} (-1)^{1-s^b_k + r^b_k} d^b_k - d^a_i \tag{4.5}$$

10: Calculate $\boldsymbol{A} \in \{0,1,2,3\}^{n_a}$ and $\boldsymbol{\Pi} \in \{0,1,2,3\}^{n_p}$ for the ancillary and primary qubits respectively, where $A_i = A^s_i + 2A^z_i \pmod 4$ and $\Pi_j = \Pi^s_j + 2\Pi^z_j \pmod 4$.

11: Send $\boldsymbol{A}$ and $\boldsymbol{\Pi}$ for the ancillary and primary qubits respectively, to the Server.

Server:

12: Measure qubits in the basis of equation (4.1), for the ancillary and primary qubits respectively, producing measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$.

13: Send measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$ to the Client.

Client:

14: Generate and output $\widetilde{x} \in \{0,1\}^{n_p}$ using equation (1.25).

like to show that the classical strings sent by the Client, and the state ρ'_B, contain no information.

As discussed in Section 1.7, to prove Protocol 4.1.2 is compositionally secure we drop the notion of a malicious server for that of a global distinguisher that has a view of all inputs and outputs of the relevant resources. To recreate the view of a malicious server, we develop a simulator σ interfacing between the ideal resource $\mathcal{S}$ of Figure 4.2 and the distinguisher. The simulator will provide the necessary resources to the ideal resource, and alter the necessary outputs, in such a way that the ideal resource is able to produce outputs that are indistinguishable from those from the real protocol. Since the simulator adds no new information of its own, this shows that the ideal and real resources are equally secure.

In particular we will use the teleportation techniques inspired by [194], to prove security in the case of a malicious server. We will prove that

$$\pi_A \mathcal{R} \equiv \mathcal{S}\sigma,$$

where $\mathcal{R}$ is the quantum and classical communication channel used by the Client and the Server in the protocol and π_A is the Client's protocol. Doing so constitutes a proof of Theorem 4.1.1. We give only an intuitive proof here, supported by the figures in the remainder of this chapter, and leave a thorough proof to Appendix C.2.

Theorem 4.1.1. *The protocol described by Protocol 4.1.2 is information-theoretically secure against a dishonest server.*

Proof. The proof consists of a pattern of transformations of the real protocol of Protocol 4.1.2, into the ideal resource plus simulator setting of Protocol 4.1.3. These transformations leave the computation unchanged, therefore ensuring the indistinguishability of the two settings and so the compositional security of Protocol 4.1.2. Where there are transformations of the protocols required, we will detail which lines are transformed, and in what way they are transformed.

The first set of transformations demonstrate an alternative state generation technique that the Client might employ. Firstly, line 2 of Protocol 4.1.2 generates at random one of the four states $|+\rangle$, $|+^{\mathsf{Y}}\rangle$, $|-\rangle$ and $|-^{\mathsf{Y}}\rangle$. The same effect is achieved by measuring one qubit of an EPR pair, $|\Phi_+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, with equal probability in one of the bases $\{|+\rangle, |-\rangle\}$ and $\left\{|+^{\mathsf{Y}}\rangle, |-^{\mathsf{Y}}\rangle\right\}$. The unmeasured qubit will be, with equal probability, in one of the four aforementioned states. Secondly the application of the $\left(\sqrt{\mathsf{Y}}\right)^{d_k^b}$ operation in line 4 of Protocol 4.1.2 decides, according to the graph to be created, if the bridge and break qubit will be drawn from the set $\{|+\rangle, |-\rangle\}$ or from the set $\{|0\rangle, |1\rangle\}$. Choosing to measure one half of an EPR pair in one of the bases $\{|+\rangle, |-\rangle\}$ or $\{|0\rangle, |1\rangle\}$, and taking the unmeasured half of the EPR pair as the bridge and break qubit, has the same effect. The random rotation $\mathsf{Y}^{r_k^b}$ on line 4 of Protocol 4.1.2 has the same effect as the randomness that is intrinsic to the EPR pair measurement, and so the overall effect is the same. These alternate state generation techniques may be visualised in Figure 4.4 which presents a simple rearrangement of the Real

Protocol 4.1.3 Blind delegated IQP computation with simulator. The proof of Theorem 4.1.1 outlines a transformation of Protocol 4.1.2 into this.

Public: $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$, $\theta \in [0,2\pi]$, $\mathcal{Q}$ (the distribution from which $\boldsymbol{Q}$ is picked)
Client input: $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$
Client output: $\widetilde{x} \in \{0,1\}^{n_p}$

The simulator:

1: Generate n_p EPR pairs $|\Phi_+\rangle_j^p$, n_a EPR pairs $|\Phi_+\rangle_i^a$ and a further n_b EPR pairs $|\Phi_+\rangle_k^b$.
2: Send half of each EPR pair to the ideal resource, through the interface ρ_B in Figure 4.2, and the other half to the distinguisher.
3: Receive the bitstring $s_b \in \{0,1\}^{n_b}$ from the distinguisher and forwards it to the ideal resource through the interface $\mathcal{E}$ in Figure 4.2. In this way s_b partially defines the function $\mathcal{E}$ describing the Server's deviation.
4: Randomly generate $\boldsymbol{\Pi} \in \{0,1,2,3\}^{n_p}$ and $\boldsymbol{A} \in \{0,1,2,3\}^{n_a}$ where n_p and n_a are the numbers of primary and ancillary qubits respectively.
5: Send $\boldsymbol{A}$ and $\boldsymbol{\Pi}$ to the ideal resource, through the interface $\mathcal{E}$ in Figure 4.2, and to the distinguisher.
6: Receive the bitstrings $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$ from the distinguisher and forward them to the ideal resource, through the interface $\mathcal{E}$ in Figure 4.2. Then s^p and s^p form part of the defintion of the deviation $\mathcal{E}$ by the Server.

The ideal resource:

1: Create $d^b \in \{0,1\}^{n_b}$ in the following way: For $i = 1,\dots,n_a$ and $j = 1,\dots,n_p$, if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 0$, then $d_k^b = 0$ else if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 1$ then $d_k^b = 1$. Keep track of the relation between k and (i,j) via the surjective function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to \mathbb{Z}_{n_b}$.
2: Measure one half of each of $|\Phi_+\rangle_k^b$ in the basis $\sqrt{Y}^{d_k^b}\{|0\rangle,|1\rangle\}$ to achieve outcome r_k^b, for $k = 1,\dots,n_b$.

Protocol continues below...

Protocol 4.1.3 Continued

3: Calculate $d_j^p \in \{0,1,2,3\}^{n_p}$ and $d_i^a \in \{0,1,2,3\}^{n_a}$ using (4.6) and (4.7) respectively.

$$d_j^p = \sum_{i,k:g(i,j)=k} (-1)^{s_k^b + r_k^b} d_k^b + 2 \sum_{i,k:g(i,j)=k} r_k^b \left(1 - d_k^b\right) - \Pi_j \tag{4.6}$$

$$d_i^a = \sum_{j,k:g(i,j)=k} (-1)^{s_k^b + r_k^b} d_k^b + 2 \sum_{j,k:g(i,j)=k} r_k^b \left(1 - d_k^b\right) - A_i \tag{4.7}$$

4: Measure one half of each of $|\Phi_+\rangle_j^p$ in the basis $\mathsf{S}^{d_j^p}\{|+\rangle, |-\rangle\}$ to achieve outcome r_j^p and one half of each of $|\Phi_+\rangle_i^a$ in the basis $\mathsf{S}^{d_i^a}\{|+\rangle, |-\rangle\}$ to achieve outcome r_i^a.

5: Generate and outputs $\widetilde{x} \in \{0,1\}^{n_p}$ using equation (4.8).

$$\widetilde{x}_j = \left(s_j^p + r_j^p\right) + \sum_{i:\boldsymbol{Q}_{ij}=1} \left(s_i^a + r_i^a\right) \tag{4.8}$$

Resource of Figure 4.3 in order to isolate the state generation phase π_A^1 and to examine an equivalent circuit based on teleportation.

The next transformation is to delay the first measurement of the EPR, pairs as implied in Figure 4.5. Since information about the measurement outcome r is not yet available to define $\boldsymbol{\Pi}$ and $\boldsymbol{A}$, the Client chooses $\boldsymbol{\Pi}$ and $\boldsymbol{A}$ at random which will then be corrected for by using these values to compute the measurement bases for the Client's half of the primary and ancillary EPR pairs.

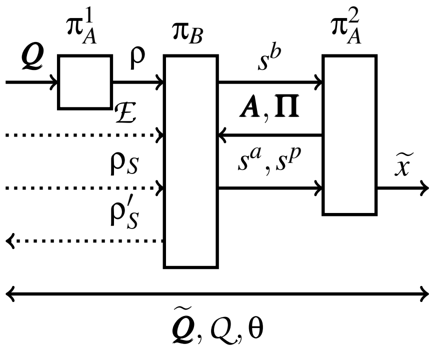
Finally, Figure 4.6 simply involves a rearrangement of the players in Figure 4.5 to match those in the simulator/distinguisher setting. The formal description of the protocol displayed by Figure 4.6 is seen in Protocol 4.1.3.

□

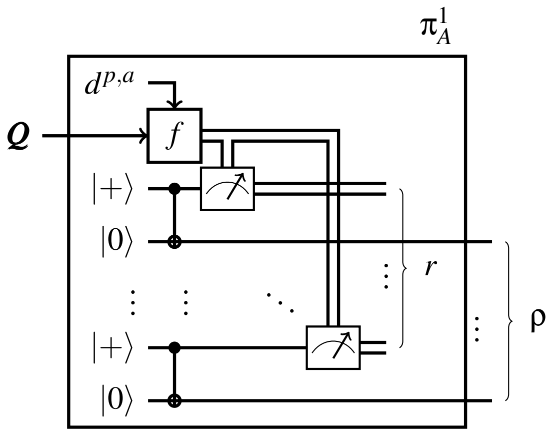
We can now be sure that our communication protocol is indistinguishable from an ideal resource of Figure 4.2 which performs an IQP computation without communicating any information to the Server which is not already public. Furthermore, this is proven in a composable framework and so can be used as part of future protocols as we will in Section 4.2. Notice also that Protocol 4.1.2 requires that the Client produce only basis states of the three Pauli operators: X, Y, and Z. This set of six states is reduced as compared to the eight required by the similar blind schemes in Section 1.6.1, of which only half are basis states of Pauli operators.

4.2 An IQP Hypothesis Test

We now have all the tools to form our IQP Hypothesis Test. Specifically, we ask the Server to sample from the output distributions of IQP circuits, and in so doing solve a problem which cannot be solved by a classical computer. The solution can, however, be



(a) The real protocol with the state generation phase of the protocol, π_A^1, isolated



(b) Expansion of state generation phase of the protocol, π_A^1. f represents the measurement angle calculation on one half of the EPR pairs.

Figure 4.4: **State preparation using teleportation**. See Protocol C.2.1 for details.

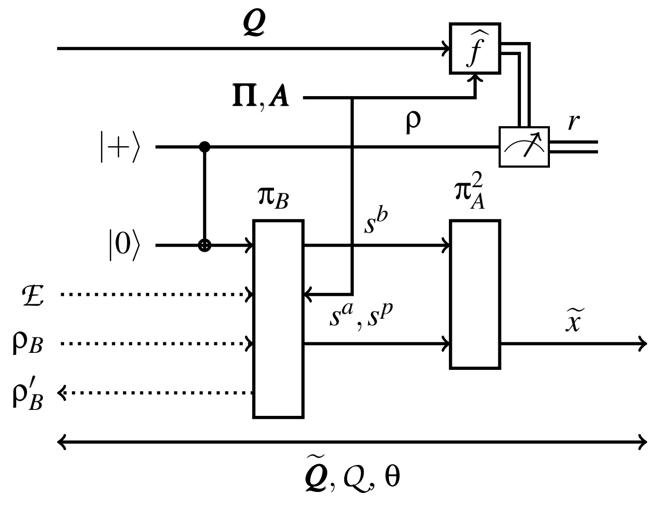


Figure 4.5: **Delayed state preparation measurement**. The real protocol with only one input qubit for simplicity. The Client sends random measurement instructions A, Π to the Server and delays the teleportation measurement until after the Server has sent the measurement outcomes $s = \{s^a, s^b, s^p\}$. Here $\widehat{f}$ represents the process of calculating measurement angles to be performed on one half of the EPR pair from equation (4.6) and equation (4.7), with details seen in Protocol C.2.2.

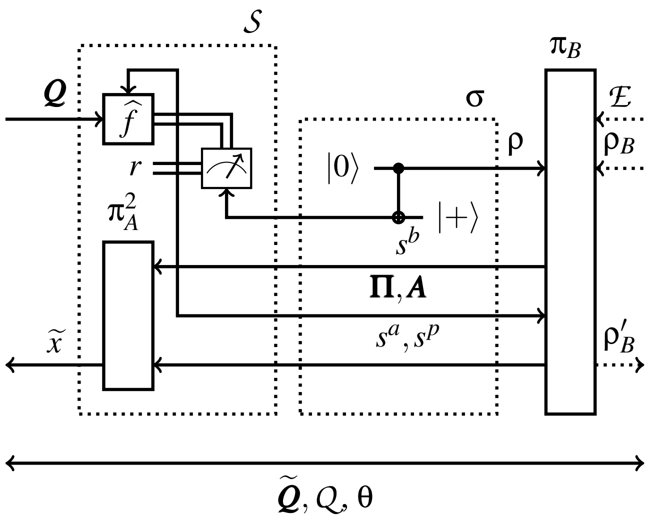


Figure 4.6: **The ideal resource $\mathcal{S}$ and the simulator σ for the malicious Server**. The simulator has no access to the private information $\boldsymbol{Q}$. A global distinguisher cannot tell the difference between this setting and the real protocol. Further details can be seen in Protocol 4.1.3.

verified by a classical client augmented with the ability to generate single qubits. To do so, we improve the security of the Shepherd-Bremner IQP Hypothesis Test discussed in Section 1.6.3 to prevent the attack on that scheme. As such, using the breakdown of the methodology used in Section 1.6.3, our approach is the following:

Hard Problem We ask the server to sample from the output distributions of X-programs generated from quadratic residue code generator matrices as described in Protocol 4.2.1. We conjecture that to do so with sufficient accuracy to produce outputs with sufficiently high bias is impossible classically without knowing the direction in which the bias is checked. We discuss the basis for this conjecture below.

Known Structure We will once again use the bias, discussed in Section 1.6.3 as a means to verify that the samples sent are from the desired distribution. Again the Client knows in which direction this bias should be checked, and so is able to do so efficiently.

Hidden Structure The direction in which the bias is checked is hidden using the blind delegated IQP protocol of Section 4.1.

In Protocol 4.2.1 we provide our IQP Hypothesis Test. Protocol 4.2.1 makes K repeated uses of the blind delegated IQP computation resource of Protocol 4.1.2, using it to run similar computations to those of Shepherd-Bremner IQP Hypothesis Test. Here K need not be any larger than is required by the Shepherd-Bremner IQP Hypothesis Test. In particular this means that, like the Shepherd-Bremner IQP Hypothesis Test of Protocol 1.6.1, Protocol 4.1.2 is sample efficient. This repeated use of Protocol 4.1.2 reveals the importance of proving its security in the Abstract Cryptography framework. Indeed by design such a proof allows us to combine uses of the blind delegated computing resource. In a slight deviation from the Shepherd-Bremner IQP Hypothesis Test, we use a different X-program for each iteration, rather than K samples from the same circuit. This is to ensure that the distribution over possible X-programs $\mathcal{Q}$, which is public, has a substantial support. By giving the Client limited quantum capabilities, we remove the computational assumption used in the Shepherd-Bremner IQP Hypothesis Test, and therefore provide unconditional security.

Our proof that Protocol 4.2.1 constitutes an information-theoretic security IQP Hypothesis Test therefore amounts to showing two things. The first is to show that the IQP computations implemented in Protocol 4.2.1 are essentially the same as those of the Shepherd-Bremner IQP Hypothesis Test scheme. Then the conjecture of their hardness made in [113] can be applied, while the output statistics, and namely the bias, can again be used for certification. The second is to show that when these computations are implemented blindly using Protocol 4.1.2, the distribution over possible X-programs, $\mathcal{Q}$, is sufficiently close to uniform so as not to reveal any information about the hidden $\boldsymbol{s}$. We present our proof formally here.

Theorem 4.2.1. *Protocol 4.2.1 presents an information-theoretically secure* IQP *hypothesis test.*

Proof. Let the matrix $\boldsymbol{Q_s}$, introduced on line 2 of Protocol 4.2.1, be the quadratic code generator matrix $\boldsymbol{Q_r}$ with a column of all ones appended to it. Since the vector with all elements set to one is in the quadratic residue code space, $\boldsymbol{Q_s}$ also generates the quadratic residue code. $\boldsymbol{Q_r}$ and $\boldsymbol{Q_s}$ can be compared by considering the respective figures, Figure 4.7 and Figure 4.8.

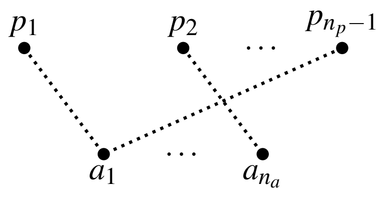
The vector $\boldsymbol{s} \in \{0,1\}^{n_p}$ with all zero entries, except the last which is set to one, is non-orthogonal to all rows of $\boldsymbol{Q_s}$. Hence, adhering to the notation here and of Section 1.6.3, $\mathcal{C}_{\boldsymbol{s}}$ is the quadratic residue code and equation (1.26), which we recall is

$$\mathbb{P}\left(X \cdot \boldsymbol{s}^T = 0\right) = \mathbb{E}_{\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{s}}}\left[\cos^2\left(\theta\left(n_s - 2 \cdot \#\boldsymbol{c}\right)\right)\right],$$

is equal to $\cos^2 \frac{\pi}{8}$.

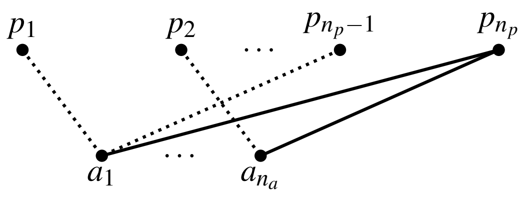
Let $\widehat{\boldsymbol{s}}^i \in \{0,1\}^{n_p - 1}$ be chosen uniformly at random, as on line 7 of Protocol 4.2.1. $\boldsymbol{A}$, defined in line 8 of Protocol 4.2.1, is the operation which adds the i^{th} column of $\boldsymbol{Q_s}$, to the last column of $\boldsymbol{Q_s}$ when $\widehat{\boldsymbol{s}}_i = 1$. We know that the resulting matrix, $\boldsymbol{Q} = \boldsymbol{Q_s}\boldsymbol{A}$, is also a generator matrix of the quadratic residue code as all the columns of $\boldsymbol{Q_s}$ are in the quadratic residue code space. We also know, from the discussion of Section 1.6.3, that all the rows of $\boldsymbol{Q}$ are non-orthogonal to $\boldsymbol{A}^{-1}\boldsymbol{s}^T$. As such $\mathcal{C}_{\boldsymbol{A}^{-1}\boldsymbol{s}^T}$, is the quadratic residue code space. Hence the bias of the X-program $\boldsymbol{Q}$ in the direction $\boldsymbol{A}^{-1}\boldsymbol{s}^T$ is $\cos^2 \frac{\pi}{8}$. This matrix may be visualised in Figure 4.9 and this fact is exploited in line 13 of Protocol 4.2.1.

From any $\boldsymbol{Q}$ we can make the extended IQP graph $\widetilde{\boldsymbol{Q}}$, which is the matrix $\boldsymbol{Q_r}$ with a column of -1 appended to the end. Observing Figure 4.10 may help to visualise this. We can now use the resource of Section 4.1.3 to perform a blind IQP computation. This conceals the known structure with information-theoretic security. This is true because, as a result of using the resource of Section 4.1.3, the Server learns only the



$$\boldsymbol{Q_r}$$

Figure 4.7: **Quadratic residue code generator matrix, $\boldsymbol{Q_r}$, and the graph that it describes**. This is illustrative and the connections in this image do not correspond to an actual quadratic residue code. This is alluded to by the dotted edges connecting the vertices.



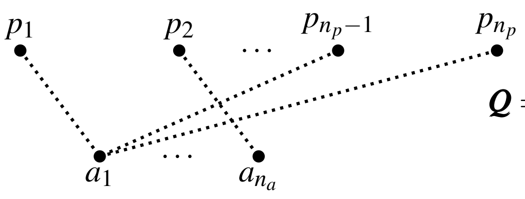
$$\boldsymbol{Q_s} = \left(\begin{array}{c|c} \boldsymbol{Q_r} & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \end{array} \right)$$

Figure 4.8: **Expanded matrix generating the quadratic residue code space**. $\boldsymbol{Q_s}$ is the matrix $\boldsymbol{Q_r}$ seen in Figure 4.7 with a column of all one appended to the last column. This corresponds to connecting all vertices a_i to the vertex p_{n_p}, which is alluded to by the solid lines.

distribution $\mathcal{Q}$ over the possible set of graphs $\boldsymbol{Q}$. By setting $\boldsymbol{Q} = \boldsymbol{Q_s A}$, Protocol 4.2.1 develops a bijection mapping $\widehat{\boldsymbol{s}} \in \{0,1\}^{n_p-1}$ to a unique matrix $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$. So $\mathcal{Q}$ is equivalent to the distribution from which $\widehat{\boldsymbol{s}}$ is drawn. In this case it is the uniform distribution over a set of size 2^{n_p-1}.

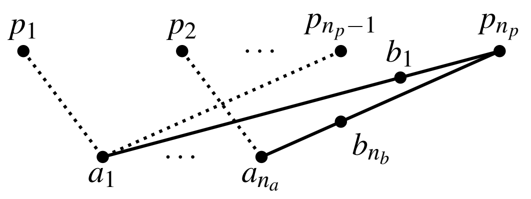
□

Note that this approach improves upon the Shepherd-Bremner IQP Hypothesis Test in two ways. Firstly it removes the conjecture they require in the case of hidden structure, replacing it with provable blindness. Secondly we are able to strengthen our belief in the very similar conjecture we make in the case of the hard problem. In particular, the 'best attempt' they give to support the conjecture that the bias cannot be made appropriately high without knowing the direction in which it will be checked requires the X-program to be implemented be known. As we are using a blind implementation



$$\boldsymbol{Q} = \left(\begin{array}{c|c} \boldsymbol{Q_r} & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \quad \bigoplus_{j=1}^{n_p-1} \ \widehat{\boldsymbol{s}}_j \boldsymbol{Q_r}_j \end{array} \right)$$

Figure 4.9: **Quadratic residue code generator matrix with randomised additional column**. Here $\boldsymbol{Q_r}_j$ is the j^{th} column of $\boldsymbol{Q_r}$. This correspond to potentially removing some of the edges connecting the vertices a_i and p_{n_p}, as seen in Figure 4.8, which we allude to with the dotted edges connected to p_{n_p}.



$$\widetilde{\mathbf{Q}} = \left(\begin{array}{c|c} & -1 \\ \mathbf{Q}_r & \vdots \\ & -1 \end{array} \right)$$

Figure 4.10: **An extended** IQP **graph of all possible $\mathcal{Q}$ of Figure 4.9**

Protocol 4.2.1 Information-theoretically secure IQP Hypothesis Test.

Client input: n_a prime such that $n_a + 1$ is a multiple of 8, $K \in \mathbb{Z}$.
Client output: $o \in \{0, 1\}$

```
 1: Set $n_p = \frac{n_a+1}{2} + 1$
 2: Take the quadratic residue code generator matrix $\boldsymbol{Q_r} \in \{0,1\}^{n_a \times (n_p - 1)}$
 3: Let $\boldsymbol{Q_s} \in \{0,1\}^{n_a \times n_p}$ be $\boldsymbol{Q_r}$ with a column of ones appended to the last column.
 4: Set $\widetilde{\boldsymbol{Q}}$ to be the matrix $\boldsymbol{Q_r}$ with a column of $-1$ appended.
 5: Let $\boldsymbol{s} \in \{0,1\}^{n_p}$ be the vector with entries all equal to zero with the exception of
    the last which is set to one.
 6: for all $i$ up to $K$ iterations do
 7:     Pick $\widehat{\boldsymbol{s}}^i \in \{0,1\}^{n_p - 1}$ chosen uniformly at random.
 8:     Define the matrix $\boldsymbol{A}^i \in \{0,1\}^{n_p \times n_p}$ according to equation (4.9).
```

$$\boldsymbol{A}^i_{l,m} = \begin{cases} 1 & \text{if } l = m \\ 0 & \text{if } l \neq m \text{ and } m < n_p \\ \widehat{\boldsymbol{s}}^i_l & \text{if } m = n_p \text{ and } l < n_p \end{cases} \tag{4.9}$$

```
 9:     Set $\boldsymbol{Q}^i = \boldsymbol{Q_s} \boldsymbol{A}^i$.
10:     Set $\mathcal{Q}$ to be the uniform distribution over all possible $\boldsymbol{Q}^j$ for different $\widehat{\boldsymbol{s}}^j$.
11:     Announce $\widetilde{\boldsymbol{Q}}$, $\mathcal{Q}$ and $\theta = \frac{\pi}{8}$. This public information is the same for each
    iteration, and so may also be announced at the beginning.
12:     Perform the IQP computation $\boldsymbol{Q}^i$ using Protocol 4.1.2; receiving $\widetilde{x}^i$.
13:     Test the orthogonality of the output $\widetilde{x}^i$ against $\left(\boldsymbol{A}^i\right)^{-1} \boldsymbol{s}^T$, setting $o^i = 0$ if it is
    not orthogonal and $o^i = 1$ if it is orthogonal.
14: end for
15: Return $o = 1$ if the fraction of $i$ with $o^i = 1$ is close to $\cos^2\left(\frac{\pi}{8}\right)$, and $o = 0$ other-
    wise.
```

of the X-program, even this best attempt is not possible in our case.

It should be the subject for future work to understand if this latter conjecture on the hardness of sampling strings with a sufficiently high bias can be broken. It seems the only attack on this conjecture which the Server could attempt would be to generate bitstrings which have a high bias in as many directions as possible. However, to the authors knowledge, no algorithm taking such an approach is known. Note that it is not unreasonable for the Client to believe that the Server did not spoof the IQP Hypothesis Test using a classical computer if no such spoofing technique has been published.

For the purpose of demonstrating quantum computational supremacy it is also of concern to understand if the particular class of X-programs being implemented are hard to sample from. Such hardness results are very rare, and it is not possible to say anything concrete here. However, we know of no reason that the distributions sampled from in Protocol 4.2.1 should be easier to sample from than general IQP distributions, and so no reason that the results of Section 1.5.2 should not apply. Note that this is identical to the argument made in the case of Shepherd-Bremner IQP Hypothesis Test in [113] and we, like them, leave this open to future proof or disproof.

It is pertinent finally to compare the information-theoretically secure IQP Hypothesis Test of Protocol 4.2.1, with Cross-Entropy Benchmarking and Heavy Output Generation Benchmarking as used in Chapter 3. While all three are sample efficient, the classical compute resource requirements of Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking grow exponentially with the circuit size. While this is not true for Protocol 4.2.1, it does require a means for the Client to send qubits to the Server. This comes with the added advantage that it can be used to benchmark much larger devices than can be benchmarked by Heavy Output Generation Benchmarking or Cross-Entropy Benchmarking.

4.3 Conclusion

We have presented a protocol that can be used by a limited quantum client, which is able to prepare single-qubit Pauli operator eigenstates, to delegate the implementation of IQP circuits to a powerful quantum server. By giving the Client these limited quantum abilities, we are able to delegate an IQP computation blindly. Our scheme is proven to be blind in the Abstract Cryptography framework, ensuring that it can be used as a component of other algorithms.

Indeed, we have used this blind delegation scheme to replace the unjustified computational complexity assumption in the Shepherd-Bremner IQP Hypothesis Test of [113]. We are therefore able to provide an information-theoretically secure IQP Hypothesis Test. This has the advantage of both repairing the fault in the Shepherd-Bremner IQP Hypothesis Test, and making our scheme less susceptible to future attacks.

Our protocol requires two rounds of measurements in order to make the appropriate corrections resulting from the blind creation of the state at the Server's side. Note that while the distributions being sampled from are IQP distributions, the resources required to implement them via the blind resource introduced are greater than the minimum one would need if blindness was not of concern. Indeed, as described in Section 1.5, only one round of measurements would typically be required. As such the hypothesis test presented here may provide a benchmark of only a subset of the computations that the device implementing them would be able to perform in theory. That said, for a small number of qubits, and short distances, computations requiring two rounds of measurements can be implemented even with present technology. However, at the time of writing, the technology to send single qubits is not sufficiently well developed to fully implement our scheme. In the meantime an interesting avenue of research would be the study of this protocol under realistic experimental errors, in view of a potential implementation. It would also be of interest to investigate the possibility of using tools such as classical client random RSP, introduced in Section 1.6.1, in order to reduce the Client to being purely classical. While implementations of random RSP by a classical client are currently very costly, it may be that the resource costs can be reduced in this specific case. This parallels the reduction in the number of different states that the Client in our IQP Hypothesis Test must produce, as compared to those required by VUBCQ.

Our information-theoretically secure IQP Hypothesis Test extends the reach of verification, characterisation and benchmarking of quantum technology into the regime where classical simulation becomes impossible. In this sense it complements the work of Chapter 2 and Chapter 3, which relied on classical simulation. In Chapter 5 we will explore how IQP circuits of sizes large enough to outperform classical simulation, which the work of this chapter allows us to benchmark, can be used to demonstrate quantum computational supremacy via an application motivated task.

Chapter 5

Quantum Learning Supremacy: Quantum Computational Supremacy with Computations of Practical Concern

In the preceding chapters of this thesis we have considered the verification, characterisation and benchmarking of NISQ technology. We have done so in order to facilitate improvements that lead to both demonstrations of quantum computational supremacy, and to the adoption of quantum technology to solve practical problems. However, treating these two themes separately risks neglecting their overlap, which is where the primary utility of NISQ technology will be found. This utility is namely in solving a practical problem which could not be solved with purely classical resources.

For example, proposals for demonstrations of quantum computational supremacy on NISQ technology typically involve sampling from the output distribution of random quantum circuits [61–63]. Such demonstrations are of great importance, both as an attestation of theoretical results, and as engineering milestones. However, it is not immediately clear that generating random samples is independently interesting. Conversely, protocols such as VQE [22, 23] and the Quantum Approximate Optimization Algorithm (QAOA) [16] use NISQ technology for practical tasks, but are less likely to provide demonstrations of quantum computational supremacy in the near-term. Combining these benefits, in this chapter we explore the utilisation of random samples in an application which may be of practical interest. We investigate instances of such applications where provable advantages of quantum computation can be exploited to define protocols that may demonstrate quantum computational supremacy.

We consider Quantum Machine Learning (QML), and generative modelling in particular. Generative modelling is the task of producing new samples from a probability distribution, called the target distribution, given a finite dataset $\left\{\boldsymbol{y}^i\right\}$ consisting of samples from said distribution. *Data-Driven Quantum Circuit Learning* is an approach taken to achieve this [90, 97]. As outlined in Figure 5.1, Data-Driven Quantum Circuit Learning is a hybrid quantum-classical algorithm with features common to many

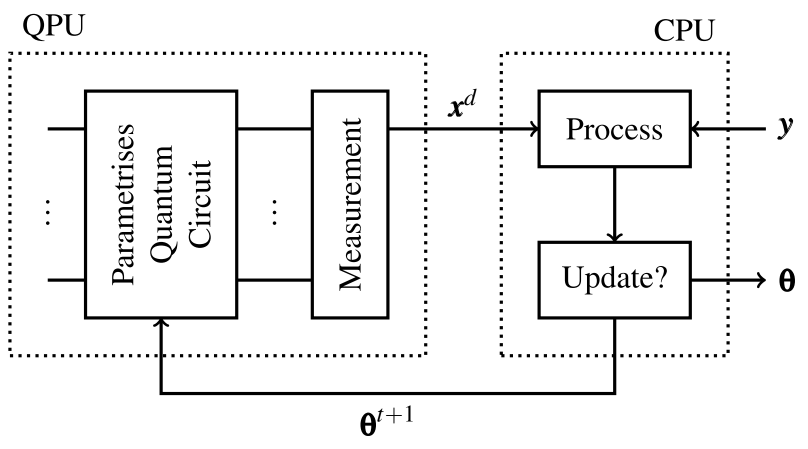


Figure 5.1: **Illustration of a Data-Driven Quantum Circuit Learning algorithm.** This is an instance of a hybrid quantum-classical algorithm, involving many rounds, called epochs, of interaction between a CPU and QPU. In each epoch t the QPU produces binary strings $\boldsymbol{x}^t$ by measuring the output from a PQC. The CPU processes $\boldsymbol{x}^t$, comparing it to samples $\boldsymbol{y}$ from the target distribution, to produce new parameters $\boldsymbol{\theta}^{t+1}$ to be used in the next epoch. This back and forth continues until $\boldsymbol{x}^t$ is sufficiently close to meeting some condition, or a maximum number of epochs is reached.

variational quantum algorithms. A parametrised quantum circuit (PQC) [14] is implemented and measured by a Quantum processing Unit (QPU), with the parameters of the circuit repeatedly updated over several epochs by an optimiser running on a Classical Processing Unit (CPU). These repeated updates continue until the measurements returned by the QPU meet some condition, or a maximum number of epochs is reached. Data-Driven Quantum Circuit Learning is a modular framework, which we exploit by choosing each component to suit our purpose.

Here we assume that the PQC in Figure 5.1 produces pure states, in which case measurement statistics are generated according to Born's measurement rule. This is to say, for a state $|\psi\rangle$, a measurement produces a sample $\boldsymbol{x}$ with probability $p(\boldsymbol{x}) = |\langle \boldsymbol{x}|\psi\rangle|^2$. Models of this form, where a pure state is produced by a circuit, are referred to as *Quantum Circuit Born Machines* (QCBM) [90, 256, 257].

In this chapter we ask if it is possible to define a Data-Driven Quantum Circuit Learning algorithm which has a provably superior performance over all classical alternatives. We formalise this question in Section 5.1 by defining a demonstration of quantum learning supremacy as an algorithm which answers our question in the affirmative. By adapting definitions from classical distribution learning [258], we can give a framework within which to prove a Data-Driven Quantum Circuit Learning algorithm should demonstrate quantum learning supremacy.[1] While such proofs exist, for example for the HHL linear equation solver [9] which is BQP-complete, a proven advantage for near-term QML algorithms is as yet out of reach.

In this work we aim to define an algorithm that might demonstrate quantum learning

[1]Provable guarantees are particularly important given recent QML algorithm 'dequantisations' [259].

supremacy specifically on NISQ technology. As such, in Section 5.2.1 we introduce the Quantum Circuit Ising Born Machine (QCIBM), which is a restricted form of a QCBM. In particular the PQC utilised by the QCIBM is sufficiently restricted to be suitable for NISQ devices, but sufficiently general to allow us to apply results on the quantum computational supremacy of IQP circuits, as discussed in Section 1.5. While QCBMs have found applications in, for example, finance [260], the practical applicaitons of QCIBMs are less well explored. However we regard their use in a demonstration of quantum learning supremacy to be a valuable application in itself.

In Section 5.2.2 we improve the differentiable training of the model over previous methods, which use the maximum mean discrepancy (MMD) [257], by using the Sinkhorn-Divergence [261]. In Section 5.2.3 we justify our conjecture that the QCIBM could be used to demonstrate quantum learning supremacy, leaving a formal proof for future work. Our justification includes showing that sampling from this model can not be simulated efficiently by any classical randomised algorithm, and that this holds for many circuit families encountered during training.

5.1 Quantum Learning Supremacy and Generative Modelling

The goal of a generative quantum machine learning algorithm is to efficiently mimic sampling from distributions in a given family. Intuitively this is what it means to 'learn' a distribution. If a generative quantum machine learning algorithm achieves this goal for some family of distributions, but it can be reasoned that there does not exist a classical learning algorithm achieving the same end, then the quantum algorithm can be said to have demonstrated quantum learning supremacy. We will use the terminology *quantum learner* and *classical learner* to refer learning algorithms with access to the corresponding computing power. Here we formalise quantum learning supremacy for distribution learning; modelling our definitions on those used in the theory of classical distribution learnability [258].

We consider learning classes of discrete distributions over binary vectors of length n. A Generator makes rigorous the notion of efficiently sampling from these distributions.

Definition 5.1.1 (Generator [258])**.** *A class of distributions, D_n has efficient* Generators *if for every distribution $\mathcal{D} \in D_n$, there is a generator $\mathsf{GEN}_{\mathcal{D}}$ which produces samples in $\{0,1\}^n$ according to the distribution $\mathcal{D}$, using polynomial resources. The generator may take a string of uniformly random bits of a size which grows polynomially in n, as input.*

Definition 5.1.1 allows for a Generator to be a classical or quantum circuit. In the case of a classical Generator a string of uniformly random bits is taken as input, and transformed into the randomness of $\mathcal{D}$ [258]. A quantum Generator could produce its own randomness and so may ignore the input string.

One may also wish to consider a generator producing quantum states [262]. We consider only classical generators to allow for a fair comparison between quantum and classical learners. However, one could imagine both a quantum device outputting a classical generator, and a classical device outputting the description of a quantum generator. Definition 5.1.1 may be generalised to this setting [263].

While we are predominately interested in efficient learning with a Generator, it is also of interest to define an Evaluator.

Definition 5.1.2 (Evaluator [258])**.** *A class of distributions, D_n has efficient* Evaluators *if for every distribution $\mathcal{D} \in D_n$, there is an evaluator $\mathsf{EVAL}_{\mathcal{D}}$ which produces the weight of an input $\mathbf{y} \in \{0,1\}^n$ under the distribution $\mathcal{D}$. This is to say, the probability of $\mathbf{y}$ according to $\mathcal{D}$. The Evaluator is* efficient *if it consumes resources growing polynomially with n.*

In the classical case it is possible to construct classes of distributions for which a generator can be learnt, but not an evaluator [258]. This parallels the comparative hardness of weak and strong simulation, discussed in Section 1.3.1. For example, the output probabilities of IQP circuits are #P-hard to compute [63], yet the distributions they produce can be sampled from efficiently by a quantum computer. In general an evaluator for a quantum circuit would be a strong simulator of it, and a generator would be a weak simulator.

As with the definitions of simulation in Section 1.3.1 it is useful to define the corresponding approximate versions of the tools introduced. We define an 'approximate generator', while an 'approximate evaluator' could be similarly defined.

Definition 5.1.3 ((d,ε)-Generator)**.** *Let d be a cost function, and $\varepsilon > 0$ a real number. Let $\mathcal{D}$ and $\mathcal{D}'$ be distributions over $\{0,1\}^n$. We say $\mathsf{GEN}_{\mathcal{D}'}$ is a (d,ε)-Generator for $\mathcal{D}$ if $d\left(\mathcal{D},\mathcal{D}'\right) \leq \varepsilon$.*

In contrast to [258], which was concerned with defining a 'good' generator to be one which achieves closeness relative to the Kullback-Leibler divergence, we have extended this notion to general cost functions. This is due to our desire to relate these ideas to the quantum circuit hardness results mentioned throughout this thesis, which typically strive for closeness in ℓ_1-norm distance.

We now have sufficient terminology to define a learnable class of distributions, illustrating the intuition in Figure 5.2.

Definition 5.1.4 ($(d,\varepsilon,\mathsf{C})$-Learnable)**.** *Take a metric d, a real number $\varepsilon > 0$, and a complexity class C. A class of distributions D_n is called* $(d,\varepsilon,\mathsf{C})$-learnable *(with a Generator) if there exists an algorithm $\mathcal{A} \in \mathsf{C}$ which, given $0 < \delta < 1$ as input, and given access to $\mathsf{GEN}_{\mathcal{D}}$ for any distribution $\mathcal{D} \in D_n$, outputs a (d,ε)-Generator for $\mathcal{D}$ with probability at least $1-\delta$. This is to say that $\mathcal{A}$ outputs $\mathsf{GEN}_{\mathcal{D}'}$ such that*

$$\mathbb{P}\left(d\left(\mathcal{D},\mathcal{D}'\right) \leq \varepsilon\right) \geq 1-\delta.$$

$\mathcal{A}$ should run in time $\mathsf{poly}(1/\varepsilon, 1/\delta, n)$*, and is called a* learning algorithm *for D_n*

In Definition 5.1.4, ε may, for example, be a function of the inputs to the learning algorithm. We may also wish to require a learnability definition which holds for all $\varepsilon > 0$. This would be too strong for our purposes as to claim quantum learning supremacy we only need to achieve closeness up to a constant ℓ_1-norm distance.

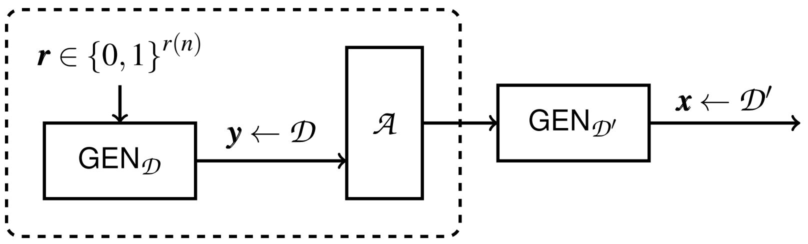

$\boldsymbol{r} \in \{0,1\}^{r(n)}$
$\mathsf{GEN}_{\mathcal{D}}$
$\boldsymbol{y} \leftarrow \mathcal{D}$
$\mathcal{A}$
$\mathsf{GEN}_{\mathcal{D}'}$
$\boldsymbol{x} \leftarrow \mathcal{D}'$

Figure 5.2: **Illustration of distribution learning**. The algorithm $\mathcal{A}$ is given access to $\mathsf{GEN}_{\mathcal{D}}$, which provides samples, $\boldsymbol{y} \leftarrow \mathcal{D}$, and must, with high probability, output an approximate generator for $\mathcal{D}$, $\mathsf{GEN}_{\mathcal{D}'}$. The target generator may take as input a string of random bits of size $r(n)$, a polynomial in n.

This framework is inspired by Probably Approximately Correct (PAC) learning [264, 265] but applies more closely to generative modelling. It is known that in certain cases, the use of quantum computers can be beneficial in PAC learning, but not generically [266]. It is therefore possible that there exist some classes of distributions which cannot be efficiently learned by classical computers, but which could be learned by quantum devices. In this spirit, we define what it would mean for a quantum algorithm to be superior to any classical algorithm for the problem of distribution learning.

Definition 5.1.5 (Quantum Learning Supremacy)**.** *An algorithm $\mathcal{A} \in$ BQP is said to have demonstrated the supremacy of quantum learning over classical learning if there exists a class of distributions D_n for which there exists d, ε such that D_n is $(d, \varepsilon, \mathsf{BQP})$-learnable, but D_n is not $(d, \varepsilon, \mathsf{BPP})$-learnable.*

A typical choice for d would be ℓ_1-norm distance, but one could imagine weaker definitions by using weaker cost functions. One may also be more restrictive and look for a demonstration of quantum learning supremacy by a class which was efficiently IQP-learnable, but not BPP-learnable, which may be more amenable for NISQ technology.

5.2 The Quantum Circuit Ising Born Machine

We now consider a special case of the formalism outlined in Section 5.1; namely generative modelling by a quantum learner with access to only NISQ technology. This will influence both the model that we define, and the power of the generators that we consider. Our choices with regards to these factors will also be guided by our desire to present a model that might be used to demonstrate quantum learning supremacy.

We will assume that $\mathsf{GEN}_{\mathcal{D}}$ in Figure 5.2, where $\mathcal{D}$ is the target distribution, is a quantum generator. This follows the line of reasoning that a proposal for demonstrating quantum learning supremacy should be built around a task that quantum computers are intrinsically better than classical devices at performing; sampling from quantum states.[2] This as a promising approach to demonstrating quantum learning supremacy with NISQ technology as sampling is particularly natural for these devices.[3]

[2]This is the same line of reasoning used when designing proposals for demonstrations of quantum computational supremacy such as RCS.

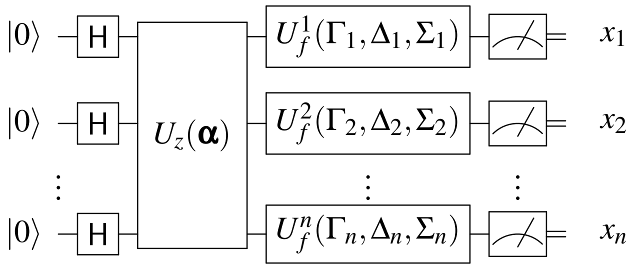
[3]Classical generators of the target distribution have been proposed as a means to demonstrate quan-

More specifically, our target distributions are a subset of the output distributions from IQP circuits. This allows us to significantly restrict our learning model, bringing it closer to realisation with NISQ technology. In particular, this section introduces the Quantum Circuit Ising Born Machine as a means to perform generative modelling. As with the Data-Driven Quantum Circuit Learning procedure outlined in Figure 5.1, a generic QCIBM comprises a PQC, drawing samples by measuring the quantum state it produces, with a classical optimisation loop used to learn a target distribution. In this section we outline and justify the choices we make for each of these components.

In Section 5.2.1 we introduce the PQC used, from which both IQP circuits and the shallowest depth version of QAOA circuits can be recovered. This is a particularly shallow learning model, with $O\left(n^2\right)$ entangling gates, and particularly limited gate set, as compared to more general purpose QCBMs [257], making it amenable NISQ devices. In Section 5.2.2 we consider the optimisation loop, and propose the use of the Sinkhorn-Divergence cost function to compare the target and model distributions. The model distribution is that produced by the QCIBM. The Sinkhorn-Divergence upper bounds the Maximum Mean Discrepancy (MMD), used to train QCBMs in the past, and interpolates between the MMD and the ℓ_1-norm distance. This means that it is at least as strong as the MMD, and possibly closer in strength to the ℓ_1-norm distance than is the MMD. This is interesting as we regard the ℓ_1-norm distance to be the 'gold-standard', due to its connection to results on quantum computational supremacy. Importantly, for some parameter values the Sinkhorn-Divergence can, like the MMD, but unlike the ℓ_1-norm distance, be calculated efficiently from samples. In Section 5.2.3 we conjecture that the QCIBM could be used to demonstrate quantum learning supremacy, outlining our reasoning. We discuss why this reasoning does not constitute a formal proof, and consider some of the hurdles in constructing such a proof.

5.2.1 Definition

The parametrised circuits and measurements used, which is to say the QPU in Figure 5.1, have the form


$|0\rangle$
H
$U_z(\boldsymbol{\alpha})$
$U_f^1(\Gamma_1,\Delta_1,\Sigma_1)$
x_1
$|0\rangle$
H
$U_f^2(\Gamma_2,\Delta_2,\Sigma_2)$
x_2
$|0\rangle$
H
$U_f^n(\Gamma_n,\Delta_n,\Sigma_n)$
x_n

where: $x_i \in \{0,1\}$; measurements are in the computational basis; $\boldsymbol{\alpha} = \{\alpha_j\}, \boldsymbol{\Gamma} = \{\Gamma_k\}$, $\boldsymbol{\Delta} = \{\Delta_k\}$, and $\boldsymbol{\Sigma} = \{\Sigma_k\}$ are sets of parameters, either fixed or to be trained; and $U_z(\boldsymbol{\alpha})$ and $U_f^k(\Gamma_i,\Delta_i,\Sigma_i)$ are defined in equation (5.1) and equation (5.2) respectively. In equation (5.1) each S_j is an element of the power set of $\{1,\dots,n\}$ and indicates the

tum learning supremacy on fault-tolerant devices [263], as we discuss in Section 5.3.

subset of qubits on which the j^{th} operator is applied.

$$U_z(\boldsymbol{\alpha}) := \prod_j U_z^j(\alpha_j, S_j) = \prod_j \exp\left(\mathrm{i}\alpha_j \bigotimes_{k\in S_j} \mathsf{Z}_k\right) \tag{5.1}$$

$$U_f^k(\Gamma_k, \Delta_k, \Sigma_k) := \exp\left(\mathrm{i}\left(\Gamma_k \mathsf{X}_k + \Delta_k \mathsf{Y}_k + \Sigma_k \mathsf{Z}_k\right)\right) \tag{5.2}$$

We will use the notation $U_f(\boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}) := \bigotimes_{k=1}^n U_f^k(\Gamma_k, \Delta_k, \Sigma_k)$.

When restricting to the case $|S_j| \leq 2$ the term in the exponential of equation (5.1) becomes an Ising Hamiltonian:

$$\mathrm{i}\sum_{i<j} J_{ij}\mathsf{Z}_i\mathsf{Z}_j + \mathrm{i}\sum_{k=1}^n b_k \mathsf{Z}_k \tag{5.3}$$

where we have separated *local* and *coupling* terms into separate sums. This inspires the name Quantum Circuit Ising Born Machine.

The samples $\boldsymbol{x} \in \{0,1\}^n$ are drawn from the distribution, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, parametrised by the set of angles, $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}\}$, and given by

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) := |\langle \boldsymbol{x}| U_f(\boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}) U_z(\boldsymbol{\alpha}) |+\rangle^n|^2. \tag{5.4}$$

We denote the above model by $\text{QCIBM}(\boldsymbol{\theta}) := \text{QCIBM}(\boldsymbol{\alpha}, \boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma})$.

We choose this structure in order to easily recover well known circuit classes. For example, to recover IQP circuits, discussed in Section 1.5, we simply need to generate the final layer of Hadamard gates (up to a global phase). To do so we set the angles of U_f to be

$$U_f^{\mathsf{IQP}}\left(\left\{\frac{\pi}{2\sqrt{2}}\right\}^n, \boldsymbol{0}, \left\{\frac{\pi}{2\sqrt{2}}\right\}^n\right) = \bigotimes_{k=1}^n \exp\left(\frac{\mathrm{i}\pi}{2\sqrt{2}}(\mathsf{X}_k + \mathsf{Z}_k)\right) = \mathrm{i}\mathsf{H}^n.$$

Then equation (5.1) defines the computation, and can be recognised as an X-program as in Definition 1.5.2, but for the change to using the equivalent Z gates between layers of H gates. It is similarly possible to recover the shallowest depth version of QAOA [16]. Both of these classes of circuits are known to be routes to demonstrate quantum computational supremacy [63–65, 267], which we utilise in this work.

5.2.2 Training

Here we consider training methods for the QCIBM. Recall, as illustrated in Figure 5.1, that the training procedure is a hybrid of classical and quantum computation. The quantum component is limited to just the model itself, making the approach favourable towards NISQ devices. The role of the CPU is to update parameters $\boldsymbol{\theta}$ of the QCIBM, in order to minimise the difference between $p_{\boldsymbol{\theta}}$, the model distribution as defined in equation (5.4), and π, the target distribution. This difference between distributions is measured by a cost function, and the optimisation procedure we use to minimise a cost function is stochastic gradient descent.

Gradient-free [90, 268] and gradient-based [91, 97, 164, 257] methods have been used to train QCBMs. Gradient based optimisers prove advantageous, both in the accuracy they achieve, and in the sampling cost [257]. During such training procedures, parameters, θ_k, are updated at each epoch, t, according to the rule $\theta_k^{t+1} \leftarrow \theta_k^t - \eta \partial_{\theta_k} \mathcal{L}_B$. Here $\partial_{\theta_k} \mathcal{L}_B$ is the gradient of a cost function $\mathcal{L}_B$ with respect to θ_k. The parameter η is the learning rate, which may be fixed or time and gradient dependent, and controls the speed of the descent.

In this section we assess two existing cost functions, and discuss their application to training the QCIBM. When assessing cost functions we are concerned with the computational resources required to calculate and minimise them, and, uniquely to this work, the implications of training using them on demonstrations of quantum learning supremacy. At a practical level, a cost function is well suited for the task of training the QCIBM via stochastic gradient descent if both the cost function and its gradient can be efficiently computed from samples from $p_{\boldsymbol{\theta}}$ and π. Here efficiency is measured both by sample and computational complexity. A good cost function would also be sensitive to differences between $p_{\boldsymbol{\theta}}$ and π, which we will assess using the ℓ_1-norm distance, $\ell_1(p_{\boldsymbol{\theta}}, \pi)$, as a benchmark. We do so as the ℓ_1-norm distance is both highly sensitive to differences between distributions, and because it relates to many theoretical results pertaining to quantum computational supremacy. The ℓ_1-norm distance cannot be calculated efficiently from samples in general, and so it not used during training.[4]

The ideal cost function would be efficient to train with, while still being of a similar sensitivity to the ℓ_1-norm distance. Note that a cost function with this combination of properties would provide a route towards a certifiable demonstration of quantum learning supremacy. In particular if a cost function, sensitive in the way outlined, can be efficiently calculated, then it can be used to certify if the model distribution is closer to the target distribution than any model distribution that could be produced by a classical computer.

MMD

The first efficient gradient-based method used to train a QCBM utilised the *squared maximum mean discrepancy* (MMD) as a cost function [257, 269], which is given by

$$\mathcal{L}_{\text{MMD}}(p_{\boldsymbol{\theta}}, \pi) := \underset{\substack{\boldsymbol{x} \leftarrow p_{\boldsymbol{\theta}} \\ \boldsymbol{y} \leftarrow p_{\boldsymbol{\theta}}}}{\mathbb{E}} (\kappa(\boldsymbol{x}, \boldsymbol{y})) + \underset{\substack{\boldsymbol{x} \leftarrow \pi \\ \boldsymbol{y} \leftarrow \pi}}{\mathbb{E}} (\kappa(\boldsymbol{x}, \boldsymbol{y})) - 2\underset{\substack{\boldsymbol{x} \leftarrow p_{\boldsymbol{\theta}} \\ \boldsymbol{y} \leftarrow \pi}}{\mathbb{E}} (\kappa(\boldsymbol{x}, \boldsymbol{y})). \tag{5.5}$$

κ is a *kernel function* which measures the similarity between points in the sample space $\{0,1\}^n$ [270].[5] A popular choice for κ is the *Gaussian mixture kernel* [257], given by

$$\kappa_G(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{c} \sum_{i=1}^{c} \exp\left(- \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}{2\sigma_i} \right).$$

[4] Very often the KL-divergence, defined in equation (1.27), is used to compare distributions during machine learning tasks. The KL-divergence upper bounds the ℓ_1-norm distance and so it is similarly sensitive to differences between distributions. Unfortunately neither its gradient, nor the KL-divergence itself, can be evaluated efficiently when training parametrised circuits [257].

[5] Recent works on the near-term advantage of using quantum computers in QML have explored quantum kernels, which can be evaluated on a quantum computer [4, 165, 271].

The parameters, σ_i, are *bandwidths* which determine the scale at which the samples are compared, and $\|\cdot\|_2$ is the ℓ_2-norm distance. The MMD with Gaussian kernels is zero if and only if the model distribution matches the target distribution [269], guaranteeing a *faithful* solution.

The gradient of the MMD is required in order to train the QCIBM. Given a PCQ composed of unitary gates of the form $V_k(\theta_k) = \exp(i\theta_k\Sigma_k)$ where $\Sigma_k^2 = \mathbb{I}$, as is the case in for the QCIBM defined in Section 5.2.1, the gradient of the MMD with respect to the kth parameter is given by [4, 257]

$$\frac{\partial \mathcal{L}_{\text{MMD}}}{\partial \theta_k} = 2\underset{\substack{\boldsymbol{a}\leftarrow p_{\boldsymbol{\theta}^-}\\ \boldsymbol{x}\leftarrow p_{\boldsymbol{\theta}}}}{\mathbb{E}}(\kappa(\boldsymbol{a},\boldsymbol{x})) - 2\underset{\substack{\boldsymbol{b}\leftarrow p_{\boldsymbol{\theta}^+}\\ \boldsymbol{x}\leftarrow p_{\boldsymbol{\theta}}}}{\mathbb{E}}(\kappa(\boldsymbol{b},\boldsymbol{x})) - 2\underset{\substack{\boldsymbol{a}\leftarrow p_{\boldsymbol{\theta}^-}\\ \boldsymbol{y}\leftarrow \pi}}{\mathbb{E}}(\kappa(\boldsymbol{a},\boldsymbol{y})) + 2\underset{\substack{\boldsymbol{b}\leftarrow p_{\boldsymbol{\theta}^+}\\ \boldsymbol{y}\leftarrow \pi}}{\mathbb{E}}(\kappa(\boldsymbol{b},\boldsymbol{y})). \tag{5.6}$$

Here $p_{\boldsymbol{\theta}^\pm}$ are output distributions generated by the same PQC with parameters $\theta_i^\pm = \theta_i \pm \frac{\pi}{2}\delta_{i,k}$, where $\delta_{i,k}$ is the Kronecker delta and $\boldsymbol{\theta}$ is the original set of parameters. This approach to calculating derivatives is called the *parameter shift rule* [97, 272].

Using collections of samples $\{\boldsymbol{x}^1,\dots,\boldsymbol{x}^N\}$ and $\{\boldsymbol{y}^1,\dots,\boldsymbol{y}^M\}$, where $\boldsymbol{x}^i \leftarrow p_{\boldsymbol{\theta}}$ and $\boldsymbol{y}^j \leftarrow \pi$, the MMD can be estimated by

$$\tilde{\mathcal{L}}_{\text{MMD}} = \frac{1}{N(N-1)}\sum_{i\neq j}^{N}\kappa(\boldsymbol{x}^i,\boldsymbol{x}^j) + \frac{1}{M(M-1)}\sum_{i\neq j}^{M}\kappa(\boldsymbol{y}^i,\boldsymbol{y}^j) - \frac{2}{MN}\sum_{i,j}^{M,N}\kappa(\boldsymbol{x}^i,\boldsymbol{y}^j)$$

which converges to equation (5.5) by the law of large numbers [4, 269]. The gradient can similarly be estimated by replacing the expectations in equation (5.6) by their empirical value [4, 257].

Importantly the rate of convergence of $\tilde{\mathcal{L}}_{\text{MMD}}$ to $\mathcal{L}_{\text{MMD}}$ depends inversely on the number of samples, and in particular [273]

$$|\sqrt{\tilde{\mathcal{L}}_{\text{MMD}}(p_{\boldsymbol{\theta}N},\pi_M)} - \sqrt{\mathcal{L}_{\text{MMD}}(p_{\boldsymbol{\theta}},\pi)}| \leq O\left(\frac{1}{\sqrt{N}}+\frac{1}{\sqrt{M}}\right).$$

The independence of this difference on the number of qubits makes the calculation of the MMD efficiently scalable. However, the MMD has the unfortunate drawback that it only provides a lower bound on the ℓ_1-norm distance [273]. As such minimising the MMD cannot be connected to many of the results on quantum computational supremacy which we have discussed throughout.

Sinkhorn-Divergence

The second cost function we consider is the *Sinkhorn-Divergence* (SHD) [261, 274–276] which has has not previously been used during gradient based training of QCBMs. The SHD is defined as

$$\mathcal{L}^{\varepsilon}_{\text{SHD}}(p_{\boldsymbol{\theta}},\pi) := \text{OT}^c_\varepsilon(p_{\boldsymbol{\theta}},\pi) - \frac{1}{2}\text{OT}^c_\varepsilon(p_{\boldsymbol{\theta}},p_{\boldsymbol{\theta}}) - \frac{1}{2}\text{OT}^c_\varepsilon(\pi,\pi) \tag{5.7}$$

where the regularised *Optimal Transport* (OT) is

$$\text{OT}^c_\varepsilon(p_{\boldsymbol{\theta}},\pi) := \min_{U\in\mathcal{U}(p_{\boldsymbol{\theta}},\pi)}\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}\sum_{\boldsymbol{y}\in\{0,1\}^n} c(\boldsymbol{x},\boldsymbol{y})U(\boldsymbol{x},\boldsymbol{y}) + \varepsilon\text{KL}(U,p_{\boldsymbol{\theta}}\otimes\pi)\right). \tag{5.8}$$

Here $\varepsilon \geq 0$ is a regularisation parameter, $c(\boldsymbol{x},\boldsymbol{y})$ is a Lipschitz function, and $\mathcal{U}(p_{\boldsymbol{\theta}},\pi)$ is the set of all joint distributions whose marginals with respect to $\boldsymbol{x},\boldsymbol{y}$ are $p_{\boldsymbol{\theta}}(\boldsymbol{x}), \pi(\boldsymbol{y})$ respectively.

For the two extreme values of ε, we recover unregularised OT, and the MMD [261, 274, 275].

$$\begin{aligned}
\varepsilon = 0: \qquad & \mathcal{L}^{0}_{\text{SHD}}(p_{\boldsymbol{\theta}},\pi) = \text{OT}^{c}(p_{\boldsymbol{\theta}},\pi) \\
& \qquad = \min_{U\in\mathcal{U}(p_{\boldsymbol{\theta}},\pi)} \left(\sum_{\boldsymbol{x}\in\{0,1\}^n} \sum_{\boldsymbol{y}\in\{0,1\}^n} c(\boldsymbol{x},\boldsymbol{y})\, U(\boldsymbol{x},\boldsymbol{y}) \right) \qquad (5.9) \\
\varepsilon \to \infty: \qquad & \mathcal{L}^{\varepsilon}_{\text{SHD}}(p_{\boldsymbol{\theta}},\pi) \to \text{MMD}(p_{\boldsymbol{\theta}},\pi) \quad \text{with} \quad \kappa(\boldsymbol{x},\boldsymbol{y}) = -c(\boldsymbol{x},\boldsymbol{y})
\end{aligned}$$

Importantly for any value of ε, $\mathcal{L}^{\varepsilon}_{\text{SHD}}(p_{\boldsymbol{\theta}},\pi)$ is positive, and equal to 0 if and only if $p_{\boldsymbol{\theta}} = \pi$ [276]. Intuitively the solution of the 'optimal transport problem' of equation (5.9) gives the optimal way to move, or transport, probability mass from one distribution to another. This gives a means of determining the similarity of distributions with $c(\boldsymbol{x},\boldsymbol{y})$ being the 'cost' of transporting an individual 'point', $\boldsymbol{x}$, to another point $\boldsymbol{y}$.

Ideally we would use OT itself to train generative models as it provides an upper bound for the ℓ_1-norm distance. Unfortunately, OT has high computational cost, and sample complexity that grows exponentially in the number of qubits [277]. For this reason, the SHD was proposed to interpolate between OT and the MMD [261, 274–276]. The hope is that this allows for the exploitation of the advantages of both the MMD and the OT, namely the low sample complexity and the greater power respectively.

Both the SHD [261] and its gradient [276] can be approximated from samples. In the later case the process of doing so includes the use of the same shifted circuits used to calculate the gradient of the MMD. The mean error between $\mathcal{L}_{\text{SHD}}$ and its estimator $\hat{\mathcal{L}}_{\text{SHD}}$ for n qubits, computed using M samples, depends on ε in general, while in the special case that $\varepsilon = O(n^2)$, this error scales as [278]

$$\mathbb{E}\left(|\mathcal{L}^{O(n^2)}_{\text{SHD}} - \hat{\mathcal{L}}^{O(n^2)}_{\text{SHD}}| \right) = O\left(\frac{1}{\sqrt{M}} \right).$$

This is the same sample complexity as the MMD [273], but exponentially better than that of unregularised OT, which scales as $O\left(\frac{1}{M^{\frac{1}{n}}} \right)$ [277]. Therefore, we can choose an optimal theoretical value for the regularisation, such that $\mathcal{L}_{\text{SHD}}$ is far enough from OT to be efficiently computable, but perhaps still retains its favourable properties. Unfortunately while bounds on the ℓ_1-norm distance by the Sinkhorn-Divergence can be found these are only known for values of ε that require an exponential sample complexity.

5.2.3 Hardness

A demonstration of quantum learning supremacy, as outlined in Definition 5.1.5, requires the existence of both a class of distributions which is not learnable by any algorithm $\mathcal{A} \in \mathsf{BPP}$, and an algorithm $\mathcal{B} \in \mathsf{BQP}$ which can learn the class of distributions. We first intuitively show the existence of the former. Suppose that the class of distributions produced by IQP circuits is $\left(\ell_1, \frac{1}{192}, \mathsf{BPP}\right)$-learnable, and that $\mathcal{A} \in \mathsf{BPP}$ is the algorithm which learns it. Notice that if a class of distributions is $\left(\ell_1, \frac{1}{192}, \mathsf{BPP}\right)$-learnable by an algorithm $\mathcal{A}$, then $\mathcal{A}$ can certainly weakly simulate this class of distributions within a ℓ_1-norm distance of $\frac{1}{192}$ by using the learnt generator. Recall from Theorem 1.5.2 that if all IQP circuits could be weakly classically simulated to within a ℓ_1-norm distance of $\frac{1}{192}$ by an algorithm $\mathcal{A} \in \mathsf{BPP}$ then PH would collapse. Assuming there is not such collapse of PH, we arrive at a contradiction, and conclude that the class of distributions produced by IQP circuits is not $\left(\ell_1, \frac{1}{192}, \mathsf{BPP}\right)$-learnable.

For the purposes of demonstrating quantum learning supremacy, it remains to give an algorithm $\mathcal{B} \in \mathsf{BQP}$ by which the class of distributions produced by IQP circuits is learnable. In particular we would like to show that the training procedure for the QCIBM described in Section 5.2.1 and Section 5.2.2 is such an algorithm. Unfortunately we are not able to formally show anything this strong.

In order to demonstrate that the output distributions of IQP circuits are $\left(\ell_1, \frac{1}{192}, \mathsf{BQP}\right)$-learnable by the QCIBM trained via gradient descent we would have to both guarantee: that an IQP distribution could be learnt to within a bound on the cost function used; and that the cost function itself bounds the ℓ_1-norm distance. The suggestion to use the SHD rather than the MMD makes progress in this direction, and presents a promising avenue of investigation. In particular it reveals the trade-off between sample complexity and sensitivity to differences in distributions, which may be exploited to achieve these objectives. However at present we can neither guarantee that the SHD will be bounded at the end of the learning procedure, nor that the SHD for those values of ε for which the SHD is efficient to compute will bound the ℓ_1-norm distance.

This contrasts with the fact that IQP circuits to which the hardness results of Section 1.5.2 are applicable can be constructed from particular parameter setting of the QCIBM. Here lies the difference between showing that a model is more expressive than any classical model [164], which is to say that it can in principle be used to sample from a larger class of distributions, and showing that it could learn a hard distribution. For the remainder of this section we will instead reason as to why the learning procedure we have described for the QCIBM in Section 5.2.1 and Section 5.2.2 is likely to be impossible to simulate using a classical device. This is to say that the circuits encountered during the gradient based training of the QCIBM would be hard to weakly simulate.

Note that while the learning procedure being hard classically is not a sufficient condition to demonstrate quantum learning supremacy, it is likely a necessary one. For example take an algorithm $\mathcal{B} \in \mathsf{BQP}$ which demonstrates quantum learning supremacy by learning all IQP distributions to within a bounded ℓ_1-norm distance. Suppose that an algorithm $\mathcal{A} \in \mathsf{BPP}$ can reproduce the distributions produced by $\mathcal{B}$ to within a bounded ℓ_1-norm distance. This includes in particular the distributions learnt by $\mathcal{B}$, and so all IQP distributions. Then, if these bounds are sufficiently small, $\mathcal{A}$ could weakly simulate all IQP distributions to within ℓ_1-norm distance of $\frac{1}{192}$. By Theorem 1.5.2 this collapses the PH and so contradicts our supposition that a classical algorithm can reproduce the learning procedure of $\mathcal{B}$.

To demonstrate that the learning procedure is hard to simulate classically, we consider the relation between the learning procedure for the QCIBM and hardness results for

IQP as discussed in Section 1.5.2.[6] It is clear from Theorem 1.5.1 that efficiently weakly sampling with multiplicative error from the output distributions of all circuits corresponding to some parameter setting of the QCIBM should be impossible for a classical device. It remains to show that the circuits which are encountered during the training procedure are also of this type. Without imposing restrictions on the parameter values which may be used by the QCIBM this too is out of reach.

However, it can be shown that the set of parameters is sufficiently large that we can be confident about the hardness of the training procedure. For example, recall from Theorem 1.5.1 and its proof that IQP circuits, with $J_{ij} = b_k = \frac{\pi}{8}$ for all i, j, k in (5.3), are hard to simulate in the worst case. This was shown by demonstrating that these circuits are universal for BQP under post selection. This is insufficient for our purposes as, if the training procedure encountered all parameter values with equal probability, it would encounter such circuits with probability 0. If the remaining parameters resulted in circuits which could be weakly classically simulated this would be of great concern.

In fact it is the case that for gates of the form $e^{iJ_{ij}\mathsf{Z}\otimes\mathsf{Z}}$, $e^{ib_k\mathsf{Z}}$ with $b_k = J_{ij}$ of either of the forms

$$\begin{array}{rl} \dfrac{(2l+1)\pi}{8m} & l, m \text{ integers} \\ 2\nu\pi & \nu \in [0,1) \text{ irrational} \end{array} \tag{5.10}$$

the resulting gate set would be universal under post selection [170]. In the case of parameters of the form $\frac{(2l+1)\pi}{8m}$ it is clear that $\frac{\pi}{8}$ can be recovered from repeated applications of the gates. The result then follows from Theorem 1.5.1. In the case of irrational parameter values, intuitively this follows since $2m\nu\pi \pmod{2\pi}$ is distributed uniformly in $[0, 2\pi)$, and so we can find an approximation of $\frac{\pi}{8}$ with an additive error with some integer $m = O\left(\frac{1}{\varepsilon}\right)$ number of repetitions of the gate [28]. While we have discussed a homogeneous choice of angle here, if we chose inhomogeneous angles, each one of the form of equation (5.10), circuits built from the resulting gate set would also be hard in the worst case. This follows as any one of the gates $e^{iJ_{ij}\mathsf{Z}\otimes\mathsf{Z}}$ and $e^{ib_k\mathsf{Z}}$ would be sufficient to recover $e^{i\frac{\pi}{8}\mathsf{Z}\otimes\mathsf{Z}}$ and $e^{i\frac{\pi}{8}\mathsf{Z}}$ using repeated applications.

Importantly, in the case of the irrational parameters, the set of values has measure 1 and so the probability of encountering them is high. This gives us confidence that a large proportion of the circuits encountered during the training procedure will be hard to weakly simulate with multiplicative error in the worst case.

5.3 Conclusion

In this chapter we have discussed the prospect of demonstrating quantum computational supremacy, via a practically motivated task, on NISQ technology. In particular we have considered generative modelling; formalising a demonstration of quantum computational supremacy through generative modelling as quantum learning supremacy. In Section 5.2 we introduced the QCIBM. We proposed stochastic

[6]The relationship between the QCIBM and the shallowest depth version of QAOA, as discussed in Section 5.2.1, also allows us to connect to yet further hardness results [4, 267].

gradient descent using the SHD as a means by which to train it, and discussed the advantages of doing so over previous approaches. We argued this could be an approach to demonstrating quantum learning supremacy on NISQ technology, but are only able to show a weaker result. This is namely that, with high probability, the circuits encountered during the training of the QCIBM are impossible to weakly simulate with a classical device up to multiplicative error in the worst case. While we are not able to formally provide a proposal to demonstrate quantum learning supremacy on NISQ technology, in other settings proposals for demonstrations of quantum computational supremacy via machine learning are possible. Conversely, there are results that isolate the difficulty in demonstrating quantum learning supremacy on NISQ technology. For the remainder of this chapter we discuss and compare results in these two directions, and consider the avenues of future research which they open up.

Using the framework introduced in Section 5.1, a route to demonstrating quantum learning supremacy with fault-tolerant universal devices has been proposed [263]. Ref.[263] provides both a class of discrete probability distributions which is provably impossible to efficiently learn with a classical generative modelling algorithm, and an efficient quantum learner of the class. This class of distributions is built from a collection of pseudorandom functions (PRF), where a PRF is a deterministic function of a key and an input which is indistinguishable from a truly random function of the input [15]. That a PRF is indistinguishable from a random function is vital, and in the case of the PRFs introduced in [263] is conditional on the Decisional Diffie-Hellman assumption (DDH).[7] From any collection of PRFs it is possible to construct a class of probability distributions for which no efficient classical learner exists [258]. However, given the key of the PRF these distributions admit an efficient classical generator. The DDH assumption can be broken using the quantum algorithm for discrete logarithm [7]. As a result the functions used to construct the class of distributions are not pseudorandom from the point of view of a quantum learner. A quantum learner which can learn the key using random samples can then be constructed [263]. As mentioned, having the key is sufficient to produce a generator for the distributions.

It is noteworthy that the generator described in [263] is classical, which deviates from our use of a quantum generator for the target distribution in Section 5.2. Further, the generator outputted by the quantum learner at the end of the protocol is exact, and can be found from one sample. This avoids the problem of finding a cost function that can be reliably trained, and which bounds the ℓ_1-norm distance. However, this also suggests that there may be room to reduce the resource requirements towards those of NISQ devices if only an approximate generator is required, and a polynomial number of samples is made use of. This would be one interesting direction of future work.

Besides distribution learning, which itself has applications in finance [260], learning boolean functions may also prove a beneficial application of quantum computers. In the classical case one wishes to learn a function $c : \{0,1\}^n \rightarrow \{0,1\}$ given random example input output pairs $(x, c(x))$ [264]. Quantum learners instead receive copies of

[7]This assumption is roughly that the product of two elements of a cyclic group looks like a random element of that group.

a superposition of labelled examples [262]:[8]

$$\frac{1}{\sqrt{2^n}} \sum_{x \in (0,1)^n} |x\rangle |c(x)\rangle \tag{5.11}$$

This formalism does not adhere to our definition of quantum learning supremacy in Section 5.1. However classes of boolean functions which can be learned from quantum examples, but not classical ones are known [262, 279, 280].

In principle boolean function learning allows a quantum learner to perform complex operations on many copies of the quantum example. This is not amenable to NISQ technology. Quantum Statistical Query (QSQ) learning is a restriction of this model [17]. In particular, a learner selects a set of observables to measure on the quantum example, and receives expectation values of these measurements. In this sense the learner receives 'statistics' of the quantum example. This setting is more representative of cloud based NISQ technology as the state will not be held or manipulated by the learner. Further, if there exists a QSQ learner of a class of functions, then there exists a noisy QSQ learner for that class [17]. This is to say the class of functions can be learnt even if the quantum example is subject to bit flip errors with some probability, further aligning the model with the power of NISQ technology.

It is known that there are functions which can be learnt by a QSQ learner but which provably cannot be learnt in the classical statistical query learning model [281]. For some classes of these functions it is also the case that there is no known learning algorithm even when given the classical examples directly, rather than statistics of the distribution. The circuits implementing the boolean functions for which these results are known appear less well suited to NISQ technology than random circuits, but it should certainly be the subject of future work to understand if these results on learning boolean functions can be transformed into the distribution learning setting of Section 5.1.

While these results present promising directions for further investigation, others reveal the pitfalls to be avoided when doing so. For example it is known that it is impossible to perform the seemingly related task of efficiently certifying that a distribution, from which samples can be drawn, is equivalent to a known IQP distribution [282].[9] More precisely, suppose $\mathcal{P}$ is the output distribution from a randomly chosen IQP circuit, and that a complete description of $\mathcal{P}$ is known. Then, with high probability, exponentially many samples from a distribution $\mathcal{Q}$ would be required to determine if $\mathcal{Q} = \mathcal{P}$ or $\ell_1(\mathcal{P}, \mathcal{Q}) > \varepsilon$, for some given ε, irrespective of the computing power of the certifier.[10] While the task of efficient certification and that of constructing an efficient learner for a class of distributions appear related, the reduction of a learner to a means of performing certification is not clear, but would be of great interest to uncover.

Finally, it is also known that in general training variational quantum algorithms is NP-hard [284]. This is shown by encoding the NP-hard MaxCut problem into the

[8]The state in equation (5.11) is a *uniform quantum example*. In principle each term in the sum may be weighted by $\sqrt{\mathcal{D}(x)}$ for some probability distribution $\mathcal{D}$.

[9]This is not only true for IQP circuits, but for any anticoncentrating classes of distributions, which includes the output distributions of BosonSampling circuits [61], and those of RCS [66].

[10]This task is referred to as *identity testing* [283].

classical optimisation step of several variational quantum algorithms. It is not clear if this encoding applies in the case of the approach outlined here, and this would be an interesting avenue of exploration. However, worst case hardness results such as this should not discourage the exploration of quantum learning supremacy, and indeed we have seen that quantum learning supremacy is possible in particular cases [263].

Chapter 6

Conclusion and Outlook

"Predictions are very difficult, especially about the future"

— Niels Bohr

At the outset, it was the goal of this thesis to:

Explore the QCVV protocols that are appropriate as quantum technology develops. In doing so, understand which applications of quantum technology are the most fruitful at each stage of the progress of technology.

In pursuing this goal we introduced schemes to guide the furtherance of quantum technology, to identify the applications to which different quantum technologies are best suited, and to both perform and certify demonstrations of quantum computational supremacy. Starting with small devices with only a few qubits, in Chapter 2 we proposed and exemplified a methodology for the use of classical simulation to predict the behaviour of larger devices, and the impact of changes to their design. In Chapter 3 we presented and utilised a suite of benchmarks to measure the performance of existing multi-qubits devices, choosing circuits so as to assess the applications to which the devices explored are well adapted. Once devices large enough to demonstrate quantum computational supremacy are available, the scalable approach to certifying a demonstration of quantum computational supremacy via IQP circuits given in Chapter 4 may be beneficially employed. Finally, by formalising and illustrating how quantum computational supremacy could be achieved through generative modelling, Chapter 5 further explores the applications that NISQ devices are best put to use in.

When concluding each chapter we have considered the ways in which the corresponding work could be fruitfully extended. Here it instead behoves us to contemplate and speculate on the prospects for the field as a whole. At the time of writing, the surety of supposed demonstrations of quantum computational supremacy is unclear. The two most compelling experiments of this kind considered demonstrations of quantum computational supremacy via RCS [60], and via a model similar to BosonSampling [176]. In the case of the former (as in time it will likely be for the latter) there has been a concerted effort to reproduce these experimental results using classical computers. There has been some success in doing so [120, 285], although it is understood that as quantum devices continue to grow and improve, these classical techniques will not be able to keep up. It is likely that this back and forth between improvements in quantum computers and classical simulation techniques will continue for some time until the advantage of quantum computers becomes insurmountable.

As this to and fro continues, the techniques used for the certification of demonstrations of quantum computational supremacy will have to adapt accordingly. This is especially true if the spoofing of techniques such as Heavy Output Generation Benchmarking and Cross-Entropy Benchmarking is achieved without the need to precisely calculate the ideal output probabilities of a quantum computation. This may mean that in practice it is more challenging to implement these certification techniques than it is to spoof them. Perhaps then the greatest problem left open is that of scalable certification techniques for NISQ technology by a classical client. It may be that the best approach to tackling this will be to highly optimise classical client verification schemes, such as those outlined in Section 1.6.2, specifically for a particular computation demonstrating quantum computational supremacy. Chapter 4 gives us some optimism that this approach may be advantageous, and presents one example of how adapting existing techniques to specific problems can be beneficial. Indeed, classical client techniques optimised for the scheme of Chapter 4 may be fruitful.

On the matter of device benchmarking, we believe that in the form of the work of Chapter 3 we have presented a highly informative standard benchmark suite. We are, however, aware that this is far from a universally accepted sentiment, and in reality many research groups use their own benchmark circuits. Such a situation makes it very difficult to assess and compare the performance of devices and compilation strategies, and so we regard it as a matter of some urgency that a consensus on the best suite of benchmarks is arrived at. In the case of machine learning, a more mature field than, for example, quantum software, there are such standard benchmark sets. These are used to measure and compare the performance of such things as language models or image recognition tools. These arose somewhat organically, and we hope, perhaps with some encouragement from the larger groups in the field, that the same can be achieved for benchmarking a quantum computing stack. Quantum volume provides a glimmer of hope that this is possible, but falls short of what we would regard as a gold-standard benchmark, as discussed in Chapter 3.

As perhaps all discussions on the prospects for quantum computing in the near-term should, we give the last word to a consideration of the prospect for quantum computers to outperform classical computers at a task of practical concern. As we have commented throughout, there are several natural ways that this could be achieved, notably through the simulation of physical systems, or via generative modelling as proposed in Chapter 5. That these application are so native to quantum computing should provide us with great confidence in the potential for these devices. That being said, it would be a fool's game to try to predict the timing of the first such demonstration. Unfortunately the development of classical computers provides little guidance on this, and it is a matter of debate as to whether what we have available now constitutes a difference engine, a parallel of Colossus, or something else. However, if I had to guess, I'd say the first practically useful quantum computer is 5 years away... as it has been throughout my PhD.

Bibliography

[1] Iskren Vankov, Daniel Mills, Petros Wallden, and Elham Kashefi. "Methods for classically simulating noisy networked quantum architectures". In: *Quantum Science and Technology* 5.1 (Nov. 2019), p. 014001. DOI: 10.1088/2058-9565/ab54a4.

[2] Daniel Mills, Seyon Sivarajah, Travis L. Scholten, and Ross Duncan. "Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack". In: *Quantum* 5 (Mar. 2021), p. 415. ISSN: 2521-327X. DOI: 10.22331/q-2021-03-22-415.

[3] Daniel Mills, Anna Pappa, Theodoros Kapourniotis, and Elham Kashefi. "Information Theoretically Secure Hypothesis Test for Temporally Unstructured Quantum Computation". In: Proceedings 14th International Conference on *Quantum Physics and Logic,* Nijmegen, The Netherlands, 3-7 July 2017. Vol. 266. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2018, pp. 209–221. DOI: 10.4204/EPTCS.266.14.

[4] Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. "The Born supremacy: quantum advantage and training of an Ising Born machine". In: *npj Quantum Information* 6.1 (July 2020), p. 60. ISSN: 2056-6387. DOI: 10.1038/s41534-020-00288-9.

[5] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theoretical Computer Science* 560 (Dec. 2014), pp. 7–11. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.05.025.

[6] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. "Proposed Experiment to Test Local Hidden-Variable Theories". In: *Phys. Rev. Lett.* 23 (15 Oct. 1969), pp. 880–884. DOI: 10.1103/PhysRevLett.23.880.

[7] Dan Boneh. "The Decision Diffie-Hellman problem". In: *Algorithmic Number Theory*. Ed. by Joe P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 48–63. ISBN: 978-3-540-69113-6.

[8] Artur K. Ekert. "Quantum cryptography based on Bell's theorem". In: *Phys. Rev. Lett.* 67 (6 Aug. 1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661.

[9] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502.

[10] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. "The LINPACK Benchmark: past, present and future". In: *Concurrency and Computation: Practice and Experience* 15.9 (2003), pp. 803–820. DOI: 10.1002/cpe.728.

[11] Oded Regev. “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography”. In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: 10.1145/1568318.1568324.

[12] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[13] *Networked Quantum Information Technologies Hub*. 2018. URL: https://nqit.ox.ac.uk/.

[14] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. “Parameterized quantum circuits as machine learning models”. In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. DOI: 10.1088/2058-9565/ab4eb5.

[15] Oded Goldreich. *Foundations of Cryptography*. Vol. 1. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511546891.

[16] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[17] Srinivasan Arunachalam, Alex B. Grilo, and Henry Yuen. *Quantum statistical query learning*. 2020. arXiv: 2002.08240 [quant-ph].

[18] Timothy Proctor et al. “What Randomized Benchmarking Actually Measures”. In: *Phys. Rev. Lett.* 119 (13 Sept. 2017), p. 130502. DOI: 10.1103/PhysRevLett.119.130502.

[19] E. Knill et al. “Randomized benchmarking of quantum gates”. In: *Phys. Rev. A* 77 (1 Jan. 2008), p. 012307. DOI: 10.1103/PhysRevA.77.012307.

[20] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342.

[21] Vedran Dunjko and Elham Kashefi. *Blind quantum computing with two almost identical states*. 2016. arXiv: 1604.01586 [quant-ph].

[22] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nature Communications* 5.1 (July 2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[23] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. “The theory of variational hybrid quantum-classical algorithms”. In: *New Journal of Physics* 18.2 (Feb. 2016), p. 023023. DOI: 10.1088/1367-2630/18/2/023023.

[24] Juan Bermejo-Vega et al. “Architectures for Quantum Simulation Showing a Quantum Speedup”. In: *Phys. Rev. X* 8 (2 Apr. 2018), p. 021010. DOI: 10.1103/PhysRevX.8.021010.

[25] Alastair I. M. Rae. *Quantum Mechanics, Sixth Edition*. Routledge, Dec. 2015. ISBN: 1482299186.

[26] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179.

[27] Seth Lloyd. “Universal Quantum Simulators”. In: *Science* 273.5278 (1996), pp. 1073–1078. ISSN: 0036-8075. DOI: 10.1126/science.273.5278.1073.

[28] Michael A. Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000, p. 676. ISBN: 9781107002173.

[29] Stephen Jordan. *Quantum Algorithm Zoo*. URL: https://quantumalgorithmzoo.org/.

[30] David Deutsch and Roger Penrose. “Quantum theory, the Church Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070.

[31] I. M. Georgescu, S. Ashhab, and Franco Nori. “Quantum simulation”. In: *Rev. Mod. Phys.* 86 (1 Mar. 2014), pp. 153–185. DOI: 10.1103/RevModPhys.86.153.

[32] Yudong Cao et al. “Quantum Chemistry in the Age of Quantum Computing”. In: *Chemical Reviews* 119.19 (2019). PMID: 31469277, pp. 10856–10915. DOI: 10.1021/acs.chemrev.8b00803.

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

[34] Lov K. Grover. “Quantum Mechanics Helps in Searching for a Needle in a Haystack”. In: *Phys. Rev. Lett.* 79 (2 July 1997), pp. 325–328. DOI: 10.1103/PhysRevLett.79.325.

[35] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. “Quantum amplitude amplification and estimation”. In: *Quantum Computation and Information* (2002), pp. 53–74. ISSN: 0271-4132. DOI: 10.1090/conm/305/05215.

[36] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000). https://arxiv.org/abs/quant-ph/0001106.

[37] Andrew M. Childs, Edward Farhi, and Sam Gutmann. “An Example of the Difference Between Quantum and Classical Random Walks”. In: *Quantum Information Processing* 1.1 (Apr. 2002), pp. 35–43. ISSN: 1573-1332. DOI: 10.1023/A:1019609420309.

[38] J. P. Buhler, H. W. Lenstra, and Carl Pomerance. “Factoring integers with the number field sieve”. In: *The development of the number field sieve*. Ed. by Arjen K. Lenstra and Hendrik W. Lenstra. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 50–94. ISBN: 978-3-540-47892-8.

[39] Aram W. Harrow and Ashley Montanaro. “Quantum computational supremacy”. In: *Nature* 549.7671 (Sept. 2017), pp. 203–209. ISSN: 1476-4687. DOI: 10.1038/nature23458.

[40] John Preskill. “Quantum computing and the entanglement frontier”. In: (2012). arXiv: 1203.5813 [quant-ph].

[41] Joe O’Gorman and Earl T. Campbell. “Quantum computation with realistic magic-state factories”. In: *Phys. Rev. A* 95 (3 Mar. 2017), p. 032338. DOI: 10.1103/PhysRevA.95.032338.

[42] T. D. Ladd et al. “Quantum computers”. In: *Nature* 464.7285 (Mar. 2010), pp. 45–53. ISSN: 1476-4687. DOI: 10.1038/nature08812.

[43] Jeremy L. O'Brien. "Optical Quantum Computing". In: *Science* 318.5856 (2007), pp. 1567–1570. ISSN: 0036-8075. DOI: 10.1126/science.1142892.

[44] E. Knill, R. Laflamme, and G. J. Milburn. "A scheme for efficient quantum computation with linear optics". In: *Nature* 409.6816 (Jan. 2001), pp. 46–52. ISSN: 1476-4687. DOI: 10.1038/35051009.

[45] Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai. "Coherent control of macroscopic quantum states in a single-Cooper-pair box". In: *Nature* 398.6730 (Apr. 1999), pp. 786–788. ISSN: 1476-4687. DOI: 10.1038/19718.

[46] D. J. Wineland et al. "Experimental Issues in Coherent Quantum-State Manipulation of Trapped Atomic Ions". eng. In: *Journal of research of the National Institute of Standards and Technology* 103.3 (1998), pp. 259–328. ISSN: 1044-677X. DOI: 10.6028/jres.103.019.

[47] J. I. Cirac and P. Zoller. "Quantum Computations with Cold Trapped Ions". In: *Phys. Rev. Lett.* 74 (20 May 1995), pp. 4091–4094. DOI: 10.1103/PhysRevLett.74.4091.

[48] David P. DiVincenzo. "The Physical Implementation of Quantum Computation". In: *Fortschritte der Physik* 48.9-11 (2000), pp. 771–783. DOI: https://doi.org/10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E.

[49] Alexander Zlokapa, Sergio Boixo, and Daniel Lidar. *Boundaries of quantum supremacy via random circuit sampling*. 2020. arXiv: 2005.02464 [quant-ph].

[50] Gil Kalai. *The Argument against Quantum Computers*. 2019. arXiv: 1908.02499 [quant-ph].

[51] Peter W. Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: 10.1103/PhysRevA.52.R2493.

[52] Andrew Steane. "Multiple-particle interference and quantum error correction". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 452.1954 (1996), pp. 2551–2577. DOI: 10.1098/rspa.1996.0136.

[53] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. "Surface codes: Towards practical large-scale quantum computation". In: *Phys. Rev. A* 86 (3 Sept. 2012), p. 032324. DOI: 10.1103/PhysRevA.86.032324.

[54] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. "Roads towards fault-tolerant universal quantum computation". In: *Nature* 549.7671 (Sept. 2017), pp. 172–179. ISSN: 1476-4687. DOI: 10.1038/nature23460.

[55] Barbara M. Terhal. "Quantum error correction for quantum memories". In: *Rev. Mod. Phys.* 87 (2 Apr. 2015), pp. 307–346. DOI: 10.1103/RevModPhys.87.307.

[56] Simon J Devitt, William J Munro, and Kae Nemoto. "Quantum error correction for beginners". In: *Reports on Progress in Physics* 76.7 (June 2013), p. 076001. DOI: 10.1088/0034-4885/76/7/076001.

[57] D. Aharonov and M. Ben-Or. "Fault-Tolerant Quantum Computation with Constant Error". In: *Proceedings of the Twenty-Ninth Annual ACM Sympo-*

sium on Theory of Computing. STOC '97. El Paso, Texas, USA: Association for Computing Machinery, 1997, pp. 176–188. ISBN: 0897918886. DOI: 10.1145/258533.258579.

[58] Craig Gidney and Martin Ekerå. *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*. Apr. 2021. DOI: 10.22331/q-2021-04-15-433.

[59] Markus Reiher et al. “Elucidating reaction mechanisms on quantum computers”. In: *Proceedings of the National Academy of Sciences* (2017). ISSN: 0027-8424. DOI: 10.1073/pnas.1619152114.

[60] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5.

[61] Scott Aaronson and Alex Arkhipov. “The Computational Complexity of Linear Optics”. In: *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*. STOC '11. San Jose, California, USA: ACM, 2011, pp. 333–342. ISBN: 978-1-4503-0691-1. DOI: 10.1145/1993636.1993682.

[62] Sergio Boixo et al. “Characterizing quantum supremacy in near-term devices”. In: *Nature Physics* 14.6 (2018), pp. 595–600. ISSN: 1745-2481. DOI: 10.1038/s41567-018-0124-x.

[63] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. “Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy”. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 467.2126 (2011), pp. 459–472. ISSN: 1364-5021. DOI: 10.1098/rspa.2010.0301.

[64] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. “Achieving quantum supremacy with sparse and noisy commuting quantum computations”. In: *Quantum* 1 (Apr. 2017), p. 8. ISSN: 2521-327X. DOI: 10.22331/q-2017-04-25-8.

[65] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. “Average-Case Complexity Versus Approximate Simulation of Commuting Quantum Computations”. In: *Phys. Rev. Lett.* 117 (8 Aug. 2016), p. 080501. DOI: 10.1103/PhysRevLett.117.080501.

[66] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. “On the complexity and verification of quantum random circuit sampling”. In: *Nature Physics* 15.2 (Feb. 2019), pp. 159–163. ISSN: 1745-2481. DOI: 10.1038/s41567-018-0318-2.

[67] Suguru Endo, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. “Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation”. In: *Journal of the Physical Society of Japan* 90.3 (2021), p. 032001. DOI: 10.7566/JPSJ.90.032001.

[68] Kishor Bharti et al. *Noisy intermediate-scale quantum (NISQ) algorithms*. 2021. arXiv: 2101.08448 [quant-ph].

[69] Eric Anschuetz, Jonathan Olson, Alán Aspuru-Guzik, and Yudong Cao. “Variational Quantum Factoring”. In: *Quantum Technology and Optimization Problems*. Ed. by Sebastian Feld and Claudia Linnhoff-Popien. Cham: Springer International Publishing, 2019, pp. 74–85. ISBN: 978-3-030-14082-3.

[70] Adam Bouland et al. *Prospects and challenges of quantum finance*. 2020. arXiv: 2011.06492 [q-fin.CP].

[71] Vedran Dunjko and Hans J Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". In: *Reports on Progress in Physics* 81.7 (June 2018), p. 074001. DOI: 10.1088/1361-6633/aab406.

[72] Tomoyuki Morimae. "Verification for measurement-only blind quantum computing". In: *Phys. Rev. A* 89 (6 June 2014), p. 060302. DOI: 10.1103/PhysRevA.89.060302.

[73] Joseph F. Fitzsimons and Elham Kashefi. "Unconditionally verifiable blind quantum computation". In: *Phys. Rev. A* 96 (1 July 2017), p. 012303. DOI: 10.1103/PhysRevA.96.012303.

[74] Anne Broadbent. "How to Verify a Quantum Computation". In: *Theory of Computing* 14.11 (2018), pp. 1–37. DOI: 10.4086/toc.2018.v014a011.

[75] D Hangleiter, M Kliesch, M Schwarz, and J Eisert. "Direct certification of a class of quantum simulations". In: *Quantum Science and Technology* 2.1 (Feb. 2017), p. 015004. DOI: 10.1088/2058-9565/2/1/015004.

[76] Joseph F. Fitzsimons, Michal Hajdu šek, and Tomoyuki Morimae. "Post hoc verification of quantum computation". In: *Phys. Rev. Lett.* 120 (4 Jan. 2018), p. 040501. DOI: 10.1103/PhysRevLett.120.040501.

[77] U. Mahadev. "Classical Verification of Quantum Computations". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. Oct. 2018, pp. 259–267. DOI: 10.1109/FOCS.2018.00033.

[78] Matthew McKague. "Interactive Proofs for BQP via Self-Tested Graph States". In: *Theory of Computing* 12.3 (2016), pp. 1–42. DOI: 10.4086/toc.2016.v012a003.

[79] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. "Classical command of quantum systems". In: *Nature* 496.7446 (Apr. 2013), pp. 456–460. ISSN: 1476-4687. DOI: 10.1038/nature12035.

[80] Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. "Robustness and device independence of verifiable blind quantum computing". In: *New Journal of Physics* 17.8 (Aug. 2015), p. 083040. DOI: 10.1088/1367-2630/17/8/083040.

[81] Jens Eisert et al. "Quantum certification and benchmarking". In: *Nature Reviews Physics* 2.7 (July 2020), pp. 382–390. ISSN: 2522-5820. DOI: 10.1038/s42254-020-0186-4.

[82] Kenneth Rudinger et al. "Probing Context-Dependent Errors in Quantum Processors". In: *Phys. Rev. X* 9 (2 June 2019), p. 021045. DOI: 10.1103/PhysRevX.9.021045.

[83] *IBM Quantum*. 2021. URL: https://quantum-computing.ibm.com/.

[84] Peter J Karalekas et al. "A quantum-classical cloud platform optimized for variational hybrid algorithms". In: *Quantum Science and Technology* 5.2 (Apr. 2020), p. 024003. DOI: 10.1088/2058-9565/ab7559.

[85] Scott Aaronson and Lijie Chen. "Complexity-Theoretic Foundations of Quantum Supremacy Experiments". In: *32nd Computational Complexity Conference (CCC 2017)*. Ed. by Ryan O'Donnell. Vol. 79. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–

Leibniz-Zentrum fuer Informatik, 2017, 22:1–22:67. ISBN: 978-3-95977-040-8. DOI: 10.4230/LIPIcs.CCC.2017.22.

[86] Alexander J. McCaskey et al. “Quantum chemistry as a benchmark for near-term quantum computers”. In: *npj Quantum Information* 5.1 (2019), p. 99. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0209-0.

[87] Pierre-Luc Dallaire-Demers et al. “An application benchmark for fermionic quantum simulations”. In: (2020). arXiv: 2003.01862 [quant-ph].

[88] Frank Arute et al. *Hartree-Fock on a superconducting qubit quantum computer*. 2020. DOI: 10.1126/science.abb9811.

[89] Norbert M. Linke et al. “Experimental comparison of two quantum computing architectures”. In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3305–3310. ISSN: 0027-8424. DOI: 10.1073/pnas.1618020114.

[90] Marcello Benedetti et al. “A generative modeling approach for benchmarking and training shallow quantum circuits”. In: *npj Quantum Information* 5.1 (2019), p. 45. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0157-8.

[91] Kathleen E. Hamilton, Eugene F. Dumitrescu, and Raphael C. Pooser. “Generative model benchmarks for superconducting qubits”. In: *Phys. Rev. A* 99 (6 June 2019), p. 062323. DOI: 10.1103/PhysRevA.99.062323.

[92] Madita Willsch et al. *Benchmarking the quantum approximate optimization algorithm*. June 2020. DOI: 10.1007/s11128-020-02692-8.

[93] Andreas Bengtsson et al. *Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm*. Sept. 2020. DOI: 10.1103/PhysRevApplied.14.034010.

[94] Guido Pagano et al. *Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator*. 2020. DOI: 10.1073/pnas.2006373117.

[95] Matthew P. Harrigan et al. *Quantum approximate optimization of non-planar graph problems on a planar superconducting processor*. Mar. 2021. DOI: 10.1038/s41567-020-01105-y.

[96] A. Broadbent, J. Fitzsimons, and E. Kashefi. “Universal Blind Quantum Computation”. In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. Oct. 2009, pp. 517–526. DOI: 10.1109/FOCS.2009.36.

[97] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. “Quantum circuit learning”. In: *Phys. Rev. A* 98 (3 Sept. 2018), p. 032309. DOI: 10.1103/PhysRevA.98.032309.

[98] A. Einstein, B. Podolsky, and N. Rosen. “Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?” In: *Phys. Rev.* 47 (10 May 1935), pp. 777–780. DOI: 10.1103/PhysRev.47.777.

[99] J. S. Bell. “On the Einstein Podolsky Rosen paradox”. In: *Physics Physique Fizika* 1 (3 Nov. 1964), pp. 195–200. DOI: 10.1103/PhysicsPhysiqueFizika.1.195.

[100] B. Hensen et al. “Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres”. In: *Nature* 526.7575 (Oct. 2015), pp. 682–686. ISSN: 1476-4687. DOI: 10.1038/nature15759.

[101] Daniel Gottesman. "Class of quantum error-correcting codes saturating the quantum Hamming bound". In: *Phys. Rev. A* 54 (3 Sept. 1996), pp. 1862–1868. DOI: 10.1103/PhysRevA.54.1862.

[102] Daniel Gottesman. "The Heisenberg Representation of Quantum Computers". In: (1998). arXiv: quant-ph/9807006 [quant-ph].

[103] A Yu Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (Dec. 1997), pp. 1191–1249. DOI: 10.1070/rm1997v052n06abeh002155.

[104] Christopher M Dawson and Michael A Nielsen. "The solovay-kitaev algorithm". In: *arXiv preprint quant-ph/0505030* (2005). https://arxiv.org/abs/quant-ph/0505030.

[105] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. 1st. USA: Cambridge University Press, 2009. ISBN: 0521424267.

[106] Scott Aaronson. *Complexity Zoo*. URL: https://complexityzoo.net/Complexity_Zoo.

[107] L.G. Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(80)90061-0.

[108] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of mathematics* (2004), pp. 781–793. DOI: https://doi.org/10.4007/annals.2004.160.781.

[109] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047.

[110] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[111] Larry J. Stockmeyer. "The polynomial-time hierarchy". In: *Theoretical Computer Science* 3.1 (1976), pp. 1–22. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(76)90061-X.

[112] Scott Aaronson. "$\mathsf{P} \overset{?}{=} \mathsf{NP}$". In: *Open problems in mathematics*. Springer, 2016, pp. 1–122.

[113] Dan Shepherd and Michael J. Bremner. "Temporally unstructured quantum computation". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2105 (2009), pp. 1413–1439. DOI: 10.1098/rspa.2008.0443.

[114] Gadi Aleksandrowicz et al. *Qiskit: An Open-source Framework for Quantum Computing*. Version 0.7.2. Jan. 2019. DOI: 10.5281/zenodo.2562111.

[115] Robert S. Smith, Michael J. Curtis, and William J. Zeng. "A Practical Quantum Instruction Set Architecture". In: (2017). arXiv: 1608.03355 [quant-ph].

[116] Quantum AI team and collaborators. *Cirq*. Oct. 2020. DOI: 10.5281/zenodo.4062499.

[117] *Qulacs*. 2018. eprint: https://github.com/qulacs/qulacs.

[118] Tyson Jones, Anna Brown, Ian Bush, and Simon C. Benjamin. "QuEST and High Performance Simulation of Quantum Computers". In: *Scientific Reports* 9.1 (2019), p. 10736. ISSN: 2045-2322. DOI: 10.1038/s41598-019-47174-9.

[119] Quantum AI team and collaborators. *qsim*. Sept. 2020. DOI: 10.5281/zenodo.4023103.

[120] Edwin Pednault et al. *Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits*. 2019. arXiv: 1910.09534 [quant-ph].

[121] Edwin Pednault et al. "Pareto-Efficient Quantum Circuit Simulation Using Tensor Contraction Deferral". In: (2017). arXiv: 1710.05867 [quant-ph].

[122] Thomas Häner and Damian S. Steiger. "0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '17. Denver, Colorado: Association for Computing Machinery, 2017. ISBN: 9781450351140. DOI: 10.1145/3126908.3126947.

[123] Barbara M Terhal and David P DiVincenzo. "Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games". In: *arXiv preprint quant-ph/0205133* (2002). https://arxiv.org/abs/quant-ph/0205133.

[124] M. Van den Nest. "Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond". In: (2008). arXiv: 0811.0898 [quant-ph].

[125] Guifré Vidal. "Efficient Classical Simulation of Slightly Entangled Quantum Computations". In: *Phys. Rev. Lett.* 91 (14 Oct. 2003), p. 147902. DOI: 10.1103/PhysRevLett.91.147902.

[126] Igor L. Markov and Yaoyun Shi. "Simulating Quantum Computation by Contracting Tensor Networks". In: *SIAM Journal on Computing* 38.3 (2008), pp. 963–981. DOI: 10.1137/050644756.

[127] Dan Stahlke. "Quantum interference as a resource for quantum speedup". In: *Phys. Rev. A* 90 (2 Aug. 2014), p. 022302. DOI: 10.1103/PhysRevA.90.022302.

[128] A. Mari and J. Eisert. "Positive Wigner Functions Render Classical Simulation of Quantum Computation Efficient". In: *Phys. Rev. Lett.* 109 (23 Dec. 2012), p. 230503. DOI: 10.1103/PhysRevLett.109.230503.

[129] Peter Clifford and Raphaël Clifford. "The Classical Complexity of Boson Sampling". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 146–155. ISBN: 978-1-6119-7503-1. DOI: 10.1137/1.9781611975031.10.

[130] Alex Neville et al. "Classical boson sampling algorithms with superior performance to near-term experiments". In: *Nature Physics* 13.12 (Dec. 2017), pp. 1153–1157. ISSN: 1745-2481. DOI: 10.1038/nphys4270.

[131] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Phys. Rev. A* 70 (5 Nov. 2004), p. 052328. DOI: 10.1103/PhysRevA.70.052328.

[132] H. J. García and I. L. Markov. "Simulation of Quantum Circuits via Stabilizer Frames". In: *IEEE Transactions on Computers* 64.8 (Aug. 2015), pp. 2323–2336. DOI: 10.1109/TC.2014.2360532.

[133] David Gosset, Daniel Grier, Alex Kerzner, and Luke Schaeffer. *Fast simulation of planar Clifford circuits*. 2020. arXiv: 2009.03218 [quant-ph].

[134] Sergey Bravyi and David Gosset. "Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates". In: *Phys. Rev. Lett.* 116 (25 June 2016), p. 250501. DOI: 10.1103/PhysRevLett.116.250501.

[135] Sergey Bravyi et al. "Simulation of quantum circuits by low-rank stabilizer decompositions". In: *Quantum* 3 (Sept. 2019), p. 181. ISSN: 2521-327X. DOI: 10.22331/q-2019-09-02-181.

[136] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". In: *Phys. Rev. A* 71 (2 Feb. 2005), p. 022316. DOI: 10.1103/PhysRevA.71.022316.

[137] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. "Methodology for quantum logic gate construction". In: *Phys. Rev. A* 62 (5 Oct. 2000), p. 052316. DOI: 10.1103/PhysRevA.62.052316.

[138] Benjamin Villalonga et al. "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware". In: *npj Quantum Information* 5.1 (2019), p. 86. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0196-1.

[139] Benjamin Villalonga et al. "Establishing the quantum supremacy frontier with a 281 Pflop/s simulation". In: *Quantum Science and Technology* 5.3 (Apr. 2020), p. 034003. DOI: 10.1088/2058-9565/ab7eeb.

[140] Akel Hashim et al. *Randomized compiling for scalable quantum computing on a noisy superconducting quantum processor*. 2020. arXiv: 2010.00215 [quant-ph].

[141] Joel J. Wallman and Joseph Emerson. "Noise tailoring for scalable quantum computation via randomized compiling". In: *Phys. Rev. A* 94 (5 Nov. 2016), p. 052325. DOI: 10.1103/PhysRevA.94.052325.

[142] K. Kraus, A. Böhm, J.D. Dollard, and W.H. Wootters. *States, Effects, and Operations: Fundamental Notions of Quantum Theory*. Lecture Notes in Physics. Springer Berlin Heidelberg, 1983. ISBN: 9780387127323.

[143] Yanzhu Chen, Maziar Farahzad, Shinjae Yoo, and Tzu-Chieh Wei. "Detector tomography on IBM quantum computers and mitigation of an imperfect measurement". In: *Phys. Rev. A* 100 (5 Nov. 2019), p. 052315. DOI: 10.1103/PhysRevA.100.052315.

[144] Mingyu Sun and Michael R. Geller. *Efficient characterization of correlated SPAM errors*. 2018. arXiv: 1810.10523 [quant-ph].

[145] Sergey Bravyi et al. *Mitigating measurement errors in multiqubit experiments*. Apr. 2021. DOI: 10.1103/PhysRevA.103.042605.

[146] Kathleen E. Hamilton and Raphael C. Pooser. "Error-mitigated data-driven circuit learning on noisy quantum hardware". In: (2019). arXiv: 1911.13289 [cs.ET].

[147] Adam Winick, Joel J. Wallman, and Joseph Emerson. *Simulating and mitigating crosstalk*. 2020. arXiv: 2006.09596 [quant-ph].

[148] Mohan Sarovar et al. "Detecting crosstalk errors in quantum information processors". In: *Quantum* 4 (Sept. 2020), p. 321. ISSN: 2521-327X. DOI: 10.22331/q-2020-09-11-321.

[149] Lorenza Viola, Emanuel Knill, and Seth Lloyd. "Dynamical Decoupling of Open Quantum Systems". In: *Phys. Rev. Lett.* 82 (12 Mar. 1999), pp. 2417–2421. DOI: 10.1103/PhysRevLett.82.2417.

[150] Robert Raussendorf and Hans J Briegel. "A one-way quantum computer". In: *Physical Review Letters* 86.22 (2001), p. 5188. DOI: 10.1103/PhysRevLett.86.5188.

[151] Robert Raussendorf, Daniel E Browne, and Hans J Briegel. "Measurement-based quantum computation on cluster states". In: *Physical review A* 68.2 (2003), p. 022312. DOI: 10.1103/PhysRevA.68.022312.

[152] H. J. Briegel et al. "Measurement-based quantum computation". In: *Nature Physics* 5.1 (2009), pp. 19–26. ISSN: 1745-2481. DOI: 10.1038/nphys1157.

[153] Richard Jozsa. *An introduction to measurement based quantum computation*. 2005. arXiv: quant-ph/0508124 [quant-ph].

[154] Sara Bartolucci et al. *Fusion-based quantum computation*. 2021. arXiv: 2101.09310 [quant-ph].

[155] Vincent Danos and Elham Kashefi. "Determinism in the one-way model". In: *Phys. Rev. A* 74 (5 Nov. 2006), p. 052310. DOI: 10.1103/PhysRevA.74.052310.

[156] Daniel E Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. "Generalized flow and determinism in measurement-based quantum computation". In: *New Journal of Physics* 9.8 (Aug. 2007), pp. 250–250. DOI: 10.1088/1367-2630/9/8/250.

[157] Atul Mantri, Tommaso F. Demarie, Nicolas C. Menicucci, and Joseph F. Fitzsimons. "Flow Ambiguity: A Path Towards Classically Driven Blind Quantum Computation". In: *Phys. Rev. X* 7 (3 July 2017), p. 031004. DOI: 10.1103/PhysRevX.7.031004.

[158] Matty J. Hoban et al. "Measurement-Based Classical Computation". In: *Phys. Rev. Lett.* 112 (14 Apr. 2014), p. 140505. DOI: 10.1103/PhysRevLett.112.140505.

[159] Tomoyuki Morimae, Keisuke Fujii, and Joseph F. Fitzsimons. "Hardness of Classically Simulating the One-Clean-Qubit Model". In: *Phys. Rev. Lett.* 112 (13 Apr. 2014), p. 130502. DOI: 10.1103/PhysRevLett.112.130502.

[160] Xun Gao, Sheng-Tao Wang, and L.-M. Duan. "Quantum Supremacy for Simulating a Translation-Invariant Ising Spin Model". In: *Phys. Rev. Lett.* 118 (4 Jan. 2017), p. 040502. DOI: 10.1103/PhysRevLett.118.040502.

[161] P Aliferis et al. "Fault-tolerant computing with biased-noise superconducting qubits: a case study". In: *New Journal of Physics* 11.1 (Jan. 2009), p. 013061. DOI: 10.1088/1367-2630/11/1/013061.

[162] Keisuke Fujii and Shuhei Tamate. “Computational quantum-classical boundary of noisy commuting quantum circuits”. In: *Scientific Reports* 6.1 (May 2016), p. 25598. ISSN: 2045-2322. DOI: 10.1038/srep25598.

[163] Alexander M. Dalzell, Aram W. Harrow, Dax Enshan Koh, and Rolando L. La Placa. “How many qubits are needed for quantum computational supremacy?” In: *Quantum* 4 (May 2020), p. 264. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-11-264.

[164] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. *Expressive power of parametrized quantum circuits*. July 2020. DOI: 10.1103/PhysRevResearch.2.033125.

[165] Vojtěch Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. ISSN: 1476-4687. DOI: 10.1038/s41586-019-0980-2.

[166] Scott Aaronson. “Quantum computing, postselection, and probabilistic polynomial-time”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 461.2063 (2005), pp. 3473–3482. DOI: 10.1098/rspa.2005.1546.

[167] Seinosuke Toda. “PP is as Hard as the Polynomial-Time Hierarchy”. In: *SIAM Journal on Computing* 20.5 (1991), pp. 865–877. DOI: 10.1137/0220053.

[168] Yenjo Han, Lane A. Hemaspaandra, and Thomas Thierauf. “Threshold Computation and Cryptographic Security”. In: *SIAM Journal on Computing* 26.1 (1997), pp. 59–78. DOI: 10.1137/S0097539792240467.

[169] Leslie Ann Goldberg and Heng Guo. “The Complexity of Approximating complex-valued Ising and Tutte partition functions”. In: *computational complexity* 26.4 (Sept. 2017), pp. 765–833. ISSN: 1420-8954. DOI: 10.1007/s00037-017-0162-2.

[170] Keisuke Fujii and Tomoyuki Morimae. “Commuting quantum circuits and complexity of Ising partition functions”. In: *New Journal of Physics* 19.3 (Mar. 2017), p. 033003. DOI: 10.1088/1367-2630/aa5fdb.

[171] Animesh Datta, Steven T. Flammia, and Carlton M. Caves. “Entanglement and the power of one qubit”. In: *Phys. Rev. A* 72 (4 Oct. 2005), p. 042316. DOI: 10.1103/PhysRevA.72.042316.

[172] Larry Stockmeyer. “On Approximation Algorithms for # P”. In: *SIAM Journal on Computing* 14.4 (1985), pp. 849–861. DOI: 10.1137/0214060.

[173] Alexander Cowtan et al. “On the Qubit Routing Problem”. In: *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*. Ed. by Wim van Dam and Laura Mancinska. Vol. 135. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 5:1–5:32. ISBN: 978-3-95977-112-2. DOI: 10.4230/LIPIcs.TQC.2019.5.

[174] Robert Beals et al. “Efficient distributed quantum computing”. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 469.2153 (2013). ISSN: 1364-5021. DOI: 10.1098/rspa.2012.0686.

[175] Xun Gao and Luming Duan. “Efficient classical simulation of noisy quantum computation”. In: (2018). arXiv: 1810.03176 [quant-ph].

[176] Han-Sen Zhong et al. “Quantum computational advantage using photons”. In: *Science* 370.6523 (2020), pp. 1460–1463. ISSN: 0036-8075. DOI: 10.1126/science.abe8770.

[177] Alex Arkhipov. “BosonSampling is robust against small errors in the network matrix”. In: *Phys. Rev. A* 92 (6 Dec. 2015), p. 062326. DOI: 10.1103/PhysRevA.92.062326.

[178] Scott Aaronson and Daniel J. Brod. “BosonSampling with lost photons”. In: *Phys. Rev. A* 93 (1 Jan. 2016), p. 012335. DOI: 10.1103/PhysRevA.93.012335.

[179] Gil Kalai and Guy Kindler. *Gaussian Noise Sensitivity and BosonSampling*. 2014. arXiv: 1409.3093 [quant-ph].

[180] Saleh Rahimi-Keshari, Timothy C. Ralph, and Carlton M. Caves. “Sufficient Conditions for Efficient Classical Simulation of Quantum Optics”. In: *Phys. Rev. X* 6 (2 June 2016), p. 021039. DOI: 10.1103/PhysRevX.6.021039.

[181] Anthony Leverrier and Raul Garcia-Patron. *Analysis of Circuit Imperfections in Bosonsampling*. Paramus, NJ, Apr. 2015. arXiv: 1309.4687 [quant-ph].

[182] Hui Wang et al. “Boson Sampling with 20 Input Photons and a 60-Mode Interferometer in a 10^{14}-Dimensional Hilbert Space”. In: *Phys. Rev. Lett.* 123 (25 Dec. 2019), p. 250503. DOI: 10.1103/PhysRevLett.123.250503.

[183] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. DOI: 10.1017/9781316848142.

[184] Dominik Hangleiter, Juan Bermejo-Vega, Martin Schwarz, and Jens Eisert. “Anticoncentration theorems for schemes showing a quantum speedup”. In: *Quantum* 2 (May 2018), p. 65. ISSN: 2521-327X. DOI: 10.22331/q-2018-05-22-65.

[185] Fernando G. S. L. Brandão and Michal Horodecki. *Exponential Quantum Speed-Ups Are Generic*. Paramus, NJ, Nov. 2013. arXiv: 1010.3654 [quant-ph].

[186] Ramis Movassagh. “Efficient unitary paths and quantum computational supremacy: A proof of average-case hardness of Random Circuit Sampling”. In: (2018). arXiv: 1810.04681 [quant-ph].

[187] Stephanie Wehner, David Elkouss, and Ronald Hanson. “Quantum internet: A vision for the road ahead”. In: *Science* 362.6412 (2018). ISSN: 0036-8075. DOI: 10.1126/science.aam9288.

[188] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. “Verification of Quantum Computation: An Overview of Existing Approaches”. In: *Theory of Computing Systems* (July 2018). ISSN: 1433-0490. DOI: 10.1007/s00224-018-9872-3.

[189] Joseph F. Fitzsimons. “Private quantum computation: an introduction to blind quantum computing and related protocols”. In: *npj Quantum Information* 3.1 (2017), p. 23. ISSN: 2056-6387. DOI: 10.1038/s41534-017-0025-3.

[190] Andrew M. Childs. “Secure Assisted Quantum Computation”. In: *Quantum Info. Comput.* 5.6 (Sept. 2005), pp. 456–466. ISSN: 1533-7146. DOI: 10.26421/QIC5.6.

[191] Dorit Aharonov, Michael Ben-Or, and Elad Eban. *Interactive Proofs For Quantum Computations*. 2008. arXiv: 0810.5375 [quant-ph].

[192] Stefanie Barz et al. “Demonstration of Blind Quantum Computing”. In: *Science* 335.6066 (2012), pp. 303–308. ISSN: 0036-8075. DOI: 10.1126/science.1214707.

[193] Tomoyuki Morimae and Keisuke Fujii. “Blind quantum computation protocol in which Alice only makes measurements”. In: *Phys. Rev. A* 87 (5 May 2013), p. 050301. DOI: 10.1103/PhysRevA.87.050301.

[194] Vedran Dunjko, Joseph F. Fitzsimons, Christopher Portmann, and Renato Renner. “Composable Security of Delegated Quantum Computation”. In: ed. by Palash Sarkar and Tetsu Iwata. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 406–425. ISBN: 9783662456088. DOI: 10.1007/978-3-662-45608-8_22.

[195] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. “QFactory: Classically-Instructed Remote Secret Qubits Preparation”. In: *Advances in Cryptology – ASIACRYPT 2019* (2019), pp. 615–645. ISSN: 1611-3349. DOI: 10.1007/978-3-030-34578-5_22.

[196] Alexandru Gheorghiu and Thomas Vidick. *Computationally-secure and composable remote state preparation*. 2019. arXiv: 1904.06320 [quant-ph].

[197] Stefanie Barz, Joseph F. Fitzsimons, Elham Kashefi, and Philip Walther. “Experimental verification of quantum computation”. In: *Nature Physics* 9.11 (Nov. 2013), pp. 727–731. ISSN: 1745-2481. DOI: 10.1038/nphys2763.

[198] David Gosset, Barbara M. Terhal, and Anna Vershynina. “Universal Adiabatic Quantum Computation via the Space-Time Circuit-to-Hamiltonian Construction”. In: *Phys. Rev. Lett.* 114 (14 Apr. 2015), p. 140501. DOI: 10.1103/PhysRevLett.114.140501.

[199] Dorit Aharonov et al. “Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation”. In: *SIAM Review* 50.4 (2008), pp. 755–787. DOI: 10.1137/080734479.

[200] Julia Kempe, Alexei Kitaev, and Oded Regev. “The Complexity of the Local Hamiltonian Problem”. In: *SIAM J. Comput.* 35.5 (May 2006), pp. 1070–1097. ISSN: 0097-5397. DOI: 10.1137/S0097539704445226.

[201] Tomoyuki Morimae, Daniel Nagaj, and Norbert Schuch. “Quantum proofs can be verified using only single-qubit measurements”. In: *Phys. Rev. A* 93 (2 Feb. 2016), p. 022326. DOI: 10.1103/PhysRevA.93.022326.

[202] Michal Hajdušek, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. *Device-Independent Verifiable Blind Quantum Computation*. 2015. arXiv: 1502.02563 [quant-ph].

[203] Theodoros Kapourniotis, Elham Kashefi, and Animesh Datta. *Verified Delegated Quantum Computing with One Pure Qubit*. 2014. arXiv: 1403.1438 [quant-ph].

[204] Theodoros Kapourniotis and Animesh Datta. “Nonadaptive fault-tolerant verification of quantum supremacy with noise”. In: *Quantum* 3 (July 2019), p. 164. ISSN: 2521-327X. DOI: 10.22331/q-2019-07-12-164.

[205] D. J. Shepherd and M. J. Bremner. *Allice’s quantum challenge*. 2008. URL: https://quantumchallenges.wordpress.com/.

[206] James G Oxley. *Matroid theory*. Vol. 3. Oxford University Press, USA, 2006.

[207] Gregory D. Kahanamoku-Meyer. *Forging quantum data: classically defeating an IQP-based quantum test*. 2019. arXiv: 1912.05547 [quant-ph].

[208] Samuele Ferracin, Theodoros Kapourniotis, and Animesh Datta. "Accrediting outputs of noisy intermediate-scale quantum computing devices". In: *New Journal of Physics* 21.11 (Nov. 2019), p. 113038. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab4fd6.

[209] Scott Aaronson and Sam Gunn. *On the Classical Hardness of Spoofing Linear Cross-Entropy Benchmarking*. 2019. arXiv: 1910.12085 [quant-ph].

[210] Boaz Barak, Chi-Ning Chou, and Xun Gao. *Spoofing Linear Cross-Entropy Benchmarking in Shallow Quantum Circuits*. 2020. arXiv: 2005.02421 [quant-ph].

[211] C. E. Porter and R. G. Thomas. "Fluctuations of Nuclear Reaction Widths". In: *Phys. Rev.* 104 (2 Oct. 1956), pp. 483–491. DOI: 10.1103/PhysRev.104.483.

[212] Andrew W. Cross et al. "Validating quantum computers using randomized model circuits". In: *Phys. Rev. A* 100 (3 Sept. 2019), p. 032328. DOI: 10.1103/PhysRevA.100.032328.

[213] Robin Blume-Kohout and Kevin C. Young. *A volumetric framework for quantum computer benchmarks*. Nov. 2020. DOI: 10.22331/q-2020-11-15-362.

[214] Timothy Proctor et al. *Measuring the Capabilities of Quantum Computers*. 2020. arXiv: 2008.11294 [quant-ph].

[215] Robert König, Renato Renner, Andor Bariska, and Ueli Maurer. "Small Accessible Quantum Information Does Not Imply Security". In: *Phys. Rev. Lett.* 98 (14 Apr. 2007), p. 140502. DOI: 10.1103/PhysRevLett.98.140502.

[216] Jonathan Barrett, Roger Colbeck, and Adrian Kent. "Memory Attacks on Device-Independent Quantum Cryptography". In: *Phys. Rev. Lett.* 110 (1 Jan. 2013), p. 010503. DOI: 10.1103/PhysRevLett.110.010503.

[217] Birgit Pfitzmann and Michael Waidner. "Composition and Integrity Preservation of Secure Reactive Systems". In: *Proceedings of the 7th ACM Conference on Computer and Communications Security*. CCS '00. Athens, Greece: Association for Computing Machinery, 2000, pp. 245–254. ISBN: 1581132034. DOI: 10.1145/352600.352639.

[218] R. Canetti. "Universally composable security: a new paradigm for cryptographic protocols". In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[219] Michael Ben-Or and Dominic Mayers. "General security definition and composability for quantum & classical protocols". In: *arXiv preprint quant-ph/0409062* (2004). https://arxiv.org/abs/quant-ph/0409062v2.

[220] Dominique Unruh. "Simulatable security for quantum protocols". In: *arXiv preprint quant-ph/0409125* (2004). https://arxiv.org/abs/quant-ph/0409125.

[221] Ueli Maurer and Renato Renner. "Abstract cryptography". In: *IN INNOVATIONS IN COMPUTER SCIENCE*. Tsinghua University Press, 2011.

[222] Christopher Portmann and Renato Renner. "Cryptographic security of quantum key distribution". In: (2014). arXiv: 1409.3525 [quant-ph].

[223] Anne Broadbent and Martti Karvonen. *Categorical composable cryptography*. 2021. arXiv: 2105.05949 [cs.CR].

[224] *Private communication*. The error model and architecture are based on communication with Chris Balance and Tom Harty, mediated through Niel de Beaudrap, early on the NQIT project. 2016.

[225] Charles H. Bennett, Herbert J. Bernstein, Sandu Popescu, and Benjamin Schumacher. "Concentrating partial entanglement by local operations". In: *Phys. Rev. A* 53 (4 Apr. 1996), pp. 2046–2052. DOI: 10.1103/PhysRevA.53.2046.

[226] Ramil Nigmatullin, Christopher J Ballance, Niel De Beaudrap, and Simon C Benjamin. "Minimally complex ion traps as modules for quantum communication and computing". In: *New Journal of Physics* 18.10 (2016). ISSN: 13672630. DOI: 10.1088/1367-2630/18/10/103028.

[227] Peter Selinger. "Efficient Clifford+T Approximation of Single-Qubit Operators". In: *Quantum Info. Comput.* 15.1–2 (Jan. 2015), pp. 159–180. ISSN: 1533-7146. arXiv: 1212.6253 [quant-ph].

[228] Brett Giles and Peter Selinger. "Exact synthesis of multiqubit Clifford+T circuits". In: *Phys. Rev. A* 87 (3 Mar. 2013), p. 032332. DOI: 10.1103/PhysRevA.87.032332.

[229] Scott Aaronson and Alex Arkhipov. "Bosonsampling is Far from Uniform". In: *Quantum Info. Comput.* 14.15–16 (Nov. 2014), pp. 1383–1423. ISSN: 1533-7146. arXiv: 1309.7460 [quant-ph].

[230] Tim Menke et al. *Automated discovery of superconducting circuits and its application to 4-local coupler design*. 2020. arXiv: 1912.03322 [quant-ph].

[231] Thi Ha Kyaw et al. *Quantum computer-aided design: digital quantum simulation of quantum processors*. 2020. arXiv: 2006.03070 [quant-ph].

[232] Petar Jurcevic et al. *Demonstration of quantum volume 64 on a superconducting quantum computing system*. 2020. arXiv: 2008.08571 [quant-ph].

[233] Thomas Alexander et al. *Qiskit pulse: programming quantum computers through the cloud with pulses*. Aug. 2020. DOI: 10.1088/2058-9565/aba404.

[234] Matthew Amy and Vlad Gheorghiu. "staq—A full-stack quantum processing toolkit". In: *Quantum Science and Technology* 5.3 (June 2020), p. 034016. DOI: 10.1088/2058-9565/ab9359.

[235] Seyon Sivarajah et al. "t|ket⟩: A retargetable compiler for NISQ devices". In: *Quantum Science and Technology* (2020). DOI: 10.1088/2058-9565/ab8e92.

[236] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms". In: *Advanced Quantum Technologies* 2.12 (2019), p. 1900070. DOI: 10.1002/qute.201900070.

[237] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". In: *Nature* 549.7671 (Sept. 2017), pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879.

[238] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. "Efficient Quantum Algorithms for Simulating Sparse Hamiltonians". In: *Communications in Mathematical Physics* 270.2 (Mar. 2007), pp. 359–371. ISSN: 1432-0916. DOI: 10.1007/s00220-006-0150-x.

[239] Jonathan Romero et al. “Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz”. In: *Quantum Science and Technology* 4.1 (Oct. 2018), p. 014008. DOI: 10.1088/2058-9565/aad3e4.

[240] Panagiotis Kl. Barkoutsos et al. “Quantum algorithms for electronic structure calculations: Particle-hole Hamiltonian and optimized wave-function expansions”. In: *Phys. Rev. A* 98 (2 Aug. 2018), p. 022322. DOI: 10.1103/PhysRevA.98.022322.

[241] Nathan Wiebe, Christopher Granade, and D G Cory. “Quantum bootstrapping via compressed quantum Hamiltonian learning”. In: *New Journal of Physics* 17.2 (Feb. 2015), p. 022005. DOI: 10.1088/1367-2630/17/2/022005.

[242] Daniel Mills, Seyon Sivarajah, Travis L. Scholten, and Ross Duncan. *Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack: Experimental Data*. Version 1.0. Zenodo, May 2020. DOI: 10.5281/zenodo.3832121.

[243] J. Misra and David Gries. “A constructive proof of Vizing’s theorem”. In: *Information Processing Letters* 41.3 (1992), pp. 131–133. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(92)90041-S.

[244] Alexander Cowtan et al. “Phase Gadget Synthesis for Shallow Circuits”. In: *Electronic Proceedings in Theoretical Computer Science* 318 (Apr. 2020), pp. 214–229. ISSN: 2075-2180. DOI: 10.4204/eptcs.318.13.

[245] *pytket documentation*. URL: https://cqcl.github.io/pytket/build/html/index.html.

[246] *qiskit documentation*. URL: https://qiskit.org/documentation/.

[247] Sumeet Khatri et al. “Quantum-assisted quantum compiling”. In: *Quantum* 3 (May 2019), p. 140. ISSN: 2521-327X. DOI: 10.22331/q-2019-05-13-140.

[248] Ali JavadiAbhari et al. “ScaffCC: A Framework for Compilation and Analysis of Quantum Computing Programs”. In: *Proceedings of the 11th ACM Conference on Computing Frontiers*. CF ’14. Cagliari, Italy: Association for Computing Machinery, 2014. ISBN: 9781450328708. DOI: 10.1145/2597917.2597939.

[249] Jeff Heckey et al. “Compiler Management of Communication and Parallelism for Quantum Computation”. In: *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS ’15. Istanbul, Turkey: Association for Computing Machinery, 2015, pp. 445–456. ISBN: 9781450328357. DOI: 10.1145/2694344.2694357.

[250] Prakash Murali et al. “Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers”. In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS ’19. Providence, RI, USA: Association for Computing Machinery, 2019, pp. 1015–1029. ISBN: 9781450362405. DOI: 10.1145/3297858.3304075.

[251] Alexandru Paler. *On the Influence of Initial Qubit Placement During NISQ Circuit Compilation*. 2018. arXiv: 1811.08985 [quant-ph].

[252] Christopher Chamberland et al. “Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits”. In: *Phys. Rev. X* 10 (1 Jan. 2020), p. 011022. DOI: 10.1103/PhysRevX.10.011022.

[253] Prakash Murali, David C. Mckay, Margaret Martonosi, and Ali Javadi-Abhari. “Software Mitigation of Crosstalk on Noisy Intermediate-Scale Quantum Computers”. In: *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS ’20. Lausanne, Switzerland: Association for Computing Machinery, 2020, pp. 1001–1016. ISBN: 9781450371025. DOI: 10.1145/3373376.3378477.

[254] J. M. Pino et al. “Demonstration of the trapped-ion quantum CCD computer architecture”. In: *Nature* 592.7853 (Apr. 2021), pp. 209–213. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03318-4.

[255] Neereja Sundaresan et al. *Reducing Unitary and Spectator Errors in Cross Resonance with Optimized Rotary Echoes*. Dec. 2020. DOI: 10.1103/PRXQuantum.1.020318.

[256] Song Cheng, Jing Chen, and Lei Wang. “Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines”. In: *Entropy* 20.8 (Aug. 2018), p. 583. ISSN: 1099-4300. DOI: 10.3390/e20080583.

[257] Jin-Guo Liu and Lei Wang. “Differentiable learning of quantum circuit Born machines”. In: *Phys. Rev. A* 98 (6 Dec. 2018), p. 062324. DOI: 10.1103/PhysRevA.98.062324.

[258] Michael Kearns et al. “On the Learnability of Discrete Distributions”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC ’94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, pp. 273–282. ISBN: 0897916638. DOI: 10.1145/195058.195155.

[259] Ewin Tang. “A Quantum-Inspired Classical Algorithm for Recommendation Systems”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 217–228. ISBN: 9781450367059. DOI: 10.1145/3313276.3316310.

[260] Brian Coyle et al. “Quantum versus classical generative modelling in finance”. In: *Quantum Science and Technology* 6.2 (Apr. 2021), p. 024013. DOI: 10.1088/2058-9565/abd3db.

[261] Marco Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transport*. Ed. by C. J. C. Burges et al. https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf. 2013.

[262] Nader H. Bshouty and Jeffrey C. Jackson. “Learning DNF over the Uniform Distribution Using a Quantum Example Oracle”. In: *Proceedings of the Eighth Annual Conference on Computational Learning Theory*. COLT ’95. Santa Cruz, California, USA: Association for Computing Machinery, 1995, pp. 118–127. ISBN: 0897917235. DOI: 10.1145/225298.225312.

[263] Ryan Sweke, Jean-Pierre Seifert, Dominik Hangleiter, and Jens Eisert. *On the Quantum versus Classical Learnability of Discrete Distributions*. Mar. 2021. DOI: 10.22331/q-2021-03-23-417.

[264] L. G. Valiant. “A Theory of the Learnable”. In: *Commun. ACM* 27.11 (Nov. 1984), pp. 1134–1142. ISSN: 0001-0782. DOI: 10.1145/1968.1972.

[265] Srinivasan Arunachalam and Ronald de Wolf. *A Survey of Quantum Learning Theory*. 2017. arXiv: 1701.06806 [quant-ph].

[266] Srinivasan Arunachalam, Alex B. Grilo, and Aarthi Sundaram. *Quantum hardness of learning shallow classical circuits*. 2019. arXiv: 1903.02840 [quant-ph].

[267] Edward Farhi and Aram W Harrow. *Quantum Supremacy through the Quantum Approximate Optimization Algorithm*. 2019. arXiv: 1602.07674 [quant-ph].

[268] Vicente Leyton-Ortega, Alejandro Perdomo-Ortiz, and Oscar Perdomo. *Robust Implementation of Generative Modeling with Parametrized Quantum Circuits*. 2019. arXiv: 1901.08047 [quant-ph].

[269] Arthur Gretton et al. *A Kernel Method for the Two-Sample Problem*. 2008. arXiv: 0805.2368 [cs.LG].

[270] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. “Kernel Methods in Machine Learning”. In: *The Annals of Statistics* 36.3 (2008), pp. 1171–1220. ISSN: 00905364.

[271] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. 1st. Springer Publishing Company, Incorporated, 2018. ISBN: 3319964232.

[272] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032331. DOI: 10.1103/PhysRevA.99.032331.

[273] Bharath K. Sriperumbudur et al. *On integral probability metrics, ϕ-divergences and binary classification*. 2009. arXiv: 0901.2698 [cs.IT].

[274] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. *On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests*. 2015. arXiv: 1509.02237 [math.ST].

[275] Aude Genevay, Gabriel Peyre, and Marco Cuturi. “Learning Generative Models with Sinkhorn Divergences”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Apr. 2018, pp. 1608–1617.

[276] Jean Feydy et al. *Interpolating between Optimal Transport and MMD using Sinkhorn Divergences*. 2018. arXiv: 1810.08278 [math.ST].

[277] Jonathan Weed and Francis Bach. *Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance*. 2017. arXiv: 1707.00087 [math.PR].

[278] Aude Genevay et al. *Sample Complexity of Sinkhorn divergences*. 2019. arXiv: 1810.02733 [math.ST].

[279] Srinivasan Arunachalam et al. *Two new results about quantum exact learning*. 2020. arXiv: 1810.00481 [quant-ph].

[280] Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra. “Learning-with-errors problem is easy with quantum samples”. In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032314. DOI: 10.1103/PhysRevA.99.032314.

[281] Michael Kearns. “Efficient Noise-Tolerant Learning from Statistical Queries”. In: *J. ACM* 45.6 (Nov. 1998), pp. 983–1006. ISSN: 0004-5411. DOI: 10.1145/293347.293351.

[282] Dominik Hangleiter, Martin Kliesch, Jens Eisert, and Christian Gogolin. “Sample Complexity of Device-Independently Certified “Quantum Supremacy””. In: *Phys. Rev. Lett.* 122 (21 May 2019), p. 210502. DOI: 10.1103/PhysRevLett.122.210502.

[283] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. DOI: 10.1017/9781108135252.

[284] Lennart Bittel and Martin Kliesch. *Training variational quantum algorithms is NP-hard – even for logarithmically many qubits and free fermionic systems*. 2021. arXiv: 2101.07267 [quant-ph].

[285] Feng Pan and Pan Zhang. *Simulating the Sycamore quantum supremacy circuits*. 2021. arXiv: 2103.03074 [quant-ph].

[286] Anne Broadbent and Elham Kashefi. “Parallelizing quantum circuits”. In: *Theoretical Computer Science* 410.26 (2009), pp. 2489–2510. ISSN: 03043975. DOI: 10.1016/j.tcs.2008.12.046.

[287] Joseph Emerson, Etera Livine, and Seth Lloyd. “Convergence conditions for random quantum circuits”. In: *Phys. Rev. A* 72 (6 Dec. 2005), p. 060302. DOI: 10.1103/PhysRevA.72.060302.

[288] Fernando G. S. L. Brandão, Aram W. Harrow, and Michał Horodecki. “Local Random Quantum Circuits are Approximate Polynomial-Designs”. In: *Communications in Mathematical Physics* 346.2 (Sept. 2016), pp. 397–434. ISSN: 1432-0916. DOI: 10.1007/s00220-016-2706-8.

[289] Andrew Fagan and Ross Duncan. “Optimising Clifford Circuits with Quantomatic”. In: *Electronic Proceedings in Theoretical Computer Science* 287 (Jan. 2019), pp. 85–105. ISSN: 2075-2180. DOI: 10.4204/eptcs.287.5.

[290] M Blaauboer and R L de Visser. “An analytical decomposition protocol for optimal implementation of two-qubit entangling gates”. In: *Journal of Physics A: Mathematical and Theoretical* 41.39 (Sept. 2008), p. 395307. DOI: 10.1088/1751-8113/41/39/395307.

[291] Easwar Magesan and Jay M. Gambetta. “Effective Hamiltonian models of the cross-resonance gate”. In: *Phys. Rev. A* 101 (5 May 2020), p. 052308. DOI: 10.1103/PhysRevA.101.052308.

[292] Timothy J. Proctor et al. “Direct Randomized Benchmarking for Multiqubit Devices”. In: *Phys. Rev. Lett.* 123 (3 July 2019), p. 030503. DOI: 10.1103/PhysRevLett.123.030503.

[293] Jay M. Gambetta et al. “Characterization of Addressability by Simultaneous Randomized Benchmarking”. In: *Phys. Rev. Lett.* 109 (24 Dec. 2012), p. 240504. DOI: 10.1103/PhysRevLett.109.240504.

[294] Arnaud Carignan-Dugas, Kristine Boone, Joel J Wallman, and Joseph Emerson. “From randomized benchmarking experiments to gate-set circuit fidelity: how to interpret randomized benchmarking decay parameters”. In: *New Jour-*

nal of Physics 20.9 (Sept. 2018), p. 092001. DOI: 10.1088/1367-2630/aadcc7.

Appendix A

Methods for Classically Simulating Noisy Networked Quantum Architectures

A.1 Expanded Circuit Descriptions

A.1.1 IQP-MBQC Circuit in NQIT Q20:20 Gate Set

As discussed in Section 1.5, for constant $\theta = \pi/8$, each IQP instance is fully defined by a binary matrix $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$. For example, $\boldsymbol{Q}$ of Figure 1.12 corresponds to

$$C = \exp\left(i\frac{\pi}{8}\mathsf{X}_1\mathsf{X}_3\right)\exp\left(i\frac{\pi}{8}\mathsf{X}_2\mathsf{X}_3\right).$$

To sample from $\langle 0^{\otimes n}|C|0^{\otimes n}\rangle$ via MBQC, as described in Protocol 1.5.2, we must measure the ancillary qubits in the $\left\{|0_{2\frac{\pi}{8}}\rangle, |1_{2\frac{\pi}{8}}\rangle\right\}$ basis of (1.24). Measurements in the basis $\{|0_{2\theta}\rangle, |1_{2\theta}\rangle\}$ can be simulated, when only measurements in the computational basis are available, by first rotating the qubit to be measured by $\mathsf{HXRZ}^{2\theta}\mathsf{XH}$. The correct rotation for $\theta = \pi/8$ is HXTXH

We can incorporate the classical corrections required by Protocol 1.5.2 into the circuit by adding CX gates according to the same pattern used to produce the resource state initially [286]. Since those corrections do not need to be physically executed, because of their equivalence to classical post-processing, we do not add any noise to them. We conclude that the corresponding MBQC pattern of Figure 1.12 can be written in circuit form as in Figure A.1. This describes an implementation of IQP using the gate set which is available to the NQIT Q20:20 device as discussed in Section 2.2.2, and is the circuit we will implement in our simulator.

A.1.2 NQIT Q20:20 Noise Functions

In Protocol A.1.1 we give the necessary tools to implement the NQIT Q20:20 noise model of Section 2.2.3 in the gate based model, which may be understood by the

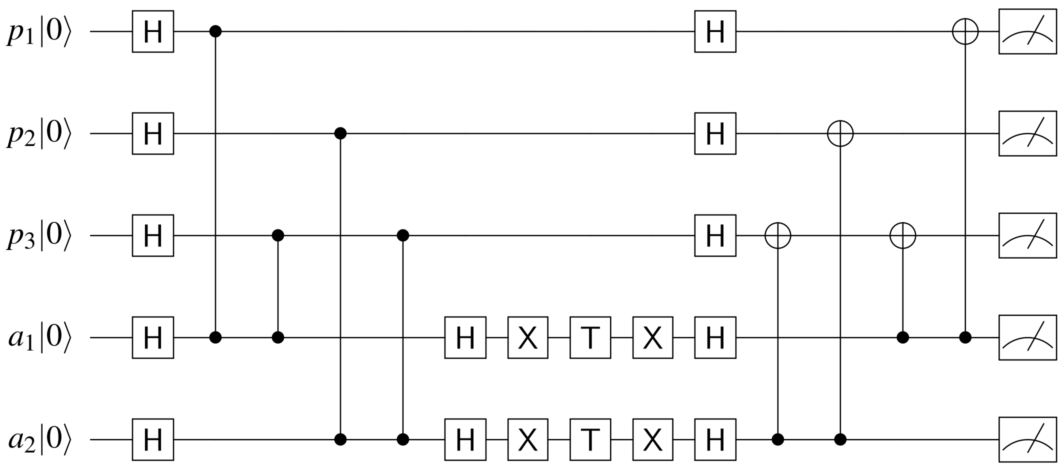


Figure A.1: **Circuit which implements the MBQC pattern of Figure 1.12**. Measurements have been delayed until the end. The final CX gates perform the necessary adaptive corrections.

simulator. Noise is added to a circuit as follows, and in accordance with the discussion on simulations of noise using pure state simulators conducted in Section 1.3.3. All operations are considered independently. Noise gates corresponding to operation based errors are inserted at an operation's position in the circuit at random, with type and probability according to the rates of Section 2.2.3. For each of those operations, a nested loop iterates over all qubits in the system and randomly applies the two time-based errors. First the execution time needed for the current operation is calculated by considering the times given in Section 2.2.3. Then, at each qubit in the loop, an appropriate noise gate is added according to a Poisson process with the rates listed, again, in Section 2.2.3.

A.2 Numerical Experiment Details

A.2.1 IQP-MBQC Experiments

Simulator Benchmarking Experiment of Section 2.3.1

Generating random unrestricted IQP-MBQC instances is equivalent to randomly populating $\boldsymbol{Q}$ with zeros and ones. The description in Appendix A.1.1 of how to convert a given X-program $\boldsymbol{Q}$ to a particular circuit lets us control the T gates count t. We saw that every individual exponential (row in $\boldsymbol{Q}$) corresponds exactly to $t = 1$, and the number of primary qubits has no effect on t. We want T gate counts of no more than 20 in order to achieve feasible run-times.

One trial consists of generating a random IQP instance, obtaining the true probability of measuring the $|0\rangle^n$ using brute-force, and solving them with the Bravyi-Gosset Simulator 20 times. Each instance is created by randomly populating with binary values a matrix $\boldsymbol{Q}$ of randomly picked dimensions in $[5,15] \times [5,12]$. This corresponds to

Protocol A.1.1 Gate based description of the NQIT Q20:20 noise to be used by the simulator. Here $\mathbb{P}(k;\lambda)$ is the probability that k events occur in a Poisson distribution with mean λ. The variables listed here assume the current NQIT Q20:20 noise levels but are altered in our experiments of Section 2.3.3 and can be set to 0 in the perfect case.

```
1:  TimeInTrapOperation = 0.5ms
2:  TimeLinkingOperation = 1.5s
3:  TimePreparation = 1.25ms
4:  TimeMeasurement = 2.25ms
5:
6:  ProbTwoQubitOperationSingleQubit = $5.5 \times 10^{-5}$
7:  ProbTwoQubitOperationTwoQubit = $6 \times 10^{-5}$
8:  ProbSingleQubitOperation = $1.5 \times 10^{-6}$
9:  ProbMeasurement = $5 \times 10^{-4}$
10: ProbPreparation = $2 \times 10^{-4}$
11: ProbDephasing = $7.2 \times 10^{-3}$
12: ProbDepolarising = $9 \times 10^{-3}$
```

Protocol continues below...

$n \in [5,12]$ and $t \in [5,15]$ where the complexity in the brute-force case is determined by n, and in the case of the Bravyi-Gosset Simulator, by t.

The experiment consists of 20 trials, with the mean of the simulator output in each trial compared to the brute-force case to give the coefficient of determination.

NQIT Q20:20 Noise and Architecture Restricted Experiment of Section 2.3.2

We again generate random IQP-MBQC circuits, but under the restrictions described in Section 2.3.2. Rather than a full matrix, $\boldsymbol{Q}$, it is now sufficient for each ancillary qubit, a_i, in an ion trap, i, to have corresponding bit strings, i^0 and i^1, indicating the entanglement patterns between itself and qubits in it and its neighbouring ion trap.

Details of the circuit simulated can be seen in Protocol A.2.1. Once the circuit is simulated, we calculate the probability that an NQIT Q20:20 implementation would measure a random bit string b. One noisy run consists of simulating the circuit produced from Protocol A.2.1, using fixed i^0, i^1, b, 20 times to calculate the mean and standard deviation. Then a new tuple i^0, i^1, b is generated and the process is repeated for the next trial. A perfect run is equivalent but with the noise values set to 0, with the perfect and noisy pair forming one trial. In total the experiment consists of 20 trials.

Notice that we are being pessimistic in Protocol A.2.1 by assuming that there is no parallelism in the gate applications. As such we apply time based noise after each gate. We have also simplified the operation of swapping to a single operation, rather than a protocol as seen in Protocol 2.3.1. This reduces the simulation time while roughly maintaining the noise impact, as the time based noise should dominate here.

Protocol A.1.1 Continued

```
13: function RandomPauli($i$,$p$)
14:     Enact a Pauli gate, selected uniformly at random, on qubit $i$ with probability $p$
15: end function
16:
17: function DephasingNoise($t$, $q$)
18:     Enact $\mathsf{Z}_q$ with probability $1-\mathbb{P}(0;t\text{ProbDephasing})$
19: end function
20:
21: function DepolarisingNoise($t$, $q$)
22:     RandomPauli($q$, $1-\mathbb{P}(0;t\text{ProbDepolarising})$)
23: end function
24:
25: function TimeBaseNoise($t$)
26:     for all $q \in$ qibits do                          ▷ Noise acts on all qubits
27:         DephasingNoise($t$, $q$)
28:         DepolarisingNoise($t$, $q$)
29:     end for
30: end function
31:
32: function TwoQubitNoise($i$, $j$)
33:     RandomPauli($i$, ProbTwoQubitOperationSingleQubit)
34:     RandomPauli($j$, ProbTwoQubitOperationSingleQubit)
35:     Enact $\mathsf{Z}_i \otimes \mathsf{Z}_j$ with probability ProbTwoQubitOperationTwoQubit
36: end function
37:
38: function SingleQubitNoise($q$)
39:     RandomPauli($q$, ProbSingleQubitOperation)
40: end function
41:
42: function PreparationNoise(q)
43:     Enact $\mathsf{X}_q$ with probability ProbPreparation
44: end function
45:
46: function MeasurementNoise($q$)
47:     Enact $\mathsf{X}_q$ with probability ProbMeasurement
48: end function
```

Protocol A.2.1 Code producing a noisy IQP-MBQC circuit, to be implemented by the simulator, as discussed in Section 2.3.2. We use i to index the ion traps, and to represent the set of $K' - 2$ available primary qubits which each trap contains (K' minus 1 qubit c_i to receive the ancillary qubit from it's neighbour, minus one ancillary qubit a_i).

Input: For every ion trap, i, two strings, i^0, i^1. Bit string b.
Output: Noisy circuit.

```
1: for all $q \in$ qubits do
2:     INITIALISE($q$)                    ▷ Recall, initialisation is in the $|+\rangle$ state
3:     PREPARATIONNOISE($q$)
4: end for
5:
6: for all $i \in$ ion traps, except the last do
7:     for all $q \in i$ do
8:         if $i^0_q = 1$ then
9:             Enact CZ between $a_i$ and $q$
10:            TWOQUBITNOISE($a_i$, $q$)
11:            TIMEBASEDNOISE(TimeInTrapOperation)
12:        end if
13:    end for
14: end for
15:
16: for all $i \in$ ion traps, except the last, such that $i$ is even do
17:     SWAP($a_i$, $c_{i+1}$)              ▷ Move ancillary qubits to neighbouring ion trap
18: end for
19: TIMEBASEDNOISE(TimeLinkingOperation + TimeMeasurement)
20:
21: for all $i \in$ ion traps, except the last, such that $i$ is odd do
22:     SWAP($a_i$, $c_{i+1}$)
23: end for
24: TIMEBASEDNOISE(TimeLinkingOperation + TimeMeasurement)
25:
26: for all $i \in$ ion traps, except the first do
27:     for all $q \in i$ do
28:         if $(i-1)^1_q = 1$ then
29:             Enact CZ between $a_{i-1}$ and $q$
30:             TWOQUBITNOISE($a_{i-1}$, $q$)
31:             TIMEBASEDNOISE(TimeInTrapOperation)
32:         end if
33:     end for
34: end for
```

Protocol continues below...

Protocol A.2.1 Continued

```
35: for all $i \in$ ion traps, except the first do        ▷ Measurement basis correction
36:     Enact $\mathsf{H}_{a_{i-1}}$
37:     SINGLEQUBITNOISE($a_{i-1}$)
38:     TIMEBASEDNOISE(TimeInTrapOperation)
39: end for
40: for all $i \in$ ion traps, except the first do
41:     Enact $\mathsf{X}_{a_{i-1}}$
42:     SINGLEQUBITNOISE($a_{i-1}$)
43:     TIMEBASEDNOISE(TimeInTrapOperation)
44: end for
45: for all $i \in$ ion traps, except the first do
46:     Enact $\mathsf{T}_{a_{i-1}}$
47:     SINGLEQUBITNOISE($a_{i-1}$)
48:     TIMEBASEDNOISE(TimeInTrapOperation)
49: end for
50: for all $i \in$ ion traps, except the first do
51:     Enact $\mathsf{X}_{a_{i-1}}$
52:     SINGLEQUBITNOISE($a_{i-1}$)
53:     TIMEBASEDNOISE(TimeInTrapOperation)
54: end for
55:
56: for all $q \in$ qubits do
57:     Enact $\mathsf{H}_q$
58:     SINGLEQUBITNOISE($q$)
59:     TIMEBASEDNOISE(TimeInTrapOperation)
60: end for
61:
62: for all $i \in$ ion traps, except the last do        ▷ CX seen at end of Figure A.1
63:     for all $q \in i$ do
64:         if $i^0_q = 1$ then
65:             Enact CX between $a_i$ and $q$
66:         end if
67:     end for
68: end for
69:
70: for all $i \in$ ion traps, except the first do
71:     for all $q \in i$ do
72:         if $(i-1)^1_q = 1$ then
73:             Enact CX between $a_{i-1}$ and $q$
74:         end if
75:     end for
76: end for
```

Protocol continues below...

Protocol A.2.1 Continued

```
77: for all $i \in$ ion traps do
78:     for all $q \in i$ and $a_{i-1}$ for all but the first ion trap do
79:         MeasurementNoise($q$)
80:     end for
81: end for
82: Measure($b$)    ▷ Give the probability of measuring $b$ in the Computational basis
```

A.2.2 2D-DQS Experiments of Section 2.3.2 and Section 2.3.3

Instead of only 2-steps, as it is in the 1D case, we need 4 steps to build a 2D grid resource state. We achieve this by entangling sequentially:

- Even-indexed columns' qubits to their right neighbours
- Odd-indexed columns' qubits to their right neighbours
- Even-indexed rows' qubits to their bottom neighbours
- Odd-indexed rows' qubits to their bottom neighbours

Having performed the entanglement we are left to apply the T gates and measure. We track the qubits on which we apply the T gates using the bit string τ which takes the value 1 at the locations where a T gate is applied.

We calculate the amplitude of a randomly selected output, b, for each instance in order to simulate sampling. We calculate several trials where for each we:

- Generate a uniformly random $\tau \in [0,1]^{20}$ to give a 4x5 circuit as in Protocol 1.5.1.
- Generate a random bit string, b, to calculate the amplitude of.
- Solve 20 times and take the mean and standard deviation. This is a perfect run.
- Generate 20 random noisy circuits, one per noisy run, based on the perfect one by inputting τ into Protocol A.2.2. In the case of Section 2.3.3 we will use different values for the variables of Protocol A.1.1, as discussed there.
- For each noisy run, solve the circuit 20 times and calculate the mean. The result is a vector of length 20 containing these mean values.

Attempts to reduce the standard deviation of the noisy runs by increasing the number of times the computation is performed during each run were not effective, suggesting the deviation is a result of the noise.

Protocol A.2.2 Code producing a noisy 2D-DQS circuit, to be implemented by the simulator, as discussed in Section 2.3.2 and Section 2.3.3. We will index traps (and equivalently, in this case, qubits) by the row, n, and column, m, where they appear in the square grid.

Input: Bit strings τ and b.
Output: Noisy circuit.

```
for all $q \in$ qubits do                              ▷ Initialise $|+\rangle$ states
    INITIALISE($q$)
    PREPARATIONNOISE($q$)
end for

for $p \in \{odd, even\}$ do                           ▷ Entangle columns of lattice
    for $\{n, m : n \in p\}$ do
        Enact CZ between $(n,m)$ and $(n+1,m)$
        TWOQUBITNOISE($(n,m),(n+1,m)$)
        TIMEBASEDNOISE(TimeInTrapOperation)
    end for
end for

for $p \in \{odd, even\}$ do                           ▷ Entangle rows of lattice
    for $\{n, m : m \in p\}$ do
        act CZ between $(n,m)$ and $(n,m+1)$
        TWOQUBITNOISE($(n,m),(n,m+1)$)
        TIMEBASEDNOISE(TimeInTrapOperation)
    end for
end for

for $q \in qubits$ do                   ▷ Enact T gate according to original circuit
    if $\tau_i = 1$ then
        Enact $\mathsf{T}_i$
        SINGLEQUBITNOISE($i$)
        TIMEBASEDNOISE(TimeInTrapOperation)
    end if
end for

for $q \in qubits$ do
    Enact $\mathsf{H}_q$                      ▷ Ajust to measure in the Hadamard basis
    SINGLEQUBITNOISE($q$)
    TIMEBASEDNOISE(TimeInTrapOperation)
end for

for $q \in qubits$ do
    MEASUREMENTNOISE($q$)
end for
MEASURE($b$)
```

Appendix B

Application-Motivated, Holistic Benchmarking of a Full Quantum Computing Stack

B.1 Exponential Distribution

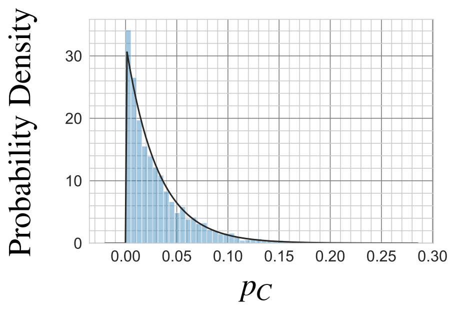
The *exponential distribution*, with *rate* λ, is a probability distribution with the probability density function

$$\Pr(x) = \lambda e^{-\lambda x}.$$

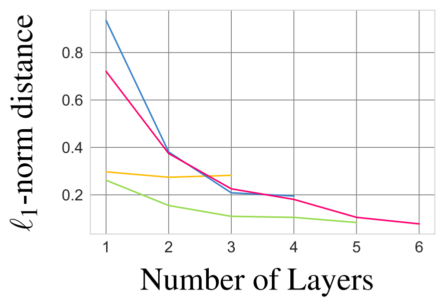
This is the distribution of waiting times between events in a Poisson process. We are concerned with showing that output probabilities of the circuits classes considered here are exponentially distributed. Such a property is a signature of quantum chaos, and that a class of circuits is approximately Haar random [62, 287]. It also allows for the calculation of both the ideal value of the cross-entropy, and the ideal heavy output probability as discussed in Section 1.6.4. This in turn allows us to fully exploit Cross-Entropy Benchmarking and Heavy Output Generation Benchmarking. Here we will argue numerically which of the circuits we introduce in Section 3.2 generate output probabilities of this form,[1] and discuss the implications when they do not.

We also demonstrate why the circuit depths used in Section 3.2 are necessary to generate output probabilities of this form. To do this we generate 100 circuits of each type and number of layers, where a layer is as defined in the respective Algorithms of Section 3.2. We then calculate the ideal output probabilities using classical simulation and compare this distribution of output probabilities to the exponential distribution. In the case of square circuits and deep circuits, we notice a better approximation of the exponential distribution by the distribution of output probabilities, measured by the ℓ_1-norm distance between the two, as the number of layers increases. We can use this to isolate the number of layers at which the difference approaches its minimum.

[1]This numerical approach to demonstrating properties of distributions of output probabilities from particular circuit classes parallels that taken in other work on benchmarking [24, 61, 62, 85].



(a) The distribution of output probabilities from a circuit C, where C is a 5 qubit circuit, from the square circuits class as defined in Protocol 3.2.2.



(b) The ℓ_1-norm distance between the distribution of output probabilities and the exponential distribution $2^n e^{-2^n x}$, where n is the number of qubits. A layer is defined as in Protocol 3.2.2. Colours correspond to numbers of qubits in the following way: 2 [🟨], 3 [🟦], 4 [🟩], 5 [🟥].

Figure B.1: **Exponential distribution fitting data for square circuits**.

B.1.1 Square Circuits

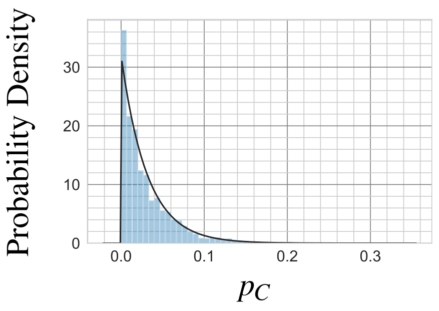
The exponential form of the distribution of the output probabilities from random circuits similar to square circuits has been established [62, 85]. As the procedure we use to generate square circuits, seen in Protocol 3.2.2, differs slightly from that used for other similar random circuits [62, 85, 212], we explore the distribution of its output probabilities here.

The relevant results are seen in Figure B.1. In particular, it can be seen from Figure B.1b that the minimum value of ℓ_1-norm distance between the distribution of output probabilities and the exponential distribution is approached at a number of layers equal to the number of qubits, justifying our choice of layer numbers in Protocol 3.2.2. It may be that asymptotically the number of layers required is sub-linear [62], although for the circuit sizes used here a linear growth in depth is appropriate. Figure B.1a illustrates the closeness of fit of the two distributions.

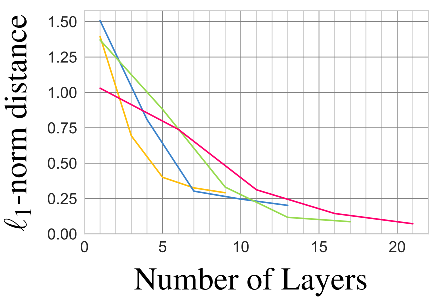
B.1.2 Deep Circuits

Unlike with square circuits, there is no precedent for utilising deep circuits to generate exponentially distributed output probabilities, as we do here. This allows us to use deep circuits as a uniquely insightful benchmark of the performance of quantum computing stacks, grounded both in the theoretical results of Section 3.1, and in pertinent applications.

The relevant results are seen in Figure B.2. In particular, it can be seen from Figure B.2b that the minimum value of ℓ_1-norm distance between the distribution of output probabilities and the exponential distribution is approached at a number of layers equal to three times the number of qubits, plus one, justifying our choice of layer numbers in Protocol 3.2.3. Figure B.2a illustrates the closeness of fit of the two distributions.



(a) The distribution of output probabilities from a circuit C, where C is a 5 qubit circuit, from the deep circuits class as defined in Protocol 3.2.3.



(b) The ℓ_1-norm distance between the distribution of output probabilities of deep circuits and the exponential distribution $2^n e^{-2^n x}$, where n is the number of qubits. A layer is defined as in Protocol 3.2.3. Colours correspond to numbers of qubits in the following way: 2 [🟨], 3 [🟦], 4 [🟩], 5 [🟥].

Figure B.2: **Exponential distribution fitting data for deep circuits**.

The depth required to achieve an exponential distribution of outcome probabilities with deep circuits is greater than is the case for square circuits. Indeed, random circuits were initially introduced as the shallowest circuits required to generate such output probabilities [62]. This sacrifice in depth is made to achieve a benchmark which is uniquely application motivated, as discussed in Section 3.2.

B.1.3 Shallow Circuits

Unlike in the case of square circuits and deep circuits, the output probabilities of shallow circuits are not exponentially distributed. This is unsurprising since random circuits with this limited connectivity are thought to require at least depth $O(\sqrt{n})$ to create such a feature [24, 66, 288]. This has the unfortunate side effect that Cross-Entropy Benchmarking cannot be used.

While it is also true that the predictions made about the ideal heavy output probability also do not apply, a study of the heavy output probability is still of interest. In particular, while we cannot connect the benchmark to the HOG problem of Problem 1, we can compare the probability of generating heavy outputs to the ideal probability of producing heavy outputs, as calculated by classical simulation.

B.2 Compilation Strategies

This section details the compilation strategies explored in each of our experiments. For the circuit families and figures of merit investigated here, the compilation strategies we used were designed and empirically confirmed to perform well at the compilation tasks at hand. The versions of each package used are listed in Table B.1.

Package	**Version**
Qiskit [114, 246]	0.12.0
pytket [235, 245]	0.3.0

Table B.1: **Packages used in this work, and their corresponding versions**.

noise-unaware pytket and noise-aware pytket The noise-unaware pytket and noise-aware pytket compilation strategies are generated using Protocol B.2.1. noise-unaware pytket is generated by passing `False` as input to Protocol B.2.1, and noise-aware pytket by passing `True`.

Of particular interest are the following functions:

`OptimiseCliffors`: Simplifies Clifford gate sequences [289].

`KAKDecomposition`: Identifies two-qubit sub-circuits with more than 3 CXs and reduces them via the KAK/Cartan decomposition [290].

`route`: Modifies the circuit to satisfy the architectural constraints [173]. This will introduce SWAP gates.

`noise_aware_placement`: Selects initial qubit placement taking in to account reported device gate error rates [235].

`line_placement`: Attempts to place qubits next to those they interact with in the first few time slices. This does not take device error rates into account.

noise-unaware Qiskit and noise-aware Qiskit The noise-unaware Qiskit and noise-aware Qiskit compilation strategies, as defined in Protocol B.2.2, are heavily inspired by `level_3_passmanager`, a preconfigured compilation strategy made available in Qiskit. noise-unaware Qiskit is generated by passing `noise_aware` as `False` in Protocol B.2.1, and noise-aware Qiskit by passing `True`.

Where possible we passed `stochastic` as `True` in order to use `StochasticSwap` instead of `BasicSwap` during the swap mapping pass. In general, `StochasticSwap` generates circuits with lower depth; however, for the versions listed in Table B.1, it proved faulty for some circuit sizes and device coupling maps used in this work. `StochasticSwap` may also result in repeated measurement of the same qubit, which cannot be implement. Repeated compilation attempts may therefore be necessary, and if this fails the circuit is not included in the plots of Section 3.4.

Of particular note are the following functions:

`NoiseAdaptiveLayout`: Selects initial qubit placement based on minimising readout error rates [250].

`DenseLayout`: Chooses placement by finding the most connected subset of qubits.

`Unroller`: Decomposes unitary operation to desired gate set.

Protocol B.2.1 pytket compilation strategies. The passes listed here are named as in the documentation for pytket [245], where additional detail on their actions can be found.

Input: `noise_aware` $\in$ {`True`,`False`}

```
1: OptimiseCliffords
2: KAKDecomposition
3:
4: RebaseToRzRx                              ▷ Convert to IBM gate set
5: CommuteRzRxThroughCX
6:
7: if noise_aware then
8:     noise_aware_placement
9: else
10:    line_placement
11: end if
12:
13: route
14: decompose_SWAP_to_CX
15: redirect_CX_gates                        ▷ Orientate CX to coupling map
16:
17: OptimisePostRouting     ▷ Optimisation preserving placement and orientation
```

`StochasticSwap`: Adds SWAP gates to adhere to coupling map using a randomised algorithm.

`BasicSwap`: Produces a circuit adhering to coupling map using a simple rule: CX gates in the circuit which are not supported by the hardware are preceded with necessary SWAP gates.

only pytket routing In this case we perform, in the order as listed, the pytket operations: `route`, `decompose_SWAP_to_CX`, and `redirect_CX_gates`. We then account for the architecture gate set, without any further optimisation.

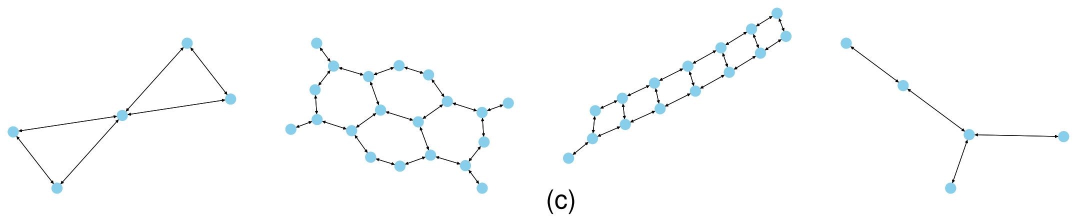
B.3 Device Data

Two device properties leveraged by our compilation strategies are the *coupling maps*, describing the connectivity of the qubits and in which directions CX gates can be performed, and the *calibration information*, describing the noise levels of the device. These properties, and devices noise levels in particular, are considered valuable benchmarks of the performance of the device in their own right.

These properties are collectively influential in noise-aware compiling, as detailed in Appendix B.2. There circuits are compiled to adhere to the device's coupling map, while also aiming to minimise some function of the calibration information. Because full quantum computing stack holistic benchmarking encompasses the circuit compi-

Protocol B.2.2 Qiskit compilation strategies. The passes listed here are named as in the documentation for Qiskit [246], where additional detail on their actions can be found.

```
Input:
  noise_aware ∈ {True,False}
  stochastic ∈ {True,False}

Unroller

if noise_aware then
    NoiseAdaptiveLayout
else
    DenseLayout
end if
AncillaAllocation                        ▷ Assign idle qubits as ancillas

if stochastic then
    StochasticSwap
else
    BasicSwap
end if

Decompose(SwapGate)                      ▷ Decompose SWAP to CX
CXDirection                              ▷ Orientate CX to coupling map

                                         ▷ Gather 2 qubit blocks
Collect2qBlocks
ConsolidateBlocks

Unroller                                 ▷ Unroll two-qubit blocks
Optimize1qGates                          ▷ Combine chains of one-qubit gates
CXDirection
```



(a) ibmqx2 (b) ibmq_singapore (c) ibmq_16_melbourne (d) ibmq_ourense

Figure B.3: **Coupling maps of the devices studied in this work.** Vertices, represented by blue circles, correspond to qubits, while edges are directed from the control to the target qubits of permitted two-qubit gates.

lation strategies, it provides a novel way of using device information to benchmark an entire system, instead of simply the physical qubits which comprise it.

B.3.1 Device Coupling Maps

A *coupling map* of a device is a graphical representation of how two-qubit gates can be applied across the device. In this representation, each qubit is represented by a vertex, with directed edges joining qubits between which a two-qubit gate can be applied. For the devices considered here, this two-qubit gate is a CX gate, implemented using the cross-resonance interaction of transmon qubits [291]. The direction of the edge is from the control to the target qubit of the CX gate, with bi-directional edges indicating that both qubits can be used as either the control or target. The coupling maps of the devices investigated in this work are shown in Figure B.3. For those devices all edges are bi-directional, although this is not typical when the asymmetric CX is employed.

As discussed in Section 3.3, a trade-off exists between the connectivity of the device and the number of two-qubit gates necessary to implement a given circuit. More highly connected coupling maps typically require fewer two-qubit gates to implement a fixed unitary than less connected ones, owing to the reduced need for SWAP gates to account for discrepancies between the coupling maps of the uncompiled circuit and the device. While this reduced depth can reduce the impact of time based noise channels, this is counterbalanced by the higher levels of crosstalk experienced by qubits corresponding to vertices with high degree in the device's coupling map [252].

B.3.2 Device Calibration Information

The noise-aware tools employed by the compilation strategies explored in this work consider three kinds of errors which can occur, namely: readout error, single-qubit gate error, and two-qubit gate error. For the devices provided through IBM Quantum, this information is contained in calibration data which is accessible using tools in the Qiskit library, and is updated twice daily. The experiments in this work were conducted between 2020-01-29 and 2020-02-10 with the calibration data in Figure B.4 and Figure B.5 aggregated over this time period.

Here we make the simplifying assumption, discussed in Section 1.3.3, that readout error acts independently on each qubit. As such, if readout error corresponds to returning

"0" when the proper label is "1", or vice-versa, then the *readout error rate*, denoted, ε^{a}, is calculated as

$$\varepsilon^{\mathrm{a}} = \frac{\Pr(\text{“0”}||1\rangle) + \Pr(\text{“1”}||0\rangle)}{2}.$$

ε^{a} is estimated by repeatedly preparing a qubit in a known state, immediately measuring it, and then counting the number of times the measurement returns the wrong label. This value, for the devices explored in this work, is reported in Figure B.4a.

Errors affecting the gates of the device correspond to an incorrect operation applied by the device. There are many ways to quantify the effect of this error, with IBM Quantum's devices reporting randomised benchmarking (RB) numbers [18, 19].[2] The RB number, ε^{C}, is estimated by running many self-inverting Clifford circuits, consisting of m layers of gates drawn from the n-qubit Clifford group, inverted at layer $m+1$. The *survival probability*, which is the probability the input state is unchanged, can then be estimated. Under a broad set of noise models and assumptions [18, 294], this survival probability can be shown to decay exponentially with m. Consequently, it can be estimated by fitting a decay curve of the form $Ap^m + B$. The RB number is related to $p \in [0,1]$, called the *depolarisation/decay rate*, by

$$\varepsilon^{\mathrm{C}} = (1-p)(1-1/D),$$

where $D = 2^n$, and n is the number of qubits acted on by the Clifford gates. ε^{C}, which is also referred to as the *error per Clifford* of the device, is minimised at $p = 1$, in which case the survival probability is constant and set by the state preparation and measurement errors.

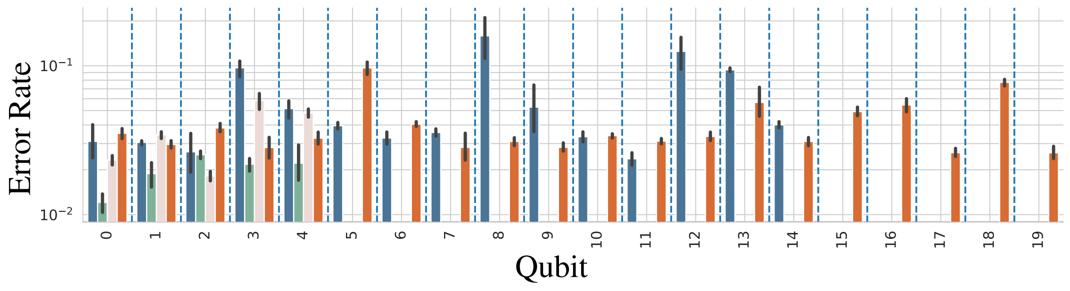
The Clifford gates necessary for RB must be compiled to the native gate set of the device. Using an estimate of ε^{C}, an estimate of the *error per gate*, $\varepsilon^{\mathrm{g}}_{G}$, for a gate G, can be obtained by multiplying ε^{C} by a factor related to the average number of uses of G when implementing a random Clifford operation:

$$\varepsilon^{\mathrm{g}}_{G} \sim \varepsilon^{\mathrm{C}} \times \#\text{ uses of } G \text{ per Clifford}.$$

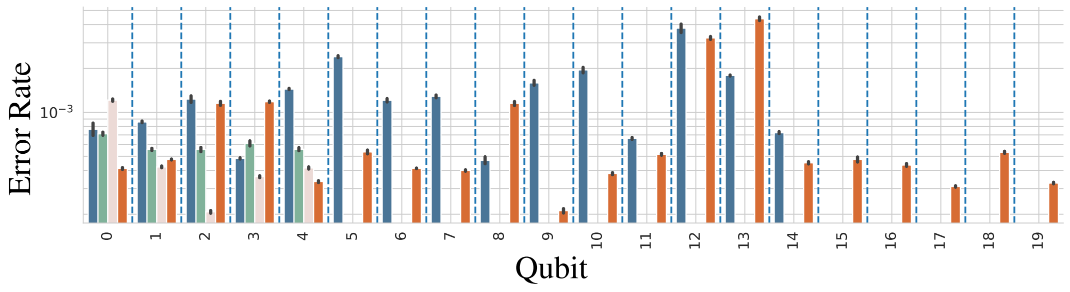
Values for $\varepsilon^{\mathrm{g}}_{\mathsf{U}_2}$, the error per gate for U_2 gates, can be found in Figure B.4b, and $\varepsilon^{\mathrm{g}}_{\mathsf{CX}}$, that for CX gates, in Figure B.5. The commonly reported average fidelity for U_3 gates is $1-\left(1-\varepsilon^{\mathrm{g}}_{\mathsf{U}_2}\right)^2$.

Several important noise channels, most notably crosstalk, are not included in the device calibration data. As shown in Section 3.4, the effects of this noise can be inferred through the application-motivated benchmarks we introduce in this work, by showing the trade-off between connectivity of the device and crosstalk [252].

[2]There are many variants of randomised benchmarking such as: direct RB [292], which scales better to larger devices than does traditional Clifford RB; and Simultaneous RB [293], which can be used to quantify crosstalk.



(a) **Average readout error.** The readout error is the probability the state of a given qubit is incorrectly labelled.



(b) **Average error per** U_2 **gate.** The error per gate is a measure of how accurately the U_2 gate is applied.

Figure B.4: **Error per single qubit operations on the devices used in this work.** Bars indicate the average error rates; error bars are one standard deviation. Data aggregated based on calibration data collected over the course of our experiments. Devices shown here are: ibmqx2 [🟩], ibmq_ourense [⬜], ibmq_singapore [🟧], ibmq_16_melbourne [🟦]. A logarithmic scale is used.

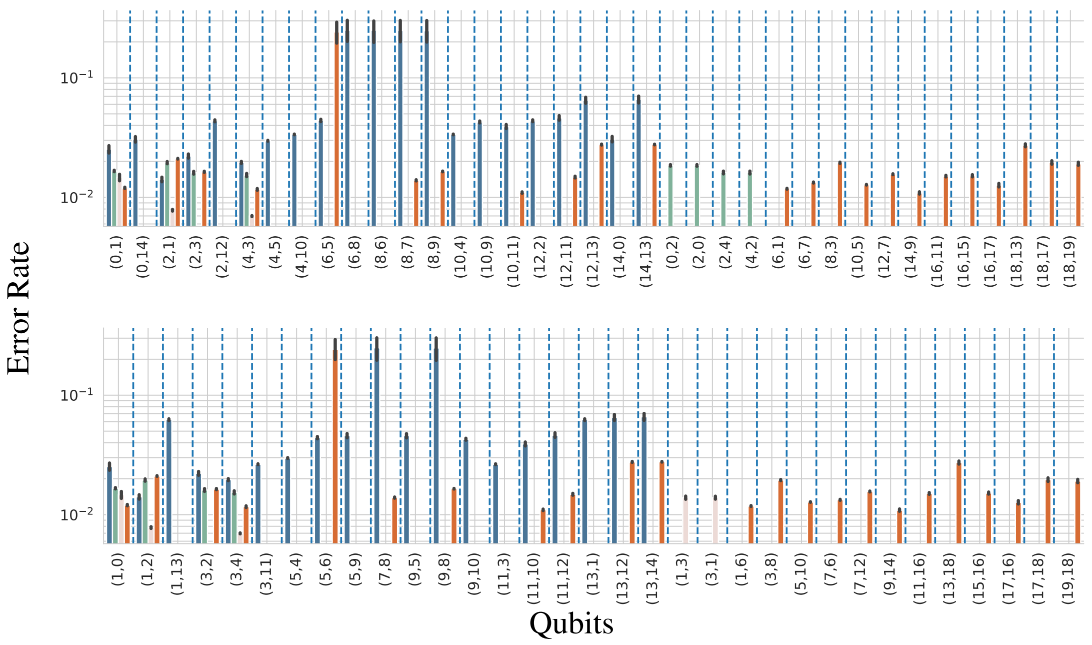


Figure B.5: **Average error per** CX **operation on the devices used in this work.** The error per CX gate is a measure of how accurately the CX gate is applied. Bars indicate the average error rates; error bars are one standard deviation. Data aggregated based on calibration data collected over the course of our experiments. Devices shown here are: ibmqx2 [🟩], ibmq_ourense [⬜], ibmq_singapore [🟧], ibmq_16_melbourne [🟦]. A logarithmic scale is used.

B.4 Empirical Relationship Between Heavy Output Probability and ℓ_1-norm Distance

As discussed in Section 1.6.4, the theoretical foundations for believing that implementing shallow circuits to within a fixed ℓ_1-norm distance constitutes a demonstration of quantum computational supremacy are stronger than for implementations with high heavy output probability. That being said, Figure 3.6 and Figure 3.14 contain similar features. For example, ibmq_16_melbourne consistently performs the worst, with ibmq_singapore and ibmq_ourense performing the best in both figures of merit. An interesting question, then, is how these two figures of merit generally relate to one another.

If the ℓ_1-norm distance was 0, the experimental outcome frequencies would equal the ideal outcome probabilities. Consequently, the heavy output probabilities would be the same between the device and an ideal quantum computer. Because the heavy output probability depends on the circuit in question, when examining the empirical relationship between ℓ_1-norm distance and heavy output probability, it is useful to normalise the latter by the heavy output probability of an ideally-implemented circuit. We define the normalised heavy output probability as the ratio of the heavy output probability of the device and the heavy output probability from an ideal quantum computer. Therefore if the ℓ_1-norm distance was 0, the normalised heavy output probability would be 1.

As the ℓ_1-norm distance increases, the experimental frequencies increasingly differ from the ideal outcome probabilities. Two things then may happen: heavy outputs are produced more regularly, in which case the normalised heavy output probability will grow above 1; or less regularly, in which case the normalised heavy output probability will fall below 1. In practice, we expect the distribution produced by the device to converge to the uniform one over all bit strings as the noise increases, so we expect the normalised heavy output probability to fall with increasing ℓ_1-norm distance.

The empirical relationship between the normalised heavy output probability and ℓ_1-norm distance is shown in Figure B.6. For each circuit, Figure B.6 plots the ℓ_1-norm distance of the distribution produced by a real device against the normalised heavy output probability. As expected, a negative correlation exists between these two figures of merit. For the deepest circuits, and in particular the widest circuits from the deep circuits class, the cluster of points can be seen to indicate that the normalised heavy output probability falls more slowly as the ℓ_1-norm distance becomes larger. This is because the minimum value of heavy output probability is being reached, which is to say that the output distribution from the real device has converged to the uniform one, while more detail can be extracted by considering the ℓ_1-norm distance.

This correlation is encouraging as, in the regime where it becomes impossible to calculate the ℓ_1-norm distance, we can be justified in believing that the correlation between the features present in the plot throughout this section persists. This line of reasoning is similar to that used when Cross-Entropy Benchmarking is used to predict demonstrations of quantum computational supremacy in the regime when it too becomes impossible to calculate [60].

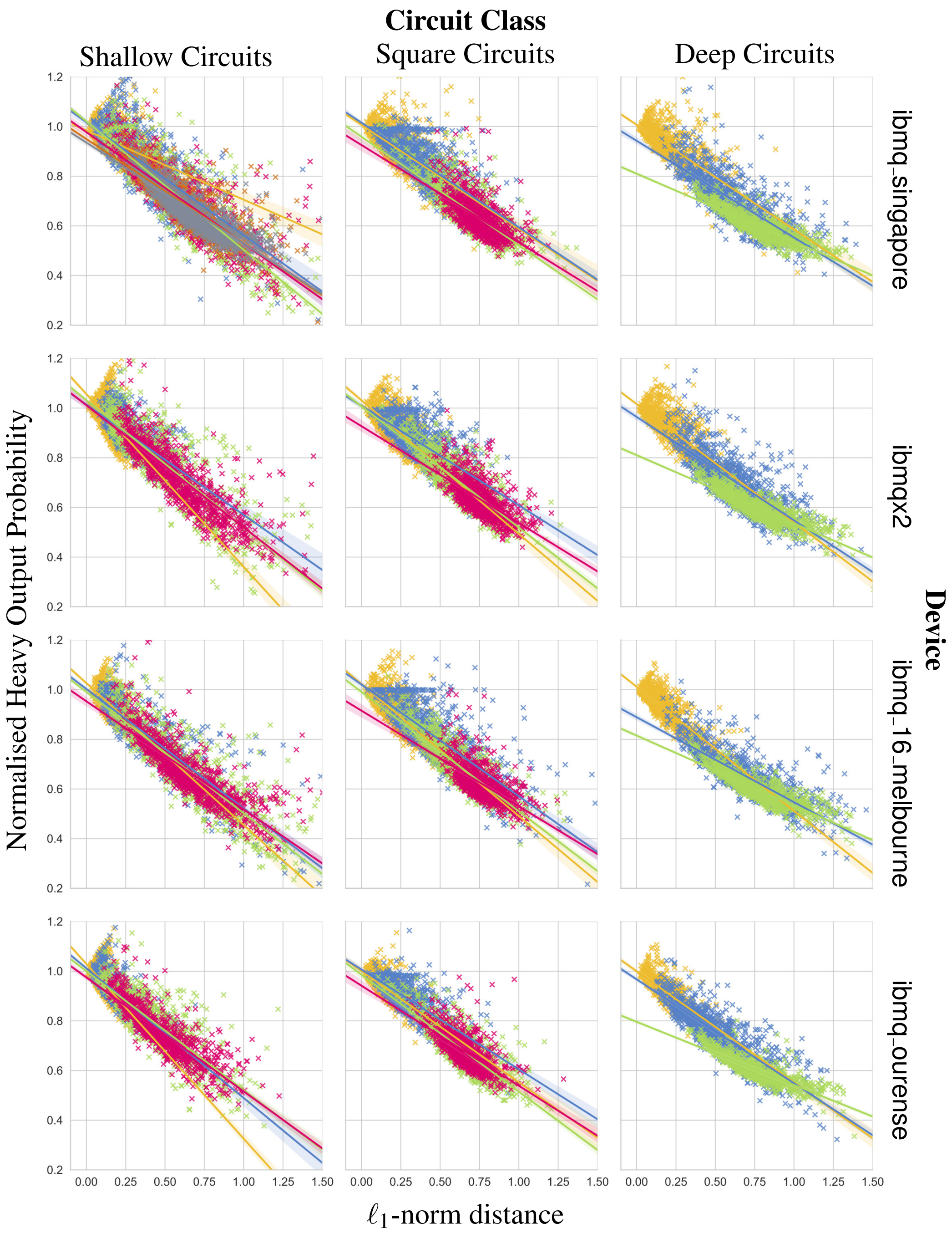


Figure B.6: **Scatter plot and linear regression line comparing the normalised heavy output probability and ℓ_1-norm distance**. Each point corresponds to one circuit of the class and width as labelled. Colours correspond to numbers of qubits in the following way: 2 [🟨], 3 [🟦], 4 [🟩], 5 [🟥], 6 [🟧], 7 [⬜].

Appendix C

Blind IQP Computation, and an IQP Hypothesis Test

C.1 Proof of Lemma 4.1.1

Lemma (Restated from Lemma 4.1.1)**.** *Consider a quantum state $E_{\boldsymbol{Q}}|\phi\rangle$, where we have used the graph state circuit notation of Definition 1.4.2, and where $|\phi\rangle$ is an arbitrary quantum state. If $\widetilde{\boldsymbol{Q}}$ is an extended* IQP *graph built from $\boldsymbol{Q}$ then there exists a state $E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle$, which can be transformed into the state $E_{\boldsymbol{Q}}|\phi\rangle$ through a sequence of Pauli-*Y *basis measurements on qubits and local rotations around the* Z *axis of the unmeasured qubits through angles $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$.*

This proof is similar to that of Lemma 1.4.1 in [73].

Proof. We will define a scheme for building the state $|\psi\rangle$ and a corresponding extended IQP graph $\widetilde{\boldsymbol{Q}}$ meeting the conditions of the lemma.

Consider the case where $\widetilde{\boldsymbol{Q}}$ was built from $\boldsymbol{Q}$ by replacing the entry (i,j) of $\boldsymbol{Q}$ with -1. $\boldsymbol{Q}$ can be built from $\widetilde{\boldsymbol{Q}}$ either by applying a break operation to the vertex $b_{1=g(i,j)}$, or by applying a bridge operation to this same vertex. We now move to consider these two separate cases.

Break: $\boldsymbol{Q}_{ij} = 0$. Define the state $E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle$ as

$$E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle = \mathsf{CZ}_{a_i,b_1}\mathsf{CZ}_{p_j,b_1}E_{\boldsymbol{Q}}|\phi\rangle|b_1\rangle,$$

where we set $|b_1\rangle = |r^b_1\rangle$ with $r^b_1 \in \{0,1\}$.

Notice then that $\mathsf{CZ}_{a_i,b_1}\mathsf{CZ}_{p_j,b_1}E_{\boldsymbol{Q}}$ indeed describes the same entanglement pattern as $E_{\widetilde{\boldsymbol{Q}}}$. Indeed, applying the CZ operations is equivalent to applying the operator $\mathsf{Z}^{r^b_1}$ to each of the qubits a_i and p_j. We can conclude:

$$E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle = \mathsf{Z}^{r^b_1}_{a_i}\mathsf{Z}^{r^b_1}_{p_j}E_{\boldsymbol{Q}}|\phi\rangle|b_1\rangle$$

Measuring the qubit b_1 in the Pauli-Y basis collapses it, with equal likelihood, to either of the Pauli-Y basis states. As the qubit b_1 is disentangled, this measurement has no other effect on the state. We are therefore left with the state $\mathsf{Z}_{a_i}^{r_1^b}\mathsf{Z}_{p_j}^{r_1^b}E_{\boldsymbol{Q}}|\phi\rangle$ which differs by only local rotations from $E_{\boldsymbol{Q}}|\phi\rangle$.

Bridge: $\boldsymbol{Q}_{ij} = 1$. Define the state $E_{\tilde{\boldsymbol{Q}}}|\psi\rangle$ as

$$E_{\tilde{\boldsymbol{Q}}}|\psi\rangle = \mathsf{CZ}_{a_i,b_1}\mathsf{CZ}_{p_j,b_1}\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}|\phi\rangle|b_1\rangle, \tag{C.1}$$

where $|b_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{r_1^b}|1\rangle\right)$ with $r_1^b \in \{0,1\}$ is a Hadamard basis state. Notice that $\mathsf{CZ}_{a_i,b_1}\mathsf{CZ}_{p_j,b_1}\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}$ describes the same operation as $E_{\tilde{\boldsymbol{Q}}}$, since CZ_{a_i,p_j} cancels the identical operation implicit in $E_{\boldsymbol{Q}}$.

Applying the operations, CZ_{a_i,b_1} and CZ_{p_j,b_1} to the state $\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}|\phi\rangle|b_1\rangle$ is equivalent to applying

$$\frac{1}{\sqrt{2}}|0\rangle \otimes \mathsf{I}_{a_i} \otimes \mathsf{I}_{p_j} + (-1)^{r_1^b}\frac{1}{\sqrt{2}}|1\rangle \otimes \mathsf{Z}_{a_i} \otimes \mathsf{Z}_{p_j}$$

to the state $\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}|\phi\rangle$. Following this by a measurement of the qubit b_1 in the Pauli-Y basis is equivalent to applying

$$\frac{1}{\sqrt{2}}\mathsf{I}_{a_i} \otimes \mathsf{I}_{p_j} + (-1)^{1-s_1^b}(-1)^{r_1^b}\, i\frac{1}{\sqrt{2}}\mathsf{Z}_{a_i} \otimes \mathsf{Z}_{p_j}$$

to $\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}|\phi\rangle$. Here we have used the notation that $s_1^b = 0$ when $|+^{\mathsf{Y}}\rangle = |0\rangle + i|1\rangle$ is measured and $s_1^b = 1$ when $|-^{\mathsf{Y}}\rangle = |0\rangle - i|1\rangle$ is measured.

The original state of equation (C.1) is transformed, by this measurement, to the state

$$\left(\frac{1}{\sqrt{2}}\mathsf{I}_{a_i} \otimes \mathsf{I}_{p_j} + (-1)^{1-s_1^b}(-1)^{r_1^b}\, i\frac{1}{\sqrt{2}}\mathsf{Z}_{a_i} \otimes \mathsf{Z}_{p_j}\right)\mathsf{CZ}_{a_i,p_j}E_{\boldsymbol{Q}}|\phi\rangle. \tag{C.2}$$

Notice that the CZ operator can be written as

$$\mathsf{CZ}_{1,2} = \frac{1}{2}\left(\mathsf{I}_1 \otimes \mathsf{I}_2 + Z_1 \otimes \mathsf{I}_2 + \mathsf{I}_1 \otimes Z_2 - Z_1 \otimes Z_2\right).$$

Using this fact, along with the knowledge that $\mathsf{S} = \frac{1}{\sqrt{2}}(\mathsf{I} + iZ)$, allows us to see

$$\mathsf{CZ}_{1,2} = (\mathsf{S}_1 \otimes \mathsf{S}_2)\left(\frac{1}{\sqrt{2}}\mathsf{I}_1 \otimes \mathsf{I}_2 - i\frac{1}{\sqrt{2}}\mathsf{Z}_1 \otimes \mathsf{Z}_2\right)$$

$$\mathsf{CZ}_{1,2} = \left(\mathsf{S}_1^{-1} \otimes \mathsf{S}_2^{-1}\right)\left(\frac{1}{\sqrt{2}}\mathsf{I}_1 \otimes \mathsf{I}_2 + i\frac{1}{\sqrt{2}}\mathsf{Z}_1 \otimes \mathsf{Z}_2\right)$$

In particular:

$$\mathsf{CZ}_{a_i,p_j} = \left(S_{a_i}^{(-1)^{1-s_1^b}(-1)^{r_1^b}} \otimes S_{p_j}^{(-1)^{1-s_1^b}(-1)^{r_1^b}}\right)$$
$$\left(\frac{1}{\sqrt{2}}\mathsf{I}_{a_i} \otimes \mathsf{I}_{p_j} - (-1)^{1-s_1^b}(-1)^{r_1^b}\, i\frac{1}{\sqrt{2}}\mathsf{Z}_{a_i} \otimes \mathsf{Z}_{p_j}\right)$$

Substituting this into (C.2), and with some rearranging, we realise the resulting state is

$$\left(\mathsf{S}_{a_1}^{(-1)^{1-s_1^b+r_1^b}} \otimes \mathsf{S}_{p_j}^{(-1)^{1-s_1^b+r_1^b}}\right) E_{\boldsymbol{Q}}|\phi\rangle.$$

Again, this differs from the state $E_{\boldsymbol{Q}}|\phi\rangle$ only by local rotations around the Z axis.

We now turn to the case where the number of break and bridge operations needed to move from $\widetilde{\boldsymbol{Q}}$ to $\boldsymbol{Q}$ is more than one. The state $E_{\widetilde{\boldsymbol{Q}}}|\psi\rangle$ can be built one step at a time by repeating the steps above (i.e. entangling and measuring the appropriate bridge and break qubits one at a time). We wish to show that all the necessary entanglement operations, all the measurement operations and all the necessary corrections can be done together, and in this order. This is to say that all entanglement is done together, all measurements are done together, and all corrections are done together.

This can be done by demonstrating that these operations appropriately commute. Since the qubits that might require corrections are never measured, and because the corrections commute with the entanglement operators, the corrections may be moved to the end of the computation. Those qubits that will be measured are not entangled after they are measured. As such all entanglement may be commuted before all the measurements. This established the appropriate order of operations. □

C.2 Proof of Theorem 4.1.1

Theorem (Restated from Theorem 4.1.1)**.** *The protocol described by Protocol 4.1.2 is information-theoretically secure against a dishonest server.*

Proof. The proof consists of a pattern of transformations of the real protocol of Protocol 4.1.2, into the ideal resource of Protocol 4.1.3, which leaves the computation unchanged, therefore ensuring the indistinguishability of the two settings.

The first transformation we perform is of the state generation phase of the Protocol 4.1.2. The new method we use for this phase is described in Protocol C.2.1 and relies on the measurement of EPR pairs to produce qubits in the correct basis, with some randomness resulting from the measurement. This may be visualised by the expansion of π_A^1 seen in Figure 4.5. While lines 1, 2 and 4 of Protocol 4.1.2 and the lines 1, 2, 3 and 5 of Protocol C.2.1 differ, the remainder of both protocols is identical. We show now that the protocols are indistinguishable.

- Firstly consider the generation of r^p and r^a. In Protocol 4.1.2 these terms are picked uniformly at random from the set of all binary stings of the appropriate length. In the case of Protocol C.2.1 they are generated by measurements on EPR pairs, the result of which is entirely random. Similarly, in both cases, r^b is picked uniformly at random from the set of all binary strings of the appropriate length.
- Line 2 of Algorithm 4.1.2 generates at random one of the four states $|+\rangle$, $|+^Y\rangle$, $|-\rangle$ and $|-^Y\rangle$. Line 3 of Algorithm C.2.1 achieves the same effect by mea-

suring an EPR pair with equal probability in one of the basis $\{|+\rangle, |-\rangle\}$ and $\{|+^Y\rangle, |-^Y\rangle\}$.

- Finally, the application of the $\left(\sqrt{\mathsf{Y}}\right)^{d_j^b}$ operation in line 4 of Algorithm 4.1.2 decides, according the graph to be created, if the bridge and break qubit will be drawn from the set $\{|+\rangle, |-\rangle\}$ or $\{|0\rangle, |1\rangle\}$. Choosing between using the measurement basis $\{|+\rangle, |-\rangle\}$ or $\{|0\rangle, |1\rangle\}$ on one half of an EPR pair has the same effect. The random rotation $\mathsf{Y}_j^{r^b}$ then has the same effect of the randomness that is intrinsic to the measurement performed in Protocol C.2.1.

Consider now the transformation from Protocol C.2.1 to Protocol C.2.2. The reader may wish to refer to Figure 4.5 for a visualisation of this new resource.

- Notice that line 3 of Protocol C.2.1 is identical to that of line 9 of Protocol C.2.2. This operation can be delayed without affecting the computation as the qubit being measured is not acted upon in any other way during the protocol.

- Consider Π and A. In Protocol C.2.2 they are generated at random from the set of all $\Pi \in [0,1,2,3]^{n_p}$ and $A \in [0,1,2,3]^{n_a}$ as stated in line 7. This is the case too for Algorithm C.2.1 because Π_i^z, Π_i^s, A_k^z and A_k^s are one time padded by r_i^p, d_i^p, r_k^a and d_k^a respectively as seen in equations (4.2), (4.3), (4.4) and (4.5).

- It remains to show that Protocol C.2.2 results in the same computation as Protocol C.2.1. This can be achieved by noting a simple rearrangement of equations (4.2), (4.3), (4.4) and (4.5) to make d_i^p and d_k^a the subject. In doing so we assume the $r_p^i, r_a^k = 0$ which is corrected for, if this is not the case, in equation (4.8).

Finally, Protocol 4.1.3 simply involves a relabeling of the players in Protocol C.2.2 to match those in the simulator distinguisher setting. This amounts to the transformation from Figure 4.5 to Figure 4.6.

This series of transformations convinces us that the following relationship is true and that the resource of Protocol 4.1.2 is composably secure against a dishonest server.

$$\pi_A \mathcal{R} \equiv \mathcal{S} \sigma$$

□

Protocol C.2.1 Blind delegated IQP computation, adapting Protocol 4.1.2 by the addition of a teleportation technique for state preparation.

Public: $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$, $\theta \in [0,2\pi]$, $\mathcal{Q}$ (the distribution from which $\boldsymbol{Q}$ is picked)
Client input: $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$
Client output: $\widetilde{x} \in \{0,1\}^{n_p}$

```
Client:
 1: Randomly generate $d^p \in \{0,1\}^{n_p}$ and $d^a \in [0,1]^{n_a}$ where $n_p$ and $n_a$ are the numbers
    of primary and ancillary qubits respectively.
 2: Generate $n_p$ EPR pairs $|\Phi_+\rangle_j^p$, $n_a$ EPR pairs $|\Phi_+\rangle_i^a$ and a further $n_b$ EPR pairs
    $|\Phi_+\rangle_k^b$.
 3: Measure one half of each of $|\Phi_+\rangle_j^p$ in the basis $\mathsf{S}^{d_j^p}\{|+\rangle,|-\rangle\}$ to achieve outcome
    $r_j^p$ and one half of each of $|\Phi_+\rangle_i^a$ in the basis $\mathsf{S}^{d_i^a}\{|+\rangle,|-\rangle\}$ to achieve outcome $r_i^a$.
 4: Create $d^b \in \{0,1\}^{n_b}$ in the following way: For $i = 1,\dots,n_a$ and $j = 1,\dots,n_p$, if
    $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 0$, then $d_k^b = 0$ else if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 1$ then $d_k^b = 1$. Keep
    track of the relation between $k$ and $(i,j)$ via the surjective function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to$
    $\mathbb{Z}_{n_b}$.
 5: Measure one half of each of $|\Phi_+\rangle_k^b$ in the basis $\sqrt{\mathsf{Y}}^{d_k^b}\{|0\rangle,|1\rangle\}$ to achieve outcome
    $r_k^b$, for $k = 1,\dots,n_b$.
 6: State $\rho$ comprising of all unmeasured states in the Client's position is sent to the
    Server.
Server:
 7: Implement $E_{\widetilde{\boldsymbol{Q}}}$.
 8: Measure qubits $b_1,\dots,b_{n_b}$ in the basis $\{|+^{\mathsf{Y}}\rangle,|-^{\mathsf{Y}}\rangle\} := \{|0\rangle + i|1\rangle, |0\rangle - i|1\rangle\}$.
    Take the outcome of measuring qubit $b_k$ to be $s_k^b \in \{0,1\}$ if the measurment
    projects the output to the state $|0\rangle + i(-1)^{s_k^b}|1\rangle$.
 9: Send the outcome $s^b \in \{0,1\}^{n_b}$ to the Client.
Client:
10: Calculate $\boldsymbol{\Pi}^z, \boldsymbol{\Pi}^s \in \{0,1\}^{n_p}$ and $\boldsymbol{A}^z, \boldsymbol{A}^s \in \{0,1\}^{n_a}$ using: (4.2), (4.3), (4.4) and
    (4.5).
11: Calculate $\boldsymbol{A} \in \{0,1,2,3\}^{n_a}$ and $\boldsymbol{\Pi} \in \{0,1,2,3\}^{n_p}$ for the ancillary and primary
    qubits respectively, where $A_i = A_i^s + 2A_i^z \pmod 4$ and $\Pi_j = \Pi_j^s + 2\Pi_j^z \pmod 4$.
12: Send $\boldsymbol{A}$ and $\boldsymbol{\Pi}$ for the ancillary and primary qubits respectively, to the Server.
Server:
13: Measure qubits in the basis of equation (4.1), for the ancillary and primary qubits
    respectively, producing measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$.
14: Send measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$ to the Client.
Client:
15: Generate and output $\widetilde{x} \in \{0,1\}^{n_p}$ using equation (1.25).
```

Protocol C.2.2 Blind delegated IQP computation, adapting Protocol C.2.1 with the transition to pre-made randomness.

Public: $\widetilde{\boldsymbol{Q}} \in \{-1,0,1\}^{n_a \times n_p}$, $\theta \in [0,2\pi]$, $\mathcal{Q}$ (the distribution from which $\boldsymbol{Q}$ is picked)
Client input: $\boldsymbol{Q} \in \{0,1\}^{n_a \times n_p}$
Client output: $\widetilde{x} \in \{0,1\}^{n_p}$

```
Client:
 1: Generate $n_p$ EPR pairs $|\Phi_+\rangle_j^p$, $n_a$ EPR pairs $|\Phi_+\rangle_i^a$ and a further $n_b$ EPR pairs $|\Phi_+\rangle_k^b$.
 2: Create $d^b \in \{0,1\}^{n_b}$ in the following way: For $i = 1,\dots,n_a$ and $j = 1,\dots,n_p$, if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 0$, then $d_k^b = 0$ else if $\widetilde{\boldsymbol{Q}}_{ij} = -1$ and $\boldsymbol{Q}_{ij} = 1$ then $d_k^b = 1$. Keep track of the relation between $k$ and $(i,j)$ via the surjective function $g_{\widetilde{\boldsymbol{Q}}} : \mathbb{Z}_{n_a \times n_p} \to \mathbb{Z}_{n_b}$.
 3: Measure one half of each of $|\Phi_+\rangle_k^b$ in the basis $\sqrt{\mathsf{Y}}^{d_k^b}\{|0\rangle,|1\rangle\}$ to achieve outcome $r_k^b$, for $k = 1,\dots,n_b$.
Server:
 4: Implement $E_{\widetilde{\boldsymbol{Q}}}$.
 5: Measure qubits $b_1,\dots,b_{n_b}$ in the basis $\{|+^{\mathsf{Y}}\rangle,|-^{\mathsf{Y}}\rangle\} := \{|0\rangle + i|1\rangle, |0\rangle - i|1\rangle\}$. Take the outcome of measuring qubit $b_k$ to be $s_k^b \in \{0,1\}$ if the measurment projects the output to the state $|0\rangle + i(-1)^{s_k^b}|1\rangle$.
 6: Send the outcome $s^b \in \{0,1\}^{n_b}$ to the Client.
Client:
 7: Randomly generate $\boldsymbol{\Pi} \in \{0,1,2,3\}^{n_p}$ and $\boldsymbol{A} \in \{0,1,2,3\}^{n_a}$ where $n_p$ and $n_a$ are the numbers of primary and ancillary qubits respectively.
 8: Calculate $d_j^p \in \{0,1,2,3\}^{n_p}$ and $d_i^a \in \{0,1,2,3\}^{n_a}$ using (4.6) and (4.7) respectively.
 9: Measure one half of each of $|\Phi_+\rangle_j^p$ in the basis $\mathsf{S}^{d_j^p}\{|+\rangle,|-\rangle\}$ to achieve outcome $r_j^p$ and one half of each of $|\Phi_+\rangle_i^a$ in the basis $\mathsf{S}^{d_i^a}\{|+\rangle,|-\rangle\}$ to achieve outcome $r_i^a$.
10: Send $\boldsymbol{A}$ and $\boldsymbol{\Pi}$ for the ancillary and primary qubits respectively, to the Server.
Server:
11: Measure qubits in the basis of equation (4.1), for the ancillary and primary qubits respectively, producing measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$.
12: Send measurement outcomes $s^p \in \{0,1\}^{n_p}$ and $s^a \in \{0,1\}^{n_a}$ to the Client.
Client:
13: Generate and outputs $\widetilde{x} \in \{0,1\}^{n_p}$ using equation (4.8).
```