Classical Verification and Enhancement of Near-Term Quantum Devices

by

Bin Cheng

A thesis submitted in satisfaction of the

requirements for the degree of

Doctor of Philosophy

under the supervision of

Prof. Zhengfeng Ji

Prof. Michael J. Bremner

A/Prof. Man-Hong Yung (external)

Faculty of Engineering and Information Technology

University of Technology Sydney

Aug 2023

Certificate of Original Authorship

I, Bin Cheng, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

Signature:

Date: January 5, 2024

To my family.

Acknowledgements

This thesis would not be possible without a number of people supporting me throughout my PhD studies. I would like to start by thanking my supervisors, Zhengfeng Ji, Michael Bremner and Man-Hong Yung. Their consistent guidance and invaluable assistance have been pivotal to my progress, and I am profoundly grateful for their mentorship.

A special acknowledgment goes to Man-Hong Yung, who not only introduced me to the exciting field of quantum information science but also served as a mentor in both my research and personal life. I would also like to express my sincere gratitude to Zhengfeng Ji for his patience and encouragement, along with his role in shaping my research interests and taste. Under his guidance, I gradually grow into a researcher. Furthermore, I am thankful to Michael Bremner for his encouragement, support and insightful discussions that enriched my academic journey.

During my PhD journey, my friends, colleagues and collaborators have been constant companions and engaging in thought-provoking discussions. I extend my gratitude to Fei Li, Xiaowei Li, Yupan Liu, Ryan Mann, Fei Meng, Mauro Morales, Gang Tang, Ximing Wang, Peng Yan, Yu-Ning Zhang, and many others. Moreover, I acknowledge the efforts from Zhengfeng Ji, Tongyin Lin, Yupan Liu, Ryan Mann, Fei Meng, and Ximing Wang, for helping me proofread my thesis.

Finally, I would like to express my appreciation to my parents for their unconditional love and support, as well as unconsciously being picky when I share good news with them. I would like to thank Tongyin Lin for her love and companionship. The happy two months we spent together in Sydney made my writing grind much more enjoyable.

This is a CONVENTIONAL THESIS. Parts of this thesis have been already published. The content have been edited to suit the formatting of the thesis and to maintain its coherence.

List of Research Outputs

Related to the Thesis:

(1) Man-Hong Yung and Bin Cheng. Anti-Forging Quantum Data: Cryptographic Verification of Quantum Cloud Computing, 2020 [YC20]

(2) Chong Ying, Bin Cheng, Youwei Zhao, He-Liang Huang, Yu-Ning Zhang, Ming Gong, Yulin Wu, Shiyu Wang, Futian Liang, Jin Lin, Yu Xu, Hui Deng, Hao Rong, Cheng-Zhi Peng, Man-Hong Yung, Xiaobo Zhu, and Jian-Wei Pan. Experimental Simulation of Larger Quantum Circuits with Fewer Superconducting Qubits. Phys. Rev. Lett., 130(11):110601, 2023 [YCZ+23]

(3) Michael J. Bremner, Bin Cheng, and Zhengfeng Ji. IQP Sampling and Verifiable Quantum Advantage: Stabilizer Scheme and Classical Security, 2023 [BCJ23]

(4) Lei Xie, Bin Cheng, Xiaodie Lin, Zhenyu Chen, Zhaohui Wei, and Zhengfeng Ji. A Machine Learning Approach to Quantum Error Mitigation, 2023 [XCL+23]

Others:

(1) Bujiao Wu, Bin Cheng, Jialin Zhang, Man-Hong Yung, and Xiaoming Sun. Speedup in Classical Simulation of Gaussian Boson Sampling. Sci. Bull., 65(10):832-841, 2020 [WCZ+20]

(2) Xi Chen, Bin Cheng, Zhaokai Li, Xinfang Nie, Nengkun Yu, Man-Hong Yung, and Xinhua Peng. Experimental Cryptographic Verification for Near-Term Quantum Cloud Computing. Sci. Bull., 66(1):23-28, 2021 [CCL+21]

(3) Kai Sun, Zi-Jian Zhang, Fei Meng, Bin Cheng, Zhu Cao, Jin-Shi Xu, Man-Hong Yung, Chuan-Feng Li, and Guang-Can Guo. Experimental Verification of Group Non-Membership in Optical Circuits. Photon. Res., 9(9):1745, 2021 [SZM+21]

(4) Fan Wang, Bin Cheng, Zi-Wei Cui, and Man-Hong Yung. Quantum Computing by Quantum Walk on Quantum Slide, 2022 [WCCY22]

(5) Bin Cheng, Xiu-Hao Deng, Xiu Gu, Yu He, Guangchong Hu, Peihao Huang, Jun Li, Ben-Chuan Lin, Dawei Lu, Yao Lu, Chudan Qiu, Hui Wang, Tao Xin, Shi Yu, Man-Hong Yung, Junkai Zeng, Song Zhang, Youpeng Zhong, Xinhua Peng, Franco Nori, and Dapeng Yu. Noisy Intermediate-Scale Quantum Computers. Front. Phys., 18(2):21308, 2023 [CDG+23]

Contents

List of Figures

List of Tables

Abstract

Near-term quantum devices are arguably able to perform computational tasks beyond classical capabilities. But a definitive claim of such a quantum computational advantage relies on classical verification. On the other hand, on the road of pursuing quantum advantage on practical tasks, classical techniques to boost the performance of near-term quantum devices are also required. In this thesis, we explore the classical verification and classical enhancement of near-term quantum devices.

In the first part of the thesis, we present results on the verification protocols based on the instantaneous quantum polynomial-time (IQP) model, which is a promising model for achieving verifiable quantum advantage on near-term quantum devices. We first study the interplay between IQP circuits, stabilizer formalism and coding theory, and give a characterization of the correlation functions from IQP circuits. Based on this, we give a new IQP-based construction, called the stabilizer scheme, which enriches the scope of IQP-based schemes while maintaining their simplicity and verifiability. To analyze the classical security, we introduce the *Hidden Structured Code* (HSC) problem as a well-defined mathematical challenge that underlies the stabilizer scheme. We explore a class of attack algorithms based on secret extraction and give evidence of the security of the stabilizer scheme, assuming the hardness of the HSC problem. Moreover, we show that the vulnerability observed in the original IQP verification protocol is primarily attributed to inappropriate parameter choices, which can be naturally rectified with proper parameter settings.

In the second part of the thesis, we first present a machine learning approach to quantum error mitigation. We propose the concept of neighborhood learning, and explore the choice of the neighbor circuits and the learning models. Based on our observations, we give an adaptive learning strategy to dynamically construct the neighbor circuits, that achieves a better tradeoff between performance and required resources compared to various quantum error mitigation techniques. Finally, we present the experimental results on simulating large linear cluster states (up to 33 qubits) with only

4 superconducting qubits. Our experiment is based on the circuit-cutting technique, and achieves a better fidelity on 12-qubit linear cluster state than simulating the state directly on a 12-qubit quantum computer.

Chapter 1

Introduction

Quantum computing represents a fundamental paradigm shift in computation, with potential speedups on problems such as integer factorization [Sho94], database search [Gro96] and quantum simulation [Fey82, Llo96]. However, these applications typically require fault-tolerant quantum computers, which are beyond the reach of our current noisy intermediate-scale quantum (NISQ) era [Pre18]. Nevertheless, experimental demonstrations have shown that we can perform random-circuit sampling [BIS+18, AAB+19, ZCC+21, WBC+21] and boson sampling [AA11, ZWD+20] at scales arguably beyond classical simulation. These efforts are known as the pursuit of quantum computational advantage (supremacy) in the literature, which are usually based on the quantum random sampling problems [HE23].

However, a definitive demonstration of quantum computational advantage relies on classical verification. Although the quantum supremacy experiments can use some benchmarking techniques such as cross-entropy benchmarking (XEB) [AAB+19] to certify the quantum devices, they cannot be efficiently verified in an adversarial setting without modification of the underlying computational task. In this respect, IQP (Instantaneous Quantum Polynomial-time) sampling could be a promising candidate, since it has been proposed to have beyond classical capabilities in some settings [BJS11, BMS16, BMS17] and a verifiable scheme based on quadratic-residue codes [SB09]. However, this verifiable scheme relies on computational assumptions that are not standard and have not been studied in depth. In fact, it was recently broken by an attack proposed by Kahanamoku-Meyer [KM19].

On the other hand, it is not anticipated that quantum computers will fully replace classical computers, and a reasonable computational model should be hybrid quantum-classical. There have already been quantum algorithms proposed that fit this framework, like the variational quantum eigensolvers for quantum chemistry problems [PMS$^+$14, YCM$^+$14], which are designed to be suitable for near-term quantum devices. It is important to explore classical techniques that can be used to enhance the quantum capability, like quantum error mitigation [TBG17, LB17] and circuit-cutting techniques [PHOW20].

In this context, this thesis will focus on these two important aspects of near-term quantum devices, classical verification and classical enhancement. We first give the necessary background for this thesis in Chapter 2. In Section 2.1, after introducing the basic components of quantum computation in Section 2.1.1, we introduce some technical ingredients, including the stabilizer formalism in Section 2.1.2, coding theory in Section 2.1.3 and matrix factorization over $\mathbb{F}_2$ in Section 2.1.4. Then, we give a review of quantum computational advantage in Section 2.2 and verifiable quantum advantage in Section 2.3.

In the first part of the thesis, we revisit and revive the IQP-based verifiable quantum advantage. Although the original scheme by Shepherd and Bremner has been broken by [KM19], the IQP-based protocols still offer a promising avenue for achieving verifiability beyond classical computing with fewer resources than Shor's algorithm or the verification protocols based on trapdoor claw-free functions [BCM$^+$18, BKVV20, KMCVY21].

Specifically, in Chapter 3, we give an overview of the IQP-based verification protocols with the language set in our previous work [YC20]. We discuss the general framework, the correlation functions from the IQP circuits, the Shepherd-Bremner construction and its loophole. In addition, we also present the heuristic construction in [YC20] and a new redundancy technique called column redundancy, which turns out to fix the recent loophole of the Shepherd-Bremner scheme.

In Chapter 4, we first study the interplay between IQP circuits, stabilizer formalism and coding theory, and give a characterization of the IQP circuit correlation functions in Section 4.1. Based on this, in Section 4.2, we give a new IQP-based construction, called the stabilizer scheme, which enriches the scope of IQP-based schemes while maintaining their simplicity and verifiability. In Section 4.3, we give another construction algorithm which arised in the early exploration of [BCJ23]. This construction algorithm is based on the matrix factorization over $\mathbb{F}_2$ and although it has been superseded by the stabilizer scheme, it provides a different perspective on the IQP-based verification protocols.

In Chapter 5, we analyze the classical security of the stabilizer scheme and discuss a class of classical attacks based on extracting secrets. In Section 5.1, we give a simplified and generalized version of the attack algorithm in [KM19], named the Linearity Attack. Then, we present analysis in Section 5.2, showing that the stabilizer scheme is secure against the Linearity Attack. In Section 5.3, we show how to fix the loophole of the Shepherd-Bremner construction by using a different set of parameters, which is achieved by the column redundancy technique.

In the second part of the thesis, we discuss the classical enhancement of near-term quantum devices. In Chapter 6, we propose a machine learning approach, called the neighborhood learning, for quantum error mitigation. We extensively explore various technical aspects of the neighborhood learning and devise an adaptive learning strategy that effectively balances accuracy and computational cost.. In Chapter 7, we report an experimental implementation of a tomography-like circuit-cutting scheme to simulate large linear-cluster states [YCZ+23]. Finally, we conclude in Chapter 8.

1.1 Contribution

All materials in this thesis are written by myself, with revisions made by my collaborators. The following is a list of my contributions in each chapter.

- Chapter 2 is written from fresh, except for Section 2.3, which is adapted from the background material in [BCJ23]. This is work done in collaboration with Michael Bremner and Zhengfeng Ji. My contribution in [BCJ23] includes conceiving the initial idea, contributing to all technical results, performing the numerical simulations and writing the majority of the paper.
- Chapter 3 is written based on [YC20] and parts of [BCJ23]. [YC20] is work done in collaboration with Man-Hong Yung, who proposed this project. My contribution consists of the technical results, the numerical simulations and the majority of writing.
- Chapter 4 and Chapter 5 are based on [BCJ23]. Specifically, Section 4.3 contains an unpublished construction algorithm for IQP-based verification protocol, which was proposed and analyzed by me, with feedback from Michael Bremner and Zhengfeng Ji.
- Chapter 6 is based on [XCL+23], which is done in collaboration with Lei Xie, Xiaodie Lin, Zhenyu Chen, Zhaohui Wei and Zhengfeng Ji. My contribution in this work involves contributing ideas, performing numerical simulations and most of the writing.
- Chapter 7 is based on joint work with Chong Ying, Youwei Zhao, He-Liang Huang, Yu-Ning Zhang, Ming Gong, Yulin Wu, Shiyu Wang, Futian Liang, Jin Lin, Yu Xu, Hui Deng, Hao Rong, Cheng-Zhi Peng, Man-Hong Yung, Xiaobo Zhu, and Jian-Wei Pan [YCZ+23]. My contribution in this work includes (a) designing the experimental protocol with feedback from collaborators, (b) analyzing the experimental data with Yu-Ning Zhang and (c) most of the writing.

Chapter 2

Background

2.1 Preliminaries

2.1.1 Quantum Computation

In classical computation, information is represented by bits that can have a value of either 0 or 1. However, a classical bit can also be probabilistic, with a value of 0 occurring with probability p and a value of 1 occurring with probability $1-p$. Quantum computation, on the other hand, uses quantum bits, or qubits, to represent information. Unlike classical bits, qubits can be in a superposition of 0 and 1, and their state is represented by a vector in a two-dimensional complex Hilbert space, denoted by $\mathbb{C}^2$. The basis of $\mathbb{C}^2$ can be chosen to be the *computational basis states*, which is given by,

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

The state of a qubit can then be represented as a linear combination of the basis states: $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, where α and β are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. The probability of measuring a qubit to be in $x \in \{0, 1\}$ is given by $| \langle x|\psi\rangle |^2$.

One can use tensor product to compose quantum states. The tensor product of $|\psi\rangle$ and $|\phi\rangle$ is denoted as $|\psi\rangle \otimes |\phi\rangle$ or just $|\psi\rangle |\phi\rangle$, when the context is clear. Then, we can define the computational basis states for n qubits using tensor product, which is denoted by $|\mathbf{x}\rangle := |x_1\rangle \cdots |x_n\rangle$ with $x_j \in \{0,1\}$. We might commonly just write $|\mathbf{x}\rangle = |x_1 \cdots x_n\rangle$. The state of n qubits can be represented as $|\psi\rangle = \sum_{\mathbf{x}\in\{0,1\}^n} c_{\mathbf{x}} |\mathbf{x}\rangle$, where $c_{\mathbf{x}}$'s are complex numbers satisfying $\sum_{\mathbf{x}} |c_{\mathbf{x}}|^2 = 1$. This representation is valid for pure states; in general, a quantum state can be in a mixed state, which should be represented by a density matrix ρ. A density matrix is a positive semidefinite matrix with trace 1, and it can be interpreted as a probabilistic mixture of pure states, i.e., $\rho = \sum_k p_k |\psi_k\rangle\langle\psi_k|$, where $\langle\psi_k|$ is the conjugate transpose of $|\psi_k\rangle$.

It is anticipated that quantum computers utilizing the exotic quantum features such as quantum superposition and quantum entanglement can solve computational problems more efficiently than their classical counterparts. Often, quantum computation is described in the quantum-circuit model, where a quantum circuit is the product of a sequence of elementary quantum gates that are unitaries acting on one or two qubits. Common gates include the Hadamard gate, the T gate and the controlled-NOT (CNOT) gate, whose unitary matrices are given by,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

These gates are known to form a universal gate set, meaning that any n-qubit unitary can be approximated using these gates [NC11]. Pauli gates are also commonly used, whose matrix representations are given by,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.3}$$

Note that in the gate set $\{H, T, \text{CNOT}\}$, if one replaces the T gate with the S gate, which is given by $\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$, then one can only form the Clifford circuits. It is well-known that Clifford circuits can be classically efficiently simulated using the Gottesman-Knill algorithm [Got99]. The key insight behind the Gottesman-Knill algorithm is that the

Clifford circuit maps Pauli operators to Pauli operators under conjugation, which enables a more compact representation. This is part of the stabilizer formalism, which is a powerful tool for quantum error correction and classical simulation of quantum circuits.

2.1.2 Stabilizer Formalism

Here, we review the ingredients of stabilizer formalism that are necessary for this thesis. For a detailed treatment, we refer to [NC11, Chapter 10] or [DDM03, AG04]. We first denote the four Pauli matrices as follows,

$$\sigma_{00} = I \qquad \sigma_{10} = X \qquad \sigma_{11} = Y \qquad \sigma_{01} = Z \,. \tag{2.4}$$

In a compact form, $\sigma_{vw} = i^{vw} X^v Z^w$. For $\mathbf{v}, \mathbf{w} \in \mathbb{F}_2^n$ and $\mathbf{a} = \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} \in \mathbb{F}_2^{2n}$, we denote

$$\sigma_{\mathbf{a}} := \sigma_{v_1 w_1} \otimes \cdots \otimes \sigma_{v_n w_n} \,. \tag{2.5}$$

The n-qubit Pauli group, denoted as $\mathcal{P}_n$, consists of operators of the form $\sigma_{\mathbf{a},r,h} := i^h(-1)^r \sigma_{\mathbf{a}}$ with $h, r \in \mathbb{F}_2$; we use Pauli operators to refer to the elements of the Pauli group. The Hermitian Pauli operators are those with $h = 0$. The multiplication of two Pauli operators is given by

$$\begin{aligned} \sigma_{\mathbf{a}_1,r_1,h_1}\sigma_{\mathbf{a}_2,r_2,h_2} &= i^{h_1+h_2+\mathbf{a}_1^T\mathbf{J}\mathbf{a}_2}(-1)^{r_1+r_2+\mathbf{a}_1^T\mathbf{J}\mathbf{a}_2+\mathbf{v}_1\cdot\mathbf{w}_2}\sigma_{\mathbf{a}_1+\mathbf{a}_2} && (2.6)\\ &= i^{h_1+h_2+\mathbf{a}_1^T\mathbf{J}\mathbf{a}_2}(-1)^{r_1+r_2+\mathbf{a}_1^T\mathbf{J}\mathbf{a}_2+\mathbf{a}_1^T\mathbf{K}\mathbf{a}_2}\sigma_{\mathbf{a}_1+\mathbf{a}_2} \,, && (2.7)\end{aligned}$$

where $\mathbf{J}$ is the symplectic form,

$$\mathbf{J} = \begin{pmatrix} \mathbf{0}_n & \mathbf{I}_n \\ \mathbf{I}_n & \mathbf{0}_n \end{pmatrix} , \tag{2.8}$$

and

$$\mathbf{K} = \begin{pmatrix} \mathbf{0}_n & \mathbf{I}_n \\ \mathbf{0}_n & \mathbf{0}_n \end{pmatrix} . \tag{2.9}$$

Two Pauli operators $\sigma_{\mathbf{a}_1,r_1,h_1}$ and $\sigma_{\mathbf{a}_2,r_2,h_2}$ commute if $\mathbf{w}_1 \cdot \mathbf{v}_2 + \mathbf{v}_1 \cdot \mathbf{w}_2 = 0$, or $\mathbf{a}_1^T \mathbf{J} \mathbf{a}_2 = 0$. We say that k Pauli operators $\sigma_{\mathbf{a}_1,r_1,h_1}, \cdots, \sigma_{\mathbf{a}_2,r_2,h_2}$ are independent if there does not exist a set of coefficients $c_1, \cdots c_k \in \mathbb{F}_2$ such that $(\sigma_{\mathbf{a}_1,r_1,h_1})^{c_1} \cdots (\sigma_{\mathbf{a}_2,r_2,h_2})^{c_k} = I^{\otimes n}$ unless $c_1 = \cdots = c_k = 0$. It is not hard to see that if $\mathbf{a}_1, \cdots, \mathbf{a}_k$ are linearly independent in $\mathbb{F}_2^{2n}$, then the Pauli operators $\sigma_{\mathbf{a}_1,r_1,h_1}, \cdots, \sigma_{\mathbf{a}_2,r_2,h_2}$ are independent.

A stabilizer state $|\psi\rangle$ is the simultaneous eigenstate of n commutable and independent Hermitian Pauli operators with eigenvalue 1. These n Pauli operators generate an Abelian group, called the stabilizer group, denoted as $\mathcal{S}_n$. Note that $-I^{\otimes n}$ cannot be in the stabilizer group, as it has no eigenstate with eigenvalue 1. A Clifford operator C preserves the Pauli group under conjugation, which means that if C is a Clifford operator, then $C\sigma_{\mathbf{a},r,h}C^\dagger = \sigma_{\mathbf{a}',r',h'}$. The set of Clifford operators forms the Clifford group $\mathcal{C}_n$.

Tableau representation. Since elements in the stabilizer group are Hermitian, one can represent them with $2n+1$ bits,

$$(v_1, \ldots, v_n, w_1, \ldots, w_n, r) \ . \tag{2.10}$$

For example, the vector for $-X_1Z_2$ is $(1, 0, 0, 1, 1)$. Any stabilizer state can be specified by n stabilizer generators, which commute with each other. Therefore, the state is associated with the following tableau,

$$\bar{\mathrm{S}} = \begin{pmatrix} v_{11} & \ldots & v_{1n} & w_{11} & \ldots & w_{1n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{n1} & \ldots & v_{nn} & w_{n1} & \ldots & w_{nn} & r_n \end{pmatrix} , \tag{2.11}$$

or $\bar{\mathrm{S}} = (\mathrm{S}_x, \mathrm{S}_z, \mathbf{r})$, whose rows define the stabilizer generators. Here, $\mathrm{S}_x, \mathrm{S}_z \in \mathbb{F}_2^{n\times n}$ and $\mathbf{r} = \mathbb{F}_2^n$. We call S_x the X part, S_z the Z part, and $\mathbf{r}$ the phase column of the stabilizer tableau. The fact that $\mathcal{S}_n$ is an Abelian group yields $\mathrm{S}_x\mathrm{S}_z^T + \mathrm{S}_z\mathrm{S}_x^T = \mathbf{0}_n$. Define $\mathrm{S} = (\mathrm{S}_x, \mathrm{S}_z)$, and then we have

$$\mathrm{S}\mathbf{J}\mathrm{S}^T = \mathbf{0}_n \ . \tag{2.12}$$

As an example, the $|0^n\rangle$ state is stabilized by $\langle Z_1, \cdots, Z_n\rangle$, and its stabilizer tableau is given by,

$$\begin{pmatrix} 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 1 & 0 \end{pmatrix} . \tag{2.13}$$

We will call it the standard stabilizer tableau of $|0^n\rangle$.

Change of basis. For a stabilizer state $|\psi\rangle$, one can use different stabilizer tableaus $\bar{\mathrm{S}} = (\mathrm{S}_x, \mathrm{S}_z, \mathbf{r})$ or $\bar{\mathrm{S}}' = (\mathrm{S}'_x, \mathrm{S}'_z, \mathbf{r}')$ to represent it, which correspond to different sets of stabilizer generators. We want to find a relation between these two tableaus. We first look at the multiplication between two stabilizer operators. Let $\mathbf{s}_j^T = (\mathbf{v}_j^T, \mathbf{w}_j^T)$ be the j-th row of S. Since stabilizer operators are Hermitian, we can write them as $\sigma_{\mathbf{s}_j, r_j} := (-1)^{r_j} \sigma_{\mathbf{s}_j}$. Then,

$$\sigma_{\mathbf{s}_1, r_1} \sigma_{\mathbf{s}_2, r_2} = (-1)^{r_1 + r_2 + \mathbf{s}_1^T \mathbf{K} \mathbf{s}_2} \sigma_{\mathbf{s}_1 + \mathbf{s}_2} . \tag{2.14}$$

One can see that the basis change from $\bar{\mathrm{S}}$ to $\bar{\mathrm{S}}'$ can be represented by an invertible matrix $\mathbf{R} \in \mathbb{F}_2^{n\times n}$, such that

$$\mathrm{S}' = \mathbf{R}\mathrm{S} . \tag{2.15}$$

The change in $\mathbf{r}$ is more complicated, because we will have extra minus signs in the phase part when we multiply two Paulis (e.g., $\mathbf{s}_1^T \mathbf{K} \mathbf{s}_2$). Following the procedure in [DDM03], one can obtain

$$\mathbf{r}' = \mathbf{R}\mathbf{r} + \mathrm{diag}(\mathbf{R}^T f_{\mathrm{upper}}(\mathrm{S}\mathbf{K}\mathrm{S}^T)\mathbf{R}) , \tag{2.16}$$

where $f_{\mathrm{upper}}(\mathbf{A})$ returns the strict upper-triangular part of $\mathbf{A}$.

Overlap of two stabilizer states. Given two stabilizer states $|\psi\rangle$ and $|\phi\rangle$, let $\mathrm{Stab}(|\psi\rangle)$ and $\mathrm{Stab}(|\phi\rangle)$ be their stabilizer groups, respectively, which are subgroups of the n-qubit Pauli group. Let $\{P_1, \cdots, P_n\}$ be a choice of generators of $\mathrm{Stab}(|\psi\rangle)$

and $\{Q_1, \cdots, Q_n\}$ be those of $\mathrm{Stab}(|\phi\rangle)$. Note that the set of generators is not unique. Then, the overlap $|\langle\psi|\phi\rangle|$ is determined by their stabilizer groups [AG04].

Proposition 2.1 ([AG04]). *Let $|\psi\rangle$ and $|\phi\rangle$ be two stabilizer states. Then, $\langle\psi|\phi\rangle = 0$ if their stabilizer groups contain the same Pauli operator of the opposite sign. Otherwise, $|\langle\psi|\phi\rangle| = 2^{-g/2}$, where g is the minimum number of different generators over all possible choices.*

Proof. First, suppose $\mathrm{Stab}(|\psi\rangle) = \langle P_1, \cdots, P_n\rangle$ and $\mathrm{Stab}(|\phi\rangle) = \langle Q_1, \cdots, Q_n\rangle$. Then, we can write the states as,

$$|\psi\rangle\langle\psi| = \left(\frac{I+P_1}{2}\right)\cdots\left(\frac{I+P_n}{2}\right) \tag{2.17}$$

$$|\phi\rangle\langle\phi| = \left(\frac{I+Q_1}{2}\right)\cdots\left(\frac{I+Q_n}{2}\right) . \tag{2.18}$$

The square of the overlap is then given by,

$$|\langle\psi|\phi\rangle|^2 = \mathrm{Tr}(|\psi\rangle\langle\psi|\,|\phi\rangle\langle\phi|) = \mathrm{Tr}\left(\frac{I+P_1}{2}\cdots\frac{I+P_n}{2}\frac{I+Q_1}{2}\cdots\frac{I+Q_n}{2}\right) . \tag{2.19}$$

(1) Without loss of generality, suppose $Q_1 = -P_n$. Then, we have,

$$\frac{I+P_n}{2}\frac{I-P_n}{2} = 0 . \tag{2.20}$$

Thus, $\langle\psi|\phi\rangle = 0$ in this case.

(2) Suppose that $P_i = Q_i$ for all $i > g$ and that the group $\langle P_1, \cdots, P_g\rangle$ is not equal to $\langle Q_1, \cdots, Q_g\rangle$. By commutation, we can group the same generators, which gives,

$$\frac{I+P_i}{2}\frac{I+Q_i}{2} = \frac{I+P_i}{2}\frac{I+P_i}{2} = \frac{I+P_i}{2} , \tag{2.21}$$

for $i > g$. This will eliminate the terms related to $Q_{g+1}, \cdots, Q_n$. Then,

$$|\langle\psi|\phi\rangle|^2 = \mathrm{Tr}\left(\frac{I+P_1}{2}\cdots\frac{I+P_n}{2}\frac{I+Q_1}{2}\cdots\frac{I+Q_g}{2}\right) \tag{2.22}$$

$$= \frac{1}{2^g}\langle\psi|(I+Q_1)\cdots(I+Q_g)|\psi\rangle . \tag{2.23}$$

For every term $Q_iQ_j\cdots Q_k \neq I$ in the expansion, there exists a Pauli operator $P \in \mathrm{Stab}(|\psi\rangle)$ that anticommutes with it; otherwise, the term will be in the stabilizer group of $|\psi\rangle$. For such an operator Q, we have $\langle\psi|Q|\psi\rangle = 0$. Indeed, notice that

$$\langle\psi|Q|\psi\rangle = \langle\psi|QP|\psi\rangle = -\langle\psi|PQ|\psi\rangle = -\langle\psi|Q|\psi\rangle \;, \tag{2.24}$$

which implies $\langle\psi|Q|\psi\rangle = 0$. Finally, we have,

$$|\langle\psi|\phi\rangle|^2 = \frac{1}{2^g}\langle\psi|I|\psi\rangle = \frac{1}{2^g}\,, \tag{2.25}$$

and $|\langle\psi|\phi\rangle| = 2^{-g/2}$.

■

2.1.3 Coding Theory

Here, we give some necessary ingredients from coding theory used in this thesis and refer to [MS77] for a detailed treatment. We only consider linear codes over $\mathbb{F}_2$. A linear code, or simply a code C of length m is a linear subspace of $\mathbb{F}_2^m$. The element of a code is called a codeword. One can use a generator matrix $\mathbf{H}$ to represent a code, with its columns spanning the codespace C. The dual code is defined as $C^\perp := \{\mathbf{v} \in \mathbb{F}_2^m : \mathbf{v}\cdot\mathbf{w} = 0 \text{ for } \mathbf{w} \in C\}$, which is also a linear code. It is not hard to see that $C^\perp = \ker(\mathbf{H}^T)$, which implies $\dim(C) + \dim(C^\perp) = m$. A code C is weakly self-dual if $C \subseteq C^\perp$ and (strictly) self-dual if $C = C^\perp$, in which case $\dim(C) = m/2$.

The (Hamming) weight of a vector $\mathbf{v}$, denoted as $|\mathbf{v}|$, is the number of ones in the entries of $\mathbf{v}$. A code C is an even code if all codewords have even Hamming weight and a doubly-even code if all codewords have Hamming weight a multiple of 4. Moreover, we have the following proposition.

Proposition 2.2. *The all-ones vector is a codeword of C if and only if its dual code $C^\perp$ is an even code.*

Proof. Suppose the all-ones vector is a codeword, i.e., $\mathbf{1} \in C$. Then for every $\mathbf{c} \in C^{\perp}$, we have $\mathbf{c} \cdot \mathbf{1} = 0$, which means that $|\mathbf{c}|$ is even and hence $C^{\perp}$ is an even code. Conversely, suppose $C^{\perp}$ is an even code. Then, all codewords will be orthogonal to the all-ones vector, and thus it is in C. ■

We define the notion of (un)biased even codes, which will be useful later.

Definition 2.3. A code C is called a *biased even code* if it is an even code where the number of codewords with Hamming weight 0 modulo 4 and 2 modulo 4 are not equal. It is called an *unbiased even code* otherwise.

Let the (maximum) self-dual subspace of C be $\mathcal{D} := C \bigcap C^{\perp}$, which is itself a weakly self-dual code. Note that $\mathcal{D}$ must be an even code, since all codewords are orthogonal to themselves and hence have even Hamming weight. We have the following lemma.

Lemma 2.4. *A weakly self-dual even code is either a doubly-even code or an unbiased even code. For the former case, all columns of its generator matrix have weight 0 modulo 4 and are orthogonal to each other. For the latter case, there is at least one column in the generator matrix with weight 2 modulo 4.*

The proof of this lemma relies on the following lemma.

Lemma 2.5. *Let* $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2$, *where* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{F}_2^m$ *are of even parity and* $\mathbf{c}_1 \cdot \mathbf{c}_2 = 0$. *Then,* $|\mathbf{c}_3| = 0 \pmod 4$ *if* $|\mathbf{c}_1| = |\mathbf{c}_2| \pmod 4$ *and* $|\mathbf{c}_3| = 2 \pmod 4$ *if* $|\mathbf{c}_1| \neq |\mathbf{c}_2| \pmod 4$.

Proof. Let $|\mathbf{c}_1| = a + 4k_1$ and $|\mathbf{c}_2| = b + 4k_2$, where $a, b \in \{0, 2\}$. Let the size of joint support of $\mathbf{c}_1$ and $\mathbf{c}_2$ be k_{12}. Then, $\mathbf{c}_1 \cdot \mathbf{c}_2 = k_{12} = 0 \pmod 2$, which means that k_{12} is an even number. So,

$$|\mathbf{c}_3| = a + b - 2k_{12} + 4(k_1 + k_2) = a + b \pmod 4 . \tag{2.26}$$

- If $|\mathbf{c}_1| = |\mathbf{c}_2| \pmod 4$, we have $a = b = 0$ or 2. In either case, $|\mathbf{c}_3| = 0 \pmod 4$.

- If $|\mathbf{c}_1| \neq |\mathbf{c}_2| \pmod 4$, we have $a = 0$ and $b = 2$ or $a = 2$ and $b = 0$. In either case, $|\mathbf{c}_3| = 2 \pmod 4$.

■

One can adapt the proof of this lemma to show that a doubly-even code is a weakly self-dual code.

Proposition 2.6. *A doubly-even code is a weakly self-dual code.*

Proof. Suppose C is a doubly-even code, and $\mathbf{c}_1, \mathbf{c}_2 \in C$. Then, we have $|\mathbf{c}_1| = 4k_1$ and $|\mathbf{c}_2| = 4k_2$. Suppose $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2$, which gives $|\mathbf{c}_3| = 4(k_1 + k_2) - 2k_{12}$. Since $\mathbf{c}_3$ is also a codeword of the doubly-even code C, we have $|\mathbf{c}_3| = 0 \pmod 4$, which implies that k_{12} is even and thus $\mathbf{c}_1 \cdot \mathbf{c}_2 = 0$. ■

Now, we are ready to prove Lemma 2.4.

Proof. Let $\mathcal{D}$ be a weakly self-dual even code spanned by $\{\mathbf{c}_1, \cdots, \mathbf{c}_d\}$. Then, $\mathbf{c}_i$'s are all even-parity and orthogonal to each other. Any codeword of $\mathcal{D}$ can be written as $\mathbf{c} = a_1\mathbf{c}_1 + \cdots + a_d\mathbf{c}_d$. According to Lemma 2.5, in the linear combination of $\mathbf{c}$, if there is an odd number of $\mathbf{c}_i$'s with weight 2 modulo 4, then $\mathbf{c}$ will have weight 2 mod 4, and otherwise, $\mathbf{c}$ will have weight 0 mod 4. Therefore, if all $\mathbf{c}_i$'s have weight 0 modulo 4, then $\mathcal{D}$ is doubly-even. If there exist $\mathbf{c}_i$'s with weight 2 modulo 4, then $\mathcal{D}$ is an unbiased even code. ■

One can apply a basis change to the generator matrix $\mathbf{H}$, resulting in $\mathbf{HQ}$, where $\mathbf{Q}$ is an invertible matrix. This will not change the code C. Define the Gram matrix of the generator matrix by $\mathbf{G} := \mathbf{H}^T\mathbf{H}$. A basis change on $\mathbf{H}$ transforms $\mathbf{G}$ into $\mathbf{Q}^T\mathbf{GQ}$, which is a congruent transformation. The rank of Gram matrix is also an invariant under basis change. That is, $\mathrm{rank}(\mathbf{Q}^T\mathbf{GQ}) = \mathrm{rank}(\mathbf{G})$ for $\mathbf{Q}$ invertible. It may be tentative to consider this as a direct consequence of Sylvester's law of inertia, but this is not the case since we are working in $\mathbb{F}_2$. Nevertheless, this can be proven as follows. First, the column space of $\mathbf{GQ}$ is a subspace of $\mathbf{G}$, which implies $\mathrm{rank}(\mathbf{GQ}) \leq \mathrm{rank}(\mathbf{G})$. On the other hand, the column space of $\mathbf{G}$ is a subspace of $\mathbf{GQ}$, because $\mathbf{GQQ}^{-1} = \mathbf{G}$, which implies $\mathrm{rank}(\mathbf{G}) \leq \mathrm{rank}(\mathbf{GQ})$. Therefore, we have $\mathrm{rank}(\mathbf{G}) = \mathrm{rank}(\mathbf{GQ})$. Applying this reasoning again gives $\mathrm{rank}(\mathbf{G}) = \mathrm{rank}(\mathbf{GQ}) = \mathrm{rank}(\mathbf{Q}^T\mathbf{GQ})$.

The rank of the Gram matrix $\mathbf{G} = \mathbf{H}^T\mathbf{H}$ can be related to the code C in the following way.

Proposition 2.7. *Given a generator matrix* $\mathbf{H}$*, let its Gram matrix be* $\mathbf{G} = \mathbf{H}^T\mathbf{H}$ *and the generated code be* C*. Let* $\mathcal{D} = C \bigcap C^{\perp}$*, where* $C^{\perp}$ *is the dual code of* C*. Then,* $\mathrm{rank}(\mathbf{G}) = \dim(C) - \dim(\mathcal{D})$.

Proof. Suppose $\mathbf{H} \in \mathbb{F}_2^{m\times n}$ and let $g = \mathrm{rank}(\mathbf{G})$, where $\mathbf{G} = \mathbf{H}^T\mathbf{H}$. Let $r = \dim(C)$ and $d = \dim(\mathcal{D})$, where $d \leq r \leq n$. We first prove for the case $r = n$. In this case, every codeword $\mathbf{c} \in C$ can be expressed as $\mathbf{c} = \mathbf{Ha}$ for a unique $\mathbf{a} \in \mathbb{F}_2^n$ and the correspondence is one-to-one. Then, one can prove that $\mathbf{Ha} \in \mathcal{D}$ is equivalent to $\mathbf{a} \in \ker(\mathbf{G})$ and thus d is equal to the dimension of $\ker(\mathbf{G})$, which is $d = r - g$. Indeed, if $\mathbf{Ha} \in \mathcal{D}$, we have $\mathbf{Ga} = \mathbf{H}^T\mathbf{Ha} = \mathbf{0}$, which means that $\mathbf{a} \in \ker(\mathbf{G})$. Conversely, if $\mathbf{a} \in \ker(\mathbf{G})$, we have $\mathbf{H}^T\mathbf{Ha} = \mathbf{0}$, which means that $\mathbf{Ha} \in C^{\perp}$. Since $\mathbf{Ha} \in C$, this implies $\mathbf{Ha} \in \mathcal{D}$.

Now, we consider the case $r < n$. In this case, there always exists an invertible matrix $\mathbf{Q}$ such that $\mathbf{HQ} = (\mathbf{H}', \mathbf{0}_{m\times(n-r)})$ and $\mathbf{H}' \in \mathbb{F}_2^{m\times r}$ is a generator matrix of C that is of full column rank. Moreover,

$$\mathrm{rank}(\mathbf{G}) = \mathrm{rank}(\mathbf{Q}^T\mathbf{GQ}) = \mathrm{rank}(\mathbf{H}'^T\mathbf{H}') \ . \tag{2.27}$$

Then, applying the previous reasoning to $\mathbf{H}'$ yields that $d = r - \mathrm{rank}(\mathbf{H}'^T\mathbf{H}') = r - g$. ■

2.1.4 Matrix Factorization over $\mathbb{F}_2$

Here, we give a review of the matrix factorization over $\mathbb{F}_2$. Given a symmetric matrix $\mathbf{G}$, the goal is to find a matrix $\mathbf{H}$, so that $\mathbf{G} = \mathbf{H}^T\mathbf{H}$. The matrix $\mathbf{H}$ is called a factor of $\mathbf{G}$. Although the matrix factorization problem is well-studied over the real field, it is less-studied over $\mathbb{F}_2$. Nevertheless, Lempel gave an algorithm for finding a factor of $\mathbf{G}$

over $\mathbb{F}_2$ [Lem75]. In fact, Lempel's algorithm allows one to find a minimal factor of $\mathbf{G}$, which is a factor with the smallest number of rows.

To illustrate Lempel's algorithm, we need the following lemma.

Lemma 2.8. *Let $\mathbf{Z}$ be a binary matrix with an even number of rows, and $\mathbf{Z}^T \cdot \mathbf{1} = \mathbf{0}$. Let $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathbf{1} \cdot \mathbf{x}^T$, where $\mathbf{x}$ is an arbitrary binary vector of the same length as the number of columns of $\mathbf{Z}$. Then, $\mathbf{Z}^T\mathbf{Z} = \tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}$.*

Proof. Observe that,

$$\tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}} = \mathbf{Z}^T\mathbf{Z} + \mathbf{x} \cdot \mathbf{1}^T \cdot \mathbf{Z} + \mathbf{Z}^T \cdot \mathbf{1} \cdot \mathbf{x}^T + \mathbf{x} \cdot \mathbf{1}^T \cdot \mathbf{1} \cdot \mathbf{x}^T . \tag{2.28}$$

The second and the third terms are zero since $\mathbf{Z}^T \cdot \mathbf{1} = \mathbf{0}$. The fourth term is zero because there is an even number of rows in $\mathbf{Z}$ and thus $\mathbf{1}^T \cdot \mathbf{1} = 0$. So, $\mathbf{Z}^T\mathbf{Z} = \tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}$. ■

Lempel's algorithm starts by constructing an elementary factorization of the symmetric matrix $\mathbf{G}$. We define two sets based on $\mathbf{G}$:

$$\begin{aligned} N_1 &:= \left\{k : \sum_{j=1}^{n} \mathbf{G}_{kj} = 1\right\} \\ N_2 &:= \left\{(i, j) : \mathbf{G}_{ij} = 1 \text{ and } i < j\right\} . \end{aligned} \tag{2.29}$$

Then, we have the following lemma.

Lemma 2.9 ([Lem75]). *Given an $n \times n$ binary symmetric matrix $\mathbf{G}$, define N_1 and N_2 as in Eq. (2.29). Let $\mathbf{E}_0$ be a matrix with $|N_1| + |N_2|$ rows and n columns satisfying the following conditions. Each row in the first $|N_1|$ rows of $\mathbf{E}_0$ has one only in the k-th position for $k \in N_1$, and each row in the last $|N_2|$ rows has ones in the i-th and j-th positions for $(i, j) \in N_2$, and zeros elsewhere. Then, $\mathbf{G} = \mathbf{E}_0^T\mathbf{E}_0$.*

Here, the factor $\mathbf{E}_0$ contains at most $O(n^2)$ rows. Although this already gives a factor of $\mathbf{G}$, the goal of Lempel's algorithm is to find the minimal factor. After obtaining $\mathbf{E}_0$, Lempel's algorithm uses an iterative procedure to eliminate rows in the factor, until certain stopping condition is satisfied, resulting in the following sequence

$$\mathbf{E}_0 \to \mathbf{E}_1 \to \mathbf{E}_2 \to \cdots \to \mathbf{B} . \tag{2.30}$$

Above, the number of rows in $\mathbf{E}_i$ will be reduced at least by 1 compared to $\mathbf{E}_{i-1}$. Strictly speaking, the last factor $\mathbf{B}$ in this sequence corresponds to the minimal factor of $\mathbf{G}$ only if $\mathbf{G}$ is non-singular. But for our later purpose in Section 4.3, we do not require a minimal factor.

In the rest of this section, we give a detailed description of how this iterative procedure works. For the elementary factor $\mathbf{E}_0$ of $\mathbf{G}$, if its rows are not linearly independent (and we assume the number of rows in $\mathbf{E}_0$ is greater than 3), then one can always partition $\mathbf{E}_0$ into two part; that is,

$$\mathbf{E}_0 = \begin{pmatrix} \mathbf{F} \\ \mathbf{K} \end{pmatrix}, \tag{2.31}$$

up to a reordering of rows, where $\mathbf{K}$ contains at least two rows satisfying $\mathbf{K}^T \cdot \mathbf{1} = \mathbf{0}$. Such a $\mathbf{K}$ can be found by identifying the independent rows in $\mathbf{E}_0$ and representing other rows with these independent rows. For example, if $\mathbf{p}_1^T$ and $\mathbf{p}_2^T$ are independent, while $\mathbf{p}_3^T = \mathbf{p}_1^T + \mathbf{p}_2^T$, one may set $\mathbf{K}$ to be the first three rows and $\mathbf{F}$ to be the rest. This ensures that $\mathbf{K}^T \cdot \mathbf{1} = \mathbf{0}$ since $\mathbf{p}_3 + \mathbf{p}_1 + \mathbf{p}_2 = \mathbf{0}$.

Now, there are two cases. If $\mathbf{K}$ consists of two identical rows, then $\mathbf{K}$ actually contributes nothing to $\mathbf{G}$. One removes $\mathbf{K}$ and denote the new factor $\mathbf{E}_1$, which contains two rows less than $\mathbf{E}_0$. In the general case, define

$$\mathbf{Z} := \begin{cases} \mathbf{K} & \text{if } r(\mathbf{K}) \text{ is even,} \\ \begin{bmatrix} \mathbf{K} \\ \mathbf{0}^T \end{bmatrix} & \text{if } r(\mathbf{K}) \text{ is odd.} \end{cases} \tag{2.32}$$

Then, the number of rows in the matrix $\mathbf{Z}$ is even, $\mathbf{Z}^T\mathbf{1} = \mathbf{0}$ and $\mathbf{Z}^T\mathbf{Z} = \mathbf{K}^T\mathbf{K}$, which implies

$$\mathbf{E}_0^* := \begin{pmatrix} \mathbf{F} \\ \mathbf{Z} \end{pmatrix} \tag{2.33}$$

is also a factor of $\mathbf{G}$. Let $\mathbf{F}^{(1)}$ and $\mathbf{Z}^{(1)}$ be the first row in $\mathbf{F}$ and $\mathbf{Z}$, respectively. If $\mathbf{F}$ is null (i.e., contains zero rows), set $\mathbf{F}^{(1)}$ to be an all-zeros row. Let $\mathbf{x}^T = \mathbf{F}^{(1)} + \mathbf{Z}^{(1)}$, and define $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathbf{1} \cdot \mathbf{x}^T$. Then, according to Lemma 2.8, we have $\mathbf{Z}^T\mathbf{Z} = \tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}$ and thus

$$\tilde{\mathbf{E}}_0 = \begin{pmatrix} \mathbf{F} \\ \tilde{\mathbf{Z}} \end{pmatrix} \tag{2.34}$$

is a factor of $\mathbf{G}$. But observe that the first row of $\tilde{\mathbf{Z}}$ is actually $\mathbf{F}^{(1)}$, and their net contribution to $\mathbf{G}$ is null. Therefore, we can eliminate both rows and obtain $\mathbf{E}_1$. If $r(\mathbf{K})$ is even, then $r(\mathbf{E}_1) = r(\mathbf{E}_0) - 2$; if $r(\mathbf{K})$ is odd, then $r(\mathbf{E}_1) = r(\mathbf{E}_0) - 1$. Here, $r(\mathbf{A})$ denotes the number of rows in a matrix $\mathbf{A}$. Overall, $\mathbf{E}_1$ contains at least one row less than $\mathbf{E}_0$.

Repeat this procedure for $\mathbf{E}_1$ and so on until the number of rows in $\mathbf{E}_j$ is less than or equal to 3, or until all rows in $\mathbf{E}_j$ are independent. If all rows in $\mathbf{E}_j$ are independent, one cannot find a proper $\mathbf{K}$ and the iteration terminates. If the number of rows in $\mathbf{E}_j$ is 3 and not all rows are independent, then there will be two cases. The first case is that $\mathbf{E}_j$ contains two same rows, which means that one can remove them and reduce the number of rows to 1. We will denote this one-row matrix $\mathbf{B}$. The second case is that $\mathbf{E}_j$ contains two different rows and the third row is the sum of the other two. In this case, $\mathbf{K}$ will be $\mathbf{E}_j$ itself, while $\mathbf{F}$ is null. But it is not hard to see that applying the above procedure cannot reduce rows anymore and therefore the iteration should be terminated. In any case, we denote the last factor in this sequence $\mathbf{B}$, which corresponds to the minimal factor of $\mathbf{G}$ if $\mathbf{G}$ is non-singular. This concludes the algorithm.

2.2 Quantum Computational Advantage

In recent years, an important research topic has been the pursuit of demonstrating quantum computational advantage. To quantify the computational advantage provided by quantum computers, we need tools from computational complexity theory. Computational complexity theory formally studies the computational resources required to solve a given task; it classifies the computational problems into different classes according to the resources, and studies the relation between different classes [AB09]. Computational resources include time, space, randomness and quantum mechanics.

Basic complexity theory. One basic type of computational task is the decision problem, which only requires 0/1 output. An elementary complexity class is P, which is the class of decision problems that a deterministic Turing machine can solve efficiently; here, ‘efficiently’ means using polynomial time steps. In this definition, Turing machine is an abstract model of classical computers, and hence P captures the power of deterministic classical algorithms running in polynomial time. Another important classical complexity class is NP, which is the class of decision problems that a deterministic Turing machine can efficiently *verify*. It is clear that P $\subseteq$ NP, because if a problem can be efficiently solved, then it can be efficiently verified.

If we replace the deterministic Turing machine by a quantum Turing machine, then BQP [BB92, BV97] and QMA [Kni96, KSV02, Wat00, AN02] can be defined, which are the quantum analogues of P and NP, respectively. The notion of quantum Turing machine was first attempted by Deutsch [Deu85], and it was later defined as a deterministic Turing machine augmented with quantum coin flips in [BV97]. There are other variants of quantum Turing machines, and Yao showed that these models are (polynomially) equivalent to a model called quantum circuits [Yao93], which was also introduced by Deutsch [Deu89]. Therefore, both quantum Turing machine and quantum circuits can be used as models of quantum computers, and quantum circuits gain more popularity after Yao’s equivalence proof.

BQP **in the world of classical complexity classes.** The next question is how large BQP is. It can be shown that quantum computers are at least as powerful as classical computers, i.e., P $\subseteq$ BQP, from the results of Bernstein and Vazirani [BV97]. Moreover, quantum computers can efficiently solve some problems that are considered hard for classical computers, like factorization [Sho94], which is an instance of NP problems. But Bennett et al. [BBBV97] gave evidence that NP is unlikely contained in

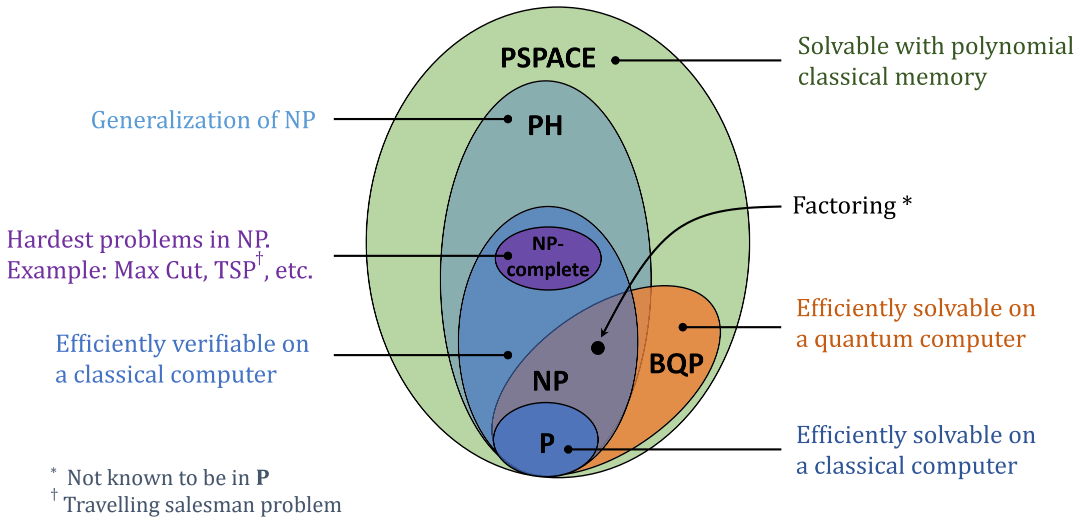

PSPACE
Solvable with polynomial classical memory
Generalization of NP
PH
Factoring *
Hardest problems in NP. Example: Max Cut, TSP†, etc.
NP-complete
Efficiently solvable on a quantum computer
BQP
Efficiently verifiable on a classical computer
NP
P
Efficiently solvable on a classical computer
* Not known to be in **P**
† Travelling salesman problem

Figure 2.1: BQP in the world of classical complexity classes. Here, PH stands for the polynomial hierarchy and it is a generalization of NP [Sto76].

BQP, meaning that quantum computers might not be able to solve all problems in NP. On the other hand, it was shown that BQP is contained in PSPACE [BV97], the set of problems decidable in polynomial space with a deterministic Turing machine. The relation of BQP with other complexity classes is shown in Fig. 2.1. Therefore, if one proves that P is not equal to BQP, one also separates P from PSPACE, thereby resolving a major open problem in complexity theory.

Notions of classical simulation. The interplay between quantum mechanics and complexity theory suggests the outstanding difficulty in establishing quantum advantage formally. Indeed, it is notoriously challenging to prove separation between complexity classes. A work by Terhal and DiVincenzo [TD02] shed light on establishing quantum advantage from another perspective. Specifically, they showed that it is impossible to classically efficiently simulate constant-depth quantum circuits, which is a restricted model of quantum computation, unless polynomial hierarchy (PH) collapses. Here, the non-collapse of polynomial hierarchy [Sto76] is a consensus in complexity theory and it is a generalization of another common belief that P $\neq$ NP. The importance of this result is that it reduces the intractability of simulating constant-depth quantum circuits to the conjecture that is not related to quantum mechanics itself. It inspires later research in quantum computational supremacy [Pre12, LBR17, HM17].

The notion of classical simulation used in [TD02] is called strong simulation in [BJS11]. Formally, it is defined as classically computing the output probabilities of any subset of qubits accurately. However, strong simulation of quantum circuits is #P-hard [FGHP99], which means that it might be hard for quantum computers as well. A more natural task would be weak simulation, which is a sampling problem and requires an algorithm to output *samples* (i.e., bit strings) from the output distribution of a quantum circuit. It can be shown that strong simulation without error implies weak simulation without error [TD02]. In reality, quantum computers are noisy and the output distribution will deviate from the ideal one. Therefore, we can also allow some error in classical simulation. Two common notions of simulation errors are multiplicative error and total variation distance, both defined in terms of the distance between the sampled distribution and the ideal distribution. It is natural to ask whether it is possible to prove the classical hardness of weak simulation, given some widely-accepted conjectures in complexity theory. This is what quantum computational supremacy concerns [Pre12].

Quantum computational supremacy. Here, we will discuss three proposals for quantum computational supremacy, namely, IQP (Instantaneous Quantum Polynomial-time) sampling [BJS11], boson sampling [AA11], and random-circuit sampling [BIS+18]. These three tasks require to sample from the output distributions of IQP circuits, random linear optical networks, and certain family of random quantum circuits, respectively. IQP sampling and boson sampling are the first two proposals with rigorous complexity-theoretic foundations. Random-circuit sampling (RCS) was first proposed because it fit well into the hardware of the Google team [BIS+18]. Its theoretical foundations were subsequently laid down by community efforts detailed later. Experimentally, Google [AAB+19] and USTC [ZCC+21, WBC+21] demonstrated

RCS with 50-60 superconducting qubits, and USTC also implemented a variant of boson sampling [ZWD+20]. The scale of these experiments are already considered to be formidable for classical simulation.

Complexity-theoretic foundations. The IQP model was first proposed in the context of cryptographic test of quantum computers [SB09]. Then, multiplicative-error weak simulation of IQP circuits was proven to be intractable for classical computers, given the conjecture that the polynomial hierarchy does not collapse [BJS11]. Similar result was also proven for boson sampling [AA11]. However, multiplicative error is a too stringent error notion, even for quantum computers. It would be more reasonable to consider total variation distance.

For both models, classical weak simulation with total variation distance can also be proven to be impossible, with extra conjectures apart from the non-collapse of PH [AA11, BMS16]. Specifically, the hardness proofs are based on mapping the output probability to certain well-studied quantities in counting problems, which is the permanent for boson sampling and Ising-model partition function for IQP sampling. Those extra conjectures are related to these quantities. For boson sampling, the two extra conjectures are the anti-concentration property of permanents and the average-case hardness of approximating permanents [AA11]. Both conjectures remain open for boson sampling and have not been widely examined. However, for IQP sampling, the anti-concentration property can be established [BMS16].

The classical hardness of RCS can be proven by similar techniques and based on similar conjectures. But unlike IQP sampling and boson sampling, the output probabilities of random quantum circuits do not correspond to well-studied quantities in counting problems. Nevertheless, the anti-concentration property of RCS was proven from the structure of random quantum circuits [HBVSE18, BCG21, DHJB20]. As for the approximate average-case hardness for RCS, recently there are a series of work towards proving this conjecture [BFNV19, Mov18, Mov19], but it still remains open.

2.3 Verifiable Quantum Advantage

The achievability of quantum advantages depends not only on the delicate design of quantum algorithms but also relies on the quality of the quantum hardware that performs the algorithms. How to verify the output of physical quantum devices is a long-standing question, which was first asked by Gottesman [1]. In the context of verifying arbitrary quantum computation, there have been a plethora of important results [BFK09, BFK10, ABOE08, ABOEM08, FK17, FHcvM18, RUV13, Mah18]. The more relevant context to this thesis is called test of quantumness in the literature, which is to verify quantum computational capability beyond classical computing. In light of NISQ, the verification protocol shall be run in a setting that uses minimal quantum and classical computing resources. A motivating example is given by Shor's algorithm for integer factorization [Sho94], which is appealing in that hard instances can be easily generated and verified classically yet finding the solution is beyond the capabilities of classical computers. However, this also has the drawback that the quantum solution also seems to be beyond the capabilities of NISQ devices.

Recently, there have been tests of quantumness that combine the power of both interactive proofs and cryptographic assumptions [BCM+18, BKVV20, KMCVY21]. This class of cryptographic verification protocols uses a primitive called trapdoor claw-free (TCF) functions, which has the following properties. First, it is a 2-to-1 function that is hard to invert, meaning that given $y = f(x) = f(x')$, it is hard for an efficient classical computer to find the preimage pair (x, x'). Second, given a trapdoor to the function $f(x)$, the preimage pair can be efficiently found on a classical computer. We will refer to this class of verification protocols as the TCF-based protocols. The TCF-based protocols require the quantum prover to prepare the state of the form $\sum_x |x\rangle |f(x)\rangle$. Although a recent experiment implemented a small-scale TCF-based protocol on a

[1]`https://www.scottaaronson.com/blog/?p=284`. At first sight, this seems a simple question. One may ask the quantum cloud to run a classical intractable task which is feasible for a quantum computer. This idea is not practical as it is equivalent to separating BQP (bounded-error quantum polynomial time) and P (polynomial time), one of the most important open problem in quantum complexity theory.

trapped-ion platform [ZKML+22], this class of protocols is still very challenging for the current technology.

Another class of verification protocols is based on IQP circuits initiated by Shepherd and Bremner [SB09]. IQP (Instantaneous Quantum Polynomial-time) circuits are a family of quantum circuits that employ only commuting gates, typically diagonal in the Pauli-X basis. In IQP-based verification protocols, the verifier generates a pair consisting of an IQP circuit U_{IQP} and a secret key $\mathbf{s} \in \{0, 1\}^n$. After transmitting the classical description of the IQP circuit to the prover, the verifier requests measurement outcomes in the computational basis. Then, the verifier uses the secret to determine whether the measurement outcomes are from a real quantum computer. Such a challenge seems hard for classical computers, as random IQP circuits are believed to be computationally difficult to simulate classically with minimal physical resources, assuming some plausible complexity-theoretic assumptions such as the non-collapse of polynomial hierarchy [BJS11, BMS16, BMS17].

But the use of random IQP circuits is not suitable for the verification protocol, due to the anti-concentration property [YC20, BMS16]. In the Shepherd-Bremner scheme, the verifier constructs the pair $(U_{\text{IQP}}, \mathbf{s})$, according to an obfuscated quadratic-residue code (QRC) [MS77]. While the Shepherd-Bremner scheme was experimentally attractive, a drawback was that in comparison with the TCF-based protocols, the cryptographic assumptions were non-standard and have had comparatively unstudied. In 2019, a loophole was found in the Shepherd-Bremner scheme, which allows a classical prover to regularly find the secret efficiently [KM19]. Once the secret is found, a classical prover can easily generate data to spoof the test. Since the IQP-based protocols offer a promising avenue for achieving verifiability beyond classical computing with fewer resources than Shor's algorithm, it is imperative to investigate whether it is possible to extend and fix the Shepherd-Bremner construction.

Part I

Classical Verification of Quantum Devices

Chapter 3

IQP-based Verification Protocols

IQP stands for instantaneous quantum polynomial-time, and it is a family of quantum circuits that consist of only commuting gates. In what follows, we focus on a specific family of IQP circuits, the X program [SB09], where all local gates are diagonal in the Pauli-X basis. One can represent this family of IQP circuits by a time evolution of the Hamiltonian H, which consists of only products of Pauli X's. For example, for $H = X_1X_2X_4 + X_3X_4 + X_1X_3$, the corresponding IQP circuit is given by $U_{\text{IQP}} = e^{i\theta H} = e^{i\theta X_1X_2X_4}e^{i\theta X_3X_4}e^{i\theta X_1X_3}$. In the general case, the evolution time for each term in H can be different, but we mainly consider the case where $\theta = \pi/8$ for all terms. One can also use an m-by-n binary matrix to represent the IQP Hamiltonian, where m is the number of local terms and n is the number of qubits. Each row of the matrix represents one local term and the locations of 1's indicate the qubits that it acts on. The matrix representation for H in the previous example is given by

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} . \tag{3.1}$$

Before proceeding, we first introduce some notations. We mainly work on the field $\mathbb{F}_2$. We use bold upper-case letters such as $\mathbf{H}$ to denote a matrix and bold lower-case letters such as $\mathbf{s}$ to denote a vector. If not stated otherwise, a vector is referred to as a column vector, and a row vector will be added the transpose symbol, like $\mathbf{p}^T$. The inner product between two vectors $\mathbf{x}$ and $\mathbf{s}$ is denoted as $\mathbf{x} \cdot \mathbf{s}$; sometimes we will also use $\mathbf{H} \cdot \mathbf{s}$ to denote the matrix multiplication. We use $\mathrm{col}(\mathbf{H})$ and $\mathrm{row}(\mathbf{H})$ to denote the collections of columns and rows of a matrix $\mathbf{H}$, respectively. We use $c(\mathbf{H})$ and $r(\mathbf{H})$ to denote the number of columns and the number of rows of a matrix $\mathbf{H}$, respectively. The rank of a matrix $\mathbf{H}$ is denoted as $\mathrm{rank}(\mathbf{H})$. We use $\ker(\mathbf{H})$ to denote the kernel space of $\mathbf{H}$, i.e., the set of vector $\mathbf{v}$ such that $\mathbf{H}\mathbf{v} = \mathbf{0}$. We call two square matrices $\mathbf{A}$ and $\mathbf{B}$ congruent if there exists an invertible matrix $\mathbf{Q}$ satisfying $\mathbf{A} = \mathbf{Q}^T\mathbf{B}\mathbf{Q}$, denoted as $\mathbf{A} \sim_c \mathbf{B}$. We call such an transformation *congruent transformation.*

The all-ones vector will be denoted as $\mathbf{1}$, with its dimension inspected from the context; the similar rule applies to the all-zeros vector (or matrix) $\mathbf{0}$. The $n \times n$ identity matrix is denoted as $\mathbf{I}_n$. For a vector $\mathbf{x}$, we define its support as $\mathrm{supp}(\mathbf{x}) := \{j : x_j = 1\}$. We define $[n] := \{1, 2, \cdots, n\}$. If not stated otherwise, a full-rank matrix is referred to a matrix with full column rank.

We denote the linear subspace spanned by a set of vectors $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ as $\langle \mathbf{c}_1, \ldots, \mathbf{c}_k \rangle$. Given linear subspaces $V = \langle \mathbf{c}_1, \ldots, \mathbf{c}_l \rangle$ and $U = \langle \mathbf{c}_1, \ldots, \mathbf{c}_k \rangle$ with $k < l$, we denote the complement subspace of U in V with respect to the basis $\{\mathbf{c}, \ldots, \mathbf{c}_l\}$ by $(V/U)_{\mathbf{c}_1,\ldots,\mathbf{c}_l}$; namely, $(V/U)_{\mathbf{c}_1,\ldots,\mathbf{c}_l} := \langle \mathbf{c}_{k+1}, \ldots, \mathbf{c}_l \rangle$. Usually, we are not interested in a specific basis, so we use V/U to denote a random complement subspace of U in V, i.e., $V/U \leftarrow_{\mathcal{R}} \{\langle \mathbf{c}_{k+1}, \ldots, \mathbf{c}_l \rangle : V = \langle \mathbf{c}_1, \ldots, \mathbf{c}_l \rangle, U = \langle \mathbf{c}_1, \ldots, \mathbf{c}_k \rangle\}$, where $\leftarrow_{\mathcal{R}}$ denotes a random instance from a set. We let $V \backslash U := \{\mathbf{v} : \mathbf{v} \in V, \mathbf{v} \notin U\}$ be the ordinary complement of two sets.

3.1 General Framework

The general framework for IQP-based verification protocol is shown in Fig. 3.1. Here, the verifier first generates the pair of IQP Hamiltonian H and the secret $\mathbf{s}$. She will pre-compute the correlation function $\langle \mathcal{Z}_{\mathbf{s}} \rangle := \langle 0^n | U_{\mathrm{IQP}}^{\dagger} \mathcal{Z}_{\mathbf{s}} U_{\mathrm{IQP}} | 0^n \rangle$, which can be achieved with the classical algorithms in [YC20]. Here, $\mathcal{Z}_{\mathbf{s}} := Z_1^{s_1} \otimes \cdots \otimes Z_n^{s_n}$ is a Pauli-Z product,

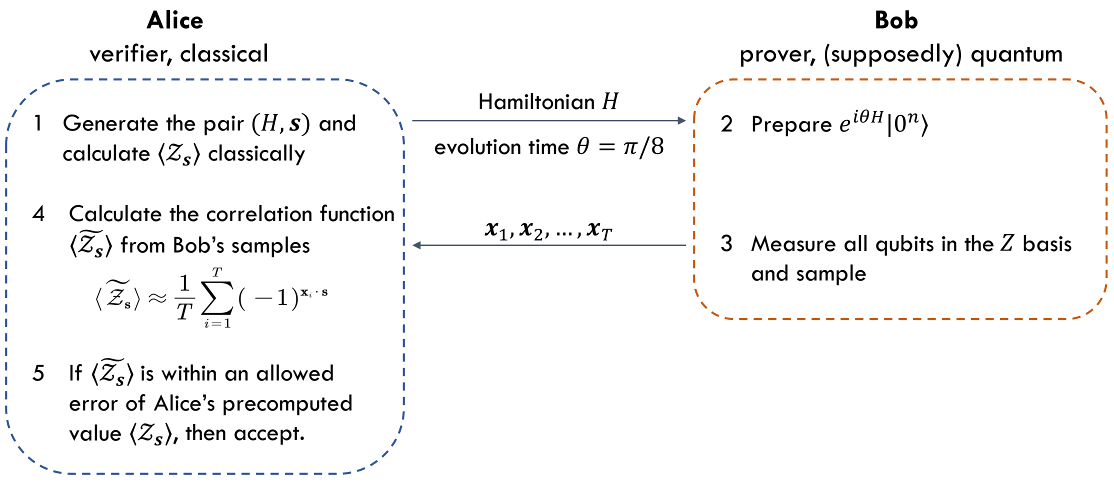

Alice
verifier, classical
1 Generate the pair (H, s) and calculate $\langle \mathcal{Z}_s \rangle$ classically
4 Calculate the correlation function $\langle \widetilde{\mathcal{Z}_s} \rangle$ from Bob's samples
$\langle \widetilde{\mathcal{Z}_s} \rangle \approx \frac{1}{T}\sum_{i=1}^{T}(-1)^{\mathbf{x}_i \cdot \mathbf{s}}$
5 If $\langle \widetilde{\mathcal{Z}_s} \rangle$ is within an allowed error of Alice's precomputed value $\langle \mathcal{Z}_s \rangle$, then accept.
Hamiltonian H
evolution time $\theta = \pi/8$
$x_1, x_2, \ldots, x_T$
Bob
prover, (supposedly) quantum
2 Prepare $e^{i\theta H}|0^n\rangle$
3 Measure all qubits in the Z basis and sample

Figure 3.1: Schematic for IQP-based verification protocol in the case $\theta = \pi/8$.

defined by the secret $\mathbf{s}$. Then, the classical description of the Hamiltonian H is sent to the prover, while the secret is kept on the verifier's side; the verifier also instructs the prover the evolution time for each term of the Hamiltonian. After that, the prover apply the time evolution $e^{i\theta H}$ to $|0^n\rangle$, and measure all qubits in the computational basis. The prover repeats this process T times and obtain a set of samples $\mathbf{x}_1, \cdots, \mathbf{x}_T$, which will be sent back to the verifier. From the prover's measurement samples, the verifier estimates the correlation function with respect to $\mathbf{s}$ by

$$\langle \widetilde{\mathcal{Z}_\mathbf{s}} \rangle := \frac{1}{T}\sum_{i=1}^{T}(-1)^{\mathbf{x}_i \cdot \mathbf{s}} . \tag{3.2}$$

If the value of $\langle \widetilde{\mathcal{Z}_\mathbf{s}} \rangle$ is within an allowed error of the ideal value $\langle \mathcal{Z}_\mathbf{s} \rangle$, then the verifier accepts the result and the prover passes the verification.

There are two important steps for the verifier in the IQP-based verification protocol. The first one is to evaluate the correlation function in advance, so that the verifier can compare the value obtained from the prover's measurement outcomes with the precomputed value. The second one is to construct a suitable pair $(H, \mathbf{s})$, so that the correlation function $\langle \mathcal{Z}_\mathbf{s} \rangle = \langle 0^n | e^{-i\theta H} \mathcal{Z}_\mathbf{s} e^{i\theta H} | 0^n \rangle$ is sufficiently away from zero. Otherwise, the verifier may need to request a super-polynomial number of samples from the prover to make the statistical error small enough, which makes the protocol inefficient.

3.2 IQP Circuit Correlation Functions

In this section, we will discuss the properties of the correlation functions $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ of IQP circuits. To evaluate the correlation function, we first note that the Hamiltonian can be divided into two part $H = H_{\mathbf{s}} + R_{\mathbf{s}}$ based on the secret $\mathbf{s}$. Here, the part $H_{\mathbf{s}}$ anti-commutes with $\mathcal{Z}_{\mathbf{s}}$, i.e., $\{\mathcal{Z}_{\mathbf{s}}, H_{\mathbf{s}}\} = 0$, and the redundant part $R_{\mathbf{s}}$ commutes with $\mathcal{Z}_{\mathbf{s}}$, i.e., $[R_{\mathbf{s}}, \mathcal{Z}_{\mathbf{s}}] = 0$. Correspondingly, the matrix representations satisfy $\mathbf{H}_{\mathbf{s}}\,\mathbf{s} = \mathbf{1}$ and $\mathbf{R}_{\mathbf{s}}\,\mathbf{s} = \mathbf{0}$. Due to these commutation relations, the value of the correction function only depends on the $H_{\mathbf{s}}$, i.e.,

$$\langle \mathcal{Z}_{\mathbf{s}} \rangle = \langle 0^n | e^{i2\theta H_{\mathbf{s}}} | 0^n \rangle \ . \tag{3.3}$$

Then, one can observe an intriguing point from this expression. When $\theta = \pi/8$, the IQP circuit is non-Clifford and there is complexity-theoretic evidence that the IQP circuits in this setting is hard to simulate classically [BMS16]. However, $e^{i2\theta H_{\mathbf{s}}}$ becomes a Clifford circuit and the correlation function can be computed efficiently [She10, YC20]. Indeed, $\langle \mathcal{Z}_{\mathbf{s}} \rangle = \langle 0^n | e^{i(\pi/4)H_{\mathbf{s}}} | 0^n \rangle$ actually corresponds to an amplitude of the Clifford circuit $e^{i(\pi/4)H_{\mathbf{s}}}$. In this way, the verifier can evaluate the correlation function efficiently using the Gottesman-Knill algorithm [Got99] and the subsequent improvement [AG04]. Specifically, the value of $|\langle \mathcal{Z}_{\mathbf{s}} \rangle|$ is either 0 or $2^{-g/2}$, where $0 \le g \le n$ is an integer determined by the stabilizer groups of $|0^n\rangle$ and $e^{i(\pi/4)H_{\mathbf{s}}}\,|0^n\rangle$, respectively. We will discuss the connection between IQP circuits and the stabilizer formalism in more details in Section 4.1.

For the general case, when the angle for each Hamiltonian term is arbitrary and possibly different, the correlation function can still be estimated to within an additive error ϵ using Monte Carlo sampling. We have the following theorem [YC20]:

Theorem 3.1. *The correlation function of any Pauli-Z product,* $\langle \mathcal{Z}_{\mathbf{s}} \rangle := \langle 0^n | U_{\mathrm{IQP}}^{\dagger} \mathcal{Z}_{\mathbf{s}} U_{\mathrm{IQP}} | 0^n \rangle$ *of an IQP circuit* U_{IQP}, *can be classically estimated to within additive error* ϵ *with probability* $1 - \delta$ *by Monte-Carlo sampling in time* $O\!\left(\frac{1}{\epsilon^2} \log \frac{2}{\delta}\right)$.

Proof. With respect to $\mathcal{Z}_\mathbf{s}$, let the non-commuting part and the commuting part of the IQP circuit be U_M and U_R, respectively. Since each gate in the IQP circuit commutes with each other, we can write without loss of generality that $U_{\text{IQP}} = U_R U_M$, which gives $\langle \mathcal{Z}_\mathbf{s} \rangle = \langle 0^n | U_M^\dagger U_R^\dagger \mathcal{Z}_\mathbf{s} U_R U_M | 0^n \rangle$. From the fact that U_R commutes with $\mathcal{Z}_\mathbf{s}$, we have,

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \langle 0^n | U_M^\dagger \mathcal{Z}_\mathbf{s} U_M | 0^n \rangle \ . \tag{3.4}$$

Furthermore, since the Hamiltonian terms in U_M anti-commute with $\mathcal{Z}_\mathbf{s}$, we have $U_M^\dagger \mathcal{Z}_\mathbf{s} U_M = \mathcal{Z}_\mathbf{s} U_M^2$ and

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \langle 0^n | U_M^2 | 0^n \rangle \ . \tag{3.5}$$

Then we apply Hadamard gates to change the basis,

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{2^n} \sum_{\mathbf{x},\mathbf{y}} \langle \mathbf{x} | (U_M^{(z)})^2 | \mathbf{y} \rangle \tag{3.6}$$

$$= \frac{1}{2^n} \sum_{\mathbf{x}} \langle \mathbf{x} | (U_M^{(z)})^2 | \mathbf{x} \rangle \ , \tag{3.7}$$

where $U_M^{(z)}$ is obtained from U_M by replacing Pauli-X with Pauli-Z and we have used the fact that $\langle \mathbf{x} | (U_M^{(z)})^2 | \mathbf{y} \rangle = 0$ if $\mathbf{x} \neq \mathbf{y}$. Each term in the summation can be efficiently calculated by tracking the phase and $\langle \mathcal{Z}_\mathbf{s} \rangle$ is the uniform average of the terms in the summation. Therefore, using the Chernoff bound, $\langle \mathcal{Z}_\mathbf{s} \rangle$ can be approximated to ϵ precision with probability $1 - \delta$ using $O\left(\frac{1}{\epsilon^2} \log \frac{2}{\delta}\right)$ samples of $\mathbf{x}$. ∎

On the other hand, even though we have efficient classical algorithms for evaluating the correlation function (either exactly or approximately), it does not directly imply an effective solution to the verification problem. For a random instance of $(U_{\text{IQP}}, \mathbf{s})$, the value of the resulting correlation function could be too small from the experimental point of view; this makes it difficult to be distinguished from the uniform distribution. For example, we have the following proposition for random 2-local IQP circuits [YC20]:

Proposition 3.2. *For random 2-local IQP circuits of the form,* $U_{\mathrm{IQP}} = e^{i\frac{\pi}{8}\left(\sum_{i<j} w_{ij}X_i\otimes X_j+\sum_i v_iX_i\right)}$ *with* $w_{ij}, v_i \in \{0,1,\cdots,7\}$*, one can show that the probability of finding a polynomial-size correlation function* $\langle \mathcal{Z}_{\mathbf{s}}\rangle$ *is exponentially small, i.e.,*

$$\Pr_{U_{\mathrm{IQP}},\mathbf{s}}\left(\langle \mathcal{Z}_{\mathbf{s}}\rangle^2 \geq \frac{1}{f(n)}\right) \leq \frac{3f(n)}{2^n}\,, \tag{3.8}$$

where $f(n)$ *is a polynomial of* n*.*

This is essentially due to the anti-concentration properties of IQP circuits [BMS16], an important ingredient for proving the quantum computational supremacy of IQP sampling.

Proof. First, for random 2-local IQP circuits of the form

$$U_{\mathrm{IQP}} = e^{i\frac{\pi}{8}\left(\sum_{i<j} w_{ij}X_i\otimes X_j+\sum_i v_iX_i\right)}\,, \tag{3.9}$$

with $w_{ij}, v_i \in \{0,1,\cdots,7\}$, the output probability is anti-concentrated. Specifically, the *anti-concentration theorem* states that [BMS16]

$$\mathbb{E}_U[p(\mathbf{x})^2] \leq \frac{3}{2^{2n}} \tag{3.10}$$

for all $\mathbf{x}$, where $\mathbb{E}_U$ denotes a uniform average over all IQP circuits of the form of (3.9), that is over uniform choices of w_{ij} and v_i. Then we have,

$$\mathbb{E}_U\left[\sum_{\mathbf{x}} p(\mathbf{x})^2\right] \leq \frac{3}{2^n}\,. \tag{3.11}$$

By definition, the correlation function can be written as,

$$\langle \mathcal{Z}_{\mathbf{s}}\rangle = \sum_{\mathbf{x}} p(\mathbf{x})(-1)^{\mathbf{s}\cdot\mathbf{x}}\,, \tag{3.12}$$

which actually holds for a general quantum circuit. This means that $p(\mathbf{x})$ is the Fourier transform of $\langle \mathcal{Z}_{\mathbf{s}}\rangle$, and that

$$p(\mathbf{x}) = \frac{1}{2^n}\sum_{\mathbf{s}} \langle \mathcal{Z}_{\mathbf{s}}\rangle\,(-1)^{\mathbf{s}\cdot\mathbf{x}}\,. \tag{3.13}$$

Then we can apply the Parseval's identity [O'D14],

$$\sum_{\mathbf{x}} p(\mathbf{x})^2 = \frac{1}{2^n} \sum_{\mathbf{s}} \langle \mathcal{Z}_{\mathbf{s}} \rangle^2 , \tag{3.14}$$

which gives,

$$\begin{aligned}
\mathbb{E}_{U,\mathbf{s}} \left[\langle \mathcal{Z}_{\mathbf{s}} \rangle^2 \right] &= \frac{1}{2^n} \sum_{\mathbf{s}} \mathbb{E}_U \left[\langle \mathcal{Z}_{\mathbf{s}} \rangle^2 \right] && (3.15) \\
&= \mathbb{E} \left[\sum_{\mathbf{x}} p(\mathbf{x})^2 \right] && (3.16) \\
&\leq \frac{3}{2^n} . && (3.17)
\end{aligned}$$

The Markov's inequality gives the following bound,

$$\Pr_{U,\mathbf{s}}(\langle \mathcal{Z}_{\mathbf{s}} \rangle^2 \geq a) \leq \frac{\mathbb{E}_{U,\mathbf{s}} \left[\langle \mathcal{Z}_{\mathbf{s}} \rangle^2 \right]}{a} \leq \frac{3}{a 2^n} , \tag{3.18}$$

for $a > 0$. Setting $a = O(1/\mathrm{poly}(n))$, we have,

$$\Pr_{U,s} \left(\langle \mathcal{Z}_{\mathbf{s}} \rangle^2 \geq \frac{1}{\mathrm{poly}(n)} \right) \leq \frac{3\,\mathrm{poly}(n)}{2^n} . \tag{3.19}$$

This means that for random 2-local IQP circuits, the probability that the correlation functions are polynomially small is exponentially small. ■

To conclude, if Alice wants to use random IQP circuits for the verification, the correlation function will generally be exponentially small. In this way, Alice will need to require an exponential number of samples from Bob, and the verification process becomes inefficient. This poses a challenge, to balance the security given by randomized constructions with the scale of the correlation function that enables easy verification. Therefore, practically, Alice needs to carefully design the IQP circuit and the secret, such that the associated correlation function is sufficiently away from zero.

3.3 Shepherd-Bremner Construction

The first explicit construction recipe of $(\mathbf{H}, \mathbf{s})$ for the case $\theta = \pi/8$ is given by Shepherd and Bremner [SB09], which can be divided into two steps, (a) constructing the pair $(\mathbf{H}_\mathbf{s}, \mathbf{s})$ and (b) adding redundancy and obfuscation. In the Shepherd-Bremner construction, the non-orthogonal part $\mathbf{H}_\mathbf{s}$ with respect to the secret $\mathbf{s}$ is constructed from a specific error-correcting code, the quadratic-residue code (QRC) [MS77]. Under this construction, the correlation function is always $1/\sqrt{2}$, which is sufficiently away from zero as desired.

Specifically, let $\mathcal{H}^{\mathrm{QRC}}_{n,m,q} = \{(\mathbf{H}, \mathbf{s})\}$ be a family of pairs of an IQP matrix $\mathbf{H} \in \mathbb{F}_2^{m\times n}$ and a secret $\mathbf{s}$ so that $\mathbf{H}_\mathbf{s}$ generates a QRC of length q (up to row permutations) and $\mathbf{H}$ is of full column rank. What the Shepherd-Bremner construction achieves is to randomly sample instances from $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$, where $n = (q+3)/2$.

Note that in the Shepherd-Bremner construction [SB09], the measure of success is given by the probability bias $\mathcal{P}_{\mathbf{s}\perp} := \sum_{\mathbf{x}\cdot\mathbf{s}=0} p(\mathbf{x})$, the probability of receiving bit strings that are orthogonal to $\mathbf{s}$, where $p(\mathbf{x})$ is the output probability of the IQP circuit. This measure is equivalent to the correlation function, since $\mathcal{P}_{\mathbf{s}\perp} = \frac{1}{2}(\langle \mathcal{Z}_\mathbf{s} \rangle + 1)$ [She10, CCL+21].

3.3.1 Constructing the Anti-commuting Part

The quadratic residue code is a cyclic code. Its cyclic generator has 1 in the j-th position if j is a non-zero quadratic residue modulo q. The size parameter q of the quadratic-residue code is a prime number and $q+1$ is required to be a multiple of eight [SB09]. For $q = 7$, the cyclic generator reads $(1, 1, 0, 1, 0, 0, 0)^T$, because $j = 1, 2, 4$ are quadratic residues modulo 7. The basis for the codespace of QRC is obtained by rotating the cyclic generator, which is the last 4 columns of the following matrix,

$$\mathbf{H}_{\mathbf{s}}^{\mathrm{QRC}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} . \tag{3.20}$$

The first column is added so that the secret is easy to find, i.e., $\mathbf{s} = (1, 0, 0, 0, 0)^T$. But note that the all-ones column does not need to be added explicitly, because it is in the linear subspace of the quadratic-residue code. For the example in Eq. (3.20), it is not hard to verified that adding the second, the fourth and the fifth columns up gives the all-ones vector.

3.3.2 Redundancy and Obfuscation

After obtaining the initial $\mathbf{H}_{\mathbf{s}}^{\mathrm{QRC}}$, the verifier needs to hide the secret and make the IQP circuit look random, while leaving the value of the correlation function unchanged. In the Shepherd-Bremner construction, the verifier will first add redundant rows $\mathbf{R}_{\mathbf{s}}$, which are rows that are orthogonal to $\mathbf{s}$, to obtain the full IQP matrix

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{\mathbf{s}}^{\mathrm{QRC}} \\ \mathbf{R}_{\mathbf{s}} \end{pmatrix} . \tag{3.21}$$

Its corresponding Hamiltonian $R_{\mathbf{s}}$ commutes with $\mathcal{Z}_{\mathbf{s}}$ and hence will not affect the correlation function, as can be seen from Eq. (3.3). After initializing $\mathbf{H}$ and $\mathbf{s}$, the verifier needs to apply obfuscation to hide the secret. The obfuscation is achieved by randomly permuting rows in $\mathbf{H}$ and performing column operations to $\mathbf{H}$ and changing $\mathbf{s}$ accordingly.

Definition 3.3 (Obfuscation). Given an instance $(\mathbf{H}, \mathbf{s})$, the obfuscation is defined as the transformation

$$\mathbf{H} \leftarrow \mathbf{PHQ} \qquad\qquad \mathbf{s} \leftarrow \mathbf{Q}^{-1}\mathbf{s}\,, \tag{3.22}$$

where $\mathbf{P}$ is a random row-permutation matrix and $\mathbf{Q}$ is a random invertible matrix.

Note that in the obfuscation, the secret $\mathbf{s}$ is changed accordingly, to preserve the inner-product relation with the rows in $\mathbf{H}$.

Performing row permutations will not change the IQP unitary, because all gates commute with each other. Therefore, the value of the correlation function is not changed under row permutations. What is less obvious is that performing column operations will also leave the correlation function relative to the new secret unchanged. When all angles are the same, we have the following theorem:

Theorem 3.4 (Adapted from Theorem 1 of [SB09]). *Given an IQP matrix* $\mathbf{H}$ *and a vector* $\mathbf{s}$*, denote the non-orthogonal part of* $\mathbf{H}$ *with respect to* $\mathbf{s}$ *by* $\mathbf{H}_\mathbf{s}$*. Denote* $C_\mathbf{s}$ *as the linear subspace spanned by the columns of* $\mathbf{H}_\mathbf{s}$*. Transforming* $\mathbf{H}$ *into an IQP Hamiltonian* H*, the correlation function* $\langle \mathcal{Z}_\mathbf{s} \rangle := \langle 0^n | e^{-i\theta H} \mathcal{Z}_\mathbf{s} e^{i\theta H} | 0^n \rangle$ *can be expressed as,*

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{2^d} \sum_{\mathbf{c} \in C_\mathbf{s}} \cos[2\theta(q - 2|\mathbf{c}|)]\,, \tag{3.23}$$

where d *is the dimension of the linear subspace* $C_\mathbf{s}$ *and* q *is the number of rows in* $\mathbf{H}_\mathbf{s}$*.*

Theorem 3.4 states that the correlation function $\langle \mathcal{Z}_\mathbf{s} \rangle$ depends only on the linear subspace spanned by columns of $\mathbf{H}_\mathbf{s}$. Therefore, as long as we change the secret $\mathbf{s}$ accordingly, so that the new secret corresponds to the same linear subspace, the correlation function relative to the new secret will be the same. For the QRC construction, this means that the correlation function relative to the new secret will also be $1/\sqrt{2}$.

Proof. First, from Eq. (3.3), we have

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \langle \mathbf{y} | e^{i2\theta H_\mathbf{s}^{(z)}} | \mathbf{y} \rangle \tag{3.24}$$

$$= \frac{1}{2^n} \sum_{\mathbf{y}\in\{0,1\}^n} \exp\left(i2\theta \sum_{\mathbf{p}^T\in\mathrm{row}(\mathbf{H_s})} (-1)^{\mathbf{p}\cdot\mathbf{y}} \right) \tag{3.25}$$

$$= \frac{1}{2^n} \sum_{\mathbf{y}\in\{0,1\}^n} \cos\left(2\theta \sum_{\mathbf{p}^T\in\mathrm{row}(\mathbf{H_s})} (-1)^{\mathbf{p}\cdot\mathbf{y}} \right) . \tag{3.26}$$

Here, $H_s^{(z)}$ is obtained by replacing the Pauli-X operators in H_s with Pauli-Z operators, where H_s is the corresponding IQP Hamiltonian of the non-orthogonal rows $\mathbf{H_s}$. We used the fact that $\langle \mathcal{Z}_s \rangle$ is real in the last line.

Define $\mathbf{c_y} := \mathbf{H_s} \cdot \mathbf{y}$ to be an encoding of $\mathbf{y}$ under $\mathbf{H_s}$. Suppose that $\mathbf{H_s}$ contains q rows, so we can write

$$\mathbf{c_y} = \begin{pmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_q^T \end{pmatrix} \cdot \mathbf{y} = \begin{pmatrix} \mathbf{p}_1 \cdot \mathbf{y} \\ \vdots \\ \mathbf{p}_q \cdot \mathbf{y} \end{pmatrix} , \tag{3.27}$$

which means that each entry in $\mathbf{c_y}$ equals $\mathbf{p} \cdot \mathbf{y}$ for $\mathbf{p}^T \in \mathrm{row}(\mathbf{H_s})$. Then $\sum_{\mathbf{p}^T\in\mathrm{row}(\mathbf{H_s})} (-1)^{\mathbf{p}\cdot\mathbf{y}}$ equals the number of zeros in $\mathbf{c_y}$ minus the number of ones (i.e., Hamming weight $|\mathbf{c_y}|$), which gives,

$$\sum_{\mathbf{p}^T\in\mathrm{row}(\mathbf{H_s})} (-1)^{\mathbf{p}\cdot\mathbf{y}} = q - 2|\mathbf{c_y}| . \tag{3.28}$$

Plugging it into Eq. (3.26), we arrive at,

$$\langle \mathcal{Z}_s \rangle = \frac{1}{2^n} \sum_{\mathbf{y}\in\{0,1\}^n} \cos[2\theta(q - 2|\mathbf{c_y}|)] . \tag{3.29}$$

Now, in the column picture, we can write $\mathbf{H_s} = (\mathbf{c}_1, \cdots, \mathbf{c}_n)$, where $\mathbf{c}_i \in \mathrm{col}(\mathbf{H_s})$ is a column vector of length q. Thus,

$$\mathbf{c_y} = \mathbf{H_s} \cdot \mathbf{y} = y_1\mathbf{c}_1 + \cdots + y_n\mathbf{c}_n . \tag{3.30}$$

Suppose the dimension of C_s is d, and without loss of generality, assume that $\{\mathbf{c}_1, \cdots, \mathbf{c}_d\}$ forms a basis. Then,

$$\mathbf{c} = y_1\mathbf{c}_1 + \cdots + y_d\mathbf{c}_d \tag{3.31}$$

for any $\mathbf{c} \in C_\mathbf{s}$. So the expression of $\langle \mathcal{Z}_\mathbf{s} \rangle$ becomes,

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{2^{n-d}} \sum_{y_{d+1},\cdots,y_n} \left(\frac{1}{2^d} \sum_{\mathbf{c}\in C_M} \cos[2\theta(q - 2|\mathbf{c}|)] \right) . \tag{3.32}$$

The first summation will give a factor of 2^{n-d}, which cancels with $\frac{1}{2^{n-d}}$. So finally, it ends up giving,

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{2^d} \sum_{\mathbf{c}\in C_M} \cos[2\theta(q - 2|\mathbf{c}|)] . \tag{3.33}$$

Every term in the summation depends on the element $\mathbf{c}$, and therefore $\langle \mathcal{Z}_\mathbf{s} \rangle$ depends only on the linear subspace $C_\mathbf{s}$. ■

3.3.3 A Loophole in the Shepherd-Bremner Construction

There are two potential ways to hack the protocol, i.e., to pass the test with a classical computer. One way is to simulate IQP circuits with classical computers. This is generally implausible, because general IQP circuits are hard to sample from classically efficiently, assuming some reasonable complexity-theoretic conjectures [BJS11, BMS16]. Therefore, one can expect that this kind of attack will fail when the system size is large enough.

Another way is to find the secret from the IQP matrix $\mathbf{H}$, which can then be used to generate correctly correlated samples that can pass the test. This is less clear whether this is possible or not. [SB09] conjectured that it is NP-hard to find the hidden $\mathbf{H}_\mathbf{s}^{\mathrm{QRC}}$ constructed from quadratic-residue code out of $\mathbf{H}$. However, while this seems plausible, the parameter regime of the Shepherd-Bremner construction may not give hard instances.

Recall that the Shepherd-Bremner construction recipe can only randomly sample instances from $\mathcal{H}_{n,m,q}^{\mathrm{QRC}}$ with $n = (q+3)/2$. This actually leads to a loophole, with which Kahanamoku-Meyer designed a classical attack targeting the QRC-based construction of IQP circuits [KM19]. In Kahanamoku-Meyer's attack, the classical prover starts by constructing a matrix $\mathbf{M}$ from rows of the IQP matrix $\mathbf{H}$. Then, the prover will iterate over vectors in the kernel of $\mathbf{M}$, and perform a property check on each vector, which checks whether the non-orthogonal part of $\mathbf{H}$ associated to that vector generates a QRC. With probability $1/2$, the real secret lies in the kernel of $\mathbf{M}$, and the numerical result shows that the kernel space will generally not be large for the Shepherd-Bremner construction [KM19]. Therefore, the attacker can find the secret efficiently. However, choosing a different encoding method other than QRC can easily invalidate this attack, since no vector will pass the property check.

We conclude this section by remarking that [She10] subsequently studied IQP circuits from the perspective of binary matroids and Tutte polynomials. Specifically, the amplitude of the IQP circuit $\langle 0^n | e^{i\theta H} | 0^n \rangle$ is expressed in terms of the normalized Tutte polynomial, and its computational complexity is studied in various cases. When $\theta = \pi/4$, the related Tutte polynomial can be efficiently evaluated using Vertigan's algorithm [Ver98], which is similar to the Gottesman-Knill algorithm [Got99]. But when $\theta = \pi/8$ (and any other values except for the multiple of $\pi/4$), computing the amplitude is $\#P$-hard in the worst case. Moreover, [She10] also derived similar relation to Eq. (3.3), in the language of the normalized Tutte polynomial. Therefore, it was suggested that the correlation function is efficiently classical computable when $\theta = \pi/8$, and could be used to perform hypothesis test, although no new construction was proposed in [She10].

3.4 A Heuristic Generalized Construction

In [YC20], we provide a heuristic construction of IQP circuits for verification, which goes beyond the Shepherd-Bremner construction. Our heuristic construction does not rely on quadratic-residue code, and hence is intrinsically immune to Kahanamoku-Meyer's attack, as we explained before. Moreover, the angles of the each Hamiltonian term can be different, and multiple secrets could be simultaneously encoded in the IQP circuits.

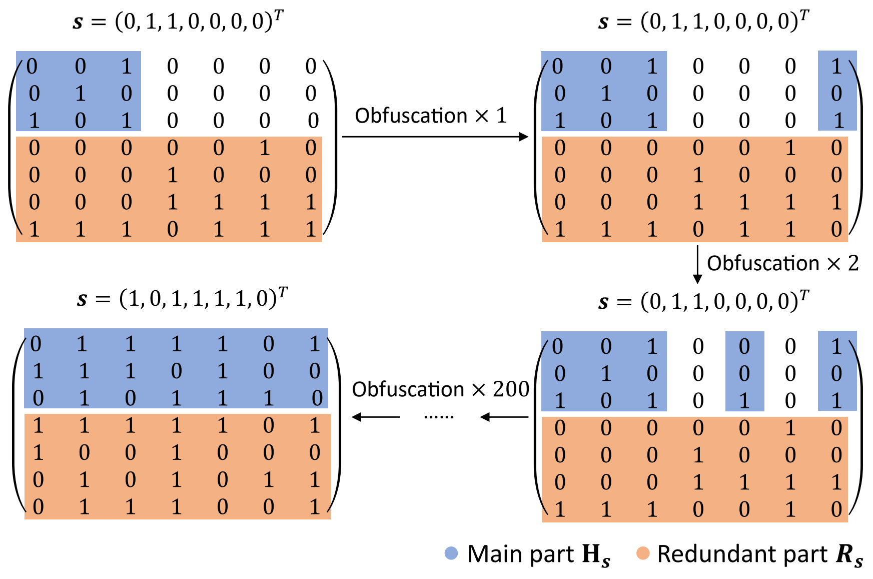


Figure 3.2: The obfuscation process in the matrix representation. Here, $\mathbf{H}_\mathbf{s}$ (the blue block) initially acts only on first few qubits. After the first obfuscation process, the third column of the matrix is added to the last column, and the last entry of $\mathbf{s}$ is added to the third one, correspondingly. Similarly, after the second obfuscation process, the first column of the matrix is added to the fifth one, and the fifth entry of $\mathbf{s}$ is added to the first one. The resulting matrix after 200 times of obfuscations is shown in the lower left corner, and the acting range of the main part extends to the whole circuit.

The construction exploits the feature that any IQP correlation function can be estimated classically. Therefore, in principle, even if Alice, the verifier uses a random IQP circuit, she can still know the ideal value of the correlation function, up to a polynomial additive error. But the anti-concentration property of IQP circuit implies that for random IQP circuits, the correlation function will be superpolynomially small with high probability (see Eq. (3.8)). One way to circumvent this is to start from a small random IQP circuit, and then extend the system size with the redundancy and obfuscation technique in the Shepherd-Bremner construction. The construction recipe is as follows.

(1) Randomly sample a desired number of secret strings $\mathbf{s}_1, \cdots, \mathbf{s}_l \in \{0,1\}^{n'}$ with a small length n'.

(2) Search for an n'-qubit IQP Hamiltonian H' and a set of angles $\boldsymbol{\theta}$ for each term, so that $|\langle \mathcal{Z}_{\mathbf{s}_i} \rangle|$ is larger than a threshold τ for all i.

(3) Let $\mathbf{H}'$ be the binary representation of H'. Append all-zeros columns to $\mathbf{H}'$ to extend the number of qubits to n. Extend the length of secret strings accordingly (not necessarily appending zeros to the secrets).

(4) Adding random redundant rows to $\mathbf{H}'$ that are orthogonal to all secrets.

(5) Obfuscate the whole IQP matrix and all secrets together by performing column operations and row permutations (as in Definition 3.3).

Fig. 3.2 gives a pictorial example, where there is only one secret, and the initial H' anti-commutes with the $\mathcal{Z}_{\mathbf{s}}$ (i.e., it gives the the non-orthogonal part $\mathbf{H}_{\mathbf{s}}$ part relative to that secret). Referring to Eq. (3.8), the initial system size should be $n' = O(\log n)$, where n is the final system size. Otherwise, it may take a long time to search for a suitable H' with sizable correlation functions. The threshold τ should be $O(1/\mathrm{poly}(n))$. We remark that once the verifier searches for a good tuple $(\mathbf{H}', \boldsymbol{\theta}, \mathbf{s}_1, \cdots, \mathbf{s}_l)$ such that all $|\langle \mathcal{Z}_{\mathbf{s}_i} \rangle|$ are larger than the threshold τ, she can save it for future use. In fact, she can first search for a large number of such tuples to construct a database, and then randomly choose one for the verification.

After adding zero columns to $\mathbf{H}'$ and appending redundant rows, the verifier uses the obfuscation technique as previously introduced in [SB09], which is the transformation defined in Definition 3.3. In particular, the column operations not only hide the secrets, but also extend the range of qubits that the IQP Hamiltonian H acts on. However, we need to make sure that the correlation functions relative to the new secrets are not changed. This is achieved by extending Theorem 3.4 to the case where the angles can be different for different terms in the Hamiltonian [YC20].

Theorem 3.5. *Given an IQP Hamiltonian H and a vector $\mathbf{s}$, denote the anti-commuting part of H with respect to $\mathcal{Z}_{\mathbf{s}}$ as $H_{\mathbf{s}}$ and its binary matrix representation as $\mathbf{H}_{\mathbf{s}}$. Then, the* *value of the correlation function $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ depends only on $C_{\mathbf{s}}$, the linear subspace spanned by the columns of* $\mathbf{H}_{\mathbf{s}}$.

Proof. First, Eq. (3.7) gives,

$$\langle \mathcal{Z}_{\mathbf{s}} \rangle = \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \langle \mathbf{y} | \prod_{\mathbf{p}^T \in \text{row}(\mathbf{H}_{\mathbf{s}})} e^{i 2\theta_{\mathbf{p}} \mathcal{Z}_{\mathbf{p}}} | \mathbf{y} \rangle \ , \tag{3.34}$$

where $\mathcal{Z}_{\mathbf{p}} := Z^{p_1} \otimes \cdots \otimes Z^{p_n}$. Then,

$$\langle \mathcal{Z}_{\mathbf{s}} \rangle = \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \prod_{\mathbf{p}^T \in \text{row}(\mathbf{H}_{\mathbf{s}})} \exp\big(i 2\theta_{\mathbf{p}} (-1)^{\mathbf{p} \cdot \mathbf{y}}\big) \tag{3.35}$$

$$= \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \exp\left(i \sum_{\mathbf{p} \in \text{row}(\mathbf{H}_{\mathbf{s}})} 2\theta_{\mathbf{p}} (-1)^{\mathbf{p} \cdot \mathbf{y}} \right) . \tag{3.36}$$

Define $\mathbf{Q}$ as the matrix for column operations, which is an invertible matrix. Then, after the column operations, we have $\mathbf{H} \to \mathbf{HQ}$, $\mathbf{H}_{\mathbf{s}} \to \mathbf{H}_{\mathbf{s}}\mathbf{Q}$ and $\mathbf{s} \to \mathbf{Q}^{-1}\mathbf{s}$, so that the inner-product relation between rows in $\mathbf{H}$ and $\mathbf{s}$ is preserved. Moreover, we have $\mathbf{p} \cdot \mathbf{y} = (\mathbf{Q}^{-T}\mathbf{p}) \cdot (\mathbf{Q}\mathbf{y})$, where $\mathbf{Q}^{-T}$ denotes the transpose of $\mathbf{Q}^{-1}$. Then, denoting the new secret string as $\mathbf{s}' := \mathbf{Q}^{-1}\mathbf{s}$, the associated correlation function is given by,

$$\langle \mathcal{Z}_{\mathbf{s}'} \rangle = \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \exp\left(i \sum_{\mathbf{p}'^T \in \text{row}(\mathbf{H}_{\mathbf{s}}\mathbf{Q})} 2\theta'_{\mathbf{p}'} (-1)^{(\mathbf{Q}^{-T}\mathbf{p}') \cdot (\mathbf{Q}\mathbf{y})} \right) \tag{3.37}$$

$$= \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \exp\left(i \sum_{\mathbf{p}^T \in \text{row}(\mathbf{H}_{\mathbf{s}})} 2\theta_{\mathbf{p}} (-1)^{\mathbf{p} \cdot (\mathbf{Q}\mathbf{y})} \right) \tag{3.38}$$

$$= \frac{1}{2^n} \sum_{\mathbf{y} \in \{0,1\}^n} \exp\left(i \sum_{\mathbf{p}^T \in \text{row}(\mathbf{H}_{\mathbf{s}})} 2\theta_{\mathbf{p}} (-1)^{\mathbf{p} \cdot \mathbf{y}} \right) \tag{3.39}$$

$$= \langle \mathcal{Z}_{\mathbf{s}} \rangle \ , \tag{3.40}$$

where in the first line, we let $\theta'_{\mathbf{p}'} = \theta_{\mathbf{p}}$ if $\mathbf{p}' = \mathbf{Q}^T\mathbf{p}$, and in the third line, we perform a relabelling $\mathbf{y} \to \mathbf{Q}\mathbf{y}$. This proves that the value of correlation function depends only on the column space of $\mathbf{H}_{\mathbf{s}}$. ■

Although this construction is immune to Kahanamoku-Meyer's attack, it is heuristic in nature, which makes it difficult to analyze its efficiency and security.

3.5 Column Redundancy

Apart from choosing a different encoding method, choosing a different parameter regime in the QRC-based construction also offers a viable solution to fixing the loophole. We give the procedure here, and defer the analysis to when we discuss the classical security.

Recall that in the Shepherd-Bremner construction, the obfuscation of the IQP circuit and the hiding of the secret are achieved by adding redundant rows followed by performing random row permutations and column operations. Let $\mathcal{H}^{\mathrm{QRC}}_{n,m,q} := \{(\mathbf{H}, \mathbf{s})\}$ be a family of pairs of an IQP matrix $\mathbf{H} \in \mathbb{F}_2^{m\times n}$ and a secret $\mathbf{s}$ so that $\mathbf{H}_\mathbf{s}$ generates a QRC of length q (up to row permutations) and $\mathbf{H}$ is of full column rank. What their construction recipe does is to randomly sample instances from $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$, where $n = (q+3)/2$ and $m \geq q$, leaving a loophole for the recent classical attack [KM19]. To see why the parameter regime is as above, we first note that the length of QRC is q, implying that the number of rows in $\mathbf{H}_\mathbf{s}$ is q and hence $m \geq q$. Moreover, the dimension of a length-q QRC is $(q+1)/2$, which implies that the rank of $\mathbf{H}_\mathbf{s}$ is $(q+1)/2$. But an all-ones column was added in the construction (see Eq. (3.20)), which is a codeword of QRC, leading to $n = (q+3)/2$.

Here, we show that the Shepherd-Bremner construction can be improved by adding column redundancy to the IQP matrix, which can achieve random sampling from families $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ with any $n \geq (q+1)/2$. As we will see in Section 5.3, adding column redundancy to the Shepherd-Bremner construction can make the recent classical attack fail and hence fix the loophole. Furthermore, adding column redundancy is not specific to the QRC-based construction, but applicable to any construction.

Essentially, adding column redundancy is to replace a full rank generator matrix of a code with a "redundant" generator matrix. The procedure of adding column redundancy is as follows:

(1) Given a full-rank $\mathbf{H}_\mathbf{s}$, (e.g., the last 4 columns in Eq. (3.20)) and the secret, we first append all-zeros columns to $\mathbf{H}_\mathbf{s}$ and extend $\mathbf{s}$ accordingly,

$$\mathbf{H}_\mathbf{s} \leftarrow (\mathbf{H}_\mathbf{s}, \mathbf{0}) \qquad \mathbf{s} \leftarrow \begin{pmatrix} \mathbf{s} \\ \mathbf{s}' \end{pmatrix}. \tag{3.41}$$

(2) Apply random column operations $\mathbf{Q}$ to obtain $\mathbf{H}_\mathbf{s} \leftarrow \mathbf{H}_\mathbf{s}\mathbf{Q}$ and $\mathbf{s} \leftarrow \mathbf{Q}^{-1}\mathbf{s}$.

Here, in the first step $\mathbf{s}'$ is an arbitrary vector whose length is the same as the number of all-zeros columns appended to $\mathbf{H}_\mathbf{s}$. Since the correlation function only depends on the linear code generated by $\mathbf{H}_\mathbf{s}$ [SB09, YC20] and adding column redundancy does not change the linear code, the correlation function with respect to the new secret is unchanged after the above two steps. We would like to remark that although there are 2^{n_2} choices for $\mathbf{s}'$ of length n_2, once we fix a choice, the only constraint to the redundant rows is to be orthogonal to the specific new secret $\mathbf{s}$. Moreover, since the final IQP matrix $\mathbf{H}$ is of full column rank, only the real secret $\mathbf{s}$ will correspond to the code generated by $\mathbf{H}_\mathbf{s}$.

Back to the case of QRC-based construction, if one chooses a redundant generator matrix of QRC by adding column redundancy, then n can be any integer larger than $(q+1)/2$, the dimension of QRC. This hides the dimension information of the hidden QRC. Combined with other obfuscation techniques in the Shepherd-Bremner construction, this achieves random sampling from $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ with any possible parameters.

Note that such a technique was used in [YC20] to scramble a small random IQP circuit into a large one, to maintain the value of the correlation function, although its connection to the classical security was not explored. In Section 5.3, we show that adding column redundancy with the procedure presented above can actually fix the recent loophole in the Shepherd-Bremner construction. Moreover, a multi-secret version was explored in [Sno20], which was shown to be more vulnerable to the classical attack instead.

Chapter 4

Stabilizer Scheme

In this chapter, we propose a new IQP-based protocol, which we refer to as the *stabilizer scheme*. Our construction allows the verifier to efficiently generate an IQP circuit, $U_{\mathrm{IQP}} = e^{i\pi H/8}$, and a secret, $\mathbf{s}$, so that the correlation function relative to the secret has a magnitude equal to $2^{-g/2}$, where g is a tunable integer. The stabilizer scheme is based on the interplay between IQP circuits, stabilizer formalism and coding theory, and it significantly strengthens previous constructions based on quadratic-residue codes [SB09] or random small IQP circuits [YC20]. Our characterization on IQP circuits, discussed in Section 4.1, builds upon and integrates previous results [She10, Man21], which tackle this problem from the perspective of binary matroids and Tutte polynomials. The construction algorithm is based on sampling generator matrices of random codes satisfying certain conditions, which is presented in Section 4.2. In addition, in Section 4.3, we give another construction algorithm based on matrix factorization, which arises in an early exploration of our work [BCJ23]. Altogether, this enriches the scope of IQP-based schemes while maintaining their simplicity and verifiability, and also provides a viable avenue for constructing protocols for verifiable quantum advantage.

4.1 Stabilizer Characterization of IQP Circuits

In this section, we establish the connection between IQP circuits, stabilizer formalism and coding theory, which turns out to be useful in constructing the IQP circuits for the verification protocol. For $\theta = \pi/8$, we show that the stabilizer tableau of the Clifford operation $e^{i2\theta H_{\mathbf{s}}}$ has a nice structure that allows us to determine the value of $\langle \mathcal{Z}_{\mathbf{s}} \rangle = \langle 0^n | e^{i2\theta H_{\mathbf{s}}} | 0^n \rangle$ efficiently. As an application, we analyze the Shepherd-Bremner construction with this framework.

4.1.1 IQP Stabilizer Tableau

We first give the form of the stabilizer tableau of $e^{i\pi H/4}\,|0^n\rangle$.

Theorem 4.1. *Given a binary matrix* $\mathbf{H} = (\mathbf{c}_1, \cdots, \mathbf{c}_n)$ *and transforming it into an IQP Hamiltonian* H*, the stabilizer tableau of the state* $|\psi\rangle = e^{i\pi H/4}\,|0^n\rangle$ *can be expressed as,*

$$\left(\begin{array}{ccc|ccc|c} \mathbf{c}_1 \cdot \mathbf{c}_1 & \cdots & \mathbf{c}_1 \cdot \mathbf{c}_n & 1 & \cdots & 0 & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{c}_n \cdot \mathbf{c}_1 & \cdots & \mathbf{c}_n \cdot \mathbf{c}_n & 0 & \cdots & 1 & r_n \end{array}\right). \tag{4.1}$$

Here, if one uses $00, 01, 10, 11$ *to represent* $|\mathbf{c}_j| = 0, 1, 2, 3 \pmod 4$*, then* r_j *is equal to the first bit.*

We call Eq. (4.1) the IQP (stabilizer) tableau and it is of the form $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$. We apply the above theorem to $\mathbf{H}_{\mathbf{s}}$, in which case the X part is $\mathbf{G}_{\mathbf{s}} = \mathbf{H}_{\mathbf{s}}^T \mathbf{H}_{\mathbf{s}}$.

This theorem can be proved by starting from the standard tableau of $|0^n\rangle$, and keeping track of the stabilizer tableau after applying each terms of $e^{i\pi H/4}$ (i.e., each row of $\mathbf{H}$). First, we start with the standard tableau of $|0^n\rangle$, which is

$$\begin{pmatrix} 0 & \dots & 0 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 1 & 0 \end{pmatrix}, \tag{4.2}$$

corresponding to the stabilizer generators $\{Z_1, \cdots, Z_n\}$; here, the X part of the tableau is an all-zeros matrix. Then, we apply the local terms in $e^{i\pi H/4}$ one by one, and keep track of the change of the stabilizer tableau. We have the following lemma which gives the form of Z_j conjugated by $e^{i\pi H/4}$.

Lemma 4.2 (Evolution of Z_j). *Let $\mathbf{H} = (\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n)$ be a binary matrix. Then, translating $\mathbf{H}$ into the IQP Hamiltonian H and after the conjugation of $e^{i\pi H/4}$, we have,*

$$e^{i\pi H/4} Z_j e^{-i\pi H/4} = i^{|\mathbf{c}_j|} \prod_{k=1}^{n} X_k^{\mathbf{c}_j \cdot \mathbf{c}_k} Z_j\ , \tag{4.3}$$

where $|\mathbf{c}_j|$ is the Hamming weight of $\mathbf{c}_j$.

For example, let

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} . \tag{4.4}$$

Then, after the conjugation of $e^{i\pi H/4}$, we have

$$Z_1 \to (-1)(X_1X_2)(X_1X_4)Z_1 = -Z_1X_2X_4\ . \tag{4.5}$$

Proof. First, note that $e^{i\pi H/4} = \prod_{\mathbf{p}^T \in \mathrm{row}(\mathbf{H})} e^{i\pi \mathcal{X}_\mathbf{p}/4}$, where $\mathcal{X}_\mathbf{p} := X^{p_1} \otimes \cdots \otimes X^{p_n}$. For each row $\mathbf{p}^T$, if $p_j = 1$, then

$$e^{i\pi \mathcal{X}_\mathbf{p}/4} Z_j e^{-i\pi \mathcal{X}_\mathbf{p}/4} = e^{i\pi \mathcal{X}_\mathbf{p}/2} Z_j = i\mathcal{X}_\mathbf{p} Z_j\ ; \tag{4.6}$$

and if $p_j = 0$, Z_j will remain unchanged. We suppose $p_j = 1$ for later illustration. Then, we apply the operator corresponding to another row $\mathbf{p}'^T$, which gives,

$$i e^{i\pi \mathcal{X}_{\mathbf{p}'}/4} \mathcal{X}_\mathbf{p} Z_j e^{-i\pi \mathcal{X}_{\mathbf{p}'}/4} = i\mathcal{X}_\mathbf{p} e^{i\pi \mathcal{X}_{\mathbf{p}'}/4} Z_j e^{-i\pi \mathcal{X}_{\mathbf{p}'}/4}\ . \tag{4.7}$$

If $p'_j = 1$, we have that the post-evolution stabilizer is given by $i^2 \mathcal{X}_\mathbf{p} \mathcal{X}_{\mathbf{p}'} Z_j$. In general, let $\mathbf{H}_j$ be the submatrix of $\mathbf{H}$ that consists of all rows whose j-th entry is 1. Then, after the conjugation of $e^{i\pi H/4}$, we have

$$e^{i\pi H/4} Z_j e^{-i\pi H/4} = i^{|\mathbf{c}_j|} \prod_{\mathbf{p}^T \in \mathrm{row}(\mathbf{H}_j)} \mathcal{X}_\mathbf{p} Z_j\ . \tag{4.8}$$

For the Pauli X's in the above, whether there is the X_k component depends on the number of 1's in both the j-th and k-th column of $\mathbf{H}$. Indeed, the exponent of X_k is equal to $\mathbf{c}_j \cdot \mathbf{c}_k$. This completes the proof. ■

Next, we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. Since Z_j is the j-th stabilizer generator fo $|0^n\rangle$, Lemma 4.2 actually gives the j-th stabilizer generator of $|\psi\rangle = e^{i\pi H/4}|0^n\rangle$. We can also write it in the following form,

$$(-1)^{r_j} \prod_{k=1}^{n} i^{\mathbf{c}_j\cdot\mathbf{c}_j} X_k^{\mathbf{c}_j\cdot\mathbf{c}_k} Z_j \ , \tag{4.9}$$

where $2r_j + \mathbf{c}_j \cdot \mathbf{c}_j = |\mathbf{c}_j| \pmod 4$ (note that the inner product is taken over $\mathbb{F}_2$). Therefore, if one uses $00, 01, 10, 11$ to represent $|\mathbf{c}_j| = 0, 1, 2, 3 \pmod 4$, then r_j is equal to the first bit. Finally, from this form of stabilizer generators, we can write down the stabilizer tableau of $|\psi\rangle$ as

$$\begin{pmatrix} \mathbf{c}_1\cdot\mathbf{c}_1 & \dots & \mathbf{c}_1\cdot\mathbf{c}_n & 1 & \dots & 0 & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{c}_n\cdot\mathbf{c}_1 & \dots & \mathbf{c}_n\cdot\mathbf{c}_n & 0 & \dots & 1 & r_n \end{pmatrix} . \tag{4.10}$$

■

4.1.2 Correlation Functions

Next, we relate the correlation function to the code generated by $\mathbf{H}_\mathbf{s}$, denoted as $C_\mathbf{s}$. Note that $\mathbf{H}_\mathbf{s}\mathbf{s} = \mathbf{1}$ means that the all-ones vector is a codeword of $C_\mathbf{s}$. From Proposition 2.2, this means that the dual code $C_\mathbf{s}^\perp$ is an even code and the intersection $\mathcal{D}_\mathbf{s} := C_\mathbf{s} \bigcap C_\mathbf{s}^\perp$ is a weakly self-dual even code. Then, $\mathcal{D}_\mathbf{s}$ will be either a doubly-even code or an unbiased even code, according to Lemma 2.4.

Theorem 4.3. *Given an IQP matrix* $\mathbf{H}_\mathbf{s}$ *and a vector* $\mathbf{s}$*, so that* $\mathbf{H}_\mathbf{s}\ \mathbf{s} = \mathbf{1}$*. Denote the code generated by columns of* $\mathbf{H}_\mathbf{s}$ *by* $C_\mathbf{s}$ *and its dual code by* $C_\mathbf{s}^\perp$*. Let* $\mathcal{D}_\mathbf{s} := C_\mathbf{s} \bigcap C_\mathbf{s}^\perp$*. Then,* *transforming* $\mathbf{H}_\mathbf{s}$ *into an IQP Hamiltonian* $H_\mathbf{s}$*, the magnitude of the correlation function* $\langle \mathcal{Z}_\mathbf{s} \rangle = \langle 0^n | e^{i\pi H_\mathbf{s}/4} | 0^n \rangle$ *is* $2^{-g/2}$ *if* $\mathcal{D}_\mathbf{s}$ *is a doubly-even code and* 0 *if* $\mathcal{D}_\mathbf{s}$ *is an unbiased even code. Here,* $g := \dim(\mathcal{C}_\mathbf{s}) - \dim(\mathcal{D}_\mathbf{s})$ *is also the rank of the Gram matrix* $\mathbf{G}_\mathbf{s} = \mathbf{H}_\mathbf{s}^T \mathbf{H}_\mathbf{s}$.

Before presenting the proof, we first give some remarks. First, from a group-theoretic perspective, the rank of the Gram matrix g is also the minimum number of different generators over all possible choices of the stabilizer groups between $|0^n\rangle$ and $e^{i\pi H_\mathbf{s}/4}|0^n\rangle$ (Proposition 2.1). Furthermore, we note that Theorem 4.3 integrates several results in [She10] concisely, with a particular focus on coding theory, so that it aligns better with our objective of constructing IQP circuits for the verification protocol. [She10] studies the IQP circuits with $\theta = \pi/4$ with a reworking of Vertigan's algorithm for evaluating the magnitude of the Tutte polynomial of a binary matroid at the point $(-i, i)$ [Ver98]. There, the amplitude $\langle \mathbf{x} | e^{i\theta H} | 0^n \rangle$ is considered for $\theta = \pi/4$ and any IQP Hamiltonian H, where the all-ones vector may not be a codeword of the code generated by the binary matrix $\mathbf{H}$. Such an amplitude has been further studied in [Man21], which gives the expression of the phase of the amplitude by applying results of [Pen14]. In the language of binary matroids, the dual intersection $\mathcal{D}_\mathbf{s}$ is the bicycle space of the matroid represented by $\mathbf{H}_\mathbf{s}$ and its dimension $\dim(\mathcal{D}_\mathbf{s})$ is also known as the bicycle dimension [Ver98, Man21]. Finally, we note that although computing the magnitude suffices for our later construction, the sign of the correlation function can also be computed efficiently, as shown in [Man21]. In addition, when $g = O(\log n)$, the correlation function has an inverse polynomial scaling. In this case, one can use the random sampling algorithm in [YC20] to determine the sign efficiently.

Proof of Theorem 4.3. First, $\mathbf{H}_\mathbf{s}\mathbf{s} = \mathbf{1}$ implies that $\mathcal{C}_\mathbf{s}^\perp$ is an even code and so is $\mathcal{D}_\mathbf{s}$. When $\theta = \pi/8$, we have

$$\begin{aligned} \langle \mathcal{Z}_\mathbf{s} \rangle &= \langle 0^n | e^{i2\theta H_\mathbf{s}} | 0^n \rangle && (4.11) \\ &= \langle 0^n | \prod_{\mathbf{p}^T \in \mathrm{row}(\mathbf{H}_\mathbf{s})} e^{i2\theta X_\mathbf{p}} | 0^n \rangle && (4.12) \end{aligned}$$

$$= \langle 0^n | \prod_{\mathbf{p}^T \in \text{row}(\mathbf{H}_\mathbf{s})} \frac{1}{\sqrt{2}} (I + i\mathcal{X}_\mathbf{p}) | 0^n \rangle \tag{4.13}$$

$$= \frac{1}{\sqrt{2^m}} \sum_{\mathbf{a} \in \{0,1\}^m} i^{|\mathbf{a}|} \langle 0^n | \mathcal{X}_{\mathbf{a}^T \mathbf{H}_\mathbf{s}} | 0^n \rangle \tag{4.14}$$

$$= \frac{1}{\sqrt{2^m}} \sum_{\mathbf{a}: \mathbf{a}^T \mathbf{H}_\mathbf{s} = \mathbf{0}} i^{|\mathbf{a}|} \tag{4.15}$$

$$= \frac{1}{\sqrt{2^m}} \sum_{\mathbf{a} \in C_\mathbf{s}^\perp} i^{|\mathbf{a}|}, \tag{4.16}$$

where m is the number of rows in $\mathbf{H}_\mathbf{s}$. Since $C_\mathbf{s}^\perp$ is an even code, we can write,

$$\langle \mathcal{Z}_\mathbf{s} \rangle = \frac{1}{\sqrt{2^m}} \left(\sum_{\substack{\mathbf{a} \in C_\mathbf{s}^\perp \\ |\mathbf{a}| = 0 \bmod 4}} 1 - \sum_{\substack{\mathbf{a} \in C_\mathbf{s}^\perp \\ |\mathbf{a}| = 2 \bmod 4}} 1 \right). \tag{4.17}$$

Let $d = \dim(\mathcal{D}_\mathbf{s})$, $r = \dim(C_\mathbf{s})$ and $g = r - d$.

One can always find an invertible matrix $\mathbf{Q}$, such that in $\mathbf{H}_\mathbf{s}\mathbf{Q}$, the first g columns are in $C_\mathbf{s} \backslash \mathcal{D}_\mathbf{s}$, the g-th to the r-th columns form a basis of $\mathcal{D}_\mathbf{s}$ and the remaining columns are all-zeros. This transformation will not change the value of the correlation function according to Eq. (4.17), because it preserves the code $C_\mathbf{s}$ and hence the dual code $C_\mathbf{s}^\perp$. Under this transformation, the stabilizer tableau related to $\mathbf{H}_\mathbf{s}\mathbf{Q}$ is given by $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$. Here, only the top-left $g \times g$ submatrix of $\mathbf{G}'$ can be nonzero, and all other entries are zero. According to Proposition 2.7, the rank of $\mathbf{G}$ is also g, which means that the $g \times g$ submatrix is full rank.

As for the phase column $\mathbf{r}'$, if the basis of $\mathcal{D}_\mathbf{s}$ have weight 0 modulo 4, then only the first g entries of $\mathbf{r}'$ can be nonzero, and all other entries are zero, according to Theorem 4.1. In this case, $\mathcal{D}_\mathbf{s}$ is a doubly-even code. For this set of generators represented by the transformed tableau, the number of non-Z generators is g, corresponding to the first g rows of $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$. This is the minimum number over all possible choices, since the top-left submatrix of $\mathbf{G}'$ is already full rank. Thus, the correlation function is nonzero and has a magnitude $2^{-g/2}$, according to Proposition 2.1.

On the other hand, if in $\mathbf{H}_\mathbf{s}\mathbf{Q}$, some basis of $\mathcal{D}_\mathbf{s}$ have weight 2 modulo 4, then the corresponding entries in $\mathbf{r}'$ are 1, which gives Z-products with minus sign in the stabilizer group of $|\psi_s\rangle := e^{i\pi H_s/4}|0^n\rangle$. This means that $|\psi_s\rangle$ has zero overlap with $|0^n\rangle$ and hence the correlation function is zero. In this case, it can be shown that $\mathcal{D}_s$ is an unbiased even code using Lemma 2.4. ■

4.1.3 Applied to Shepherd-Bremner Construction

To show the usefulness of the stabilizer characterization, we apply these two theorems to analyze the Shepherd-Bremner construction. Combined with the properties of QRC, we have the following corollary.

Corollary 4.4. *Let q be a prime such that 8 divides $q+1$. Let $\mathbf{H}_s^{\mathrm{QRC}}$ be a matrix whose first column is $\mathbf{1}$ (of length q), and whose remaining columns are the basis of the quadratic-residue code of length q, formed by the cyclic generator (i.e., in the form of Eq. (3.20)). Then, translating $\mathbf{H}_s^{\mathrm{QRC}}$ into an IQP Hamiltonian H_s, the stabilizer tableau of $|\psi_s\rangle = e^{i\pi H_s/4}|0^n\rangle$ can be expressed as the following form,*

$$\left(\begin{array}{ccc|ccc|c} 1 & \cdots & 1 & 1 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \cdots & 1 & 1 & \cdots & 1 & 1 \end{array}\right) . \tag{4.18}$$

As a result, the corresponding stabilizer group is given by,

$$\langle -Y_1X_2\cdots X_n, -X_1Y_2X_3\cdots X_n, \cdots, -X_1X_2\cdots X_{n-1}Y_n\rangle , \tag{4.19}$$

where $n=(q+3)/2$. Moreover, the correlation function $\langle \mathcal{Z}_s\rangle = \langle 0^n|\psi_s\rangle$ has a magnitude $1/\sqrt{2}$.

Proof. The rank of QRC is $(q+1)/2$, which means that there are $n=(q+3)/2$ columns in $\mathbf{H}_s^{\mathrm{QRC}}$. To prove this corollary, it suffices to prove the following,

$$|\mathbf{c}_j| = 3 \pmod 4 \qquad\qquad \mathbf{c}_j\cdot\mathbf{c}_k = 1 \pmod 2 , \tag{4.20}$$

according to Theorem 4.1.

First, the number of non-zero quadratic residues modulo q is $(q-1)/2$. Since $q+1$ is a multiple of 8, we have $|\mathbf{c}_j| = (q-1)/2 = 3 \pmod 4$ for $j \neq 1$. For $j = 1$, $|\mathbf{c}_1| = q = 3 \pmod 4$.

As for the second formula, the cases (a) $j = k$, (b) $j = 1$ but $k \neq 1$ and (c) $j \neq 1$ but $k = 1$ follow the proof of the first formula. So, we focus on proving it for $j \neq k \neq 1$. Define the extended QRC by appending an extra parity bit to the codeword of QRC, which equals the Hamming weight of the codeword modulo 2. From classical coding theory, the extended QRC is self-dual [MS77]. That is, every two codewords of the extended QRC is orthogonal to each other. For $\mathbf{c}_j$, the added parity bit is 1, since these columns are odd-parity. Then, the fact that the extended codewords are orthogonal to each other implies that $\mathbf{c}_j \cdot \mathbf{c}_k = 1 \pmod 2$. This proves the form of the stabilizer tableau, which represents the generators $\{-Y_1X_2\cdots X_n, -X_1Y_2X_3\cdots X_n, \cdots, -X_1X_2\cdots X_{n-1}Y_n\}$.

Multiplying the first generator to the remaining $n-1$ generators gives the same stabilizer group with a different set of generators $\langle -Y_1X_2\cdots X_n, Z_1Z_2, Z_1Z_3\cdots, Z_1Z_n\rangle$. In this representation, the Z-type stabilizer generators have a positive phase and the number of non-Z generator is $g = 1$. According to Proposition 2.1, the correlation function has a magnitude $1/\sqrt{2}$ (i.e. 0.854 in probability bias) with respect to the secret, regardless of the size parameter q. ■

4.2 Construction Algorithm

In this section, we present the stabilizer construction, which is a systematic way to construct IQP circuits with $\theta = \pi/8$ for verification. In fact, the goal is to generate a pair $(\mathbf{H}, \mathbf{s})$, such that they satisfy certain conditions, which stem from Theorem 4.3. We first define the family of pairs that we would like to sample from.

Definition 4.5. Let $\mathcal{H}_{n,m,g} = \{(\mathbf{H}, \mathbf{s})\}$ be a family of pairs of an IQP matrix $\mathbf{H} \in \mathbb{F}_2^{m\times n}$ and a secret $\mathbf{s} \in \mathbb{F}_2^n$ satisfying the following conditions. (1) $\mathcal{D}_{\mathbf{s}} = C_{\mathbf{s}} \bigcap C_{\mathbf{s}}^{\perp}$ is a doubly-

```
Parameters: n, m, g
Output: $(\mathbf{H}, \mathbf{s}) \in \mathcal{H}_{n,m,g}$
1: Randomly sample $m_1$ and $d$ with certain constraints        ▷ Proposition 4.7
2: Sample $\mathbf{D} \in \mathbb{F}_2^{m_1 \times d}$ and $\mathbf{F} \in \mathbb{F}_2^{m_1 \times g}$ satisfying certain conditions ▷ Section 4.2.2
3: Initialize $\mathbf{H_s} \leftarrow (\mathbf{F}, \mathbf{D}, \mathbf{0}_{m_1 \times (n-r)})$, where $r = g + d$
4: Sample a secret $\mathbf{s}$ from the solutions of $\mathbf{H_s}\ \mathbf{s} = \mathbf{1}$
5: $\mathbf{H} \leftarrow \begin{pmatrix} \mathbf{H_s} \\ \mathbf{R_s} \end{pmatrix}$, where $\mathbf{R_s}$ is a random matrix with $m - m_1$ rows satisfying $\mathbf{R_s s} = \mathbf{0}$
   and $\mathrm{rank}(\mathbf{H}) = n$
6: Perform obfuscation as in Definition 3.3
```

Meta-Algorithm 1: Stabilizer construction

even code, where $C_\mathbf{s}$ is the code generated by columns of $\mathbf{H_s}$ and $C_\mathbf{s}^\perp$ is its dual code; (2) $\mathrm{rank}(\mathbf{H}_\mathbf{s}^T \mathbf{H_s}) = g$; (3) $\mathrm{rank}(\mathbf{H}) = n$.

In this definition, the size of the IQP circuits are determined by n and m, which correspond to the number of qubits and gates, respectively. Additionally, condition (1) is to guarantee that the correlation function $\langle \mathcal{Z}_\mathbf{s} \rangle$ corresponding to instances of $\mathcal{H}_{n,m,g}$ is nonzero, and condition (2) states that its magnitude is given by $2^{-g/2}$. Therefore, the family $\mathcal{H}_{n,m,g}$ includes all instances of IQP circuits of a certain size that have correlation function $\pm 2^{-g/2}$ with respect to some secret $\mathbf{s}$. Note that the rank of the Gram matrix $\mathbf{H}_\mathbf{s}^T \mathbf{H_s}$ should be $g = O(\log n)$ for the protocol to be practical. The reason for considering IQP matrices $\mathbf{H}$ with full column rank will be made clear when we discuss the classical security of the IQP-based verification protocol (Section 5.1.2).

Moreover, we give an efficient classical sampling algorithm to sample instances from $\mathcal{H}_{n,m,g}$, which is the stabilizer construction (Meta-Algorithm 1).

Theorem 4.6. *There exists an efficient classical sampling algorithm that sample from $\mathcal{H}_{n,m,g}$, given the parameters n, m and g.*

For the algorithmic purpose, we set two additional parameters, m_1 and d, which are the number of rows in $\mathbf{H_s}$ and the dimension of $\mathcal{D}_\mathbf{s}$, respectively. These are random integers satisfying certain natural constraints. The rank of $\mathbf{H_s}$ is then equal to $r = g+d$.

Proposition 4.7 (Parameter constraints). *Given* $(\mathbf{H}, \mathbf{s}) \in \mathcal{H}_{n,m,g}$, *let* $\mathcal{D}_\mathbf{s} = C_\mathbf{s} \bigcap C_\mathbf{s}^\perp$, *where* $C_\mathbf{s}$ *is the code generated by* $\mathbf{H}_\mathbf{s}$ *and* $C_\mathbf{s}^\perp$ *is the dual code. Let* m_1 *be the number of rows in* $\mathbf{H}_\mathbf{s}$ *and* $d = \dim(\mathcal{D}_\mathbf{s})$, *which means* $\dim(C_\mathbf{s}) = g + d$. *Then, we have*

- $g + d \leq n$;
- $0 < m_1 \leq m$;
- $n - g - d \leq m - m_1$;
- $g + 2d \leq m_1$;
- $m_1 = g \bmod 2$.

Proof. The first constraint is because $\operatorname{rank}(\mathbf{H}_\mathbf{s}) \leq n$. The second one is trivial. The third one is due to the fact that $\mathbf{H}$ is of full column rank, which means that the number of redundant rows should be $m - m_1 \geq n - \operatorname{rank}(\mathbf{H}_\mathbf{s}) = n - g - d$. The fourth one is because $\dim(C_\mathbf{s}) + \dim(C_\mathbf{s}^\perp) = m_1$ and $\dim(\mathcal{D}_\mathbf{s}) \leq \dim(C_\mathbf{s}^\perp)$. The fifth one is from Theorem 4.9. ■

The stabilizer construction works by sampling $\mathbf{H}_\mathbf{s}$ and $\mathbf{R}_\mathbf{s}$ in certain 'standard forms', up to row permutations and column operations. Note that the 'standard forms' of $\mathbf{H}_\mathbf{s}$ and $\mathbf{R}_\mathbf{s}$ are not necessarily unique.

4.2.1 Standard Form

We first discuss $\mathbf{R}_\mathbf{s}$. To ensure that $\operatorname{rank}(\mathbf{H}) = n$, observe that in any $\mathbf{H}$ of full column rank, the redundant rows $\mathbf{R}_\mathbf{s}$ can always be transformed by row permutations into a form, where the first $n - r$ rows form a basis of $\mathbb{F}_2^n$ together with the rows in $\mathbf{H}_\mathbf{s}$. Therefore, up to row permutations, the first $n - r$ rows of $\mathbf{R}_\mathbf{s}$ are sampled to be random independent rows that are orthogonal to $\mathbf{s}$ and lie outside the row space of $\mathbf{H}_\mathbf{s}$. The remaining rows in $\mathbf{R}_\mathbf{s}$ are random rows orthogonal to $\mathbf{s}$.

Next, we discuss sampling $(\mathbf{H}_\mathbf{s}, \mathbf{s})$, which is the core of the stabilizer construction. Essentially, we want to randomly generate a (possibly redundant) generator matrix $\mathbf{H}_\mathbf{s}$ of a code $C_\mathbf{s}$, so that its dimension is r, its intersection $\mathcal{D}_\mathbf{s}$ with the dual code is a doubly-even code with dimension $d = r - g$ and the all-ones vector is a codeword. The last condition guarantees that a secret $\mathbf{s}$ can always be found. Note that, we allow $\mathrm{rank}(\mathbf{H_s}) < n$. That is, we allow $\mathbf{H_s}$ to be a "redundant" generator matrix of $\mathcal{C}_\mathbf{s}$, instead of a full-rank one. This is called adding column redundancy to the full-rank generator matrix of $\mathcal{C}_\mathbf{s}$, because after the obfuscation process, there will be redundant linear combinations in the columns of $\mathbf{H_s}$ (see Section 3.5).

For such a generator matrix, there is an invertible matrix $\mathbf{Q}$ to perform a basis change so that

$$\mathbf{H_s Q} = (\mathbf{F}, \mathbf{D}, \mathbf{0}_{m_1 \times (n-r)}) , \tag{4.21}$$

where $\mathbf{D} \in \mathbb{F}_2^{m_1 \times d}$ is a generator matrix of the doubly-even code $\mathcal{D}_\mathbf{s}$, and columns in $\mathbf{F} \in \mathbb{F}_2^{m_1 \times g}$ span $\mathcal{C}_\mathbf{s}/\mathcal{D}_\mathbf{s}$. In addition, it can be shown that $\mathrm{rank}(\mathbf{F}^T\mathbf{F}) = \mathrm{rank}(\mathbf{Q}^T\mathbf{H}_\mathbf{s}^T\mathbf{H_s Q}) = \mathrm{rank}(\mathbf{H}_\mathbf{s}^T\mathbf{H_s}) = g$. In more details, $\mathbf{D}$ and $\mathbf{F}$ shall satisfy the following conditions.

Proposition 4.8 (Conditions of D and F). *Given $(\mathbf{H}, \mathbf{s}) \in \mathcal{H}_{n,m,g}$, let $\mathbf{H_s}$ be the rows of $\mathbf{H}$ that are not orthogonal to $\mathbf{s}$. Then, there exists an invertible $\mathbf{Q}$, so that $\mathbf{H_s Q} = (\mathbf{F}, \mathbf{D}, \mathbf{0})$ and*

- *$\mathbf{D}$ consists of $d = r - g$ independent vectors with weight 0 modulo 4, which are orthogonal to each other, with $r = \mathrm{rank}(\mathbf{H_s})$.*
- *$\mathbf{F}$ consists of g independent columns from $\ker(\mathbf{D}^T)$ which lie outside the column space of $\mathbf{D}$.*
- *$\mathbf{F}^T\mathbf{F}$ is a random g-by-g symmetric matrix with rank g.*
- *The all-ones vector $\mathbf{1}$ either explicitly appears as the first column of $\mathbf{D}$ or $\mathbf{F}$, or it can be written as the sum of the first two columns of $\mathbf{F}$.*

Proof. The matrix $\mathbf{D}$ is taken as the generator matrix of the dual intersection $\mathcal{D}_\mathbf{s}$, which is a doubly-even code. The form of $\mathbf{D}$ follows from Proposition 2.4. The second condition is because $\mathcal{D}_\mathbf{s} \subset \mathcal{C}_\mathbf{s}^\perp$, which implies $\mathbf{D}^T\mathbf{F} = \mathbf{0}$. So, columns of $\mathbf{F}$ lie in $\ker(\mathbf{D}^T)$. The third condition is because $\mathrm{rank}(\mathbf{F}^T\mathbf{F}) = \mathrm{rank}(\mathbf{Q}^T\mathbf{H}_\mathbf{s}^T\mathbf{H_s Q}) = \mathrm{rank}(\mathbf{H}_\mathbf{s}^T\mathbf{H_s}) = g$. As for the last condition, if $\mathbf{1} \in \mathcal{C}_\mathbf{s}$, one can always perform basis change so that $\mathbf{1}$ explicitly appears in the columns of $\mathbf{H_s}$. More specifically, if $\mathbf{1} \in \mathcal{D}_\mathbf{s}$, the column operation $\mathbf{Q}$ can transform it as the first column of $\mathbf{D}$. If not, it can be made as the first column of $\mathbf{F}$. The second part of this condition can be achieved by adding the second column of $\mathbf{F}$ to the first. ■

Moreover, although there might be no unique standard form of $\mathbf{H_s}$, the Gram matrix has a unique standard form. First note that row permutations have no effect on the Gram matrix, since $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ for a permutation matrix $\mathbf{P}$. So we focus on column operations. As shown in [KS08], there exists an invertible matrix $\mathbf{Q}$, so that

$$\mathbf{Q}^T\mathbf{H}_\mathbf{s}^T\mathbf{H}_\mathbf{s}\mathbf{Q} = \operatorname{diag}\left(\mathbf{I}_g, \mathbf{0}\right) \text{ or } \operatorname{diag}\left(\bigoplus_{i=1}^{g/2}\mathbf{J}, \mathbf{0}\right), \tag{4.22}$$

depending on whether at least one diagonal element of $\mathbf{H}_\mathbf{s}^T\mathbf{H}_\mathbf{s}$ is 1 or not, where $\mathbf{J} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. However, for the construction purpose, we need to ensure that the all-ones vector is a codeword of $\mathcal{C}_\mathbf{s}$. Therefore, we give a slightly different standard form of $\mathbf{H}_\mathbf{s}^T\mathbf{H}_\mathbf{s}$, which can be achieved by $\mathbf{H_s}$ in the form of $(\mathbf{F}, \mathbf{D}, \mathbf{0})$.

Theorem 4.9. *Let* $(\mathbf{H}, \mathbf{s})$ *be a random instance from* $\mathcal{H}_{n,m,g}$ *and let* $\mathbf{H_s}$ *be the rows of* $\mathbf{H}$ *satisfying* $\mathbf{H_s}\,\mathbf{s} = \mathbf{1}$. *Then, there exists an invertible matrix* $\mathbf{Q}$, *so that* $\mathbf{H_s}\mathbf{Q}$ *is in the form of* $(\mathbf{F}, \mathbf{D}, \mathbf{0})$ *with* $\mathbf{D}$ *and* $\mathbf{F}$ *satisfying the conditions in Proposition 4.8 and*

$$\mathbf{Q}^T\mathbf{H}_\mathbf{s}^T\mathbf{H}_\mathbf{s}\mathbf{Q} = \begin{pmatrix} \mathbf{I} & & & & \\ & \mathbf{J} & & & \\ & & \ddots & & \\ & & & \mathbf{J} & \\ & & & & \mathbf{0}_{(n-g)\times(n-g)} \end{pmatrix} \textit{ or } \begin{pmatrix} \mathbf{J} & & & \\ & \ddots & & \\ & & \mathbf{J} & \\ & & & \mathbf{0}_{(n-g)\times(n-g)} \end{pmatrix}, \tag{4.23}$$

where $\mathbf{I}$ *is either 1 or* $\mathbf{I}_2$ *and* $\mathbf{J} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. *In addition,* $m_1 = g \bmod 2$, *where* m_1 *is the number of rows in* $\mathbf{H_s}$.

Proof. First, according to Proposition 4.8, up to the column operations, $\mathbf{H}_\mathbf{s}^T\mathbf{H}_\mathbf{s} \sim_c \mathbf{F}^T\mathbf{F} \oplus \mathbf{D}^T\mathbf{D} \oplus \mathbf{0}$. For the $\mathbf{D}$ matrix, we have $\mathbf{D}^T\mathbf{D} = \mathbf{0}_{d\times d}$, which already matches standard form, where $d = \operatorname{rank}(\mathbf{H}_s) - g$. So, we focus on $\mathbf{G}' := \mathbf{F}^T\mathbf{F}$. The matrix $\mathbf{F}$ satisfies $\mathbf{D}^T\mathbf{F} = \mathbf{0}$ and $\operatorname{rank}(\mathbf{F}^T\mathbf{F}) = g$, where g is the number of columns in $\mathbf{F}$. We want to find an invertible matrix $\mathbf{Q}'$, which leaves the $\mathbf{D}$ matrix unchanged and only changes the $\mathbf{F}$ matrix, so that $\mathbf{G}'$ is equal to the top-left $g \times g$ submatrix in the standard form. We first discuss the congruent standard form of general full-rank symmetric matrix.

(1) First, suppose that not all diagonal elements of $\mathbf{G}'$ are zero. In this case, we can assume $\mathbf{G}'_{11} = 1$, because otherwise, we can always apply a permutation matrix to $\mathbf{F}$, so that the nonzero diagonal element of $\mathbf{G}'$ is moved to the $(1, 1)$-location. Then, up to congruent transformations,

$$\mathbf{G}' = \begin{pmatrix} 1 & \mathbf{g}^T \\ \mathbf{g} & \mathbf{G}_1 \end{pmatrix}. \tag{4.24}$$

Let,

$$\mathbf{Q}_1 = \begin{pmatrix} 1 & \mathbf{g}^T \\ 0 & \mathbf{I} \end{pmatrix}. \tag{4.25}$$

We have,

$$\mathbf{Q}_1^T\mathbf{G}'\mathbf{Q}_1 = \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{g}\mathbf{g}^T + \mathbf{G}_1 \end{pmatrix}. \tag{4.26}$$

(2) If $\mathbf{G}'_{jj} = 0$ for $1 \leq j \leq g$, then without loss of generality, we can assume $\mathbf{G}'_{12} = 1$; otherwise, we can apply a permutation matrix to swap the the non-zero entry to the $(1, 2)$ and $(2, 1)$ positions. In this case, up to congruent transformations,

$$\mathbf{G}' = \begin{pmatrix} \mathbf{J} & \mathbf{G}_2 \\ \mathbf{G}_2^T & \mathbf{G}_3 \end{pmatrix}, \tag{4.27}$$

where $\mathbf{J} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Let

$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{I}_2 & \mathbf{J}\mathbf{G}_2 \\ \mathbf{0} & \mathbf{I}_{g-2} \end{pmatrix}. \tag{4.28}$$

Then,

$$\mathbf{Q}_1^T\mathbf{G}'\mathbf{Q}_1 = \begin{pmatrix} \mathbf{J} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{G}_2^T\mathbf{J}\mathbf{G}_2 + \mathbf{G}_3 \end{pmatrix} . \tag{4.29}$$

Therefore, in the congruent standard form, $\mathbf{F}^T\mathbf{F}$ is a block-diagonal matrix of the form

$$(\mathbf{I}\oplus)\mathbf{J} \oplus \cdots \oplus \mathbf{J} . \tag{4.30}$$

- If m_1 is odd, then $\mathbf{1}$ must be in $\mathcal{C}_\mathbf{s}\backslash\mathcal{D}_\mathbf{s}$. In this case, $\mathbf{1}$ is the first column of $\mathbf{F}$, according to Proposition 4.8. Then, $\mathbf{G}'_{11} = 1$ and applying the transformation of Eq. (4.25) leads to a matrix in the form of Eq. (4.26). This implies that all other columns in the new $\mathbf{F}$ are orthogonal to $\mathbf{1}$ and hence have even parity. Therefore, we have $\mathbf{G}'_{jj} = 0$ for $j > 1$ and the congruent standard form is $\mathbf{G}' \sim_c 1 \oplus \left(\bigoplus_{i=1}^{(g-1)/2} \mathbf{J}\right)$.
- If m_1 is even and $\mathbf{1} \notin \mathcal{D}_\mathbf{s}$, then $\mathbf{1}$ is also the first column of $\mathbf{F}$, according to Proposition 4.8. Then, we can assume that $\mathbf{G}'_{12} = 1$, as what we did in proving the general congruent standard form. This implies that the second column of $\mathbf{F}$ must be odd-parity, and so $\mathbf{G}'_{22} = 1$. As as result, up to congruent transformations,

$$\mathbf{G}' = \begin{pmatrix} \mathbf{J}_1 & \mathbf{G}_2 \\ \mathbf{G}_2^T & \mathbf{G}_3 \end{pmatrix} , \tag{4.31}$$

where $\mathbf{J}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$. Let

$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{I}_2 & \mathbf{J}_3\mathbf{G}_2 \\ \mathbf{0} & \mathbf{I}_{g-2} \end{pmatrix} , \tag{4.32}$$

where $\mathbf{J}_3 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. Then,

$$\mathbf{Q}_1^T\mathbf{G}'\mathbf{Q}_1 = \begin{pmatrix} \mathbf{J}_1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{G}_2^T\mathbf{J}_3\mathbf{G}_2 + \mathbf{G}_3 \end{pmatrix} . \tag{4.33}$$

From the second row (column) of $\mathbf{Q}_1^T\mathbf{G}'\mathbf{Q}_1$, one can see that only the second column of the new $\mathbf{F}$ is odd-parity and all other columns are even parity, which means that the diagonal elements of $\mathbf{G}_2^T\mathbf{J}_3\mathbf{G}_2 + \mathbf{G}_3$ are zero. Then, Eq. (4.28) is repeatedly applied, so that $\mathbf{G}' = \mathbf{J}_1 \oplus \mathbf{J} \oplus \cdots \oplus \mathbf{J}$. Finally, let $\mathbf{Q}_2$ be an invertible matrix that adds the second column to the first. We have $\mathbf{Q}_2^T\mathbf{G}'\mathbf{Q}_2 = \mathbf{I}_2 \oplus \mathbf{J} \oplus \cdots \oplus \mathbf{J}$.

- If m_1 is even and $\mathbf{1} \in \mathcal{D}_{\mathbf{s}}$, then m_1 must be a multiple of 4 and $\mathbf{1}$ is the first column of $\mathbf{D}$, according to Proposition 4.8. In this case, $\mathbf{D}^T\mathbf{F} = \mathbf{0}$ implies that all columns of $\mathbf{F}$ will be even-parity, which means that the diagonal elements of $\mathbf{G}'$ will be zero. Moreover, the diagonal elements will remain zero if the congruent transformation only acts nontrivially on $\mathbf{G}'$. Therefore, in the congruent standard form, $\mathbf{F}^T\mathbf{F} \sim_c \bigoplus_{i=1}^{g/2} \mathbf{J}$.

Above, the all-ones vector appears as the first column of $\mathbf{D}$ or $\mathbf{F}$ except for the third case, where $\mathbf{1} = \mathbf{c}_1 + \mathbf{c}_2$ can be obtained by adding up the first two columns of $\mathbf{F}$. Finally, in all of the above cases, $m_1 = g \bmod 2$. ■

According to this theorem, sampling $(\mathbf{H}_{\mathbf{s}}, \mathbf{s})$ is reduced to generating an $\mathbf{H}_{\mathbf{s}} = (\mathbf{F}, \mathbf{D}, \mathbf{0})$ so that the Gram matrix $\mathbf{H}_{\mathbf{s}}^T\mathbf{H}_{\mathbf{s}}$ is in the form of Eq. (4.23). Then, a secret $\mathbf{s}$ is sampled from the solutions of $\mathbf{H}_{\mathbf{s}}\,\mathbf{s} = \mathbf{1}$. Sampling such an $\mathbf{H}_{\mathbf{s}}$ is further reduced to sampling $\mathbf{D}$ and $\mathbf{F}$, so that $\mathbf{D}$ is a generator matrix for a random doubly-even code and $\mathbf{F}$ is a random matrix satisfying $\mathbf{D}^T\mathbf{F} = \mathbf{0}$, $\operatorname{rank}(\mathbf{F}^T\mathbf{F}) = g$ and that $\mathbf{1}$ is in the column space of $(\mathbf{F}, \mathbf{D})$.

4.2.2 Sampling D and F

We now discuss the sampling of $\mathbf{D}$ and $\mathbf{F}$, which can be implemented efficiently.

Sampling D. Here, the goal is to sample a $\mathbf{D} = (\mathbf{c}_1, \cdots, \mathbf{c}_d)$ with $d \le (m_1 - g)/2$, where $m_1 = g \bmod 2$. Columns in $\mathbf{D}$ are orthogonal to each other and have weight a multiple of 4, according to Proposition 4.8. The algorithm is shown in Algorithm 1, which works as follows. First, $\mathbf{c}_1$ can be a random vector with weight 0 modulo 4; $\mathbf{D}$ is

```
Parameters: $m_1$ and $d$
Require: $d \leq m_1/2$
 1: $\mathbf{c}_1 \leftarrow$ a random vector with weight 0 modulo 4
 2: $\mathbf{D} \leftarrow (\mathbf{c}_1)$
 3: for $t = 1, \cdots, d-1$ do
 4:     $\mathbf{c}_{t+1} \leftarrow$ a random vector from $\ker(\mathbf{D}^T)/\langle \mathbf{c}_1, \cdots, \mathbf{c}_t \rangle$ with weight 0 modulo 4
 5:     if $\mathbf{c}_{t+1}$ does not exist then
 6:         break
 7:     end if
 8:     $\mathbf{D} \leftarrow (\mathbf{D}, \mathbf{c}_{t+1})$
 9: end for
10: if $\mathbf{1}$ lies in the column space of $\mathbf{D}$ then
11:     Apply column operations so that $\mathbf{1}$ is the first column of $\mathbf{D}$
12: end if
13: return $\mathbf{D}$
```

Algorithm 1: Algorithm to sample a $\mathbf{D} = (\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_d)$ so that $\mathbf{c}_i \cdot \mathbf{c}_j = 0$ and $|\mathbf{c}_i| = 0 \bmod 4$.

initialized as $\mathbf{D} = (\mathbf{c}_1)$. Then, the second column $\mathbf{c}_2$ is sampled with the constraint that $\mathbf{c}_1 \cdot \mathbf{c}_2 = 0$ and $|\mathbf{c}_2| = 0 \bmod 4$; $\mathbf{D}$ is updated to be $\mathbf{D} = (\mathbf{c}_1, \mathbf{c}_2)$. Next, the third column $\mathbf{c}_3$ is sampled so that it is orthogonal the first two columns and $|\mathbf{c}_3| = 0 \bmod 4$. This process is iterated until all d columns are sampled, or until no vector satisfying the condition can be sampled, in which case a matrix with $d-1$ columns will be returned.

In the t-th iteration, the vector $\mathbf{c}_t$ is sampled from $\ker(\mathbf{D}^T)/\langle \mathbf{c}_1, \cdots, \mathbf{c}_{t-1} \rangle$ with $\mathbf{D} = (\mathbf{c}_1, \cdots, \mathbf{c}_{t-1})$. That is, we want $\mathbf{c}_t$ to be orthogonal to the first $t-1$ columns and outside the linear subspace that they span. This can be achieved as follows. We first solve for a basis of $\ker(\mathbf{D}^T)$, and then the first $t-1$ of the basis vectors are set as $\{\mathbf{c}_1, \cdots, \mathbf{c}_{t-1}\}$, with the remaining basis vectors changed accordingly. The vector $\mathbf{c}_t$ is sampled to be the random linear combination of the remaining basis vectors. In this way, the orthogonality and independence of $\mathbf{c}_t$ with respect to $\mathbf{c}_1, \cdots, \mathbf{c}_{t-1}$ are guaranteed.

In addition, to ensure that $|\mathbf{c}_t| = 0 \bmod 4$, we can first sample an even-parity vector from $\ker(\mathbf{D}^T)/\langle \mathbf{c}_1, \cdots, \mathbf{c}_{t-1}\rangle$. It is well-known that for a linear subspace over $\mathbb{F}_2$, either all vectors are even-parity or half the vectors are even-parity. Therefore, the sampling of even-parity vector can be efficiently done and we denote resulted vector as $\mathbf{a}_1$. The weight of $\mathbf{a}_1$ will be either 0 or 2 modulo 4. If $|\mathbf{a}_1| = 0 \bmod 4$, then it is set to be $\mathbf{c}_t$. Otherwise, we sample a vector $\mathbf{a}_2$ from $\ker(\mathbf{D}^T)/\langle \mathbf{c}_1, \cdots, \mathbf{c}_{t-1}, \mathbf{a}_1\rangle$ that is orthogonal to $\mathbf{a}_1$; that is, $\mathbf{a}_2$ is a random vector from $\ker(\mathbf{D}^T)$ that is orthogonal to and outside $\langle \mathbf{c}_1, \cdots, \mathbf{c}_{t-1}, \mathbf{a}_1\rangle$. Then, if $|\mathbf{a}_2| = 0 \bmod 4$, it is set to be $\mathbf{c}_t$ and if not, it follows from Lemma 2.5 that $\mathbf{a}_1 + \mathbf{a}_2$ must have a weight that is a multiple of 4, and thus we assign it as $\mathbf{c}_t$. With this approach, a $\mathbf{c}_t$ with weight a multiple of 4 can be guaranteed to be sampled except for the final iteration of two extremal cases.

We now turn to discuss the cases $d = m_1/2$ and $d = (m_1 - 1)/2$, where in the last iteration, the column $\mathbf{c}_d$ *may* not exist. In such a case, only a matrix $\mathbf{D}$ with $d - 1$ columns will be returned. However, we would like to emphasize that it is the g parameter that affects the value of the correlation function. For the subspace $\mathcal{D}_\mathrm{s}$, we only require it to be doubly-even, and its dimension does not matter. Therefore, we do not require the sampling algorithm succeed every time when applied to these two extremal cases.

When $d = m_1/2$ with m_1 even, the resulting $\mathcal{D}_\mathrm{s}$ forms a doubly-even self-dual code, which implies $g = 0$. In this scenario, $\mathbf{1}$ is included in $\mathcal{D}_\mathrm{s}$ because $\mathbf{1}$ will be orthogonal to all vectors in $\mathcal{D}_\mathrm{s}$ and itself. This implies that m_1 must be a multiple of 4. An example of this case is the extended QRC [MS77]. On the other hand, when $d = (m_1 - 1)/2$ with m_1 odd, we have $g = 1$ along with $\mathcal{D}_\mathrm{s} = C_\mathrm{s}^\perp$. In this situation, the $\mathbf{F}$ matrix must be $\mathbf{1}$. The QRC serves as an example of this particular case.

The reason why the last iteration of Algorithm 1 may break on these two cases is as follows. We only discuss $d = m_1/2$, but the reasoning for $d = (m_1 - 1)/2$ is similar. When $t = d - 1$, $\mathbf{D} = (\mathbf{c}_1, \cdots, \mathbf{c}_{d-1})$. If $\mathbf{1}$ is not in $\langle \mathbf{c}_1, \cdots, \mathbf{c}_{d-1}\rangle$, then the algorithm will assign it as $\mathbf{c}_d$, and this iteration ends normally. However, if $\mathbf{1} \in \langle \mathbf{c}_1, \cdots, \mathbf{c}_{d-1}\rangle$, the dimension of $\ker(\mathbf{D}^T)$ is $m_1 - (d - 1) = m_1/2 + 1$. So, the subspace that $\mathbf{a}_1$ is

```
Input: $m_1, g$ and $\mathbf{D}$
Require: $g \leq m_1 - 2d$ with $d$ the number of columns in $\mathbf{D}$; $g = m_1 \bmod 2$
 1: $\mathcal{D} \leftarrow$ column space of $\mathbf{D}$
 2: if $m_1$ is odd then                                  ▷ $\mathbf{G}' = \mathrm{diag}(1, \mathbf{J}, \cdots, \mathbf{J})$ and $g$ is odd
 3:     $\mathbf{c}_1 \leftarrow \mathbf{1}_{m_1}$
 4:     $\mathbf{F} \leftarrow (\mathbf{c}_1)$
 5: else if $m_1$ is even and $\mathbf{1}_{m_1} \notin \mathcal{D}$ then          ▷ $\mathbf{G}' = \mathrm{diag}(\mathbf{I}_2, \mathbf{J}, \cdots, \mathbf{J})$
 6:     $\mathbf{c}_2 \leftarrow$ a random odd-parity vector in $\ker(\mathbf{D}^T)/\mathcal{D}$
 7:     $\mathbf{c}_1 \leftarrow \mathbf{1}_{m_1} + \mathbf{c}_2$
 8:     $\mathbf{F} \leftarrow (\mathbf{c}_1, \mathbf{c}_2)$
 9: else                                                  ▷ $\mathbf{G}' = \mathrm{diag}(\mathbf{J}, \cdots, \mathbf{J})$
10:     $\mathbf{c}_1 \leftarrow$ a random vector in $\ker(\mathbf{D}^T)/\mathcal{D}$
11:     $\mathbf{c}_2 \leftarrow$ a random vector in $\ker(\mathbf{D}^T)/\mathcal{D}$ that satisfies $\mathbf{c}_1 \cdot \mathbf{c}_2 = 1$
12:     $\mathbf{F} \leftarrow (\mathbf{c}_1, \mathbf{c}_2)$
13: end if
14: while number of columns in $\mathbf{F} < g$ do
15:     $C \leftarrow \mathcal{D} \oplus$ column space of $\mathbf{F}$
16:     $\mathbf{a} \leftarrow$ a random vector in $C^\perp/\mathcal{D}$
17:     $\mathbf{b} \leftarrow$ a random vector in $C^\perp/\mathcal{D}$ that satisfies $\mathbf{a} \cdot \mathbf{b} = 1$
18:     $\mathbf{F} \leftarrow (\mathbf{F}, \mathbf{a}, \mathbf{b})$
19: end while
20: return $\mathbf{F}$
```

Algorithm 2: Algorithm to sample $\mathbf{F} = (\mathbf{c}_1, \cdots, \mathbf{c}_g)$ so that $\mathbf{D}^T\mathbf{F} = \mathbf{0}$ and $\mathrm{rank}(\mathbf{F}^T\mathbf{F}) = g$.

sampled from has dimension $m_1/2 + 1 - t = 2$. If this subspace has a nonzero vector with weight 0 modulo 4, then this vector will be assigned as $\mathbf{c}_d$ and the iteration will also end normally. However, if all vectors in this subspace have weight 2 modulo 4, except for the all-zeros vector, then $\mathbf{c}_d$ could not be found. In this case, the iteration breaks.

Sampling F. Next, we give the algorithm to sample $\mathbf{F} = (\mathbf{c}_1, \cdots, \mathbf{c}_g)$ (Algorithm 2). The algorithm takes m_1, g and $\mathbf{D}$ as inputs, and outputs a matrix $\mathbf{F}$ so that $\mathbf{F}^T\mathbf{F}$ is a rank-g symmetric matrix in the standard form as in Eq. (4.23), $\mathbf{D}^T\mathbf{F} = \mathbf{0}$, and $\mathbf{1}$ lies in the span of columns in $\mathbf{D}$ and $\mathbf{F}$. This implies that all columns of $\mathbf{F}$ should be sampled from $\ker(\mathbf{D}^T)/\mathcal{D}_{\mathbf{s}}$, with the additional orthogonality constraints imposed by $\mathbf{F}^T\mathbf{F}$.

There are three cases for $\mathbf{F}^T\mathbf{F}$. First, if m_1 is odd, then $\mathbf{1}$ cannot lie in $\mathcal{D}_s$. According to Proposition 4.8 and Theorem 4.9, $\mathbf{1}$ can be set as the first column of $\mathbf{F}$ and $\mathbf{F}^T\mathbf{F} = \mathrm{diag}(1, \mathbf{J}, \cdots, \mathbf{J})$. Second, if m_1 is even but $\mathbf{1}$ is not in $\mathcal{D}_s$, then $\mathbf{F}^T\mathbf{F} = \mathrm{diag}(\mathbf{I}_2, \mathbf{J}, \cdots, \mathbf{J})$, according to Theorem 4.9. In this case, $\mathbf{c}_1$ and $\mathbf{c}_2$ are odd-parity vectors, and $\mathbf{c}_1 + \mathbf{c}_2 = \mathbf{1}$. Third, if m_1 is even and $\mathbf{1}$ lies in $\mathcal{D}_s$, then $\mathbf{F}^T\mathbf{F} = \mathrm{diag}(\mathbf{J}, \cdots, \mathbf{J})$. In this case, $\mathbf{c}_1$ is a random vector from $\ker(\mathbf{D}^T)/\mathcal{D}_s$ and $\mathbf{c}_2$ is a random vector from $\ker(\mathbf{D}^T)/\mathcal{D}_s$ satisfying $\mathbf{c}_2 \cdot \mathbf{c}_1 = 1$. Note that $\mathbf{c}_1$ and $\mathbf{c}_2$ are automatically even-parity, since they are orthogonal to $\mathbf{1}$. Moreover, $\mathbf{c}_2$ must lie outside the space $\mathcal{D}_s \oplus \langle \mathbf{c}_1 \rangle$, due to the constraint $\mathbf{c}_2 \cdot \mathbf{c}_1 = 1$.

After the initialization of $\mathbf{c}_1$ (and $\mathbf{c}_2$), the algorithm proceeds to sample other columns of $\mathbf{F}$, if $g > 1$ (or $g > 2$). We only illustrate the case when m_1 is odd below, but the sampling process for an even m_1 follows a similar pattern. For m_1 odd, $\mathbf{F}^T\mathbf{F} = \mathrm{diag}(1, \mathbf{J}, \cdots, \mathbf{J})$. We first initialize $C_s \leftarrow \mathcal{D}_s \oplus \langle \mathbf{c}_1 \rangle$, which is the subspace C in Algorithm 2. The block diagonal form of $\mathbf{F}^T\mathbf{F}$ implies that $\mathbf{c}_2$ and $\mathbf{c}_3$ are vectors from $\ker(\mathbf{D}^T)/\mathcal{D}_s$ that are orthogonal to $\mathbf{c}_1 = \mathbf{1}$, i.e., $\mathbf{c}_2, \mathbf{c}_3 \in C_s^{\perp}/\mathcal{D}_s$. For $\mathbf{c}_2$, it is sampled as a random vector from $C_s^{\perp}/\mathcal{D}_s$, which is the vector $\mathbf{a}$ in Algorithm 2. For $\mathbf{c}_3$, it is sampled as a random vector from $C_s^{\perp}/\mathcal{D}_s$ satisfying $\mathbf{c}_2 \cdot \mathbf{c}_3 = 1$, which is the vector $\mathbf{b}$ in Algorithm 2. This finishes the sampling of columns corresponding to the first $\mathbf{J}$ block. Then, the subspace C_s is updated to include $\mathbf{c}_2$ and $\mathbf{c}_3$ into its basis, with its dimension increased by 2. This process is repeated for other columns, until all g columns are sampled. Finally, $\mathbf{F} = (\mathbf{c}_1, \cdots, \mathbf{c}_g)$ and it can be verified that $\mathbf{F}^T\mathbf{F}$ is indeed equal to the standard form $\mathrm{diag}(1, \mathbf{J}, \cdots, \mathbf{J})$.

4.3 Construction Based on Matrix Factorization

Here, we present another construction of IQP circuits for verification. This construction was derived in an early exploration of the stabilizer construction, and it is based on the matrix factorization problem instead of the sampling of random generator matrices. In fact, the codes do not explicitly appear in the construction. Although this construction algorithm has been superseded by the sampling-based construction as in Section 4.2, it is still interesting in its own right, and it provides a different perspective on the construction of IQP circuits for verification.

Recall that the goal of the construction is to sample the pair $(\mathbf{H}, \mathbf{s})$, so that the correlation function $\langle \mathcal{Z}_{\mathbf{s}} \rangle = \langle 0^n | e^{-i\pi H/8} \mathcal{Z}_{\mathbf{s}} e^{i\pi H/8} | 0^n \rangle$ has a magnitude equal to a desired value $2^{-g/2}$, where g is an integer. Similar to the Shepherd-Bremner construction and the stabilizer construction in Meta-Algorithm 1, the matrix-factorization construction also consists of two steps, namely, (*i*) sampling $(\mathbf{H}_{\mathbf{s}}, \mathbf{s})$ and (*ii*) adding redundancy and obfuscation. The second step is the same as in the stabilizer construction, and we will not repeat it here.

As for the first step, in Section 4.1, we show how to transform $\mathbf{H}_{\mathbf{s}}$ (or equivalently, $e^{i\pi H_{\mathbf{s}}/4}$) into a stabilizer tableau and analyze the correlation function. Here, we reverse this process, and show that starting from a random stabilizer tableau of the form (4.1) and a desired value of the correlation function, one can *efficiently* find the corresponding $\mathbf{H}_{\mathbf{s}}$ (and $\mathbf{s}$).

4.3.1 Problem Formulation and Overview

Recall that the IQP tableau of $e^{i\pi H_{\mathbf{s}/4}}$ is of the form $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$, where $G_{jk} = \mathbf{c}_j \cdot \mathbf{c}_k$ is the inner product between columns of $\mathbf{H}_{\mathbf{s}}$. The inverse statement of Theorem 4.1 is that, given $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$, find $\mathbf{H}_{\mathbf{s}}$ so that $e^{i\pi H_{\mathbf{s}}/4}$ can be represented by this stabilizer tableau. According to Theorem 4.1, r_j and G_{jj} jointly determine the Hamming weight of the j-th column of $\mathbf{H}_{\mathbf{s}}$ modulo 4, which is $2r_j + G_{jj}$ explicitly. This imposes the **weight constraint** on $\mathbf{H}_{\mathbf{s}}$. Furthermore, $\mathbf{H}_{\mathbf{s}}$ needs to satisfy $\mathbf{H}_{\mathbf{s}} \cdot \mathbf{s} = \mathbf{1}$ for some nonzero $\mathbf{s}$, which is called the **codeword constraint**, because it means that the all-ones vector $\mathbf{1}$ needs to be a codeword in the codespace generated by $\mathbf{H}_{\mathbf{s}}$. Altogether, the problem of sampling $(\mathbf{H}_{\mathbf{s}}, \mathbf{s})$ is formulated as follows.

```
**Parameters:** $n, g$
**Output:** $(\mathbf{H_s}, \mathbf{s})$
 1: Set $\mathbf{G}$ in the form of Eq. (4.23)
 2: $\mathbf{r} \leftarrow (r_1, \cdots, r_g, 0, \cdots, 0)^T$, where $r_j$ is random bit
 3: Sample $\mathbf{s}$ from the solutions of $\mathbf{Gs} = (G_{11}, \cdots, G_{nn})^T$
 4: Random $\mathbf{H}_{\text{rand}}$ satisfying $\mathbf{H}_{\text{rand}}\mathbf{s} = \mathbf{1}$        ▷ Randomization procedure
 5: Transform $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ into $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$ according to $\mathbf{H}_{\text{rand}}$        ▷ Eq. (4.36)
 6: Construct the Lempel sequence (4.39) for $\mathbf{G}'$        ▷ Begin AdaptedLempel
 7: $\mathbf{H}_0 \leftarrow$ a random factor from the Lempel sequence        ▷ Initial factorization
 8: Obtain $\mathbf{H}_1$ from $\mathbf{H}_0$ to satisfy the weight constraint        ▷ Eq. (4.40)
 9: **if** $\mathbf{H}_1\ \mathbf{s} = \mathbf{1}$ **then**
10:     $\mathbf{H}'_{\mathbf{s}} \leftarrow \mathbf{H}_1$
11: **else**
12:     Obtain $\mathbf{H}_2$ from $\mathbf{H}_1$ to satisfy the codeword constraint        ▷ Eq. (4.46)
13:     Obtain $\mathbf{H}_3$ from $\mathbf{H}_2$ as in Eq. (4.47). Set $\mathbf{H}'_{\mathbf{s}} \leftarrow \mathbf{H}_3$
14: **end if**
15: $\mathbf{H_s} \leftarrow \begin{pmatrix} \mathbf{H}_{\text{rand}} \\ \mathbf{H}'_{\mathbf{s}} \end{pmatrix}$
16: **return** $(\mathbf{H_s}, \mathbf{s})$
```

Meta-Algorithm 2: Matrix-factorization construction: sampling $(\mathbf{H_s}, \mathbf{s})$.

Problem 4.10. *Given a symmetric matrix* $\mathbf{G} \in \mathbb{F}_2^{n\times n}$, *a vector* $\mathbf{r} \in \mathbb{F}_2^n$ *and a vector* $\mathbf{s} \in \mathbb{F}_2^n$, *sample* $\mathbf{H_s} = (\mathbf{c}_1, \cdots, \mathbf{c}_n)$, *such that (a)* $\mathbf{G} = \mathbf{H_s}^T\mathbf{H_s}$, *(b)* $|\mathbf{c}_j| \equiv 2r_j + G_{jj} \bmod 4$ *for all* $j \in [n]$ *and (c)* $\mathbf{H_s} \cdot \mathbf{s} = \mathbf{1}$.

This is a constrained matrix factorization problem, and $\mathbf{H_s}$ is a factor of $\mathbf{G}$ satisfying the two constraints. Note that different from the stabilizer construction, where the secret $\mathbf{s}$ is obtained by solving the linear system $\mathbf{H_s} \cdot \mathbf{s} = \mathbf{1}$, here $\mathbf{s}$ is randomly sampled and given as part of the input.

This constrained matrix factorization problem can be efficiently solved if $\mathbf{G}$ and $\mathbf{s}$ satisfy certain relation (Meta-Algorithm 2).

Theorem 4.11. *Consider a stabilizer tableau* $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ *for n qubits with* $\mathbf{G}$ *symmetric, and let* $\mathbf{s} \in \mathbb{F}_2^n$ *satisfy* $\mathbf{Gs} = (G_{11}, \cdots, G_{nn})^T$. *Under these conditions, Meta-Algorithm 2 efficiently generates a random solution to the constrained matrix factorization problem* *described in Problem 4.10, which is a random factor* $\mathbf{H_s}$ *of* $\mathbf{G}$ *while satisfying both weight and codeword constraints.*

The relation between $\mathbf{G}$ and $\mathbf{s}$ is called the self-consistent equation, which is a necessary condition for $\mathbf{G}$ to have at least one factor satisfying the codeword constraint. To see this, suppose $\mathbf{E} = (\mathbf{c}_1, \cdots, \mathbf{c}_n)$ is any factor of $\mathbf{G}$ satisfying $\mathbf{E} \cdot \mathbf{s} = \mathbf{1}$. Then, $\mathbf{G}\mathbf{s} = \mathbf{E}^T\mathbf{E} \cdot \mathbf{s} \equiv (|\mathbf{c}_i|, \cdots, |\mathbf{c}_n|)^T \bmod 2$, which is just the diagonal element of $\mathbf{G}$.

Below, we first give an overview of Meta-Algorithm 2, and present the details in the following sections. First, the algorithm sample a random stabilizer tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$, so that the underlying stabilizer state has overlap $2^{-g/2}$ with $|0^n\rangle$. Then, a secret $\mathbf{s}$ is sampled from the solutions of $\mathbf{G}\mathbf{s} = (G_{11}, \cdots, G_{nn})^T$. The stabilizer tableau defines a constrained matrix factorization problem, which can be solved by an adapted Lempel's algorithm in Section 4.3.4. But given the number of rows in $\mathbf{H_s}$, the output of the adapted Lempel's algorithm is not random enough, which is only comprised of a small subset of all possible solutions. In order to add randomness to the final factor $\mathbf{H_s}$ so that every possible instances of $\mathcal{H}_{n,m,g}$ can be reached, the algorithm performs a randomization procedure in Section 4.3.3 before the adapted Lempel's algorithm. This randomization procedure transforms $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ into $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$ and outputs $\mathbf{H}_{\text{rand}}$, which is a random matrix satisfying $\mathbf{H}_{\text{rand}}\mathbf{s} = \mathbf{1}$. The new stabilizer tableau $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$ defines a new matrix factorization problem, which will then be solved by the adapted Lempel's algorithm. Denoting the factor for $\mathbf{G}'$ by $\mathbf{H}'_\mathbf{s}$, the final factor associated with $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ will be given by $\mathbf{H_s} = \begin{pmatrix} \mathbf{H}_{\text{rand}} \\ \mathbf{H}'_\mathbf{s} \end{pmatrix}$.

We would like to remark that the number of rows in $\mathbf{H_s}$ can be controlled by setting the number of rows in $\mathbf{H}_{\text{rand}}$ and $\mathbf{H}_0$ in Meta-Algorithm 2. However, the number of rows in $\mathbf{H}'_\mathbf{s}$ may not appear to be the same as $\mathbf{H}_0$. Indeed, as we will see in Section 4.3.4, they may differ by at most 4 rows, which stems from the additional blocks added for satisfying the weight constraint. Therefore, it is not convenient to control the number of rows in $\mathbf{H_s}$ as precisely as in the stabilizer construction algorithm. This is a drawback of the matrix-factorization construction, although its impact is minor when n and m are large enough.

4.3.2 Sampling Secret and Stabilizer tableau

Meta-Algorithm 2 starts by setting the Gram matrix $\mathbf{G}$ in the standard form [KS08]. In principle, $\mathbf{G}$ can be a random symmetric matrix of rank g. But due to the effect of obfuscation (see Eq. (3.22)), we can just consider the standard form. When g is odd, the standard form of $\mathbf{G}$ is $\operatorname{diag}\left(\mathbf{I}_g, \mathbf{0}\right)$. When g is even, the standard form of $\mathbf{G}$ is randomly chosen to be $\operatorname{diag}\left(\mathbf{I}_g, \mathbf{0}\right)$ or $\operatorname{diag}\left(\bigoplus_{i=1}^{g/2} \mathbf{J}, \mathbf{0}\right)$. After generating the Gram matrix $\mathbf{G}$, the secret can be sampled from the solutions of the self-consistent equation,

$$\mathbf{G}\mathbf{s} = \begin{pmatrix} G_{11} \\ \vdots \\ G_{nn} \end{pmatrix} . \tag{4.34}$$

Then, the phase column is sampled so that the stabilizer state represented by $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ has nonzero overlap with $|0^n\rangle$. Note that for the Gram matrix $\mathbf{G}$ in the standard form, only the top-left $g \times g$ submatrix is nonzero and all other entries are zeros. So, for the overlap to be nonzero, $\mathbf{r}$ should be set as $\mathbf{r} = (r_1, \cdots, r_g, 0, \cdots, 0)^T$, where r_j's are random bits, according to Proposition 2.1.

4.3.3 Randomization

For a given stabilizer tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$, there are actually many factors $\mathbf{H}_\mathbf{s}$ satisfying the constraints in Problem 4.10. However, as we will see later, the output of the adapted Lempel's algorithm can only 'hit' a small subset of all possible solutions. To add randomness to the final factor $\mathbf{H}_\mathbf{s}$, we perform a randomization procedure before the adapted Lempel's algorithm. The randomization procedure is as follows.

(1) Given $\mathbf{s}$, randomly sample $\mathbf{H}_{\text{rand}}$ satisfying $\mathbf{H}_{\text{rand}}\, \mathbf{s} = \mathbf{1}$;

(2) If $\mathbf{H}_{\text{rand}}$ is already a solution, return it. Otherwise, proceed with the following steps;

(3) Use $\mathbf{H}_{\text{rand}}$ to transform $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ into $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$;

(4) Solve the constrained matrix factorization problem defined by $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$ and $\mathbf{s}$. Denote the resulting solution as $\mathbf{H}'_{\mathbf{s}}$;

(5) Return

$$\mathbf{H}_{\mathbf{s}} := \begin{pmatrix} \mathbf{H}_{\text{rand}} \\ \mathbf{H}'_{\mathbf{s}} \end{pmatrix} . \tag{4.35}$$

It is clear that this randomization procedure allows Meta-Algorithm 2 to 'hit' all possible solutions of the constrained matrix factorization problem, due to $\mathbf{H}_{\text{rand}}$.

Now, we explain this procedure in more details. Given the tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$, we denote the underlying stabilizer state by $|\psi_{\mathbf{s}}\rangle$. We sample a random matrix $\mathbf{H}_{\text{rand}}$ satisfying the constraint $\mathbf{H}_{\text{rand}}\,\mathbf{s} = \mathbf{1}$, and transform it into an IQP Hamiltonian H_{rand}. Then, we apply the inverse evolution $e^{-i\pi H_{\text{rand}}/4}$ to $|\psi_{\mathbf{s}}\rangle$, resulting in a new state $|\psi'_{\mathbf{s}}\rangle$:

$$|\psi'_{\mathbf{s}}\rangle := e^{-i\frac{\pi}{4}H_{\text{rand}}} |\psi_{\mathbf{s}}\rangle \ , \tag{4.36}$$

represented by a new tableau $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$. The new tableau can be obtained by applying Lemma 4.2. Note that after the randomization process, $\mathbf{G}'$ in the new stabilizer tableau is *not necessarily* low-rank.

Then, we solve Problem 4.10 defined by the new stabilizer tableau $(\mathbf{G}', \mathbf{I}_n, \mathbf{r}')$ and the original secret $\mathbf{s}$, using the adapted Lempel's algorithm described in Section 4.3.4; we denote the resulting matrix be $\mathbf{H}'_{\mathbf{s}}$. In the language of quantum circuit and quantum state, we have

$$e^{i\frac{\pi}{4}H'_{\mathbf{s}}} |0^n\rangle = e^{-i\frac{\pi}{4}H_{\text{rand}}} |\psi_{\mathbf{s}}\rangle \ . \tag{4.37}$$

If we define $H_{\mathbf{s}} := H_{\text{rand}} + H'_{\mathbf{s}}$, then $|\psi_{\mathbf{s}}\rangle = e^{i\frac{\pi}{4}H_{\mathbf{s}}} |0^n\rangle$, which is exactly the state represented by the tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$. In terms of binary matrix, let $\mathbf{G}_{\text{rand}} := \mathbf{H}_{\text{rand}}^T \mathbf{H}_{\text{rand}}$,

and

$$\mathbf{H_s} := \begin{pmatrix} \mathbf{H}_{\text{rand}} \\ \mathbf{H}'_{\mathbf{s}} \end{pmatrix} . \tag{4.38}$$

Then $\mathbf{G} = \mathbf{G}' + \mathbf{G}_{\text{rand}}$ and $\mathbf{H_s}$ is the solution of Problem 4.10 defined by $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ and $\mathbf{s}$.

4.3.4 Adapted Lempel's Algorithm

In this section, we will present the adapted Lempel's algorithm. Recall that given a stabilizer tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$ and a secret $\mathbf{s}$, our goal is to solve Problem 4.10 for a factor $\mathbf{H_s}$ of $\mathbf{G}$, satisfying the weight and codeword constraints. Note that $\mathbf{G}$ is not necessarily low-rank, due to the randomization procedure.

Without the constraints, matrix factorization over $\mathbb{F}_2$ has been studied by Lempel in 1975 [Lem75], who gave an efficient algorithm to find a minimal factorization for any symmetric matrix $\mathbf{G}$, i.e., a factor $\mathbf{B}$ with minimal rows so that $\mathbf{G} = \mathbf{B}^T\mathbf{B}$ (see also Section 2.1.4). Lempel's algorithm has been used several times in the context of the stabilizer formalism [CH17a, CH17b, HC18, FRCB22]. We will use Lempel's result as a starting point, and show how to adapt it to satisfy the two constraints with extra subroutines.

Initial factorization

We first give a brief review of Lempel's algorithm; see Section 2.1.4 for more details. First, as shown in Lemma 2.9, any symmetric matrix $\mathbf{G}$ has an elementary factorization $\mathbf{E}_0$ that contains at most $O(n^2)$ rows [Lem75]. Then, one can use an iterative procedure to eliminate rows in the factor, until certain stopping condition is satisfied, resulting in the following sequence

$$\mathbf{E}_0 \longrightarrow \mathbf{E}_1 \longrightarrow \mathbf{E}_2 \longrightarrow \cdots \longrightarrow \mathbf{B} \, , \tag{4.39}$$

which we will call the Lempel sequence. Above, the number of rows in $\mathbf{E}_i$ will be reduced at least by 1 compared to $\mathbf{E}_{i-1}$.

Strictly speaking, the last factor $\mathbf{B}$ in this sequence corresponds to the minimal factor of $\mathbf{G}$ only if $\mathbf{G}$ is non-singular. But our purpose is not to find the minimal factor, and we can choose any factor in this sequence as the starting point for the construction, which we denote as $\mathbf{H}_0$. To conclude, the initial factorization is as follows.

(1) Find an elementary factorization $\mathbf{E}_0$ of $\mathbf{G}$;

(2) Perform the iterative procedure to obtain the Lempel sequence (4.39);

(3) Sample a factor from the Lempel sequence as the initial factor $\mathbf{H}_0$.

Below, we will show how to tweak $\mathbf{H}_0$ to satisfy the two constraints imposed by the stabilizer tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r})$.

Satisfying the weight constraint

First, given $\mathbf{G} = \mathbf{H}_0^T\mathbf{H}_0$, we would like to obtain a new factor $\mathbf{H}_1$ that satisfies the weight constraint. Recall from Theorem 4.1 that the weight constraint (modulo 4) is imposed by $\mathbf{r}$ and the diagonal element of $\mathbf{G}$. We would like to append a block $\mathbf{H}_a$ to $\mathbf{H}_0$, so that

$$\mathbf{H}_1 := \begin{pmatrix} \mathbf{H}_0 \\ \mathbf{H}_a \end{pmatrix} \tag{4.40}$$

is a factor of $\mathbf{G}$ satisfying the weight constraint. Note that the appended block must satisfy $\mathbf{H}_a^T\mathbf{H}_a = \mathbf{0}$ since $\mathbf{H}_1^T\mathbf{H}_1 = \mathbf{H}_0^T\mathbf{H}_0 + \mathbf{H}_a^T\mathbf{H}_a = \mathbf{G}$. Below is an explicit construction of $\mathbf{H}_a$.

Suppose that the weight constraint is given by $\mathbf{w} = (w_1, \cdots, w_n)$ with $w_j = 2r_j + G_{jj}$. Given the matrix $\mathbf{H}_0$, we can also transform it into a stabilizer tableau $(\mathbf{G}, \mathbf{I}_n, \mathbf{r}')$, which represents the Clifford operator $e^{i\pi H_0/4}$. Then, the weights of columns in $\mathbf{H}_0$ modulo 4 is $\mathbf{w}' = (w_1', \cdots, w_n')$, with $w_j' = 2r_j' + G_{jj}$. It is not hard to see that either $r_j' = r_j$ or $r_j' + r_j = 1$; equivalently, w_j' will be either equal to w_j, or equal to $w_j \pm 2$. Let $\mathbf{a} \in \mathbb{F}_2^n$ be an indicator vector such that $a_j = 1$ if $w_j \neq w_j'$ and $a_j = 0$ otherwise.

Let $\mathbf{H}_a := (\mathbf{a}, \mathbf{a})^T$ and one can verify that $\mathbf{H}_a^T \mathbf{H}_a = \mathbf{0}$ since it contains two identical rows. Using the language of stabilizers, $\mathbf{H}_a$ represents the gates $iX_1^{a_1} \otimes \cdots \otimes X_n^{a_n}$ when $\theta = \pi/4$, which will only flip the phase part r'_j where $r_j \neq r'_j$. In this way, columns in $\mathbf{H}_1$ will have the same Hamming weights modulo 4 as $\mathbf{w}$, and thus satisfy the weight constraint.

Satisfying the codeword constraint

Here, we present the subroutine to make *any* factor of $\mathbf{G}$ satisfy the codeword constraint. First, we give a relation that holds for any factor of $\mathbf{G}$. Recall that $\mathbf{G}$ satisfies the self-consistent equation (4.34), which can also be written as,

$$\mathbf{G}\mathbf{s} \equiv \begin{pmatrix} |\mathbf{c}_1| \\ \vdots \\ |\mathbf{c}_n| \end{pmatrix} \bmod 2 \,. \tag{4.41}$$

In a slight abuse of notations, we use $(\mathbf{c}_1, \cdots, \mathbf{c}_n)$ to denote any factor of $\mathbf{G}$. Then, in the component form, the self-consistent equation reads,

$$\mathbf{c}_i \cdot \left(\sum_j s_j \mathbf{c}_j \right) = \mathbf{c}_i \cdot \mathbf{c}_i \,, \tag{4.42}$$

even if $\sum_j s_j \mathbf{c}_j \neq \mathbf{1}$.

Now, for the factor $\mathbf{H}_1$ obtained in the last step, if it already satisfies $\mathbf{H}_1 \cdot \mathbf{s} = \mathbf{1}$, then we can just skip this step. If not, without loss of generality, assume

$$\mathbf{H}_1 \cdot \mathbf{s} = \sum_j s_j \mathbf{c}_j = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} , \tag{4.43}$$

where the first k entries of the right-hand side is 1 and the remaining $m' - k$ entries is 0 (let m' be the number of rows in $\mathbf{H}_{\text{init}}$). We can always do such row permutation, because the row order does not affect the inner product between columns and the row permutation can be absorbed into the row permutations in the obfuscation (Eq. (3.22)). Then, we partition $\mathbf{H}_1$ into two parts $\mathbf{H}_1 = \begin{pmatrix} \mathbf{F} \\ \mathbf{Z}_1 \end{pmatrix}$ according to Eq. (4.43), where $\mathbf{F}$ consists of the first k rows of $\mathbf{H}_1$, and $\mathbf{Z}_1$ consists of the last $m' - k$ rows. We have,

$$\mathbf{F}\mathbf{s} = \mathbf{1} \qquad\qquad \mathbf{Z}_1\mathbf{s} = \mathbf{0}\,. \tag{4.44}$$

Next, since $\mathbf{H}_1\mathbf{s}$ has ones only on the first k entries, the self-consistent equation (4.42) implies that the first k entries of any column in $\mathbf{H}_1$ have the same parity of the whole column itself. Thus, the remaining entries must have even parity, which gives $\mathbf{Z}_1^T \cdot \mathbf{1} = \mathbf{0}$. We can always assume that $\mathbf{Z}_1$ has an even number of rows, because otherwise, we can append an all-zeros row to $\mathbf{Z}_1$ without affecting $\mathbf{Z}_1^T\mathbf{Z}_1$ and $\mathbf{H}_1^T\mathbf{H}_1$.

We would like to apply Lemma 2.8 to 'inject' the vector $\mathbf{1}$ into the column space of $\mathbf{Z}_1$. Specifically, let $\mathbf{Z}_2 := \mathbf{Z}_1 + \mathbf{1} \cdot \mathbf{x}^T$, where $\mathbf{x}$ satisfies $\mathbf{x} \cdot \mathbf{s} = 1$ and $|\mathbf{x}| = 1$. Then,

$$\mathbf{Z}_2 \cdot \mathbf{s} = (\mathbf{Z}_1 + \mathbf{1} \cdot \mathbf{x}^T) \cdot \mathbf{s} = \mathbf{1}\,, \tag{4.45}$$

and $\mathbf{Z}_2^T\mathbf{Z}_2 = \mathbf{Z}_1^T\mathbf{Z}_1$ according to Lemma 2.8. Note that $\mathbf{Z}_2$ is derived from $\mathbf{Z}_1$ by flipping all entries in the j-th column, where j is the index such that $x_j = 1$. Other columns are left unchanged, since $|\mathbf{x}| = 1$. Therefore, define

$$\mathbf{H}_2 := \begin{pmatrix} \mathbf{F} \\ \tilde{\mathbf{Z}} \end{pmatrix}\,, \tag{4.46}$$

and although we have $\mathbf{G} = \mathbf{H}_2^T\mathbf{H}_2$ and $\mathbf{H}_2\ \mathbf{s} = \mathbf{1}$, the j-th column of $\mathbf{H}_2$ may not satisfy the weight constraint.

To resolve this issue, define

$$\mathbf{H}_3 := \begin{cases} \mathbf{H}_2 & \text{if } \mathbf{H}_2 \text{ satisfies the weight constraint;} \\ (\mathbf{H}_2^T, \mathbf{x}, \mathbf{x})^T & \text{otherwise.} \end{cases} \tag{4.47}$$

We claim that $\mathbf{G} = \mathbf{H}_3^T\mathbf{H}_3$, $\mathbf{H}_3\ \mathbf{s} = \mathbf{1}$ and $\mathbf{H}_3$ satisfies the weight constraint. That is, $\mathbf{H}_3$ is a valid solution to Problem 4.10. Indeed, it can be verified that $\mathbf{H}_3^T\mathbf{H}_3 = \mathbf{H}_2^T\mathbf{H}_2$ since the two additional rows, if added, will not affect the inner products between columns. Moreover, $\mathbf{H}_3\ \mathbf{s} = \mathbf{1}$ since $\mathbf{H}_2\ \mathbf{s} = \mathbf{1}$ and $\mathbf{x} \cdot \mathbf{s} = 1$. Finally, if $\mathbf{H}_2$ does not satisfy the weight constraint, then $\mathbf{H}_3$ is obtained by modifying $\mathbf{H}_2$ in a similar way to Eq. (4.40). According to our discussion there, $\mathbf{H}_3$ will satisfy the weight constraint.

Putting everything together

To wrap up, the adapted Lempel's algorithm is as follows.

(1) Find an elementary factorization $\mathbf{E}_0$ of $\mathbf{G}$;

(2) Construct the Lempel sequence (4.39);

(3) Sample a random factor $\mathbf{H}_0$ from the Lempel sequence;

(4) Obtain $\mathbf{H}_1$ from $\mathbf{H}_0$, so that $\mathbf{H}_1$ satisfies the weight constraint and $\mathbf{G} = \mathbf{H}_1^T\mathbf{H}_1$;

(5) Obtain $\mathbf{H}_2$ from $\mathbf{H}_1$, so that $\mathbf{H}_2$ satisfies the codeword constraint and $\mathbf{G} = \mathbf{H}_2^T\mathbf{H}_2$;

(6) Obtain $\mathbf{H}_3$ from $\mathbf{H}_2$, so that $\mathbf{H}_3$ satisfies the weight constraint and the codeword constraint simultaneously and $\mathbf{G} = \mathbf{H}_3^T\mathbf{H}_3$.

Chapter 5

Classical Attacks and Security

In this chapter, we examine the classical security of our protocol, i.e., the possibility that an efficient classical prover can pass the test. A straightforward classical attack is to simulate the IQP circuit sent by the verifier. We do not expect this to be efficient, since there is generally no structure to be exploited by a classical simulation algorithm. For example, due to the obfuscation as in Eq. (3.22), the geometry of the IQP circuit can be arbitrary, which implies that the treewidth in a tensor network algorithm cannot be easily reduced [MS08].

Here, we focus on another class of classical attacks based on extracting secrets. Given an IQP matrix $\mathbf{H}$, once the hidden secret $\mathbf{s}$ is found, a classical prover can first calculate the correlation function $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ efficiently. Then, he generates a sample $\mathbf{x}$ which is orthogonal to $\mathbf{s}$ with probability $(1 + \langle \mathcal{Z}_{\mathbf{s}} \rangle)/2$ and not orthogonal to $\mathbf{s}$ with probability $(1 - \langle \mathcal{Z}_{\mathbf{s}} \rangle)/2$. The generated samples will have the correct correlation with the secret $\mathbf{s}$ and hence pass the test. Kahanamoku-Meyer's attack algorithm for the Shepherd-Bremner construction is an instance of this class [KM19].

But generally, this attack may not be efficient. From a code perspective, the stabilizer construction is to sample a random code satisfying certain constraints, and hide it by adding redundancy and performing obfuscation. Finding the secret allows one to find the hidden subcode, which should be a hard problem in general. In particular, we formulate the following conjecture.

Conjecture 5.1 (Hidden Structured Code (HSC) Problem). *For certain appropriate choices of n, m, g, there exists an efficiently samplable distribution over instances $(\mathbf{H}, \mathbf{s})$ from the family $\mathcal{H}_{n,m,g}$, so that no polynomial-time classical algorithm can find the secret $\mathbf{s}$ given n, m and $\mathbf{H}$ as input, with high probability over the distribution on $\mathcal{H}_{n,m,g}$.*

Naturally, sampling instances with uniform distribution from $\mathcal{H}_{n,m,g}$ is more favorable, since it does not put any bias on specific instances. For the underlying distribution induced by the stabilizer construction (Meta-Algorithm 1), it seems that it is uniform or close to uniform, as the output instances are random instances satisfying certain natural constraints imposed by the structure of the family $\mathcal{H}_{n,m,g}$. Though, we do not have a rigorous proof for this claim. Moreover, a similar conjecture was given in [SB09] for the family $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$, where the problem is to decide whether a given $\mathbf{H}$ is from the family $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ or not. They conjectured that such a problem is NP-complete. Here, to better align with the classical attack, we consider the problem of finding the secret $\mathbf{s}$ instead.

To support Conjecture 5.1, we first generalize Kahanamoku-Meyer's attack algorithm to target any IQP-based verification protocols with $\theta = \pi/8$. We show that this generalized attack, named the Linearity Attack, fails to break our construction. Furthermore, our analysis reveals that the loophole of the original Shepherd-Bremner construction stems from an improper choice of parameters. The Shepherd-Bremner construction can be improved by the column redundancy technique, which enables random sampling from the family $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ with any possible parameters and thereby fixes the loophole.

5.1 Linearity Attack

Classical attacks based on secret extraction aim to mimic the quantum behavior on certain candidate set S. Observe that given an IQP circuit represented by the binary matrix $\mathbf{H}$, a quantum prover can output a sample $\mathbf{x}$, which has the correlation function $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ in the direction of $\mathbf{s}$ for every $\mathbf{s}$, even if it is not the secret of the verifier. If a classical prover can also generate samples that have the correct correlation with every $\mathbf{s}$, then he has the power to classically sample from an IQP circuit, which is implausible [BJS11, BMS16]. However, he has the knowledge that the verifier will only check one secret. Therefore, a general attack strategy for him is to first reduce the set of candidate secrets from $\{0,1\}^n$ to a (polynomial-sized) subset S, and then generate samples that have the correct correlation with every vector in the candidate set.

Here, we discuss Linearity Attack, which is an instance of classical attacks based on secret extraction and generalizes the attack algorithm in [KM19] It consists of two steps. First, it uses linear algebraic techniques to construct a candidate set S. Then, the prover calculates the correlation function for every vector in S, and outputs samples that have the correct correlation with those vectors.

5.1.1 Secret Extraction

Overview of the algorithm

The secret extraction procedure in the Linearity Attack is presented in Meta-Algorithm 3, which is a generalized version of the procedure described in [KM19]. The algorithm begins by randomly selecting a vector $\mathbf{d}$ and eliminating rows in $\mathbf{H}$ that are orthogonal to $\mathbf{d}$, resulting in $\mathbf{H}_{\mathbf{d}}$. Subsequently, the algorithm searches for vectors that satisfy certain property check in $\ker(\mathbf{G}_{\mathbf{d}})$, where $\mathbf{G}_{\mathbf{d}} = \mathbf{H}_{\mathbf{d}}^T \mathbf{H}_{\mathbf{d}}$ represents the Gram matrix associated with $\mathbf{d}$. In what follows, we discuss some technical details and defer the analysis to Section 5.2.

```
1: procedure ExtractSecret($\mathbf{H}$)
2:     Initialize $S \leftarrow \emptyset$.                                ▷ candidate set
3:     repeat
4:         Uniformly randomly pick $\mathbf{d} \in \mathbb{F}_2^n$.
5:         Construct $\mathbf{H_d}$ and $\mathbf{G_d} = \mathbf{H_d}^T \mathbf{H_d}$
6:         for each vector $\mathbf{s}_i \in \ker(\mathbf{G_d})$ do
7:             if $\mathbf{s}_i$ passes certain property check then        ▷ To be specified
8:                 Add $\mathbf{s}_i$ to $S$.
9:             end if
10:        end for
11:    until some stopping criterion is met.
12:    return $S$
13: end procedure
```

Meta-Algorithm 3: The ExtractSecret($\mathbf{H}$) procedure of Linearity Attack.

Secret extraction in Kahanamoku-Meyer's attack

Meta-Algorithm 3 differs slightly from the approach described in [KM19]. In the original algorithm, the classical prover begins by constructing a matrix $\mathbf{M} \in \mathbb{F}_2^{l \times n}$ through linear combinations of rows in $\mathbf{H}$. Specifically, after sampling the vector $\mathbf{d}$, the classical prover proceeds to sample l random vectors $\mathbf{e}_1, \cdots, \mathbf{e}_l$. Then, the j-th row of $\mathbf{M}$ is defined by,

$$\mathbf{m}_j^T := \sum_{\substack{\mathbf{p}^T \in \mathrm{row}(\mathbf{H}) \\ \mathbf{p}\cdot\mathbf{d} = \mathbf{p}\cdot\mathbf{e}_j = 1}} \mathbf{p}^T . \tag{5.1}$$

After that, the original algorithm searches for the vectors that can pass certain property check in $\ker(\mathbf{M})$ instead. The secret extraction procedure of [KM19] is presented in Meta-Algorithm 4.

Our secret extraction algorithm is a generalization and simplification to the original approach. It can be shown that rows in $\mathbf{M}$ belong to the row space of $\mathbf{G_d}$. Therefore, to minimize the size of $\ker(\mathbf{M})$, one can simply set $\mathbf{M} = \mathbf{G_d}$, eliminating the need to sample the vectors $\mathbf{e}_1, \cdots, \mathbf{e}_l$.

Parameter: number of linear equations l

```
 1: procedure ExtractSecret($\mathbf{H}$)
 2:     Uniformly randomly pick $\mathbf{d} \in \mathbb{F}_2^n$.
 3:     for $j = 1, 2, \cdots, l$ do                ▷ construct the linear-system matrix $\mathbf{M}$
 4:         Uniformly randomly pick $\mathbf{e}_j \in \mathbb{F}_2^n$.
 5:         $\mathbf{m}_j^T \leftarrow$ RowSum($\mathbf{H}_{\mathbf{d},\mathbf{e}_j}$).
 6:     end for
 7:     $\mathbf{M} \leftarrow (\mathbf{m}_1, \cdots, \mathbf{m}_l)^T$.
 8:     for each vector $\mathbf{s}_i \in \ker(\mathbf{M})$ do
 9:         if $\mathbf{s}_i$ passes the QRC check then       ▷ discussed in the main text
10:             return $\mathbf{s}_i$
11:         end if
12:     end for
13: end procedure
```

Meta-Algorithm 4: The ExtractSecret($\mathbf{H}$) subroutine in [KM19]. Here, given $\mathbf{H}$ and two vectors $\mathbf{d}$ and $\mathbf{e}_t$, we define $\mathbf{H}_{\mathbf{d},\mathbf{e}_j}$ to be a submatrix from $\mathbf{H}$ by deleting rows orthogonal to either $\mathbf{d}$ or $\mathbf{e}_j$.

Proposition 5.2. *The matrix* $\mathbf{M}$ *obtained from Meta-Algorithm 4 consists of rows from the row space of* $\mathbf{G}_\mathbf{d} = \mathbf{H}_\mathbf{d}^T\mathbf{H}_\mathbf{d}$.

From this proposition, it is clear that to minimize the size of $\ker(\mathbf{M})$, one can choose $\mathbf{M} = \mathbf{G}_\mathbf{d}$. In this way, the sampling of $\mathbf{e}_j$'s can be removed.

Proof. Recall that the j-th row of $\mathbf{M}$ is obtained in the following way. First, we eliminate rows in $\mathbf{H}$ that are orthogonal to $\mathbf{d}$, which gives $\mathbf{H}_\mathbf{d}$. Then, we eliminate rows in $\mathbf{H}_\mathbf{d}$ that are orthogonal to $\mathbf{e}_j$, which gives $\mathbf{H}_{\mathbf{d},\mathbf{e}_j}$. Finally, we sum up the rows in $\mathbf{H}_{\mathbf{d},\mathbf{e}_j}$, which gives $\mathbf{m}_j^T$. Equivalently, we have

$$\mathbf{m}_j^T = (\mathbf{H}_\mathbf{d}\ \mathbf{e})^T\mathbf{H}_\mathbf{d} = \mathbf{e}^T\mathbf{G}_\mathbf{d}\ . \tag{5.2}$$

To see this, first observe that $\mathbf{H}_\mathbf{d}\ \mathbf{e}$ has ones in the positions where the corresponding rows are not orthogonal to $\mathbf{e}_j$. Then, $(\mathbf{H}_\mathbf{d}\ \mathbf{e})^T\mathbf{H}_\mathbf{d}$ selects and sums up the rows in $\mathbf{H}_{\mathbf{d},\mathbf{e}_j}$.

According to Eq. (5.2), the rows of $\mathbf{M}$ are linear combinations of rows of $\mathbf{G}_\mathbf{d}$ and thus are in the row space of $\mathbf{G}_\mathbf{d}$. ■

Property check

Next, we discuss the property checks designed to determine whether a vector in $\ker(\mathbf{G_d})$ can serve as a potential secret or not. In the context of the Shepherd-Bremner construction targeted in [KM19], the property check is to check whether $\mathbf{s}_i$ in $\ker(\mathbf{M})$ corresponds to a quadratic-residue code or not. To accomplish this, the prover constructs $\mathbf{H}_{\mathbf{s}_i}$ for the vector $\mathbf{s}_i$ and performs what we refer to as the QRC check, which examines whether $\mathbf{H}_{\mathbf{s}_i}$ generates a quadratic-residue code (with possible row reordering). However, determining whether a generator matrix generates a quadratic-residue code is a nontrivial task. Consequently, the algorithm in [KM19] attempts to achieve this by assessing the weight of the codewords in the code generated by $\mathbf{H}_{\mathbf{s}_i}$. In a quadratic-residue code, the weight of the codewords will be either 0 or 3 (mod 4). But still, there will be exponentially many codewords, and checking the weights of the basis vectors is not sufficient to ensure that all codewords have weight either 0 or 3 (mod 4). So in practice, the prover can only check a small number of the codewords.

For instances derived from the stabilizer construction, the prover will have less information about the code $\mathcal{C}_\mathbf{s}$; he only has the knowledge that this code has a large doubly-even subcode, as quantified by the rank of $\mathbf{G_s}$. Therefore, the property check for Meta-Algorithm 3 involves checking whether the rank of $\mathbf{H}_{\mathbf{s}_i}^T \mathbf{H}_{\mathbf{s}_i}$ falls below certain threshold and whether self-dual intersection $\mathcal{D}_{\mathbf{s}_i}$ is doubly-even. However, determining an appropriate threshold presents a challenge for the classical prover, who can generally only make educated guesses. If the chosen threshold is smaller that the rank of $\mathbf{G_s}$, then the secret extraction algorithm will miss the real secret, even if it lies within $\ker(\mathbf{G_d})$.

Stopping criteria

Lastly, various stopping criteria can be employed in the secret extraction procedure. One approach is to halt the procedure once a vector successfully passes the property check, as adopted in [KM19]. Alternatively, the procedure can be stopped after a specific number of repetitions or checks. In our implementation, we utilize a combination of these two criteria. If no vectors are able to pass the property check before the stopping criterion is reached, an empty candidate set S is returned, indicating a failed attack. Conversely, if the candidate set S is non-empty, the attack proceeds to the classical sampling step to generate classical samples.

5.1.2 Classical Sampling

Classical sampling based on multiple candidate secrets is nontrivial. Mathematically, the problem is formulated as follows.

Problem 5.3. *Given an IQP circuit C and a candidate set $S = \{\mathbf{s}_1, \cdots, \mathbf{s}_t\}$, outputs a sample $\mathbf{x}$ so that*

$$\mathbb{E}\left[(-1)^{\mathbf{x}\cdot\mathbf{s}_i}\right] = \left\langle \mathcal{Z}_{\mathbf{s}_i} \right\rangle , \tag{5.3}$$

for $i = 1, \cdots, t$, where $\mathbb{E}[\cdot]$ is over the randomness of the algorithm.

Note that $\mathbb{E}\left[(-1)^{\mathbf{x}\cdot\mathbf{s}_i}\right]$ is the expectation value of Eq. (3.2). We may allow a polynomially-bounded additive error in the problem formulation, considering the inevitable shot noise due to finite samples. The complexity of this problem depends on various situations. To the best of our knowledge, we are not aware of an efficient classical algorithm that solves this problem in general.

Single candidate secret

We first focus on the case $|S| = 1$, in which case the problem is easy to solve, yet remains worth discussing.

Naive sampling algorithm. A naive sampling algorithm is as follows. To generate samples with the correct correlation on $\mathbf{s}$, one just needs to output samples that are orthogonal to the candidate vector $\mathbf{s}'$ with probability $\beta_{\mathbf{s}'} = (\langle \mathcal{Z}_{\mathbf{s}'} \rangle + 1)/2$ and otherwise with probability $1 - \beta_{\mathbf{s}'}$. One can prove that if the candidate secret from the

ExtractSecret procedure is the real secret $\mathbf{s}$, then the generated samples using this strategy will have the correlation function approximately $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ with the real secret. Otherwise, the correlation function with the real secret will be zero. We have the following lemma.

Lemma 5.4. *Given a matrix* $\mathbf{H}$ *and two vectors* $\mathbf{s} \neq \mathbf{s}'$, *let* $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ *and* $\langle \mathcal{Z}_{\mathbf{s}'} \rangle$ *be their corresponding correlation functions, as defined in Eq. (3.3). If a sample* $\mathbf{x}$ *is generated to be a vector orthogonal to* $\mathbf{s}'$ *with probability* $\beta_{\mathbf{s}'} = (\langle \mathcal{Z}_{\mathbf{s}'} \rangle + 1)/2$ *and otherwise with probability* $1 - \beta_{\mathbf{s}'}$, *then* $\mathbb{E}\left[(-1)^{\mathbf{x}\cdot\mathbf{s}}\right] = 0$.

Before giving the proof, we first give some remarks. The above lemma holds even if $\mathbf{H}\mathbf{s} = \mathbf{H}\mathbf{s}'$, in which case $\mathbf{s}$ and $\mathbf{s}'$ are said to be *equivalent secrets*. Equivalent secrets have the same non-orthogonal and redundant part, and the correlation functions $\langle \mathcal{Z}_{\mathbf{s}} \rangle$ and $\langle \mathcal{Z}_{\mathbf{s}'} \rangle$ are the same. It is clear that the number of equivalent secrets is given by $2^{n-\mathrm{rank}(\mathbf{H})}$, which will be 1 if $\mathbf{H}$ is of full column rank. When there are multiple equivalent secrets, it could be the case that the vector $\mathbf{s}'$ is returned by the secret extraction procedure, because it can also pass the property check, even if it is not the real secret itself. In this case, our previous classical sampling algorithm can only give samples with zero correlation function on the real secret $\mathbf{s}$, according to Lemma 5.4.

Proof of Lemma 5.4. We first prove the following proposition.

Proposition 5.5. *For a nonzero* $\mathbf{s} \neq \mathbf{1}$, *if we randomly sample a vector* $\mathbf{d}$ *of even parity, then*

$$\Pr_{|\mathbf{d}|\ even}(\mathbf{s} \cdot \mathbf{d} = 1) = \Pr_{|\mathbf{d}|\ even}(\mathbf{s} \cdot \mathbf{d} = 0) = \frac{1}{2}\,. \tag{5.4}$$

Given $\mathbf{H} = \begin{pmatrix} \mathbf{H}_{\mathbf{s}} \\ \mathbf{R}_{\mathbf{s}} \end{pmatrix}$ and $\mathbf{s}$, if $\mathbf{s}'$ is equivalent to $\mathbf{s}$, then they have the same inner-product relations with rows in $\mathbf{H}$. The following lemma shows that a random row orthogonal to $\mathbf{s}'$ will have probability $1/2$ to have inner product 1 with $\mathbf{s}$, even if $\mathbf{H}\mathbf{s} = \mathbf{H}\mathbf{s}'$.

Lemma 5.6. *For* $\mathbf{s}' \neq \mathbf{s}$*, if we uniformly randomly sample a vector* $\mathbf{p}$ *orthogonal to* $\mathbf{s}'$*, then*

$$\Pr_{\mathbf{p}\cdot\mathbf{s}'=0}(\mathbf{p}\cdot\mathbf{s}=\mathbf{1}) = \frac{1}{2}\,. \tag{5.5}$$

Proof. Without loss of generality, assume $\mathbf{s}'$ has ones in the first k entries and zeros elsewhere. We can split $\mathbf{p} = \begin{pmatrix}\mathbf{p}_1\\ \mathbf{p}_2\end{pmatrix}$ and $\mathbf{s} = \begin{pmatrix}\mathbf{s}_1\\ \mathbf{s}_2\end{pmatrix}$, where $\mathbf{p}_1$ is a random even-parity string and $\mathbf{p}_2$ is uniformly random over $\mathbb{F}_2^{n-k}$. Then, $\mathbf{p}\cdot\mathbf{s} = \mathbf{p}_1\cdot\mathbf{s}_1 + \mathbf{p}_2\cdot\mathbf{s}_2$.

(1) If $\mathbf{s}_2 = \mathbf{0}$ and $\mathbf{s}_1 \neq \mathbf{1}$,

$$\Pr_{\mathbf{p}\cdot\mathbf{s}'=0}(\mathbf{p}\cdot\mathbf{s}=\mathbf{1}) = \Pr_{\mathbf{p}_1\text{ even}}(\mathbf{p}_1\cdot\mathbf{s}_1=\mathbf{1}) = \frac{1}{2}\,, \tag{5.6}$$

according to Proposition 5.5.

(2) If $\mathbf{s}_2 \neq \mathbf{0}$ and $\mathbf{s}_1 = \mathbf{1}$,

$$\Pr_{\mathbf{p}\cdot\mathbf{s}'=0}(\mathbf{p}\cdot\mathbf{s}=\mathbf{1}) = \Pr_{\mathbf{p}_2}(\mathbf{p}_2\cdot\mathbf{s}_2=\mathbf{1}) = \frac{1}{2}\,, \tag{5.7}$$

because $\mathbf{p}_2$ is uniformly random.

(3) If $\mathbf{s}_2 \neq \mathbf{0}$ and $\mathbf{s}_1 \neq \mathbf{1}$,

$$\Pr_{\mathbf{p}\cdot\mathbf{s}'=0}(\mathbf{p}\cdot\mathbf{s}=\mathbf{1}) = \Pr_{\mathbf{p}_1,\mathbf{p}_2}(\mathbf{p}_1\cdot\mathbf{s}_1=\mathbf{1},\mathbf{p}_2\cdot\mathbf{s}_2=\mathbf{0}) + \Pr_{\mathbf{p}_1,\mathbf{p}_2}(\mathbf{p}_1\cdot\mathbf{s}_1=\mathbf{0},\mathbf{p}_2\cdot\mathbf{s}_2=\mathbf{1}) \tag{5.8}$$

$$= \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{2}\,, \tag{5.9}$$

where we used the independence of $\mathbf{p}_1$ and $\mathbf{p}_2$.

■

Now, we are ready to prove Lemma 5.4.

Proof. With similar derivations to Lemma 5.6, one can show that

$$\Pr_{\mathbf{p}\cdot\mathbf{s}'=0}(\mathbf{p}\cdot\mathbf{s}=\mathbf{0}) = \Pr_{\mathbf{p}\cdot\mathbf{s}'=1}(\mathbf{p}\cdot\mathbf{s}=\mathbf{0}) = \Pr_{\mathbf{p}\cdot\mathbf{s}'=1}(\mathbf{p}\cdot\mathbf{s}=\mathbf{1}) = \frac{1}{2}\,. \tag{5.10}$$

That means, if one samples a random $\mathbf{p}$ to be orthogonal to $\mathbf{s}'$ with probability β, and not orthogonal to $\mathbf{s}'$ with probability $1 - \beta$, then

$$\Pr_{\mathbf{p}}(\mathbf{p} \cdot \mathbf{s} = 0) = \beta \Pr_{\mathbf{p} \cdot \mathbf{s} = 0}(\mathbf{p} \cdot \mathbf{s} = 0) + (1 - \beta) \Pr_{\mathbf{p} \cdot \mathbf{s} = 1}(\mathbf{p} \cdot \mathbf{s} = 0) = \frac{1}{2} . \tag{5.11}$$

That is, $\mathbf{p}$ is uncorrelated with $\mathbf{s}$ and the correlation function is

$$\mathbb{E}\left[(-1)^{\mathbf{p} \cdot \mathbf{s}}\right] = \Pr_{\mathbf{p}}(\mathbf{p} \cdot \mathbf{s} = 0) - \Pr_{\mathbf{p}}(\mathbf{p} \cdot \mathbf{s} = 1) = 0 . \tag{5.12}$$

Therefore, if the secret extraction procedure returns a vector $\mathbf{s}' \neq \mathbf{s}$ and the classical prover uses the naive classical sampling algorithm to generate samples, then the samples will produce zero correlation function on the real secret. ■

Sampling according to H. To address this issue, we propose a second classical sampling algorithm. Observe that linear combination of rows in $\mathbf{R}_\mathbf{s}$ gives vectors that are orthogonal to $\mathbf{s}$ and summation of an odd number of rows in $\mathbf{H}_\mathbf{s}$ gives vectors that are not orthogonal to $\mathbf{s}$. We denote the former set of vectors $\mathbb{S}_0(\mathbf{s})$ and the latter $\mathbb{S}_1(\mathbf{s})$. The identification of these sets relies on determining the submatrices $\mathbf{H}_\mathbf{s}$ and $\mathbf{R}_\mathbf{s}$. To achieve this, it suffices to find a vector $\mathbf{s}'$ that is equivalent to the real secret $\mathbf{s}$. Therefore, upon receiving the candidate secret $\mathbf{s}'$ from the secret extraction procedure, the classical prover proceeds by computing $\langle \mathcal{Z}_{\mathbf{s}'} \rangle$ and $\beta_{\mathbf{s}'}$, followed by identifying $\mathbb{S}_0(\mathbf{s}')$ and $\mathbb{S}_1(\mathbf{s}')$. A sample $\mathbf{x}$ is drawn from $\mathbb{S}_0(\mathbf{s}')$ with probability $\beta_{\mathbf{s}'}$ and from $\mathbb{S}_1(\mathbf{s}')$ with probability $1 - \beta_{\mathbf{s}'}$. If the vector $\mathbf{s}'$ is equivalent to $\mathbf{s}$, then $\mathbf{H}_\mathbf{s} = \mathbf{H}_{\mathbf{s}'}$ and $\mathbf{R}_\mathbf{s} = \mathbf{R}_{\mathbf{s}'}$. So, this sampling algorithm will generate samples with the correct correlation function with respect to the real secret $\mathbf{s}$, as opposed to the naive sampling algorithm.

This also explains why we consider IQP matrices of full column rank in the stabilizer construction. If the classical prover is given an IQP matrix $\mathbf{H}$ that is not full-rank, he can always apply an invertible matrix $\mathbf{Q}$ so that $\mathbf{HQ} = (\mathbf{H}', \mathbf{0})$, where $\mathbf{H}'$ is of full column rank. Then, he runs the secret extraction algorithm on $\mathbf{H}'$. Once a candidate secret is found, he can use it to identify the corresponding $\mathbb{S}_0$ and $\mathbb{S}_1$ from the original matrix $\mathbf{H}$, as well as computing the correlation function. Finally, if the identification matches that of the real secret, then using the second classical sampling algorithm will allow him to pass the test.

Multiple candidate secrets

Here, we give two classical sampling algorithms that given an IQP circuit C and a candidate set $S = \{\mathbf{s}_1, \cdots, \mathbf{s}_t\}$ with $t \leq n$ as input, output samples that have the correct correlation function on all candidate secrets in the set. We first consider a simple case here, where the Gram matrix $\mathbf{G}_{\mathbf{s}_i} = \mathbf{H}_{\mathbf{s}_i}^T \mathbf{H}_{\mathbf{s}_i}$ associated with each candidate secret $\mathbf{s}_i$ has the same rank. Then, if the samples are from a quantum computer, the probability bias relative to every candidate secret should be the same, denoted as β. Given this candidate set, a classical prover can use Algorithm 3 to mimic the quantum behavior. Here, the matrix $\mathbf{S}$ is defined to be a $t \times n$ matrix whose i-th row is $\mathbf{s}_i^T$. The output bit strings will have probability β to be orthogonal to all vectors in the candidate set S, and probability $1 - \beta$ to have inner product 1 with them. Therefore, the generated samples will have correct bias with every vector in the candidate set and hence the correct correlation function. The condition for Algorithm 3 to work is that the all-ones vector $\mathbf{1}$ needs to be in the column space of $\mathbf{S}$. Otherwise, the specific solution $\mathbf{y}'$ cannot be found. A sufficient condition is that the candidate vectors are linearly independent. Then, the matrix $\mathbf{S}$ will have full row rank, and the all-ones vector $\mathbf{1}$ will be in the column space of $\mathbf{S}$.

Next, we do not require the associated biases $\beta_1, \cdots, \beta_t$ to be the same. We present a similar sampling algorithm to Algorithm 3 to output samples that mimic what a quantum computer will output. For the sake of illustration, we assume that the associated biases are all different, denoted as $\{\beta_1, \beta_2, \cdots, \beta_t\}$, but the following discussion can be easily generalized to the case where some β_i's are the same. As before, the attacker does not have extra information to judge which one is the correct secret, even though the correct secret is in the candidate set. So he would have to generate samples that have bias β_1 with $\mathbf{s}_1$, β_2 with $\mathbf{s}_2$, and so on. Below, the algorithm for generating

```
Parameter: number of samples $T$.
1: procedure ClassicalSampling($\mathbf{S}, \beta$)
2:     Solve $\mathbf{Sy} = \mathbf{1}$ for a specific solution $\mathbf{y}'$.
3:     Find the basis $\{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$ of $\ker(\mathbf{S})$      ▷ $k$ is the dimension of $\ker(\mathbf{S})$.
4:     for $j = 1, 2, \cdots, T$ do
5:         Randomly sample $(c_1, \cdots, c_k) \in \mathbb{F}^k$.
6:         With probability $\beta$, set $\mathbf{x}_j \leftarrow \sum_{i=1}^k c_i \mathbf{y}_i$.
7:         With probability $1 - \beta$, set $\mathbf{x}_j \leftarrow \mathbf{y}' + \sum_{i=1}^k c_i \mathbf{y}_i$.
8:     end for
9:     return $\mathbf{x}_1, \cdots, \mathbf{x}_T$
10: end procedure
```

Algorithm 3: The ClassicalSampling subroutine for the candidate set where all secrets are associated with the same bias.

such samples is shown in Algorithm 4. Again, we transform the set S into a $t \times n$ matrix $\mathbf{S}$.

The correctness of the sampling algorithm can be easily seen via a sanity check. But one problem is whether the linear system $\mathbf{Sy} = \mathbf{b}_j$ has solutions or not. If nonzero solutions can be found for every linear systems, then $\mathbf{b}_j$'s are all in the column space of $\mathbf{S}$, which implies that the rank of $\mathbf{S}$ is t. Since there are t rows in $\mathbf{S}$, a necessary and sufficient condition for the sampling algorithm to work is that $\{\mathbf{s}_1, \cdots, \mathbf{s}_t\}$ are linearly independent. This condition can be relaxed if some of the β_j's are the same.

5.2 Analysis

Here, we present analysis on the secret extraction of Linearity Attack.

Probability of sampling a good d. First, we have the following proposition.

Proposition 5.7. *Given an IQP matrix* $\mathbf{H}$ *and two vectors* $\mathbf{d}$ *and* $\mathbf{s}$*, we have* $\mathbf{G_s d} = \mathbf{G_d s}$*, where* $\mathbf{G_s} = \mathbf{H_s}^T \mathbf{H_s}$ *and* $\mathbf{G_d} = \mathbf{H_d}^T \mathbf{H_d}$*. Therefore,* $\mathbf{s}$ *lies in* $\ker(\mathbf{G_d})$ *if and only if* $\mathbf{G_s d} = \mathbf{0}$*,*

```
**Input:** a binary matrix $\mathbf{S} \in \mathbb{F}^{t\times n}$; biases $\beta_1 > \beta_2 > \cdots > \beta_t$.
**Parameter:** number of samples $T$.
**Output:** $\mathbf{x}_1, \cdots, \mathbf{x}_T \in \mathbb{F}^n$.
 1: Find the basis of $\ker(\mathbf{S})$, denoted as $\{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$.
 2: for $j = 1, 2, \cdots, t$ do
 3:     Define $\mathbf{b}_j$ to be a binary vector whose last $j$ entries are all zero.
 4:     Solve a specific solution $\mathbf{y}'_j$ for $\mathbf{S}\mathbf{y} = \mathbf{b}_j$.
 5: end for
 6: for $i = 1, 2, \cdots, T$ do
 7:     Randomly sample $(c_1, \cdots, c_k) \in \mathbb{F}^k$.
 8:     With probability $\beta_t$, set $\mathbf{x}_i \leftarrow \sum_{i=1}^k c_i\mathbf{y}_i$.
 9:     With probability $\beta_{t-1} - \beta_t$, set $\mathbf{x}_i \leftarrow \mathbf{y}'_t + \sum_{i=1}^k c_i\mathbf{y}_i$.
10:     With probability $\beta_{t-2} - \beta_{t-1}$, set $\mathbf{x}_i \leftarrow \mathbf{y}'_{t-1} + \sum_{i=1}^k c_i\mathbf{y}_i$.
11:     ⋮
12:     With probability $\beta_1 - \beta_2$, set $\mathbf{x}_i \leftarrow \mathbf{y}'_2 + \sum_{i=1}^k c_i\mathbf{y}_i$.
13:     With probability $1 - \beta_1$, set $\mathbf{x}_i \leftarrow \mathbf{y}'_1 + \sum_{i=1}^k c_i\mathbf{y}_i$.
14: end for
15: return $\mathbf{x}_1, \cdots, \mathbf{x}_T$
```

Algorithm 4: The ClassicalSampling subroutine for the candidate set where all vectors are associated with different biases.

which happens with probability 2^{-g} *over all choices of* $\mathbf{d}$*, where* $g = \operatorname{rank}(\mathbf{G_s})$ *is the rank of* $\mathbf{G_s}$.

Proof. First, $\mathbf{G_s d} = \mathbf{H_s^T H_s d}$ and $\mathbf{H_s d}$ is a vector, where the positions of ones gives the indices of the rows in $\mathbf{H_s}$ that have inner product 1 with $\mathbf{d}$. Therefore, the ones of $\mathbf{H_s d}$ correspond to the rows in $\mathbf{H}$ that have inner product 1 with both $\mathbf{s}$ and $\mathbf{d}$. Moreover, if the vector $\mathbf{H_s d}$ is multiplied to $\mathbf{H_s^T}$ on the right, then those rows are summed up, i.e.,

$$\mathbf{G_s d} = \mathbf{H_s^T H_s d} = \sum_{\substack{\mathbf{p}^T \in \operatorname{row}(\mathbf{H}) \\ \mathbf{p}\cdot\mathbf{d} = \mathbf{p}\cdot\mathbf{s} = 1}} \mathbf{p}\,. \tag{5.13}$$

Similarly, we have

$$\mathbf{G_d s} = \mathbf{H_d^T H_d s} = \sum_{\substack{\mathbf{p}^T \in \operatorname{row}(\mathbf{H}) \\ \mathbf{p}\cdot\mathbf{d} = \mathbf{p}\cdot\mathbf{s} = 1}} \mathbf{p}\,. \tag{5.14}$$

Thus, $\mathbf{G_s d} = \mathbf{G_d s}$. If we want $\mathbf{s}$ to lie in $\ker(\mathbf{G_d})$, then $\mathbf{d}$ needs to lie in $\ker(\mathbf{G_s})$, which happens with probability

$$\frac{2^{n-g}}{2^n} = 2^{-g} , \tag{5.15}$$

for a random $\mathbf{d}$. ■

This proposition tells us that if the random $\mathbf{d}$ does not satisfy $\mathbf{G_s d} = \mathbf{0}$, then the verifier's secret $\mathbf{s}$ will not lie in $\ker(\mathbf{G_d})$. In this case, Meta-Algorithm 3 will not be able to find the correct secret from the kernel of $\mathbf{G_d}$, and it has to be started over with a new $\mathbf{d}$.

For completeness, we also give the success probability that the real secret $\mathbf{s}$ lies in $\ker(\mathbf{M})$ in Meta-Algorithm 4.

Proposition 5.8 (Theorem 3.1 in Ref. [KM19] restated). *Given* $(\mathrm{H}, \mathrm{s}) \in \mathcal{H}^{\mathrm{QRC}}_{n,m,q}$, *randomly sample a vector* $\mathbf{d} \in \{0,1\}^n$ *and let* $\mathbf{M}$ *be the binary matrix obtained from Meta-Algorithm 4. If* $\mathbf{G_s d} = 0$, *then we have* $\mathbf{Ms} = \mathbf{0}$, *which happens with probability* $1/2$ *over all choices of* $\mathbf{d}$.

Proof. First, note that for the i-th row of $\mathbf{M}$,

$$\mathbf{m}_i \cdot \mathbf{s} = \sum_{\substack{\mathbf{p}^T \in \mathrm{row}(\mathbf{H}) \\ \mathbf{p}\cdot\mathbf{d} = \mathbf{p}\cdot\mathbf{e}_i = 1}} \mathbf{p} \cdot \mathbf{s} = \sum_{\mathbf{p} \in \mathrm{row}(\mathbf{H})} (\mathbf{p}\cdot\mathbf{s})(\mathbf{p}\cdot\mathbf{d})(\mathbf{p}\cdot\mathbf{e}_i) , \tag{5.16}$$

since each term equals 1 if and only if it has inner product 1 with $\mathbf{d}$, $\mathbf{e}$ and $\mathbf{s}$ simultaneously. The above transformation is to take the conditions in the summation up to the summand, and we can take the term $\mathbf{p} \cdot \mathbf{s} = 1$ down to the summation. That is, we can write

$$\mathbf{m}_i \cdot \mathbf{s} = \sum_{\mathbf{p} \in \mathrm{row}(\mathbf{H_s})} (\mathbf{p}\cdot\mathbf{d})(\mathbf{p}\cdot\mathbf{e}_i) , \tag{5.17}$$

which can be seen to be a quantity only depending on $\mathbf{H_s}$. Further observe that the above is the inner product between $\mathbf{H_s d}$ and $\mathbf{H_s e}_i$, i.e.,

$$\mathbf{m}_i \cdot \mathbf{s} = (\mathbf{H_s e}_i) \cdot (\mathbf{H_s d}) = \mathbf{e}_i^T \mathbf{G_s d} . \tag{5.18}$$

Therefore, if $\mathbf{G_s d} = \mathbf{0}$, then $\mathbf{m}_i \cdot \mathbf{s} = 0$ for every i, which means $\mathbf{Ms} = \mathbf{0}$. That is, the verifier's secret lies in the kernel of $\mathbf{M}$ if $\mathbf{d}$ lies in the kernel of $\mathbf{G_s}$. If $\mathbf{H_s}$ generates a QRC, then $\operatorname{rank}(\mathbf{G_s}) = 1$. Then, the probability that $\mathbf{d}$ lies in the kernel of $\mathbf{G_s}$ is $2^{n-1}/2^n = 1/2$. ■

If the correlation function with respect to the real secret has inverse polynomial scaling, i.e., $2^{-g/2} = \Omega(1/\mathrm{poly}(n))$, then the probability of sampling a good $\mathbf{d}$ is also large, which is $2^{-g} = \Omega(1/\mathrm{poly}(n))$. This might appear advantageous for the attacker. But note that a classical attack cannot determine whether the sampled $\mathbf{d}$ is good or not before he can find the real secret. In fact, he even cannot *definitively* determine whether a vector $\mathbf{s}_i$ in $\ker(\mathbf{G_d})$ that passes the property check is the real secret or not.

Size of $\ker(\mathbf{G_d})$**.** The next question is, how large is the size of $\ker(\mathbf{G_d})$. This is important because the steps before the property check takes $O(n^3)$ time, which comes from the Gaussian elimination used to solve the linear system to find the kernel of $\mathbf{G_d}$. However, for the property check, the prover will potentially need to check every vectors in $\ker(\mathbf{G_d})$, which takes time proportional to its size. It is important to note that checking the basis vectors of $\ker(\mathbf{G_d})$ is not sufficient to find the real secret $\mathbf{s}$, because the linearity structure is not preserved under taking the Gram matrix. Even if $\mathbf{s} \in \ker(\mathbf{G_d})$, the basis vectors of the kernel space can have high ranks for their associated Gram matrices. Below, we give an expected lower bound for the size of $\ker(\mathbf{G_d})$.

Theorem 5.9. *Given* $(\mathbf{H}, \mathbf{s}) \in \mathcal{H}_{n,m,g}$*, randomly sample a vector* $\mathbf{d}$*. Then, the size of* $\ker(\mathbf{G_d})$ *is greater than* $2^{n-m/2}$ *in expectation over the choice of* $\mathbf{d}$*.*

Proof. First, observe that the rows in $\mathbf{G_d}$ are formed by linear combination of rows in $\mathbf{H_d}$, which means the rows space of $\mathbf{G_d}$ is no larger than that of $\mathbf{H_d}$, and $\operatorname{rank}(\mathbf{G_d}) \leq \operatorname{rank}(\mathbf{H_d})$. So, the dimension of $\ker(\mathbf{G_d})$ is

$$n - \operatorname{rank}(\mathbf{G_d}) \geq n - \operatorname{rank}(\mathbf{H_d}) \; . \tag{5.19}$$

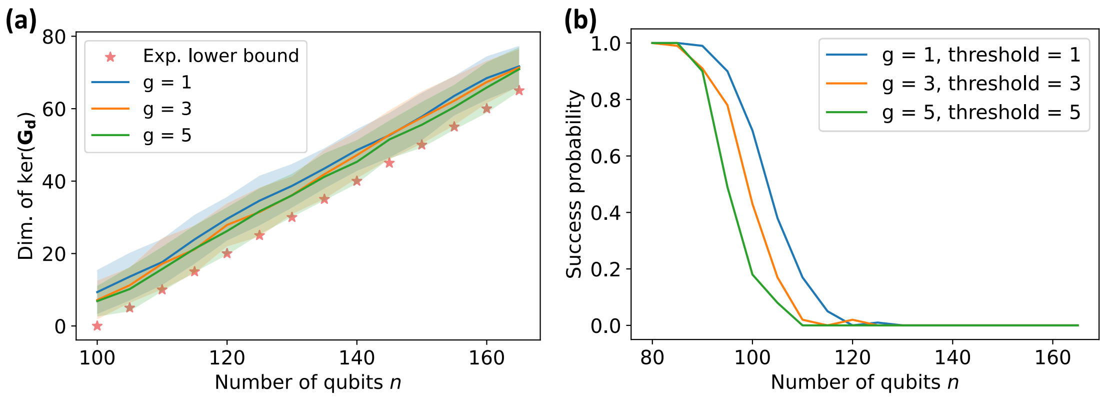


Figure 5.1: **(a)** The dimension of $\ker(\mathbf{G_d})$ for $g = 1, 3, 5$ and the number of rows $m = 200$. The asterisks indicate the expected lower bound $n - m/2$. **(b)** The success probability of the attack. Here, we set the threshold for the rank in the property check to be the same as g.

In expectation, the number of rows $r(\mathbf{H_d})$ in $\mathbf{H_d}$ is

$$\mathbb{E}_{\mathbf{d}}[r(\mathbf{H_d})] = \mathbb{E}_{\mathbf{d}}\left[\sum_{\mathbf{p}^T \in \mathrm{row}(\mathbf{H})} \mathbf{p} \cdot \mathbf{d}\right] = \sum_{\mathbf{p}^T \in \mathrm{row}(\mathbf{H})} \mathbb{E}_{\mathbf{d}}\left[\mathbf{p} \cdot \mathbf{d}\right] = \frac{m}{2}\,, \tag{5.20}$$

since $\mathbb{E}_{\mathbf{d}}\left[\mathbf{p} \cdot \mathbf{d}\right] = 1/2$ for every row $\mathbf{p}^T$. Since $\mathrm{rank}(\mathbf{H_d}) \le r(\mathbf{H_d})$, the number of rows in $\mathbf{H_d}$, we have $\dim(\ker(\mathbf{G_d})) \ge n - m/2$ in expectation. ■

Therefore, the size of $\ker(\mathbf{G_d})$ is increased exponentially by increasing n. The increase of n can be achieved by adding column redundancy, i.e., adding more all-zeros columns in Eq. (4.21). But in the stabilizer construction, the column redundancy cannot be arbitrarily large. Recall that to make the IQP matrix $\mathbf{H}$ full rank, one needs to add at least $n-r$ redundant rows, where $r = \mathrm{rank}(\mathbf{H_s})$. If $\mathbf{H}$ is not full rank, then as we discussed in Section 5.1.2, the classical prover can always perform column operations to effectively reduce the number of columns n, and hence reduce the dimension of $\ker(\mathbf{G_d})$.

Suggested parameter regime. Based on the above analysis, it is important to choose a good parameter regime to invalidate the Linearity Attack. Suppose the expected security parameter is λ, meaning that the expected time complexity of a classical prover is $\Omega(2^\lambda)$. Then, generally we require $n - m/2 \ge \lambda$ for $\ker(\mathbf{G_d})$ to be sufficiently large, and the number of redundant rows $m - m_1 \geq n - r$ for $\mathbf{H}$ to be full-rank, where m_1 is the number of rows in $\mathbf{H_s}$. Specifically, for the stabilizer construction, given n and g, we randomly choose the parameter $r \geq g$. Then, we require that the number of rows in $\mathbf{H_s}$ and $\mathbf{H}$ satisfies

$$m_1 \leq n - 2\lambda + r \qquad\qquad m_1 + n - r \leq m \leq 2(n - \lambda) \, , \tag{5.21}$$

respectively. In addition, since m is the number of gates in the IQP circuit, we will require sufficiently large n and $m = \Omega(n)$ to invalidate classical simulation.

Numerical simulation. In Fig. 5.1 (a), we plot the dimension of $\ker(\mathbf{G_d})$ for $g = 1, 3, 5$ and $m = 200$. For each number of columns n, we sample 100 instances from $\mathcal{H}_{n,m,g}$ with the stabilizer construction (Meta-Algorithm 1). Then, a random $\mathbf{d}$ is sampled and we calculate the dimension of $\ker(\mathbf{G_d})$. The asterisks are the expected lower bound $n - m/2$, as shown in Theorem 5.9. The numerical experiment demonstrates good agreement with the theoretical prediction. In Fig. 5.1 (b), we present the numerical results for the success probability of the attack. Although to invalidate the attack, the maximum number of property checks should be 2^{50} or larger, we set it to be 2^{15} for a proof of principle in the numerical experiment. For each number of columns n, we sample 100 random instances from $\mathcal{H}_{n,m,g}$, where $m = 200$. Then, the Linearity Attack is applied to each instance and the success probability is defined as the fraction of successfully attacked instances, which is the instance that the attacker can classically generate samples to spoof the test. As one can see, the success probability decreases to zero as n exceeds $m/2 + 15 = 115$, as expected.

Challenge. In addition, we have posted a challenge problem as well as the source code for generation and verification on GitHub [1], to motivate further study. The challenge problem is given by the $\mathbf{H}$ matrix of a random instance from $\mathcal{H}_{n,m,g}$ with $n = 300$ and $m = 360$; the g parameter is hidden because in practice, the prover can only guess

[1]https://github.com/AlaricCheng/stabilizer_protocol_sim

a value. One needs to generate samples with the correct correlation function in the direction of the hidden secret to win the challenge.

5.3 A Fix of the Shepherd-Bremner Construction

Finally, we would like to remark why the attack in [KM19] can break the Shepherd-Bremner construction and how we can fix it by adding column redundancy. Let $\mathcal{H}_{n,m,q}^{\mathrm{QRC}} = \{(\mathbf{H}, \mathbf{s})\}$ be a family of pairs of an IQP matrix $\mathbf{H} \in \mathbb{F}_2^{m\times n}$ and a secret $\mathbf{s}$ so that $\mathbf{H}_\mathbf{s}$ generates a QRC of length q (up to row permutations) and $\mathbf{H}$ is of full column rank. What the construction recipe of [SB09] does is to randomly sample instances from $\mathcal{H}_{n,m,q}^{\mathrm{QRC}}$, where $n = (q+3)/2$ and $m \geq q$, leaving a loophole for the recent classical attack [KM19]. To see why the parameter regime is as above, we first note that the length of QRC is q, implying that the number of rows in $\mathbf{H}_\mathbf{s}$ is q and hence $m \geq q$. Moreover, the dimension of a length-q QRC is $(q+1)/2$, which implies that the rank of $\mathbf{H}_\mathbf{s}$ is $(q+1)/2$. But an all-ones column was added in the construction (see Eq. (3.20)), which is a codeword of QRC, leading to $n = (q+3)/2$.

In the Shepherd-Bremner construction, the rank of Gram matrix $\mathbf{G}_\mathbf{s}$ associated with the real secret $\mathbf{s}$ is 1 according to Corollary 4.4. Therefore, the probability of choosing a good $\mathbf{d}$ is $1/2$ (as also shown in Theorem 3.1 of [KM19]). However, since the number of columns and the number of rows in $\mathbf{H}$ is $n = (q+3)/2$ and $m \geq q$, respectively, the size of $\ker(\mathbf{G}_\mathbf{d})$ is generally small. As a result, the prover can efficiently explore the entire $\ker(\mathbf{G}_\mathbf{d})$, and if no vector passes the property check, the prover can simply regenerate $\mathbf{d}$ and repeat the secret extraction procedure. The numerical results in [KM19] indicated that the size of $\ker(\mathbf{G}_\mathbf{d})$ is indeed constant when applied to the Shepherd-Bremner construction, which suggests that an efficient classical prover can pass the test and hence break the original construction. Specifically, for the challenge instance posted in [SB09], m is taken to be $2q$. Then, according to Theorem 5.9, the dimension of $\ker(\mathbf{G}_\mathbf{d})$ is expected to be constant, making it susceptible to the attack.

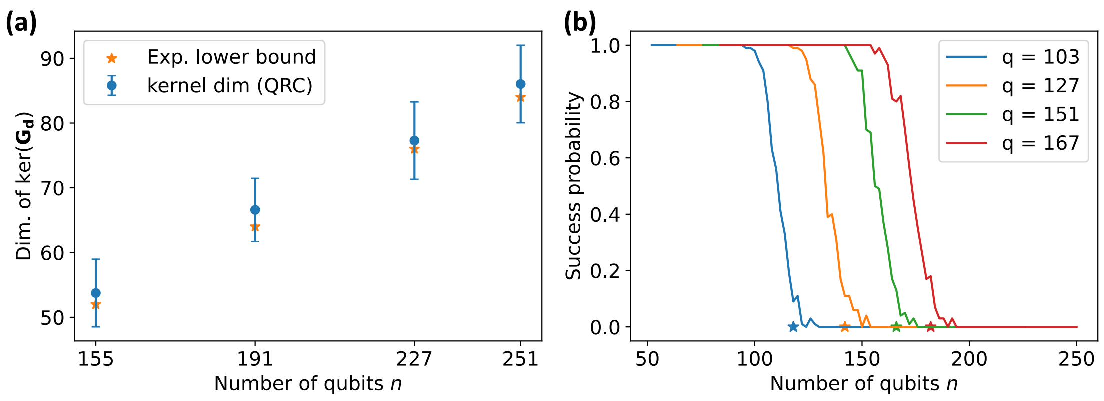


Figure 5.2: (a) The dimension of $\ker(\mathbf{G_d})$ for $q = 103, 127, 151, 167$. Here, the number of rows and columns are $m = 2q$ and $n = r + q$, where $r = (q+1)/2$ is the dimension of QRC. **(b)** The success probability of the attack. The asterisks denote the points $(q + 15, 0)$.

To address this issue, the original Shepherd-Bremner construction can be enhanced by introducing additional column redundancy to extend the number of columns n, which can achieve random sampling from families $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ with any $n \geq (q + 1)/2$ (Section 3.5). This hides the dimension information of the hidden QRC. Combined with other obfuscation techniques in the Shepherd-Bremner construction, this achieves random sampling from $\mathcal{H}^{\mathrm{QRC}}_{n,m,q}$ with any possible parameters.

Below, we propose a parameter regime that can invalidate the attack in [KM19]. Given the length q of the QRC, we have $r = (q+1)/2$ and $m_1 = q$ [MS77]. So, the first formula in Eq. (5.21) gives $n \geq (q - 1)/2 + 2\lambda$ and the second formula gives the range of the number of redundant rows $n - (q+1)/2 \leq m_2 \leq 2n - 2\lambda - q$. In this way, the size of $\ker(\mathbf{G_d})$ will be larger than 2^λ in general, offering a viable solution to fortify the Shepherd-Bremner construction against the attack. Note that the column redundancy technique was used in [YC20] to scramble a small random IQP circuit into a large one, to maintain the value of the correlation function, although its connection to the classical security was not explored. Moreover, a multi-secret version was explored in [Sno20], which was shown to be more vulnerable to the classical attack instead.

We perform numerical experiment to support our previous analysis. When $m = 2q$, n can be as large as $r + q$ and the expected kernel dimension of $\mathbf{G_d}$ is r. In Fig. 5.2 (a), we plot the kernel dimensions under the setting $n = r + q$ and $m = 2q$, with $q = 103, 127, 151$ and 167. For each parameter set, 100 instances are sampled from $\mathcal{H}_{n,m,q}^{\mathrm{QRC}}$, and then a random $\mathbf{d}$ is sampled for each instance and we evaluate the dimension of $\ker(\mathbf{G}_{\mathbf{d}})$. We also plot the expected lower bound $n - m/2$ for a comparison. In Fig. 5.2 (b), we plot the success probability versus the number of columns (qubits) n. Here, m is set to be $2q$ and n is increased from $r = (q + 1)/2$ to $r + q$. For each value of n, 100 random instances from $\mathcal{H}_{n,m,q}^{\mathrm{QRC}}$ are sampled, and the success probability is the fraction of successful attacks among them. We set the security parameter to be 15 for a proof of principle, meaning that the maximum number of QRC checks is set to be 2^{15}. The success probabilities drop down to zero when $n > q+15$, as expected. Our analysis and numerical results demonstrate that Claim 3.1 in [KM19], which originally states that the QRC-based construction can be broken efficiently by the KM attack, turns out to be false under appropriate choices of parameters.

Part II

Classical Enhancement of Quantum Devices

Chapter 6

A Machine Learning Approach to Quantum Error Mitigation

6.1 Overview

Quantum error mitigation is typically concerned with the scenario of denoising quantum expectation values, which is essential for quantum computing in the NISQ era. The first two techniques for quantum error mitigation are zero-noise extrapolation (ZNE) [TBG17, LB17] and probabilistic error cancellation (PEC) [TBG17]. Zero-noise extrapolation involves artificially increasing the noise level of a quantum circuit and then extrapolating the expectation value at the zero-noise point to estimate the expectation value in the absence of noise. This technique has been implemented on superconducting devices [KTC+19, KWY+23]. Probabilistic error cancellation, on the other hand, involves inverting well-characterized noise channels [TBG17]. Although the inverse noise channel is generally not a physically realizable channel, it can be implemented by writing the inverse channel as a quasiprobability decomposition of realizable channels, followed by classical post-processing to perform the combination. Subsequently, other quantum error mitigation techniques have been developed, including symmetry verification [BMSSO18, MYB19], purification methods [HMO+21, Koc21], and learning-based methods [CACC21, SQC+21].

Probabilistic error cancellation is of particular interest, since it has been shown to be optimal [Tak21, TEMG21]. However, implementing this technique presents several practical challenges. One challenge is accurately characterizing the noise model, which is still a difficult task even though techniques for noise characterization have been developed for localized Markovian errors [EBL18] and sparse Pauli-Lindblad noise models [BMKT22]. Additionally, it is desirable for the noise model to have a convenient inversion that incurs minimal sampling overhead. To overcome these challenges, [SQC+21] proposes to use a learning model to learn the quasiprobability coefficients. The coefficients are learned in a training set consisting of Clifford circuits, for which expectation values can be efficiently computed classically using the Gottesman-Knill algorithm [Got99, AG04]. However, there are an exponential number of coefficients to be learned. To address this issue, [SQC+21] suggested either truncating the configuration space to only keep low-weight configurations or using efficient representations.

In this chapter, we introduce the concept of neighborhood learning and demonstrate its potential as a quantum error mitigation technique. The basic idea behind neighborhood learning is to construct a list of the so-called "neighbor circuits" $C_1, \dots, C_k$ derived from a given circuit C and send neighbor circuits instead of the original circuit C to the quantum device. The average value for circuit C is then computed by combining the noisy average values of C_j for $j = 1, 2, \dots, k$ using a map named combine. That is, we need $\langle 0^n | C^\dagger O C | 0^n \rangle \approx$ combine($\langle 0^n | C_1^\dagger O C_1 | 0^n \rangle , \dots , \langle 0^n | C_k^\dagger O C_k | 0^n \rangle$), where O is the observable of interest. There are freedom in choosing the neighbor circuits and constructing post-processing map combine, which will be discussed in later sections.

As we will see, neighborhood learning is a very general framework and many existing quantum error mitigation schemes such as ZNE and PEC can be seen as special cases of this framework. We develop methods to construct the map combine by leveraging a training set consisting of circuits of the form $C = UV$, where U is a Clifford circuit and V is a low-depth general circuit. Such quantum circuits can be efficiently simulated. In addition, we also investigate technical issues such as the construction of the neighbor circuits and the choice of learning model. We also propose an adaptive learning strategy to learn the neighbor circuits. Our numerical results demonstrate that our proposed method achieves a better tradeoff between performance and required resources compared to various quantum error mitigation techniques.

6.2 Neighborhood Learning

Before delving into the discussion, we establish a convention that will be consistently used throughout this chapter. Specifically, we represent quantum channels of unitary circuits or gates using calligraphic font for the same letter representing the unitary circuit. For example, if we have a circuit denoted as C, the corresponding quantum channel will be denoted as $\mathcal{C}$ with $\mathcal{C}(\rho) = C\rho C^\dagger$.

6.2.1 General Framework

In the context of error mitigation, the setup involves a quantum circuit C, and the goal is to determine the average value of an observable O on the output state of the circuit C. For exmple, in various quantum algorithms such as QAOA [FGG14] and VQE [PMS+14, YCM+14], we are interested in computing the expectation value of O after the circuit is applied to a simple initial state ρ_{in}. This expectation value of interest is expressed as:

$$\langle C \rangle = \operatorname{Tr}\big(O\, \mathcal{C}(\rho_{\text{in}})\big). \tag{6.1}$$

Here, we do not explicitly include the dependence on ρ_{in} and O in the notation $\langle C \rangle$. In many cases, the initial state ρ_{in} is chosen to be $|0^n\rangle\langle 0^n|$ and O is some fixed Pauli operator.

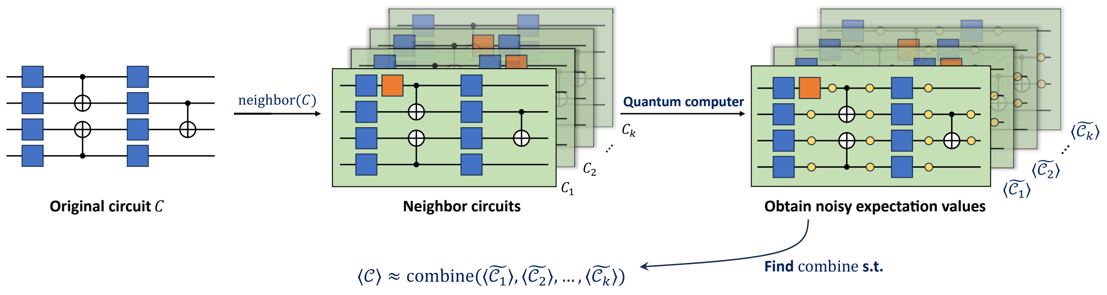

Original circuit C
neighbor(C)
Neighbor circuits
C_1
C_2
C_k
Quantum computer
Obtain noisy expectation values
$\langle\widetilde{C_1}\rangle$
$\langle\widetilde{C_2}\rangle$
$\langle\widetilde{C_k}\rangle$
Find combine s.t.
$\langle C\rangle \approx \text{combine}(\langle\widetilde{C_1}\rangle, \langle\widetilde{C_2}\rangle, \dots, \langle\widetilde{C_k}\rangle)$

Figure 6.1: Schematic for neighborhood learning. The single-qubit gates in the original circuit are represented by blue blocks, and the inserted gates are represented by orange blocks. The noisy expectation values $\langle\widetilde{C_j}\rangle$ are to be obtained on a real quantum device. We use yellow circles to represent the noise channels.

Suppose that the original quantum circuit C consists of m local gates: $C = U_m U_{m-1} \cdots U_1$, where U_j represents the j-th gate in the circuit C, acting on one or two qubits. In a real experiment, however, the circuit is affected by noise. Consequently, when we implement the quantum channel $\mathcal{U}_j$ in the circuit, the actual implementation could be a noisy version denoted as $\widetilde{\mathcal{U}}_j$. As a result, the overall noisy version of C implemented in the experiment is given by

$$\widetilde{C} = \widetilde{\mathcal{U}}_m \circ \widetilde{\mathcal{U}}_{m-1} \circ \cdots \circ \widetilde{\mathcal{U}}_1, \tag{6.2}$$

and what we can effectively compute with the quantum device is

$$\langle\widetilde{C}\rangle := \mathrm{Tr}\Big(O\,\widetilde{C}(\rho_{\mathrm{in}})\Big). \tag{6.3}$$

A simple noise model is that for each gate U_j the actual implementation is $\widetilde{\mathcal{U}}_j = \mathcal{E}_{\mathrm{dep}} \circ \mathcal{U}_j$, where $\mathcal{E}_{\mathrm{dep}}$ is the depolarizing channel

$$\mathcal{E}_{\mathrm{dep}}(\rho) = (1-\varepsilon)\rho + \frac{\varepsilon I}{2}. \tag{6.4}$$

But our discussion is not limited to such simple noise models and is designed to be much more flexible.

From the preceding discussion, our objective is to compute $\langle C\rangle$, but due to experimental noise, we obtain a noisy version denoted as $\langle\widetilde{C}\rangle$ from the quantum computing device. To address this, we propose the following approach: instead of sending a single circuit C to the quantum device, we send a collection of circuits $C_1, C_2, \dots, C_k$ that are different but related to the circuit of interest C. These circuits are specifically designed with modifications to help collectively sense the noise. We call this collection of circuits the *neighbor circuits* of C. The quantum device provides us with a list of numbers $\widetilde{\langle C_1 \rangle}, \widetilde{\langle C_2 \rangle}, \ldots, \widetilde{\langle C_k \rangle}$, the noisy average values for the circuit collection. Our next step involves classical post-processing, where we combine these noisy measurement outcomes to approximate $\langle C \rangle$ as

$$\langle C \rangle \approx \mathsf{combine}(\widetilde{\langle C_1 \rangle}, \ldots, \widetilde{\langle C_k \rangle}).$$

The challenge now lies in designing the function combine, which will effectively integrate the noisy outcomes obtained from the quantum device. Here, machine learning offers a promising solution. By providing a machine learning algorithm with a substantial number of examples, it can learn to perform the function combine effectively. To generate the required training dataset, we define a map neighbor that systematically generates neighbor circuits

$$\mathsf{neighbor}(C) = (C_1, \ldots, C_k).$$

This map takes the classical description of a circuit within a specific circuit architecture and computes the corresponding neighbor circuits.

To form the training data set, we choose a list of circuits $C^{(i)}$ for $i = 1, 2, \ldots, T$ and compute the neighbor circuits for $C^{(i)}$ as

$$\mathsf{neighbor}\big(C^{(i)}\big) = \big(C_1^{(i)}, \ldots, C_k^{(i)}\big).$$

These neighbor circuits are then sent to the quantum device to compute $\widetilde{\langle C_j^{(i)} \rangle}$ for $i = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, k$. We can now define the training set as $\big(x^{(i)}, y^{(i)}\big)$ for $i = 1, 2, \ldots, T$ where

$$x^{(i)} = \left(\widetilde{\langle C_j^{(i)} \rangle}\right)_{j=1}^{k}, \tag{6.5}$$

and

$$y^{(i)} = \langle C^{(i)} \rangle.$$

For this approach of constructing the training data to be effective, it is essential to efficiently compute the average value $\langle C^{(i)} \rangle$ for circuit $C^{(i)}$. A common and effective solution is to choose $C^{(i)}$ to be a Clifford circuit. Clifford circuits have the desirable property that their classical simulation can be performed efficiently, meaning we can efficiently compute the average value exactly [Got99, AG04].

We name the above general framework of error mitigation (illustrated in Fig. 6.1) neighborhood learning as it chooses a collection of neighbor circuits. Indeed, the neighborhood learning framework involves various technical details, including the design of the neighborhood map neighbor and the algorithms used for learning the function combine. These important discussions and analyses will be presented in Section 6.3.

6.2.2 Relation to Other Methods

The framework of neighborhood learning incorporates various quantum error mitigation techniques as special cases. For example, in zero-noise extrapolation, one deliberately increases the noise rate ϵ, obtains the noisy expectation values $\langle \widetilde{C} \rangle_\epsilon$ for different ϵ, and then extrapolates the result to the noiseless limit, where $\epsilon = 0$. Therefore, C_j is the same as the original circuit C, and its noisy version $\tilde{C}_k$ will have varying noise rates. The noise rates can be controlled by subcircuit repetitions [DMH+18, HNDJB20, GTHL+20], i.e., replacing U_j with $U_j(U_j U_j^\dagger)^l$. If the quantum circuit is noiseless, then the inserted subcircuit is effectively the identity gate. However, when the noise is present, the noise rate will be amplified by inserting such subcircuits. The noisy expectation values from the neighbor circuits are a function of the effective noise rate (or a function of l). Therefore, the map combine for ZNE is an extrapolation function, which can be chosen to be the polynomial extrapolation [TBG17] or exponential extrapolation [EBL18].

Probabilistic error cancellation can also be regarded as a special case. The concept of PEC involves finding an inverse of the noise channel $\mathcal{E}$, which can be mathematically expressed as a linear combination of physically realizable quantum channels. In fact, an arbitrary quantum channel can be expanded with a basis set of quantum channels [EBL18, Tak21]. For depolarizing channels given in Eq. (6.4), [TBG17] gives the decomposition as $\mathcal{E}_{\text{dep}}^{-1} = \eta_0\mathcal{I} + \eta_1\mathcal{X} + \eta_2\mathcal{Y} + \eta_3\mathcal{Z}$, where $\eta_0 = 1 + 3\varepsilon/4(1-\varepsilon)$ and $\eta_k = -\varepsilon/4(1-\varepsilon)$ for $k = 1, 2, 3$. Here, $\mathcal{I}, \mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ are the quantum channels for the identity, Pauli-X, Pauli-Y and Pauli-Z gate, respectively, and $\sum_k \eta_k = 1$. Let $j = (j_1, \ldots, j_m)$ and the neighbor circuit indexed by j is defined as follows,

$$C_j = (P_{j_m}U_m)\cdots(P_{j_1}U_1).$$

Here, after the ℓ-th gate U_ℓ a Pauli gate P_{j_ℓ} is inserted. P_{j_ℓ} either single-qubit or two-qubit Pauli gate, depending on the index j_ℓ. Specifically, $j_\ell \in \{0, 1, 2, 3\}$ represents $\{I, X, Y, Z\}$ if U_ℓ is a single-qubit gate and $j_\ell \in \{0, 1, 2, 3\}^2$ if U_ℓ is a two-qubit gate. Consequently, the length of the (unpacked) j is given by $m' = t + 2(m-t)$, where t is the number of single-qubit gates in C.

In PEC, the noisy expectation values $\langle\widetilde{C}_j\rangle$ of neighbor circuits are estimated on a real quantum computer. Suppose that $\mathcal{E}_\ell^{-1} = \sum_{j_\ell} \eta_{j_\ell}\mathcal{P}_{j_\ell}$ is expanded with Pauli channels, which is always possible for a Pauli noise channel $\mathcal{E}_\ell$. Then, we have

$$\mathcal{E}_\ell \circ \mathcal{E}_\ell^{-1} = \sum_{j_\ell} \eta_{j_\ell}\mathcal{E}_\ell \circ \mathcal{P}_{j_\ell} = \mathcal{I},$$

and $\sum_j \eta_j\widetilde{C}_j = C$, where $\eta_j = \eta_{j_m}\cdots\eta_{j_1}$. Therefore, the noise effect can be perfectly cancelled out in principle, and the ideal expectation value $\langle C\rangle$ can be obtained from the noisy expectation values $\langle\widetilde{C}_j\rangle$ as follows,

$$\sum_j \eta_j\langle\widetilde{C}_j\rangle = \text{Tr}\left(O\sum_j \eta_j\widetilde{C}_j(\rho_{\text{in}})\right) = \langle C\rangle. \tag{6.6}$$

In the language of neighborhood learning, the neighbor circuits are C_j and the function map combine is a linear function, with coefficients η_j.

To conclude, although the inverse $\mathcal{E}_\ell^{-1}$ may not be physically realizable, one can effectively apply $\mathcal{E}_\ell^{-1}$ by classically combining the noisy expectation values $\langle\widetilde{C}_j\rangle$ according to Eq. (6.6) to obtain the ideal expectation value $\langle C\rangle$. In this way, an effective inverse channel $\mathcal{E}_\ell^{-1}$ is applied after the ℓ-th quantum gate. Note that unlike ZNE, in PEC, the inserted gate P_{j_ℓ} is grouped into the original gate U_ℓ, and they are implemented together. An implicit assumption is made that the unitary channels $\mathcal{U}_\ell$ and $\mathcal{P}_{j_\ell} \circ \mathcal{U}_\ell$ will undergo the same noise process $\mathcal{E}_\ell$.

In practice, one can perform quasiprobability sampling instead of summing over the exponentially many j in Eq. (6.6). Define the negativity of the quasiprobability decomposition in Eq. (6.6) as the ℓ_1 norm of the coefficients $\Gamma := \sum_j |\eta_j|$. Then, one can rewrite Eq. (6.6) as,

$$\langle C \rangle = \Gamma \sum_j \frac{|\eta_j| \operatorname{sgn}(\eta_j)}{\Gamma} \langle \widetilde{C}_j \rangle = \Gamma \mathop{\mathbb{E}}_j \left[\operatorname{sgn}(\eta_j) \langle \widetilde{C}_j \rangle \right], \tag{6.7}$$

where the expectation is taken with respect to the underlying probability distribution given by $\{|\eta_j|/\Gamma\}$. One can estimate the expectation above by sampling the index j with probability $|\eta_j|/\Gamma$, and then compute the value of the random variable $\operatorname{sgn}(\eta_j)\langle \widetilde{C}_j \rangle$ with a quantum computer. According to the Chernoff bound, to estimate $\langle C \rangle$ to within additive error ϵ, one needs to sample $O(\Gamma^2/\epsilon^2)$ times. Therefore, the negativity Γ characterizes the cost of PEC, which however scales exponentially with the number of local noise channels in general. For the local depolarizing noise model, $\Gamma = \gamma^{nm'}$, where $\gamma = (1+\varepsilon/2)/(1-\varepsilon)$ is the negativity for the single-qubit depolarizing channel [TBG17] and m' is the number of depolarizing channels.

However, to implement PEC, one needs to learn the coefficients η_j's, which amounts to characterizing the noise model and is very challenging in practice. To circumvent this challenge, a learning-based variant of PEC was proposed in [SQC+21], which can also be described in our framework. They defined a significant error set SigE, which is used to truncate the set of j. One can choose SigE to be the set of j so that the inserted Pauli gates only act on a small number of qubits. Then, one chooses a linear function combine with coefficients β_j so that the error between

$$\langle C \rangle_{\text{t-pec}} = \sum_{j \in \mathrm{SigE}} \beta_j \langle \widetilde{C}_j \rangle \tag{6.8}$$

and the ideal expectation value $\langle C\rangle$ is minimized. The coefficients β_j's are the parameters to be learned, in order to minimize the error between $\langle C\rangle$ and $\langle C\rangle_{\text{t-pec}}$ over a family of quantum circuits (training set). One can choose the mean squared error (MSE) $\mathbb{E}_i\left[\langle C^{(i)}\rangle - \langle C^{(i)}\rangle_{\text{t-pec}}\right]^2$ to measure the distance, where $\mathbb{E}_i\,[\,\cdot\,]$ denotes the expectation over the training set circuits $C^{(i)}$. In order to efficiently evaluate the ideal expectation values $\langle C^{(i)}\rangle$, the training set circuits $C^{(i)}$ are chosen to be Clifford circuits.

6.3 Technical Details

In this section, we discuss the details of a neighborhood learning method missing from Section 6.2.1.

6.3.1 Test Set

The test circuits are the set of circuits that we want to evaluate the expectation values and mitigate the noise effect. Usually, we are interested in a family of quantum circuits with the same structure, instead of a single quantum circuit, which is common for near-term quantum algorithms such as variation quantum algorithms [PMS+14, FGG14]. In that case, one constructs an ansatz circuit $C(\boldsymbol{\theta}) = U_m(\theta_m)\cdots U_1(\theta_1)$ and varies the parameters $\boldsymbol{\theta}$ to minimize the expectation value of certain Hamiltonian H. The circuit structure, such as the location of the gate U_j, is fixed for the whole family. In the case of variational quantum circuits, the test circuits can be circuits $C(\boldsymbol{\theta})$ with randomly chosen parameters, or they can be generated on the fly during the classical optimization process. Regardless of the method used, the test circuit defines the circuit structure with respect to which the training circuits are constructed.

6.3.2 Training Set

From the discussion of the general framework, the construction of the training set is largely determined by the choice of circuits $C^{(i)}$. Given the choices $C^{(i)}$, one can generate the training set in the following way. First, we apply the neighborhood map neighbor to each $C^{(i)}$ obtaining $C_j^{(i)}$ for each $j = 1, 2, \ldots, k$. Second, we use the quantum device to compute all the mean values $\langle \widetilde{C_j^{(i)}} \rangle$ and form the feature vectors $x^{(i)}$ given in Eq. (6.5). Next, we use a classical algorithm to compute $\langle C^{(i)} \rangle$ and set it as the label $y^{(i)}$. We call the circuits $C^{(i)}$ the training circuits. We can see that in the process of generating the training set, we used the quantum devices to compute the feature $x^{(i)}$ and classical simulation algorithm to compute the label $y^{(i)}$.

A crucial requirement for a circuit to be a training circuit $C^{(i)}$ is that we need an efficient classical algorithm to compute $\langle C^{(i)} \rangle$. Previous works on learning-based quantum error mitigation usually employ purely Clifford circuits for this purpose [CACC21, SQC+21]. Under this choice, the ideal expectation values for the training circuit $C^{(i)}$ can only take discrete values in $\{0, \pm 1\}$ if the observable O is a Pauli operator. However, the original quantum circuit C can be general quantum circuits, and the ideal expectation can take general continuous values. Moreover, under the Pauli noise model, it can be proved that the noisy expectation value from the neighbor circuit $C_j^{(i)}$ is given by $\langle \widetilde{C_j^{(i)}} \rangle = \pm \langle \widetilde{C^{(i)}} \rangle$, if $C_j^{(i)}$ is obtained by inserting Pauli gates to $C^{(i)}$, a popular choice used in the neighborhood map.

Indeed, for $C = U_m \cdots U_1$, the noisy version of C_j is given by,

$$\widetilde{C_j} = \mathcal{E}_m \circ \mathcal{P}_m \circ \mathcal{U}_m \cdots \mathcal{E}_1 \circ \mathcal{P}_1 \circ \mathcal{U}_1,$$

where $\mathcal{E}_j$ is a Pauli noise channel, $\mathcal{U}_j$ is a Clifford gate, and $\mathcal{P}_j$ is a Pauli gate. First, considering $\mathcal{P}_m$, we have $\mathcal{E}_m \circ \mathcal{P}_m = \mathcal{P}_m \circ \mathcal{E}_m$ if $\mathcal{E}_m$ is a Pauli noise channel. In this way, $\mathcal{P}_m$ is moved to the end of the circuit. Then, consider $\mathcal{P}_{m-1}$, which commutes with $\mathcal{E}_{m-1}$. After that, when it commutes through $\mathcal{U}_m$, it becomes another Pauli operator, since U_m is a Clifford gate; that is, $U_m P_{m-1} \rho P_{m-1} U_m^\dagger = P'_{m-1} U_m \rho U_m^\dagger P'_{m-1}$, where P'_{m-1} is also a Pauli operator. Then, the quantum channel $\mathcal{P}'_{m-1}$ commutes with $\mathcal{E}_m$ and is moved to the end of the circuit, which is combined with $\mathcal{P}_m$ to become another Pauli operator. Repeating this process, we move all the inserted Pauli gates to the end of the circuit, resulting in

$$\widetilde{C}_j = \mathcal{Q} \circ \mathcal{E}_m \circ \mathcal{U}_m \cdots \mathcal{E}_1 \circ \mathcal{U}_1,$$

where $\mathcal{Q}$ is the resulting Pauli channel. For the noisy expectation value $\langle \widetilde{C}_j \rangle$, we have,

$$\begin{aligned}
\langle \widetilde{C}_j \rangle &= \operatorname{Tr}\big(O\, \widetilde{C}_j(\rho_{\text{in}})\big) \\
&= \operatorname{Tr}(O\, \mathcal{Q} \circ \mathcal{E}_m \circ \mathcal{U}_m \cdots \mathcal{E}_1 \circ \mathcal{U}_1(\rho_{\text{in}})) \\
&= \operatorname{Tr}\Big((Q^\dagger O Q)\, \mathcal{E}_m \circ \mathcal{U}_m \cdots \mathcal{E}_1 \circ \mathcal{U}_1(\rho_{\text{in}})\Big) \\
&= \mu(Q, O) \operatorname{Tr}\big(O\, \widetilde{C}(\rho_{\text{in}})\big) \\
&= \mu(Q, O) \langle \widetilde{C} \rangle,
\end{aligned}$$

where $\mu(Q, O)$ is 1 if Q and O commute and -1 if they anti-commute.

Therefore, when the neighbor circuits are obtained by inserting Pauli gates and when the noise is modelled by Pauli noise channels, the noisy expectation values from the training circuit $C^{(i)}$ and the neighbor circuit $C_j^{(i)}$ are either the same or the opposite. Consequently, all data points in the training set are of the form $(a, \pm a, \pm a, \cdots)$ with $a = 0, \pm 1$. This creates a significant discrepancy between the training set and the test set and could potentially hinder the learning process.

A possible alternative method is to consider circuits of the form $C^{(i)} = U^{(i)} V^{(i)}$. where $U^{(i)}$ is a Clifford circuit and $V^{(i)}$ can be a general low-depth quantum circuit. If we further suppose that the observable O is a Pauli operator, or a sum of polynomially many Pauli operators, then the expectation value $\langle C^{(i)} \rangle$ can be classically computed. Indeed, suppose for simplicity that O is a Pauli operator. Then, the ideal expectation value is given by

$$\langle C^{(i)} \rangle = \operatorname{Tr}\Big(O\, \mathcal{U}^{(i)} \circ \mathcal{V}^{(i)}(\rho_{\text{in}})\Big) = \operatorname{Tr}\Big(O'\, \mathcal{V}^{(i)}(\rho_{\text{in}})\Big),$$

where $O' := \big(U^{(i)}\big)^\dagger O U^{(i)}$ is also a Pauli operator by the definition of Clifford circuits. In this case, computing expectation values from the quantum circuit $C^{(i)}$ is reduced to computing that from a low-depth quantum circuit $V^{(i)}$, and the expectation value $\langle C^{(i)} \rangle$ can be classically computed by tensor network methods [MS08]. To make the tensor network simulation algorithm efficient, the depth of $V^{(i)}$ should be chosen to be $O(\log n)$.

6.3.3 Neighborhood Map

The way to construct the neighbor circuits is to perturb the original circuit by inserting or replacing a set of gates in C with gates from certain gate set. Inserting a gate P after the j-th gate is to replace U_j with PU_j, and leave the rest of the circuit unchanged, while gate replacement is to simply replace U_j with P.

We consider two different gate sets. One is the Pauli gate set, $\mathbb{G}_0 := \{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$. We call the neighbor scheme constructed by inserting (or replacing) gates from $\mathbb{G}_0$ as *Pauli insertion (or replacement) neighbor*. The other is the reduced set of basis for single-qubit CPTP (completely positive and trace-preserving) maps, denoted as $\mathbb{G}_1$. It is comprised of the basis for the single-qubit CPTP maps, with the identity channel and the 3 state-preparation channels removed [Tak21]. Specifically,

$$\mathbb{G}_1 := \left\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{K}^\dagger \mathcal{S}^\dagger \mathcal{K}, \mathcal{K} \mathcal{S}^\dagger \mathcal{K}^\dagger, \mathcal{S}^\dagger, \mathcal{K} \mathcal{H} \mathcal{K}^\dagger, \mathcal{H}, \mathcal{K}^\dagger \mathcal{H} \mathcal{K}\right\}, \tag{6.9}$$

where $\mathcal{S}, \mathcal{H}$ and $\mathcal{K}$ are the quantum channels for the S gate, the H gate and the $K := SH$ gate, respectively. We call the neighbor scheme constructed by inserting (or replacing) gates from $\mathbb{G}_1$ as *CPTP-basis insertion (or replacement) neighbor*. Note that all gates from $\mathbb{G}_0$ or $\mathbb{G}_1$ are single-qubit Clifford gates and $\mathbb{G}_0 \subseteq \mathbb{G}_1$.

Truncation. In the full PEC expansion (see Eq. (6.6)), the number of neighbor circuits is exponential in the number of noise positions. [SQC⁺21] proposed a truncation strategy that only keeps the low-weight neighbor circuits. A weight-ℓ neighbor circuit is constructed by inserting ℓ gates in ℓ positions of the original circuit (one for each position). For a quantum circuit C with m possible inserted positions (or m positions of noise), a weight-1 strategy gives a set of neighbor circuits of size $O(m)$, while a weight-2 strategy gives a set of size $O(m^2)$. For an intermediate-sized quantum circuit, it is already impractical to run the full weight-2 neighbor circuits. For example, for a quantum circuit with 50 qubits and 50 layers, the number of noise positions m is of the order 10^3, which means the number of weight-2 neighbor circuits will be the order of 10^6. It would be favorable to construct the set neighbor circuits of size sublinear in m.

Using knowledge of the noise model. In addition, one can also use the knowledge of the noise model to construct the neighborhood map $\mathsf{neighbor}$. First, we obtain a rough estimate of the noise model, which, for example, can be achieved by performing gate-set tomography or the learning protocol in [BMKT22]. Then, we work out its inverse map and get the associated quasiprobabilities. Given the number of neighbor circuits as a parameter, we perform quasiprobability sampling to generate the neighbor circuits. Then, instead of setting the function map $\mathsf{combine}$ as in Eq. (6.7), we choose the appropriate form of the function and learn the concrete map on the training set.

This strategy could be advantageous. For example, suppose that on the inverse map of the real noise channel, the neighbor circuits are of weight 3 on average. In this case, choosing a weight-1 or weight-2 neighbor might not work well, but this randomization strategy will output weight-3 neighbor circuits with high probability.

6.3.4 Learning Algorithms

An important design choice in neighborhood learning for error mitigation is the model and algorithm used for representing and learning the function $\mathsf{combine}$. We consider two choices here, including linear regression and neural networks.

Given a list of neighbor circuit C_j for $j = 1, 2, \ldots, k$, the linear regression method postulates that the function $\mathsf{combine}$ takes the following form

$$\mathsf{combine}\left(\langle \widetilde{C_1} \rangle, \ldots, \langle \widetilde{C_k} \rangle\right) = \sum_{j=1}^{k} a_j \langle \widetilde{C_j} \rangle + b,$$

As for the neural network, we consider two kinds of all-connected networks in the numerical experiment, $K \to K \to 1$ and $K \to 2K \to 1$. Here, $K \to K \to 1$ represents a two-layer network, where the hidden layer has the same number of neurons as the input layer, and the output layer has one neuron. The activation function is taken to be tanh, and we do not add activation function to the output layer. The structure of the other network is defined similarly.

6.3.5 Comparison of Different Settings

Here, we use QAOA circuits for Max Cut as our testbed. Given a graph $G = (V, E)$, the Max Cut problem asks for a cut that partitions the vertices of the graph into two disjoint subsets, such that the number of edges that cross the cut is maximized. Formally, one needs to find the maximum eigenvalue and the corresponding eigenstate of the following Hamiltonian,

$$H_C = \frac{1}{2} \sum_{(j,k)\in E} (I - Z_j Z_k) \, . \tag{6.10}$$

A p-level QAOA circuit is defined by,

$$C(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \prod_{j=1}^{p} e^{-i\beta_j H_B} e^{-i\gamma_j H_C} \, , \tag{6.11}$$

where $H_B = \sum_{k=1}^{|V|} X_k$, $\boldsymbol{\gamma} := (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_p)$. Each local gate is either of the form $e^{-i\beta X}$ or $e^{-i\gamma Z_j Z_k}$, where the latter gate can be further decomposed into two CNOT gates and one Z rotation gate. In our numerical experiments, the graph is a random 3-regular graph with 4 nodes, and the QAOA circuit has 2 levels. The noise model is set to be local depolarizing noise with strength $\varepsilon = 0.01$ for single-qubit gates and $\varepsilon = 0.1$ for two-qubit gates.

Training circuits. We first compare the two choices of training circuits. The purely Clifford training circuits are generated by replacing all the single-qubit gates in the test circuits (which are either X or Z rotations) with randomly selected Clifford gates. In

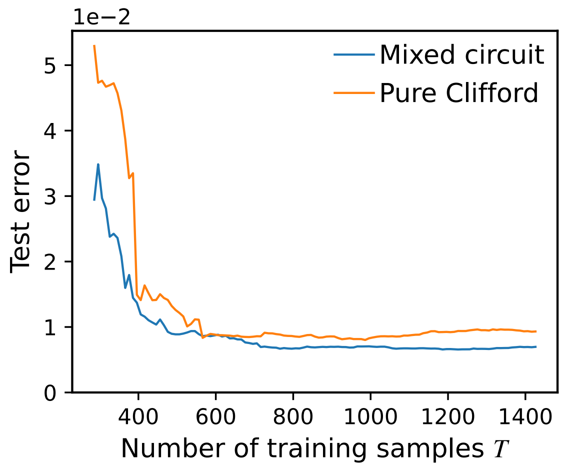
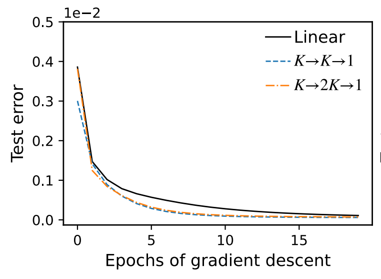


Figure 6.2: Comparison of choices of training circuits, which are circuits of the form $C = UV$ or purely Clifford circuits. The former is labelled as 'mixed circuit' while the latter is labelled as 'pure Clifford'.

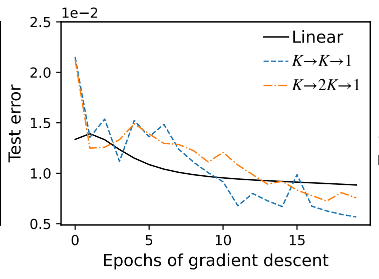
contrast, the training circuits of the form $C = UV$ are constructed by initially assigning random angles to the first few layers of gates. Subsequently, we replace the single-qubit gates in the remaining layers with random Clifford gates.

The neighbor circuits are obtained by inserting up to 3 random Pauli operators at random positions of the training circuit, forming the weight-(0, 1, 2, 3) neighbor circuits. Here, all weight-0 and weight-1 neighbor circuits are included, while 200 weight-2 and weight-3 neighbor circuits are randomly sampled. The total number of neighbor circuits (or features) under this construction is $k = 554$. The reason for restricting the number of weight-(2, 3) neighbor circuits is because the full number of weight-2 and weight-3 neighbor circuits for the system size in our numerical experiment is already of order 10^3 and 10^4, respectively. It would be too daunting to include all of them in practice. Finally, the learning algorithm combine is chosen to be linear regression.

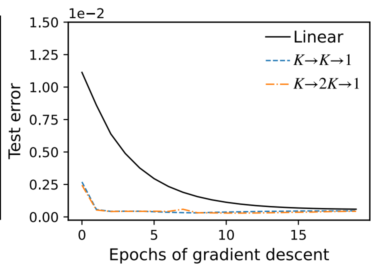
As depicted in Fig. 6.2, a substantial reduction in test error is observed for the training circuits structured as $C = UV$ compared to the purely Clifford circuits, when the training set size remains modest. Here, the test error is the MSE between the mitigated expectation value and the ideal expectation value on the test set. The size



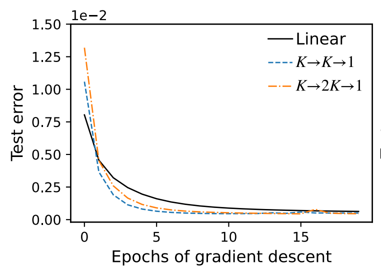
(a) Pauli insertion



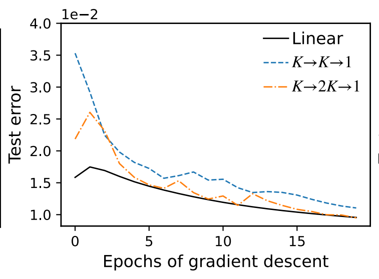
(b) Pauli replacement



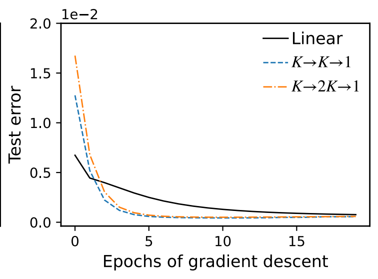
(c) Pauli insertion (weight-1)



(d) CPTP-basis insertion



(e) CPTP-basis replacement



(f) CPTP-basis insertion (weight-1)

Figure 6.3: Performance for different neighborhood maps and learning algorithms.

of training set is varied from k to $3k$. As the size of training set T increases, the test error associated with purely Clifford training circuits decreases quickly, although it continues to remain larger than that observed in the case of $C = UV$.

Neighbor circuits. Next, we compare the performance of the four neighborhood maps. As before, the test circuits are taken to be QAOA circuits illustrated in Section 6.3.1. The training circuits are of the form $C = UV$, since it outperforms the purely Clifford circuits. The neighbor circuits are also constructed to be the same weight-(0, 1, 2, 3) neighbor circuits as in the numerical simulation of Fig. 6.2.

The results are shown in Fig. 6.3 (a)-(b) and (d)-(e). We observe that the insertion neighbors outperform the replacement neighbors and the Pauli insertion neighbor outperforms the CPTP-basis insertion neighbor, regardless of the learning algorithms. This suggests that the Pauli insertion neighbor is a good choice for the neighborhood map.

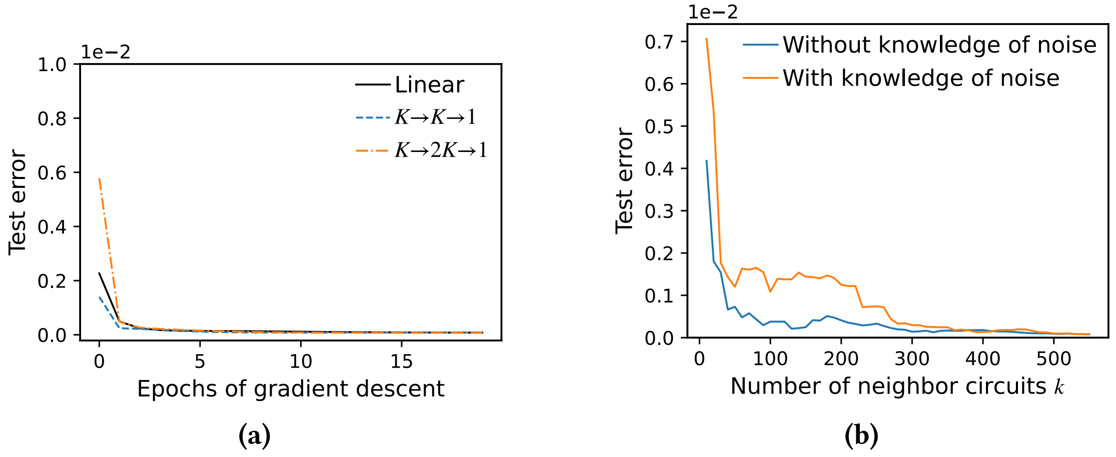


Figure 6.4: Performance of constructing the map neighbor with the knowledge of the noise model

In Fig. 6.3 (c) and (f), we present similar result with only weight-0 and weight-1 neighbor circuits. We did not present the results from replacement neighbors, since their performance is worse than the insertion neighbors. Here, the number of neighbor circuits (features) is 154 for the Pauli insertion neighbor and 460 for the CPTP-basis insertion neighbor. Consistent with previously observed, the Pauli insertion neighbor still outperforms the CPTP-basis insertion neighbor in this case. Moreover, the Pauli insertion neighbor with weight-(0, 1) neighbor circuits achieves a slightly worse performance compared to the same neighbor scheme with weight-(0, 1, 2, 3) neighbor circuits, despite using less features in the learning.

In addition, we also test the performance of constructing the map neighbor with the knowledge of the noise model (Fig. 6.4 (a)). Here, the number of neighbor circuits is set to be $k = 1000$. To model the imperfection in the noise characterization, we tweak the noise rates in the learned noise model by a small Gaussian-distributed inaccuracy. This choice of neighborhood map achieves test errors in the order of 10^{-5} in terms of MSE.

However, we also note that this result is actually similar to the result in Fig. 6.3 (a), which uses about half the number of neighbor circuits. There, all weight-0 and weight-1 neighbor circuits are used and then 200 weight-2 and weight-3 neighbor circuits are

```
Input: Pool of neighbor schemes $\mathbb{N}_0$; training set $\mathbb{T}$ of Clifford circuits
Output: $k$ learned neighbor schemes
1: Initialization: Draw the first $k$ schemes from $\mathbb{N}_0$
2: repeat
3:     Learn the coefficients for the $k$ schemes on the training set $\mathbb{T}$
4:     Inheritance: Retain the schemes with coefficients exceeding the critical
       value $c$.
5:     Mutation: Replace remaining schemes by drawing new schemes from
       the pool.
6: until Stopping conditions are met
```

Algorithm 5: Adaptive learning strategy for constructing neighbor schemes

randomly sampled, without using the knowledge of the noise model. In Fig. 6.3 (b), we compare the performance of learning with and without the knowledge of the noise model. We observe that the test error of learning with random weight-(0, 1, 2, 3) neighbor circuits decreases more quickly with the number of neighbor circuits. This may be due to the fact that under the system size and noise level in our numerical experiment, weight-(0, 1) neighbor circuits are more important and MSE of order 10^{-5} has already reached the limit of quantum error mitigation.

Learning algorithms. In Fig. 6.3, we also compare the performance of different learning algorithms. The training set size T is set to be $5k$, where k is the number of features. For insertion neighbor schemes, neural networks slightly outperform linear regression, although the difference is small. In terms of MSE, the test error associated with neural networks is about $2/3$ of that associated with linear regression.

6.4 Adaptive Learning Strategy

Based on the above observations, a natural question is find resource-efficient but effective neighbor schemes for error mitigation. One of the collaborators in our work [XCL+23], Lei Xie proposed an adaptive learning strategy for constructing the neighbor schemes (Algorithm 5). The strategy operates on the principles of an evolutionary algorithm. We first initialize a set of k neighbor schemes. Then, at each iteration, we retain pivotal neighbor schemes and replace the others based on certain criteria. The outlined procedure is depicted in Algorithm 5.

Specifically, we begin by constructing a pool of neighbor schemes. Based on previous observations, this pool can be constructed from low-weight Pauli-insertion neighbor schemes. Assuming that an output of k neighbor schemes is targeted, we draw k schemes from the pool and train the mapping function $\mathsf{combine}$ on the training data set. We opt for a linear model for $\mathsf{combine}$ due to its resource efficiency while delivering performance on par with neural networks in neighborhood learning. In this way, each neighbor scheme is associated with a coefficient in the linear combination. Subsequently, for coefficients exceeding a certain threshold c in absolute value, the corresponding neighbor schemes are retained for the next round, a process termed *inheritance*. Otherwise, we replace it with a new scheme from the pool, which is referred to as *mutation*. In this way, the number of neighbor schemes is kept to be k. This procedure is repeated until certain stopping conditions are met. The stopping conditions can be one of the following: (1) when no schemes undergo replacement in the iteration, (2) when the MAE metric on the training set exhibits no further decrease over a series of iterations, or (3) when the MAE metric falls below a predetermined threshold, signifying the achievement of the desired error mitigation scheme performance.

In addition, we decide whether to retain or discard a scheme according to the absolute value of its associated coefficient in the learned map $\mathsf{combine}$. In a linear map, the absolute coefficient value signifies the significance of the corresponding element. Consequently, a more crucial scheme will exhibit a coefficient with a larger absolute value. Moreover, instead of setting a fixed critical value c, we dynamically set it to be the MAE metric on the training set for the current iteration. This is due to the following two reasons. Firstly, the MAE metric serves as a direct reflection of an error mitigation scheme's effectiveness. Secondly, in a linear map, the coefficient and the

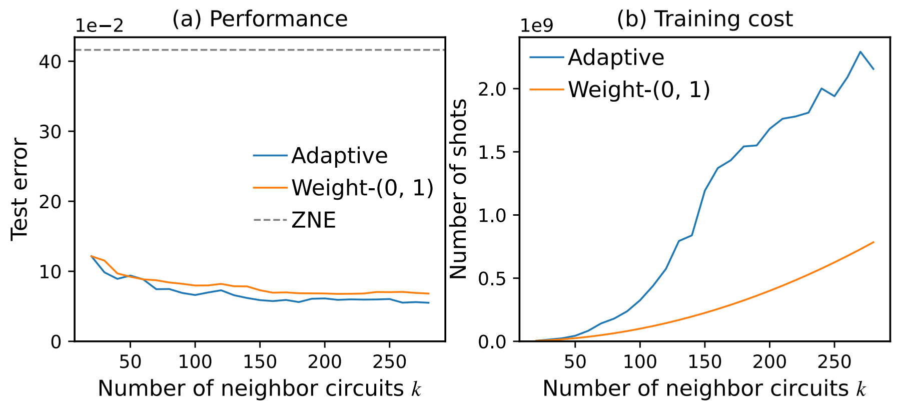


Figure 6.5: Comparison of performance and training cost between adaptive learning, weight-(0,1) neighbors and ZNE.

MAE metric are of the same order of magnitude. Thus, the MAE metric serves as a good choice of the critical value for evaluating a neighbor scheme's merit.

In order for the neighborhood learning protocol to be resource-efficient, we want to construct neighbor so that the number of the output neighbor circuits k is sublinear in the number of weight-1 neighbor circuits, while achieving a comparable performance. Here, we set the pool to consist of all weight-(0, 1, 2) neighbor schemes. In Fig. 6.5 (a), we compare adaptive learning with using only weight-0 and weight-1 neighbors. The quantum circuits are 5-qubit QAOA circuits with 3 levels and the expectation values are obtained from 1000 shots. The result from the adaptive learning strategy is shown as the blue line and that from weight-(0, 1) neighbors is shown as the orange line. The performance of ZNE is also shown as a reference. We observe that neighbor learning significantly outperforms ZNE and that adaptive learning achieves a better performance than using only weight-(0, 1) neighbors. Fig. 6.5 (b) shows the number of shots consumed during the training phase. The cost of ZNE is not presented since it does not require training. It should be noted that the cost of neighborhood learning is quite huge, which is an intrinsic drawback of learning-based error mitigation.

6.5 Discussion

In this chapter, we have presented a general framework for learning-based quantum error mitigation, named neighborhood learning. We proposed a new construction of training circuits, that is more advantageous than the purely Clifford circuits used in previous works. To make this framework practical, we numerically compared different settings and found that the Pauli insertion neighbor is a good choice for the neighborhood map. Surprisingly, while employing neural networks in our numerical simulations, their advantages compared to linear models are somewhat constrained, which could be attributed to the relatively small system size within our numerical setting. Furthermore, we present an adaptive learning strategy for constructing neighbor schemes, which achieves better performance than weight-(0, 1) neighbor circuits and significantly outperforms the ZNE technique. This highlights the potential of the adaptive learning strategy as a flexible and promising approach to achieving an optimal balance between performance and resource utilization. However, as an intrinsic limitation of learning-based error mitigation, the cost for training is large, since it requires constructing a large training set and running quantum circuits for all training data points. In the future, it is worth exploring whether efficient classical representation, such as classical shadow tomography [HKP20], can benefit the neighborhood learning framework and lower the cost.

Chapter 7

Experimental Circuit Cutting

7.1 Overview

Circuit-cutting aims at solving large problems with smaller quantum devices, with a tradeoff of using more classical resources. A related technique is to decompose a large problem into smaller subproblems, each of which is solved by a small quantum computer. Examples include quantizing classical divide-and-conquer algorithms to solve combinatorial optimization problems [DGC18, GD20], and Fujii *et al*'s deep variational quantum eigensolver framework [FMU+22], which is suitable for simulating physical systems when interactions between subsystems are weak. Partially quantizing a tensor network may also fall into this category [LZWW19, YSL+21].

In contrast, the circuit-level schemes intend to decompose a large quantum circuit into smaller pieces, implement each piece independently and finally use classical computers to combine the results. For example, Bravyi *et al*. [BSS16] discussed methods of using classical postprocessing to add virtual qubits for sparse circuits and Pauli-based computation. Mitarai and Fujii [MF21] proposed a method to add virtual two-qubit gates, which means that a remote two-qubit gate can be simulated by a quasiprobability decomposition of local single-qubit gates, thus cutting the large quantum circuit. Their work is for general quantum circuits and has been extended in a recent work [MF20] to allow decomposing non-local quantum channels into local ones. On the other hand, using the language of tensor network, Peng *et al*. [PHOW20] proposed a tomography-like circuit-cutting scheme, which is endowed with a rigorous analysis of the required quantum and classical resources to simulate general quantum circuits. The circuit-cutting scheme is further analyzed and improved in later works [PSSO21, ALRS+20, ARS+21, TTL+21].

In this chapter, we experimentally demonstrated a circuit-cutting method for simulating quantum circuits involving many logical qubits, using only a few physical superconducting qubits. By exploiting the symmetry of linear-cluster states, we can estimate the effectiveness of circuit-cutting for simulating up to 33-qubit linear-cluster states, using at most 4 physical qubits for each subcircuit. Specifically, for the 12-qubit linear-cluster state, we found that the experimental fidelity bound can reach as much as 0.734, which is about 19% higher than a direct implementation on the same 12-qubit superconducting processor. Our results indicate that circuit-cutting represents a feasible approach of simulating quantum circuits using much fewer qubits, while achieving a much higher circuit fidelity.

7.2 Cutting Large Quantum Circuits

The basic idea is to cut a qubit wire and then simulate the propagation of quantum information by classical means. We illustrate this with a toy example in Fig. 7.1 (a). First, observe that at the time slice of the cutting point (the red cross), the reduced density matrix of the first two qubits can be decomposed as,

$$\rho^{ab} = \frac{1}{2}\sum_{j=0}^{3} \mathrm{Tr}_b(\rho^{ab}\sigma_j^b) \otimes \sigma_j^b \ . \tag{7.1}$$

where we use superscripts to indicate the qubit labels and $\sigma_j \in \{I, X, Y, Z\}$. Each Pauli operator can be further decomposed into its eigenstates, e.g., $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$. Note that the identity operator can be written as $I = |0\rangle\langle 0| + |1\rangle\langle 1|$, which can be combined with Z [PSSO21]. Then, we have,

$$\rho^{ab} = \sum_{i=1}^{6} c_i \, \mathrm{Tr}_b(\rho^{ab} O_i^b) \otimes \rho_i^b \, , \tag{7.2}$$

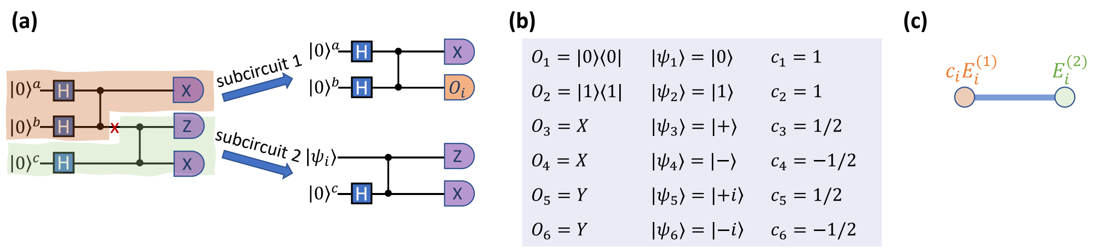
where the c_i, O_i, and $\rho_i := |\psi_i\rangle\langle\psi_i|$ are listed in Fig. 7.1 (b). At this point, the physical meaning of the above formula becomes clear. To simulate the original circuit, we first cut it into two subcircuits. The partial trace operation in Eq. (7.2) can be interpreted as measuring O_i in the qubit b of subcircuit 1, and then ρ_i is prepared and passed as input to subcircuit 2.

Suppose that we are interested in measuring the expectation of $X \otimes Z \otimes X$ of the 3-qubit circuit, denoted as $\langle XZX\rangle$. In subcircuit 1, one needs to collect the expectation values of $X \otimes O_i$, defined by $E_i^{(1)} = \mathrm{Tr}\big(\rho^{ab} X \otimes O_i\big)$. In subcircuit 2, one needs to collect the expectation values of $Z \otimes X$, denoted as $E_i^{(2)}$, from circuits with varying initial state $|\psi_i\rangle$ in the first qubit (see Fig. 7.1 (a)). We denote the expectation values from subcircuit 2 as $E_i^{(2)}$. Then, according to Eq. (7.2), $\langle XZX\rangle$ can be recovered by [PHOW20]

$$\langle XZX\rangle = \sum_{i=1}^{6} c_i E_i^{(1)} E_i^{(2)} \, . \tag{7.3}$$

This circuit-cutting procedure works for any observable in the form $A \otimes B$, where A is an observable of the qubit a and B is an observable of the qubits b and c. We remark that the combination of expectation values is achieved with a classical computer. In this process, we do not create a 3-qubit entangled state; instead, the 3-qubit state is simulated by a hybrid scheme of a 2-qubit quantum computer and a classical computer.

For more general and larger quantum circuits, one can apply this cutting scheme iteratively to multiple cutting points, to partition the whole circuit into several *disconnected* pieces of subcircuits. By running the subcircuits *independently*, and classically combining the subcircuit expectations with appropriate coefficients, one obtains expectations from the large quantum circuits. Moreover, those disconnected subcircuits can be viewed as nodes in a tensor network [PHOW20]. For example, the correspond-



$O_1 = \lvert 0\rangle\langle 0\rvert$	$\lvert\psi_1\rangle = \lvert 0\rangle$	$c_1 = 1$
$O_2 = \lvert 1\rangle\langle 1\rvert$	$\lvert\psi_2\rangle = \lvert 1\rangle$	$c_2 = 1$
$O_3 = X$	$\lvert\psi_3\rangle = \lvert +\rangle$	$c_3 = 1/2$
$O_4 = X$	$\lvert\psi_4\rangle = \lvert -\rangle$	$c_4 = -1/2$
$O_5 = Y$	$\lvert\psi_5\rangle = \lvert +i\rangle$	$c_5 = 1/2$
$O_6 = Y$	$\lvert\psi_6\rangle = \lvert -i\rangle$	$c_6 = -1/2$

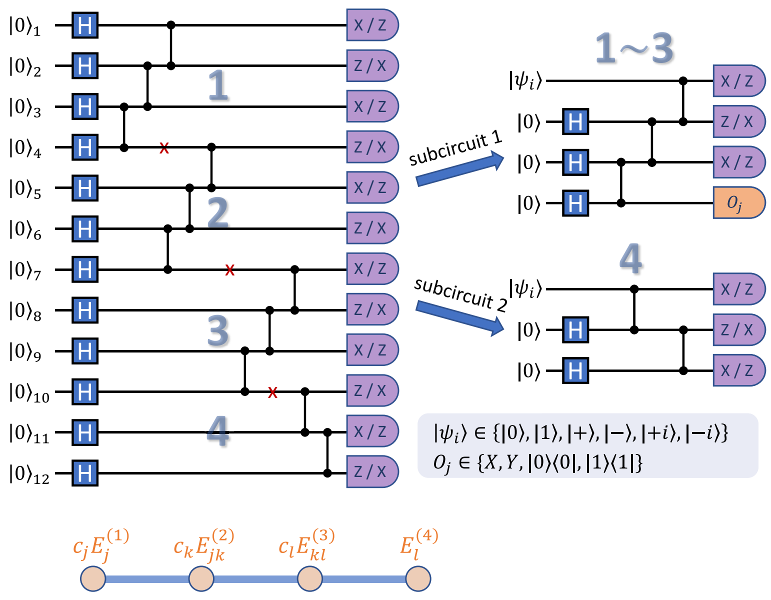
Figure 7.1: (a) Example illustrating the circuit-cutting scheme. The quantum circuit on the left is cut at the red cross, and partitioned into two subcircuits. The original circuit can be simulated by combining the quantum measurement statistics of two subcircuits in different basis and of different inputs. **(b)** The list of c_i, O_i and ρ_i for Eq. (7.2). **(c)** The tensor network representing the summation of Eq. (7.3).

ing tensor network for Fig. 7.1 (a) is a line with two nodes, and the edge has bond dimension 6, corresponding to the 6 terms in Eq. (7.3). The coefficient c_i can be absorbed into the node representing $E_i^{(1)}$ or $E_i^{(2)}$; in Fig. 7.1 (c), we absorb it into $E_i^{(1)}$. Then, one can use tensor-network contraction to perform the combination to obtain quantities of the large circuit, with classical running time exponential in the treewidth of the tensor network [PHOW20].

To summarize, the protocol is as follows. *(a)* Identify appropriate cutting points to partition the large circuit into disconnected subcircuits. *(b)* Obtain the subcircuit expectations by enumerating the possible choices of $|\psi_i\rangle$ and O_i. *(c)* Construct a tensor network from these subcircuit expectations and coefficients c_i. *(d)* Contract the tensor network to obtain the expectation value with respect to the large circuit.

7.3 Linear-Cluster States

Cluster states are a family of highly-entangled states, which can be used to achieve measurement-based quantum computation [RB01, Nie06]. That is, universal quantum computation can be performed by only making measurements on the cluster state. Linear-cluster state is a specific example of cluster states, where all qubits are aligned



$|0\rangle_1$ … $|0\rangle_{12}$
H
X/Z
Z/X
1
2
3
4
subcircuit 1
subcircuit 2
1~3
$|\psi_i\rangle$
$|0\rangle$
O_j
$|\psi_i\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+i\rangle, |-i\rangle\}$
$O_j \in \{X, Y, |0\rangle\langle 0|, |1\rangle\langle 1|\}$
$c_j E_j^{(1)}$
$c_k E_{jk}^{(2)}$
$c_l E_{kl}^{(3)}$
$E_l^{(4)}$

Figure 7.2: Circuit-cutting scheme for the linear-cluster state. *Left.* A 12-qubit linear-cluster state, which is cut into 4 pieces. *Right.* The 12-qubit linear-cluster state can be simulated by combining measurement data from these two types of subcircuits. *Bottom.* The tensor network representing the classical combination of subcircuits.

in one dimension. Explicitly, a linear-cluster state with n qubits can be expressed as,

$$|\mathrm{LC}_n\rangle = \left(\prod_{i=1}^{n-1} \mathrm{CZ}^{i,i+1} \right) |+\rangle^{\otimes n} \ , \tag{7.4}$$

where the superscripts in the CZ gates indicate the qubits that they act on.

In [YCZ+23], we experimentally simulate a 12-qubit linear-cluster state, with 4 qubits of a superconducting quantum processor, which is the same processor as in [GCZ+19]. As in Fig. 7.2, there are 3 cutting points on the 12-qubit circuit, partitioning it into 4 subcircuits. The first 3 subcircuits are all in the form of subcircuit 1, while the last subcircuit is in the form of subcircuit 2. The reuse of measurement data (or expectation values) from subcircuit 1 is due to the symmetry of linear-cluster states. Note that the sequence of CZ gates on the left of Fig. 7.2 is chosen such that pieces 1-3 can be represented by the same subcircuit 1.

To compare the performance of the circuit cutting scheme with that of running the 12-qubit circuit directly, we need to estimate their fidelities. We follow the approach in [GCZ+19], which uses techniques from entanglement detection in the stabilizer formalism [TG05, GT09]. Let $s_1 = X_1 Z_2$, $s_n = Z_{n-1} X_n$ and $s_i = Z_{i-1} X_i Z_{i+1}$ for $i \neq 1$ or n. It can be shown that a linear-cluster state is a stabilizer state with a stabilizer group spanned by $\{s_1, \cdots, s_n\}$, i.e., $s_i |\mathsf{LC}_n\rangle = |\mathsf{LC}_n\rangle$ for $i = 1, \cdots, n$. Let

$$\mathsf{ODD}_n := \prod_{i \text{ odd}} \frac{1+s_i}{2} \qquad \qquad \mathsf{EVEN}_n := \prod_{i \text{ even}} \frac{1+s_i}{2} \,. \tag{7.5}$$

For a linear-cluster state, one has $|\mathsf{LC}_n\rangle\langle\mathsf{LC}_n| \geq \mathsf{ODD}_n + \mathsf{EVEN}_n - I$ [TG05, Theorem 6]. Therefore, for an unknown quantum state ρ, its fidelity relative to the linear-cluster state is lower bounded by,

$$\mathrm{Tr}(\rho\, |\mathsf{LC}_n\rangle\langle\mathsf{LC}_n|) \geq \mathrm{Tr}(\rho \mathsf{ODD}_n) + \mathrm{Tr}(\rho \mathsf{EVEN}_n) - 1 \,, \tag{7.6}$$

which can be estimated by measuring ODD_n and EVEN_n. Observe that every term in the expansion of ODD_n can be measured in the basis $XZXZ\cdots$, while every term in the expansion of EVEN_n can be measured in the basis $ZXZX\cdots$. Therefore, to estimate the fidelity, one only needs to perform measurements in two bases. For simplicity, we will refer to them as XZ measurement and ZX measurement, respectively.

Below, we illustrate how to simulate the 12-qubit linear-cluster state with the circuit-cutting scheme. Suppose we want to obtain the expectation value $\langle P^{(1)} \otimes P^{(2)} \otimes P^{(3)} \otimes P^{(4)} \rangle$, where $P^{(i)}$ can be any 3-qubit observable of the i-th 3-qubit group. We take $P^{(1)} = XZI, P^{(2)} = ZIZ, P^{(3)} = IZX$ and $P^{(4)} = ZXZ$ as an example, whose expectation value can be obtained from the XZ measurement in the 12-qubit circuit. As shown in Fig. 7.2, the 12-qubit circuit is cut into 4 pieces, where pieces 1-3 can be represented by subcircuit 1, and the last piece can be represented by subcircuit 2. In Fig. 7.2, we denote the final state of subcircuit 1 and 2 as $\left|\Phi_{1,i}\right\rangle$ and $\left|\Phi_{2,i}\right\rangle$, respectively, where the index i indicates one of the 6 states $|\psi_i\rangle$ in the first qubit. Then, define

$$\begin{aligned} E^{(1)}_j &:= \langle \Phi_{1,3} | P^{(1)} \otimes O_j | \Phi_{1,3} \rangle & \qquad E^{(2)}_{jk} &:= \langle \Phi_{1,j} | P^{(2)} \otimes O_k | \Phi_{1,j} \rangle \\ E^{(3)}_{kl} &:= \langle \Phi_{1,k} | P^{(3)} \otimes O_l | \Phi_{1,k} \rangle & \qquad E^{(4)}_l &:= \langle \Phi_{2,l} | P^{(4)} | \Phi_{2,l} \rangle \end{aligned} \tag{7.7}$$

to be the subcircuit expectations from pieces 1-4. According to the circuit-cutting scheme, we have,

$$\langle P^{(1)} \otimes P^{(2)} \otimes P^{(3)} \otimes P^{(4)} \rangle = \sum_{j,k,l=1}^{6} c_j c_k c_l E_j^{(1)} E_{jk}^{(2)} E_{kl}^{(3)} E_l^{(4)} \,, \tag{7.8}$$

where the coefficients c_i's are shown in Fig. 7.1 (b). Again, this summation can be viewed as tensor network contraction as in the bottom of Fig. 7.2.

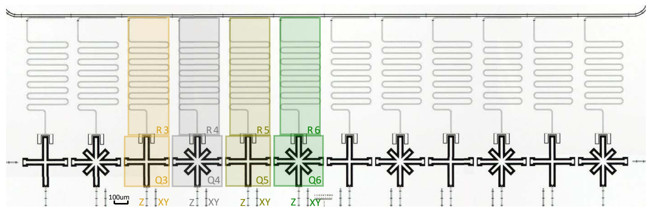
To compute $\mathrm{Tr}(\rho\mathrm{ODD})$ and $\mathrm{Tr}(\rho\mathrm{EVEN})$, one needs to first expand ODD and EVEN, and then apply Eq. (7.8). In our experiment, we used the fidelity lower bound as a measure, which requires measuring the observables involved in the expansion of ODD_{12} and EVEN_{12}. The experimental procedure for estimating the fidelity is as follows. (*a*) Identify the observables in the expansion of ODD_{12} and EVEN_{12}. (*b*) For each observable, define $P^{(i)}$ for $i = 1, 2, 3, 4$. Measure $E_j^{(1)}$, $E_{jk}^{(2)}$, $E_{kl}^{(3)}$ and $E_l^{(4)}$, and use Eq. (7.8) to obtain the expectation value of that observable. (*c*) Calculate the fidelity lower bound according to Eq. (7.6). Note that this procedure can be easily generalize to larger linear-cluster states. For example, to simulate a 15-qubit linear-cluster state, one only needs to add one more cutting point on the 13-th qubit.

The expectation values in Eq. (7.8) can be obtained from subcircuits in Fig. 7.2. For subcircuit 1, we need to prepare the circuits with 6 different $|\psi_i\rangle$. The measurement bases for the first three qubits are XZX and ZXZ, and for the last qubit are X, Y or Z; the expectation value of $|0\rangle\langle 0|$ or $|1\rangle\langle 1|$ can be obtained from Z measurement. Therefore, we need to implement $6 \times 2 \times 3 = 36$ different circuits in the form of subcircuit 1. As for subcircuit 2, similar argument shows that we need to implement 12 different circuits. Thus, a total of 48 subcircuits needs to be run.

7.4 Experiment

To verify the feasibility and evaluate the actual performance of the scheme in the experiment, we run the subcircuits in Fig. 7.2 on a 12-qubit superconducting quantum processor. As shown in Fig. 7.3 (a), the qubits are arranged in a one-dimensional chain.

(a)



(b)

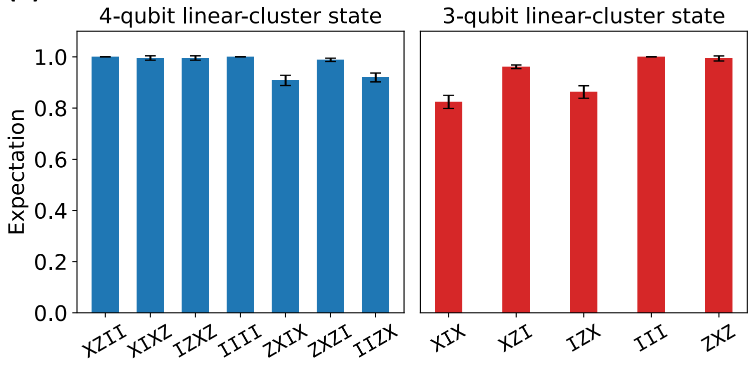


Figure 7.3: (a) Schematic of the 12-qubit superconducting processor, where we used Q3 to Q6 for the circuit-cutting experiment. **(b)** Expectation from XZ and ZX measurements of the 4-qubit and 3-qubit linear-cluster states. Ideal values are one. The error bars are due to the repeated experiments.

Each qubit has two control lines to provide full control of the qubit: a microwave XY control line to drive excitations between $|0\rangle$ and $|1\rangle$, and a magnetic flux bias line to tune the qubit resonance frequency. As the near-neighbor qubits are capacitively coupled, the fast adiabatic CZ gates [BKM+14, MG14] can be applied. The measurements of qubit are done through dispersively coupling to a readout resonator. We choose four adjacent qubits from a 12-qubit superconducting quantum processor to implement the experiments. The average performance of the chosen qubits are: $T_1 \approx 36.1$ μs, $T_2^* \approx 4.3$ μs, single-qubit gate fidelity $\approx 99.93\%$ and CZ gate fidelity $\approx 98.5\%$. More detailed data can be found in the Supplemental Material of [YCZ+23].

All the experimental results are processed using the transition matrix error mitigation (TMEM) method [BSK+21, Gel21], to suppress the readout noise. However, negative entries may appear in the probability distributions of the subcircuits after the TMEM. To make these distributions physical, we first transform those probability

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
P_{the}^{XZ}	0.25	0	0	0	0	0	0	0.25	0	0	0	0.25	0.25	0	0	0
P_{exp}^{XZ}	0.245	0	0	0	0	0	0	0.274	0.002	0	0	0.213	0.266	0	0	0.000
P_{the}^{ZX}	0.25	0	0	0.25	0	0	0	0	0	0	0	0	0	0.25	0.25	0
P_{exp}^{ZX}	0.226	0.000	0.000	0.248	0	0.004	0.009	0	0.024	0	0	0.003	0.005	0.245	0.235	0

Table 7.1: Theoretical and experimental distributions of 4-qubit linear-cluster state in the XZ measurement basis and ZX measurement basis

	000	001	010	011	100	101	110	111
P_{the}^{XZ}	0.5	0	0	0	0	0	0	0.5
P_{exp}^{XZ}	0.444	0.010	0	0.026	0.043	0	0.010	0.468
P_{the}^{ZX}	0.25	0	0	0.25	0	0.25	0.25	0
P_{exp}^{ZX}	0.247	0	0	0.291	0.003	0.217	0.241	0

Table 7.2: Theoretical and experimental distributions of 3-qubit linear-cluster state in the XZ measurement basis and ZX measurement basis

distributions into diagonal operators, and then use the maximum likelihood method to find a density operator that is the closest to them [MZO20, JKMW05]. The final distributions of the subcircuits are then extracted from these density operators. Before and after the experiment of circuit cutting, additional quantum state tomography on the final state of the circuit is performed to evaluate the performance of the experiments. The average fidelity of the 36 subcircuits in the form of subcircuit 1 is 0.944, and the average fidelity of the 12 subcircuits in the form of subcircuit 2 is 0.955, showing the high quality of the experiments.

As a warm-up, we show how to estimate the fidelity lower bounds for the 4-qubit and 3-qubit linear-cluster states. Note that if we take $|\psi_i\rangle = |+\rangle$ for the subcircuits, then they correspond to a 4-qubit and 3-qubit linear-cluster state, respectively. The concrete values for the average distributions of these two states in the XZ and ZX measurement bases are shown in Table 7.1 and Table 7.2 (the right-most bit is identified as the first bit). The probability distributions are also visualized in Fig. 7.4 (a) and (b). Here, each cell represents one bit string (after the basis transformation) and the number gives the corresponding output probability. In the 4-qubit case, the bit strings for the first row are 0000, 0001, 0010, 0011, and for the second row are 0100, 0101,

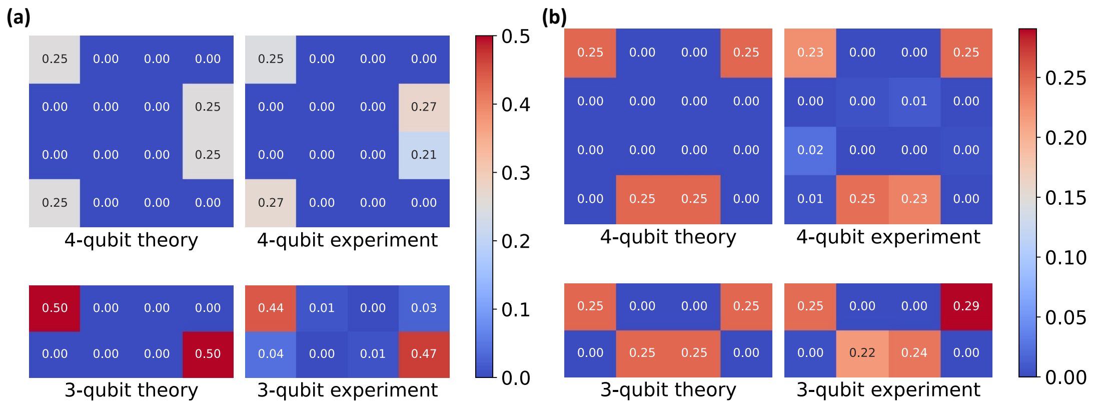


Figure 7.4: (a) The output distributions from XZ measurement of the 4-qubit and 3-qubit linear-cluster states. **(b)** The output distributions from ZX measurement of the 4-qubit and 3-qubit linear cluster states. See Supplemental Material of [YCZ+23] for the labelling of each cell.

0110, 0111, and so on. But note that, the bit strings are the measurement outcomes after the basis transformation. That means, for example, in the ZX measurement, 0011 will correspond to the output state $|1\rangle_1 \otimes |-\rangle_2 \otimes |0\rangle_3 \otimes |+\rangle_4$ (recall that in the bit-string representation, the right-most bit is identified as the first bit).

From these distributions, one can obtain the expectations of terms in the ODD and EVEN operators of the 4-qubit and 3-qubit LC states, as shown in Fig. 7.3 (b). For the 4-qubit LC state, its stabilizer group are spanned by $\{X_1Z_2, Z_1X_2Z_3, Z_2X_3Z_4, Z_3X_4\}$, and its ODD and EVEN operators are given by,

$$\mathrm{ODD}_4 = \left(\frac{I + X_1Z_2}{2}\right)\left(\frac{I + Z_2X_3Z_4}{2}\right) = \frac{I + X_1Z_2 + Z_2X_3Z_4 + X_1X_3Z_4}{4} \tag{7.9}$$

$$\mathrm{EVEN}_4 = \left(\frac{I + Z_1X_2Z_3}{2}\right)\left(\frac{I + Z_3X_4}{2}\right) = \frac{I + Z_1X_2Z_3 + Z_3X_4 + Z_1X_2X_4}{4}\ . \tag{7.10}$$

The nontrivial terms of ODD_4 are shown in the first 3 bars in the left of Fig. 7.3 (b), and those of EVEN_4 are shown in the last 3 bars. The expectation value of I is just a normalization condition, which will be always satisfied by a probability distribution. The expectation values of the non-trivial terms (the height of the bars) can be obtained from the distributions shown in Table 7.2. For example, $\langle X_1Z_2\rangle$ can be obtained from P^{XZ}_{exp}, and the observable is 1 if the parity of the first two bits are even, and -1 if the

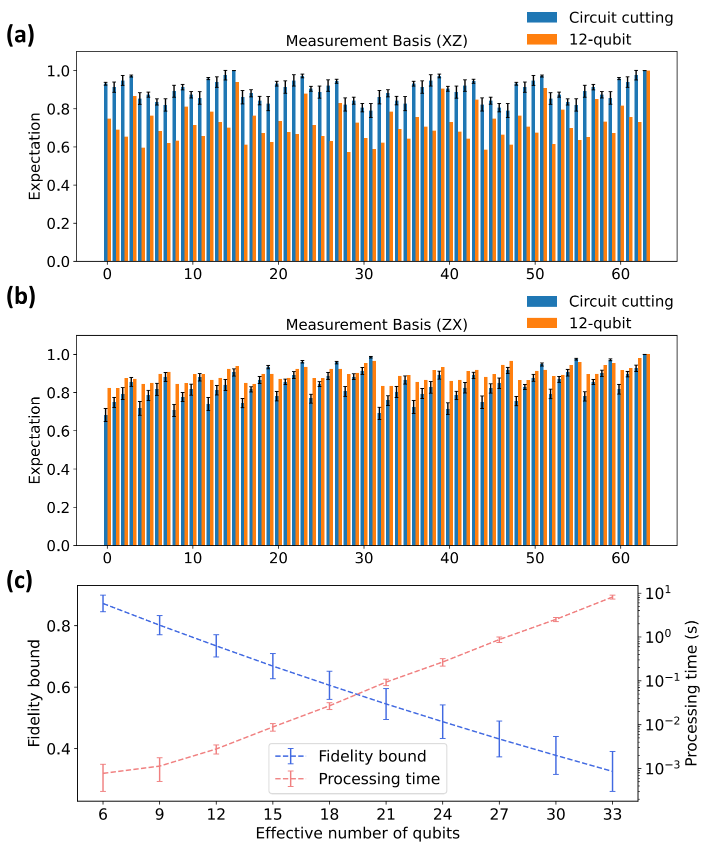


Figure 7.5: (a) Comparison of the expectations with XZ measurement obtained from the 12-qubit circuit [GCZ+19] (orange) and from the circuit-cutting scheme (blue). Each bar corresponds to one specific observable in ODD and there are 64 expectations for each group. The ideal values of all these expectations are one. **(b)** Similar data for the ZX measurement. **(c)** Fidelity bound (blue) and processing time (red) for simulating larger linear-cluster state using the same experimental data. The error bars are due to repeated experiments.

parity of the first two bits are odd. Therefore, we have,

$$\langle X_1 Z_2 \rangle = 0.245 + 0.274 + 0.002 + 0.213 + 0.266 = 1 \ . \tag{7.11}$$

The value of other bars (observables), including those from 3-qubit linear-cluster state, can be calculated with similar procedure. The fidelity lower bound then follows from these expectations according to Eq. (7.6), which is 0.952 and 0.909 for the 4-qubit and 3-qubit LC states, respectively. These bounds match the average fidelity (of all 36 circuits for subcircuit 1 and 12 circuits for subcircuit 2) from quantum state tomography.

We now turn our discussion to simulating large linear-cluster state with the circuit-cutting scheme. With the measurement data from the subcircuits, one can simulate larger linear-cluster states, and the fidelity bounds can be derived with similar procedures. Fig. 7.5 (a) and (b) present the expectations of terms in ODD_{12} and EVEN_{12} for the 12-qubit state obtained by the circuit-cutting scheme and a direct implementation, which is an analogue of Fig. 7.3 (c). Those expectations are from the XZ and ZX measurement, respectively, and there are 2^6 expectations in total for each subfigure. The blue bars are reconstructed from the circuit-cutting scheme, while the orange bars are from the experimental data in [GCZ+19]. Each bar corresponds to one specific terms in the expansion of ODD_{12} (and EVEN_{12}) and the ideal value is one. The labelling is obtained in the following way. First, as in the previous discussion, we expand

$$\mathsf{EVEN}_{12} = \left(\frac{I + Z_1X_2Z_3}{2}\right)\left(\frac{I + Z_3X_4Z_5}{2}\right)\cdots\left(\frac{I + Z_{11}X_{12}}{2}\right) \tag{7.12}$$

into $2^6 = 64$ terms. Each term can be represented by a binary vector (called the mask vector), with the 1's indicating the qubits that are acted nontrivially on. For example, in the 4-qubit case, $Z_1X_2X_4$ is represented by $(1, 1, 0, 1)$ and I is represented by $(0, 0, 0, 0)$. Now, go back to the 12-qubit case and label every term in the expansion of EVEN_{12} by a binary vector of length 12. Each bar in Fig. 7.5 (b) is associated with one term, and hence one binary vector. From right to left, the binary vector is in a lexicographic order, which means that the 64-th bar is for $(0, 0, \cdots, 0)$ (or I), the 63-th bar is for $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1)$ (or $Z_{11}X_{12}$), and so on. We wrote a computer program to automate such labelling and its ordering [CYZZ23].

We remark that the distributions from the 12-qubit experiment are also processed with the same procedure (the fidelity bound after processing is 0.615), i.e., TMEM followed by a maximum likelihood method, for a fair comparison. From these expectations, we can similarly use Eq. (7.6) to estimate the fidelity. The estimated fidelity bound from the circuit-cutting scheme is 0.734, about 19% higher than that from the experiment of [GCZ+19]. Note that the experiment of [GCZ+19] implemented CZ gates in parallel, which will incur more severe crosstalk errors compared to our current implementation, where CZ gates are applied individually (one for each layer; see the right of Fig. 7.2). Moreover, smaller circuits are easier to calibrate and control. Therefore, the circuit-cutting experiment achieves a better fidelity bound than [GCZ+19].

As for the tradeoff, the circuit-cutting scheme saves qubits and allows for better control of the quantum system, at the cost of increasing the both the quantum and classical running time. Both running times depend on the number of the cutting points, and the quantum running time is more expensive. In the 12-qubit experiment, 25000 shots are used for each basis, and thus a total of 50000 shots are consumed; there is no repetition in the 12-qubit experiment. In contrast, in the circuit-cutting experiment, there are 48 subcircuits to run, and 40000 shots are used for each subcircuits (1920000 shots in total); we perform 25 repeated experiments for the circuit-cutting scheme. Although in a single experiment (repetition), the circuit-cutting scheme can consume far more shots than the direct implementation, it is nevertheless a valuable technique for pushing the limits of near-term quantum devices.

Moreover, the symmetry in linear-cluster states allows us to reuse the measurement data from subcircuit 1 to simulate larger linear-cluster states, at a cost of increasing overhead in classical postprocessing. Specifically, we need to add more internal nodes to the tensor network in Fig. 7.2, to represent larger circuits (5 nodes for 15 qubits, 6 nodes for 18 qubits and so on). This allows us to simulate linear-cluster states of size $6 + 3k$, where k is a positive integer. We need to contract a longer chain to obtain one expectations of the large circuit, and there will be more expectations to be computed in order to obtain the fidelity lower bound. The obtained fidelity bound is expected to decay as the number of qubits increases, since the error accumulates in the classical postprocessing. The fidelity decay and classical postprocessing time are shown in Fig. 7.5 (b). Here, the classical postprocessing is done on a conventional laptop, and the processing time shows the running time of the program for calculating the fidelity lower bound of larger circuits [CYZZ23].

7.5 Discussion

In this chapter, we experimentally demonstrate a circuit-cutting scheme and simulate larger linear-cluster state with size scaling up to 33 qubits, using at most 4 qubits. In the case of 12 qubits, we achieve a higher fidelity compared to that of a previous work that prepared the 12-qubit state directly [GCZ+19], giving supportive evidence to the applicability of the circuit-cutting scheme.

Simulating large quantum circuits with small quantum devices is a promising direction in the NISQ era. Currently, there exist several circuit-cutting schemes [BSS16, PHOW20, MF21, MF20]; it is necessary to further perform experimental benchmarking on these schemes, in order to evaluate their applicability in practice. On the other hand, although circuit-cutting schemes provide systematic methods to cutting quantum circuits into smaller pieces, to the best of our knowledge, there is no general method for determining the optimal cutting points. Therefore, we believe that the potential of circuit-cutting has not yet been fully explored.

Chapter 8

Discussion and Conclusion

In this thesis, we studied the classical verification and classical enhancement of near-term quantum devices. In the first part of the thesis, we presented results on the verification protocols based on the instantaneous quantum polynomial-time (IQP) model, which is a promising model for achieving verifiable quantum advantage on near-term quantum devices. Specifically, we review the basic concepts of the IQP-based verification protocols in Chapter 3, including the Shepherd-Bremner construction and the its recent loophole. We then study the interplay between IQP circuits, stabilizer formalism and coding theory, and give a characterization of the correlation functions from IQP circuits in Chapter 4.

Based on this, we give a new IQP-based construction, called the stabilizer scheme, which enriches the scope of IQP-based schemes while maintaining their simplicity and verifiability. The construction in the stabilizer scheme is achieved by sampling generator matrices of random codes satisfying certain conditions. We also present another construction algorithm based on solving constrained matrix factorization problems, which is an early version of the stabilizer scheme and may provide insight from a different perspective.

In Chapter 5, we explore the classical security of the stabilizer scheme. We formulate the Hidden Structured Code conjecture, which states that no polynomial-time classical algorithm can generate samples that can pass the test, when the instances are randomly generated with the stabilizer scheme. To support this conjecture, we study a class of attack algorithms based on secret extraction and give evidence that the stabilizer scheme is secure against such attacks. We also provide a fix to the Shepherd-Bremner construction using the column redundancy technique, which invalidates the recent classical attack by Kahanamoku-Meyer. Our work paves the way for cryptographic verification of quantum computation advantage in the NISQ era.

There are several open problems for future research on classical verification of near-term quantum devices. The most important one is to rigorously prove the security of the IQP-based verification protocols. In Conjecture 5.1, we state that classical attacks based on secret extraction is on average hard. It would be favorable to prove the random self-reducibility of the problem, so that the hardness conjecture can be relaxed to the worst-case scenario. For example, recently a worst-to-average-case reduction was found for computing the probabilities of IQP circuits and it would be interesting to see if the techniques of [Mov23] could be leveraged to gain insight into the validity of Conjecture 5.1. Before one can rigorously prove the hardness of classical attacks, one might gain intuition by considering other possible classical attacks. In terms of implementing the protocol in practice, generating instances according to a given architecture and noise analysis are also important open problems. We believe that the mathematical structure of the stabilizer scheme provides a promising avenue for the use of certain cryptographic techniques to improve the security of IQP-based protocols, and to construct instances that can be readily implemented with current technology.

In the second part of the thesis, we presented results on classical enhancement of near-term quantum devices, with a focus on quantum error mitigation and circuit cutting. In Chapter 6, we propose a framework called neighborhood learning for quantum error mitigation, which incorporates several existing quantum error mitigation methods as special cases. We explore the performance of the neighborhood learning framework under different settings. Based on these investigations, we give an adaptive learning strategy that offers a good trade-off between the accuracy and the computational cost. In Chapter 7, we present the experimental results on a circuit-cutting scheme. Specifically, we simulate larger linear-cluster states with size up to 33 qubits, using only 4 superconducting qubits, and in the case of 12 qubits, we achieve a higher fidelity compared to preparing the state directly. Classical techniques to enhance the capability of near-term quantum devices are important and more research is needed in this direction to achieve quantum computational advantage on practical problems with the near-term quantum devices.

Bibliography

[AA11] 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, pages 333–342, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993682.

[AAB$^+$19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. doi:10.1038/s41586-019-1666-5.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, Cambridge, 2009. doi:10.1017/CBO9780511804090.

[ABOE08] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive Proofs For Quantum Computations, 2008, arXiv:0810.5375.

[ABOEM08] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive Proofs For Quantum Computations, 2008, arXiv:1704.04487.

[AG04] Scott Aaronson and Daniel Gottesman. Improved Simulation of Stabilizer Circuits. *Phys. Rev. A*, 70(5):052328, November 2004. doi:10.1103/PhysRevA.70.052328. arXiv: quant-ph/0406196.

[ALRS$^+$20] Thomas Ayral, François-Marie Le Régent, Zain Saleem, Yuri Alexeev, and Martin Suchara. Quantum divide and compute: Hardware demonstrations and noisy simulations. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 138–140. IEEE, 2020. doi:DOI:10.1109/ISVLSI49217.2020.00034.

[AN02] Dorit Aharonov and Tomer Naveh. Quantum NP - A Survey, October 2002. URL `http://arxiv.org/abs/quant-ph/0210077`. arXiv: quant-ph/0210077.

[ARS$^+$21] Thomas Ayral, François-Marie Le Régent, Zain Saleem, Yuri Alexeev, and Martin Suchara. Quantum Divide and Compute: Exploring The Effect of Different Noise Sources. *arXiv:2102.03788 [quant-ph]*, February 2021. URL http://arxiv.org/abs/2102.03788. arXiv: 2102.03788.

[BB92] A. Berthiaume and G. Brassard. The quantum challenge to structural complexity theory. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 132–137, Boston, MA, USA, 1992. IEEE Comput. Soc. Press. doi:10.1109/SCT.1992.215388.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM J. Comput.*, 26(5):1510–1523, October 1997. doi:10.1137/S0097539796300933.

[BCG21] Boaz Barak, Chi-Ning Chou, and Xun Gao. Spoofing Linear Cross-Entropy Benchmarking in Shallow Quantum Circuits. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:20, 2021. doi:10.4230/LIPIcs.ITCS.2021.30. arXiv: 2005.02421.

[BCJ23] Michael J. Bremner, Bin Cheng, and Zhengfeng Ji. IQP Sampling and Verifiable Quantum Advantage: Stabilizer Scheme and Classical Security, 2023. URL https://arxiv.org/abs/2308.07152. arXiv:2308.07152.

[BCM$^+$18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. A Cryptographic Test of Quantumness and Certifiable Randomness from a Single Quantum Device. *arXiv:1804.00640 [quant-ph]*, April 2018. URL http://arxiv.org/abs/1804.00640. arXiv: 1804.00640.

[BFK09] A. Broadbent, J. Fitzsimons, and E. Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526, Oct 2009. doi:10.1109/FOCS.2009.36.

[BFK10] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Measurement-based and universal blind quantum computation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6154 LNCS:43–86, 2010, arXiv:0807.4154. doi:10.1007/978-3-642-13678-8_2.

[BFNV19] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nat. Phys.*, 15(2):159–163, February 2019. doi:10.1038/s41567-018-0318-2. arXiv:1803.04402.

[BIS$^+$18] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nat. Phys.*, 14(6):595–600, June 2018. doi:10.1038/s41567-018-0124-x.

[BJS11] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. R. Soc. A*, 467(2126):459–472, feb 2011. doi:10.1098/rspa.2010.0301.

[BKM$^+$14] Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, 2014. doi:https://doi.org/10.1038/nature13171.

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. Simpler Proofs of Quantumness. *arXiv:2005.04826*, May 2020. URL `https://arxiv.org/abs/2005.04826`. arXiv: 2005.04826.

[BMKT22] Ewout van den Berg, Zlatko K. Minev, Abhinav Kandala, and Kristan Temme. Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors, January 2022. URL https://arxiv.org/abs/2201.09866. arXiv:2201.09866.

[BMS16] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-Case Complexity Versus Approximate Simulation of Commuting Quantum Computations. *Phys. Rev. Lett.*, 117(8):080501, August 2016. doi:10.1103/PhysRevLett.117.080501.

[BMS17] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum*, 1:8, 2017. doi:https://doi.org/10.22331/q-2017-04-25-8.

[BMSSO18] X. Bonet-Monroig, R. Sagastizabal, M. Singh, and T. E. O'Brien. Low-cost error mitigation by symmetry verification. *Phys. Rev. A*, 98(6):062339, December 2018. doi:10.1103/PhysRevA.98.062339.

[BSK$^+$21] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C. Mckay, and Jay M. Gambetta. Mitigating measurement errors in multiqubit experiments. *Phys. Rev. A*, 103:042605, Apr 2021. doi:10.1103/PhysRevA.103.042605.

[BSS16] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading Classical and Quantum Computational Resources. *Phys. Rev. X*, 6(2):021043, June 2016. doi:10.1103/PhysRevX.6.021043.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum Complexity Theory. *SIAM J. Comput.*, 26(5):1411–1473, October 1997. doi:10.1137/S0097539796300921.

[CACC21] Piotr Czarnik, Andrew Arrasmith, Patrick J. Coles, and Lukasz Cincio. Error mitigation with Clifford quantum-circuit data. *arXiv:2005.10189*, February 2021. URL `http://arxiv.org/abs/2005.10189`. arXiv: 2005.10189.

[CCL$^+$21] Xi Chen, Bin Cheng, Zhaokai Li, Xinfang Nie, Nengkun Yu, Man-Hong Yung, and Xinhua Peng. Experimental Cryptographic Verification for Near-Term Quantum Cloud Computing. *Sci. Bull.*, 66(1):23–28, 2021. doi:10.1016/j.scib.2020.08.013.

[CDG$^+$23] Bin Cheng, Xiu-Hao Deng, Xiu Gu, Yu He, Guangchong Hu, Peihao Huang, Jun Li, Ben-Chuan Lin, Dawei Lu, Yao Lu, Chudan Qiu, Hui Wang, Tao Xin, Shi Yu, Man-Hong Yung, Junkai Zeng, Song Zhang, Youpeng Zhong, Xinhua Peng, Franco Nori, and Dapeng Yu. Noisy Intermediate-Scale Quantum Computers. *Front. Phys.*, 18(2):21308, April 2023. doi:10.1007/s11467-022-1249-z.

[CH17a] Earl T. Campbell and Mark Howard. Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost. *Phys. Rev. A*, 95:022316, Feb 2017. doi:10.1103/PhysRevA.95.022316.

[CH17b] Earl T. Campbell and Mark Howard. Unifying gate synthesis and magic state distillation. *Phys. Rev. Lett.*, 118:060501, Feb 2017. doi:10.1103/PhysRevLett.118.060501.

[CYZZ23] Bin Cheng, Chong Ying, Yu-Ning Zhang, and Youwei Zhao. Alariccheng/simulating_large_lc_states: v1.0.0, March 2023. doi:10.5281/zenodo.7693877.

[DDM03] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, 68(4):042318, October 2003. doi:10.1103/PhysRevA.68.042318.

[Deu85] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400(1818):97–117, July 1985. doi:10.1098/rspa.1985.0070.

[Deu89] David Deutsch. Quantum computational networks. *Proc. R. Soc. Lond. A*, 425(1868):73–90, September 1989. doi:10.1098/rspa.1989.0099.

[DGC18] Vedran Dunjko, Yimin Ge, and J. Ignacio Cirac. Computational speedups using small quantum devices. *Phys. Rev. Lett.*, 121:250501, Dec 2018. doi:10.1103/PhysRevLett.121.250501.

[DHJB20] Alexander M. Dalzell, Nicholas Hunter-Jones, and Fernando G. S. L. Brandão. Random quantum circuits anti-concentrate in log depth. *arXiv:2011.12277 [cond-mat, physics:quant-ph]*, November 2020. URL http://arxiv.org/abs/2011.12277. arXiv: 2011.12277.

[DMH+18] E.F. Dumitrescu, A.J. McCaskey, G. Hagen, G.R. Jansen, T.D. Morris, T. Papenbrock, R.C. Pooser, D.J. Dean, and P. Lougovski. Cloud Quantum Computing of an Atomic Nucleus. *Phys. Rev. Lett.*, 120(21):210501, May 2018. doi:10.1103/PhysRevLett.120.210501.

[EBL18] Suguru Endo, Simon C. Benjamin, and Ying Li. Practical Quantum Error Mitigation for Near-Future Applications. *Phys. Rev. X*, 8(3):031027, July 2018. doi:10.1103/PhysRevX.8.031027.

[Fey82] Richard P. Feynman. Simulating physics with computers. *Int. J. Theor. Phys.*, 21(6-7):467–488, jun 1982. doi:10.1007/bf02650179.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028*, November 2014. URL http://arxiv.org/abs/1411.4028. arXiv: 1411.4028.

[FGHP99] Stephen Fenner, Frederic Green, Steven Homer, and Randall Pruim. Determining acceptance possibility for a quantum computation is hard for the polynomial hierarchy. *Proc. R. Soc. Lond. A*, 455(1991):3953–3966, November 1999. doi:10.1098/rspa.1999.0485.

[FHcvM18] Joseph F. Fitzsimons, Michal Hajdušek, and Tomoyuki Morimae. Post hoc verification of quantum computation. *Phys. Rev. Lett.*, 120:040501, Jan 2018. doi:10.1103/PhysRevLett.120.040501.

[FK17] Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Physical Review A*, 96(1), jul 2017. doi:10.1103/physreva.96.012303.

[FMU$^+$22] Keisuke Fujii, Kaoru Mizuta, Hiroshi Ueda, Kosuke Mitarai, Wataru Mizukami, and Yuya O. Nakagawa. Deep variational quantum eigensolver: A divide-and-conquer method for solving a larger problem with smaller size quantum computers. *PRX Quantum*, 3:010346, Mar 2022. doi:10.1103/PRXQuantum.3.010346.

[FRCB22] Markus Frembs, Sam Roberts, Earl Campbell, and Stephen D Bartlett. Hierarchies of resources for measurement-based quantum computation. *New Journal of Physics*, 2022. doi:10.1088/1367-2630/acaee2.

[GCZ$^+$19] Ming Gong, Ming-Cheng Chen, Yarui Zheng, Shiyu Wang, Chen Zha, Hui Deng, Zhiguang Yan, Hao Rong, Yulin Wu, Shaowei Li, Fusheng Chen, Youwei Zhao, Futian Liang, Jin Lin, Yu Xu, Cheng Guo, Lihua Sun, Anthony D. Castellano, Haohua Wang, Chengzhi Peng, Chao-Yang Lu, Xiaobo Zhu, and Jian-Wei Pan. Genuine 12-Qubit Entanglement on a Superconducting Quantum Processor. *Phys. Rev. Lett.*, 122(11), March 2019. URL `https://link.aps.org/doi/10.1103/PhysRevLett.122.110501`.

[GD20] Yimin Ge and Vedran Dunjko. A hybrid algorithm framework for small quantum computers with application to finding hamiltonian cycles. *J. Math. Phys.*, 61(1):012201, 2020. doi:https://doi.org/10.1063/1.5119235.

[Gel21] Michael R. Geller. Conditionally rigorous mitigation of multi-qubit measurement errors. *Phys. Rev. Lett.*, 127:090502, Aug 2021. doi:10.1103/PhysRevLett.127.090502.

[Got99] Daniel Gottesman. The Heisenberg Representation of Quantum Computers. In *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, Cambridge, MA, 1999. URL `http://arxiv.org/abs/quant-ph/9807006`. arXiv: quant-ph/9807006.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pages 212–219, Philadelphia, Pennsylvania, United States, 1996. ACM Press. doi:10.1145/237814.237866.

[GT09] Otfried Gühne and Géza Tóth. Entanglement detection. *Phys. Rep.*, 474(1-6):1–75, April 2009. doi:10.1016/j.physrep.2009.02.004.

[GTHL$^+$20] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J. Zeng. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 306–316, Denver, CO, USA, October 2020. IEEE. doi:10.1109/QCE49297.2020.00045.

[HBVSE18] Dominik Hangleiter, Juan Bermejo-Vega, Martin Schwarz, and Jens Eisert. Anticoncentration theorems for schemes showing a quantum speedup. *Quantum*, 2:65, May 2018. doi:10.22331/q-2018-05-22-65. arXiv: 1706.03786.

[HC18] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, 2018. doi:10.1088/2058-9565/aad604.

[HE23] Dominik Hangleiter and Jens Eisert. Computational advantage of quantum random sampling. *Rev. Mod. Phys.*, 95(3):035001, July 2023. doi:10.1103/RevModPhys.95.035001. arXiv:2206.04079.

[HKP20] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting Many Properties of a Quantum System from Very Few Measurements. *Nat. Phys.*, 16:1050–1057, February 2020. URL https://doi.org/10.1038/s41567-020-0932-7. arXiv: 2002.08953.

[HM17] Aram W. Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, September 2017. doi:10.1038/nature23458.

[HMO$^+$21] William J. Huggins, Sam McArdle, Thomas E. O'Brien, Joonho Lee, Nicholas C. Rubin, Sergio Boixo, K. Birgitta Whaley, Ryan Babbush, and Jarrod R. McClean. Virtual Distillation for Quantum Error Mitigation. *Phys. Rev. X*, 11(4):041036, November 2021. doi:10.1103/PhysRevX.11.041036.

[HNDJB20] Andre He, Benjamin Nachman, Wibe A. De Jong, and Christian W. Bauer. Zero-noise extrapolation for quantum-gate error mitigation with identity insertions. *Phys. Rev. A*, 102(1):012426, July 2020. doi:10.1103/PhysRevA.102.012426.

[JKMW05] Daniel FV James, Paul G Kwiat, William J Munro, and Andrew G White. On the measurement of qubits. In *Asymptotic Theory of Quantum Statistical Inference: Selected Papers*, pages 509–538. World Scientific, 2005. doi:https://doi.org/10.1142/9789812563071_0035.

[KM19] Gregory D. Kahanamoku-Meyer. Forging quantum data: classically defeating an IQP-based quantum test. *arXiv:1912.05547*, December 2019. URL http://arxiv.org/abs/1912.05547. arXiv: 1912.05547.

[KMCVY21] Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani, and Norman Y. Yao. Classically-Verifiable Quantum Advantage from a Computational Bell Test. *arXiv:2104.00687*, April 2021. URL http://arxiv.org/abs/2104.00687. arXiv: 2104.00687.

[Kni96] Emanuel Knill. Quantum Randomness and Nondeterminism. LANL report LAUR-96-2186, 1996. URL https://arxiv.org/abs/quant-ph/9610012. arXiv: quant-ph/96100 12.

[Koc21] Bálint Koczor. Exponential Error Suppression for Near-Term Quantum Devices. *Phys. Rev. X*, 11(3):031057, September 2021. doi:10.1103/PhysRevX.11.031057.

[KS08] Yong-Hyuk Kim and Keomkyo Seo. Two Congruence Classes for Symmetric Binary Matrices over F2. *WSEAS Trans. Math.*, 7(6):339–343, June 2008. URL https://dl.acm.org/doi/abs/10.5555/1466915.1466917.

[KSV02] A. Yu Kitaev, A. Shen, and M. N. Vyalyi. *Classical and quantum computation*. Number v. 47 in Graduate studies in mathematics. American Mathematical Soc., Providence, R.I, 2002.

[KTC$^+$19] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491–495, March 2019. doi:10.1038/s41586-019-1040-7.

[KWY+23] Youngseok Kim, Christopher J. Wood, Theodore J. Yoder, Seth T. Merkel, Jay M. Gambetta, Kristan Temme, and Abhinav Kandala. Scalable error mitigation for noisy quantum circuits produces competitive expectation values. *Nat. Phys.*, February 2023. doi:10.1038/s41567-022-01914-3. arXiv:2108.09197.

[LB17] Ying Li and Simon C. Benjamin. Efficient Variational Quantum Simulator Incorporating Active Error Minimization. *Phys. Rev. X*, 7(2):021050, June 2017. doi:10.1103/PhysRevX.7.021050.

[LBR17] A. P. Lund, Michael J. Bremner, and T. C. Ralph. Quantum sampling problems, BosonSampling and quantum supremacy. *npj Quantum Inf*, 3(1):15, December 2017. doi:10.1038/s41534-017-0018-2.

[Lem75] Abraham Lempel. Matrix Factorization over $GF(2)$ and Trace-Orthogonal Bases of $GF(2^n)$. *SIAM J. Comput.*, 4(2):175–186, June 1975. doi:10.1137/0204014.

[Llo96] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, aug 1996. doi:10.1126/science.273.5278.1073.

[LZWW19] Jin-Guo Liu, Yi-Hong Zhang, Yuan Wan, and Lei Wang. Variational quantum eigensolver with fewer qubits. *Phys. Rev. Research*, 1:023025, Sep 2019. doi:10.1103/PhysRevResearch.1.023025.

[Mah18] Urmila Mahadev. Classical Verification of Quantum Computations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–267, Paris, October 2018. IEEE. doi:10.1109/FOCS.2018.00033. arXiv: 1804.01082.

[Man21] Ryan L. Mann. Simulating quantum computations with Tutte polynomials. *npj Quantum Inf*, 7(1):141, September 2021. doi:10.1038/s41534-021-00477-0. arXiv:2101.00211.

[MF20] Kosuke Mitarai and Keisuke Fujii. Overhead of the non-local-to-local channel decomposition by quasiprobability sampling. *arXiv:2006.11174*, page 7, 2020. URL `https://arxiv.org/abs/2006.11174`. arXiv:2006.11174.

[MF21] Kosuke Mitarai and Keisuke Fujii. Constructing a virtual two-qubit gate by sampling single-qubit operations. *New J. Phys.*, 23(2):023021, 2021. doi:https://doi.org/10.1088/1367-2630/abd7bc.

[MG14] John M. Martinis and Michael R. Geller. Fast adiabatic qubit gates using only σ_z control. *Phys. Rev. A*, 90:022307, Aug 2014. doi:10.1103/PhysRevA.90.022307.

[Mov18] Ramis Movassagh. Efficient unitary paths and quantum computational supremacy: A proof of average-case hardness of Random Circuit Sampling. *arXiv:1810.04681 [cond-mat, physics:hep-th, physics:math-ph, physics:quant-ph]*, October 2018. URL `http://arxiv.org/abs/1810.04681`. arXiv: 1810.04681.

[Mov19] Ramis Movassagh. Cayley path and quantum computational supremacy: A proof of average-case $\#P-$hardness of Random Circuit Sampling with quantified robustness. *arXiv:1909.06210*, September 2019. URL `https://arxiv.org/abs/1909.06210`. arXiv: 1909.06210.

[Mov23] Ramis Movassagh. The hardness of random quantum circuits. *Nat. Phys.*, July 2023. doi:10.1038/s41567-023-02131-2.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes.* Elsevier, 1977.

[MS08] Igor L. Markov and Yaoyun Shi. Simulating Quantum Computation by Contracting Tensor Networks. *SIAM J. Comput.*, 38(3):963–981, January 2008. doi:10.1137/050644756. arXiv: quant-ph/0511069.

[MYB19] Sam McArdle, Xiao Yuan, and Simon Benjamin. Error-Mitigated Digital Quantum Simulation. *Phys. Rev. Lett.*, 122(18):180501, May 2019. doi:10.1103/PhysRevLett.122.180501.

[MZO20] Filip B Maciejewski, Zoltán Zimborás, and Michał Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography. *Quantum*, 4:257, 2020. doi:https://doi.org/10.22331/q-2020-04-24-257.

[NC11] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, New York, 10th ed. edition, 2011.

[Nie06] Michael A. Nielsen. Cluster-state quantum computation. *Rep. Math. Phys.*, 57(1):147–161, Feb 2006. doi:10.1016/s0034-4877(06)80014-5.

[O'D14] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[Pen14] R. A. Pendavingh. On the evaluation at (-i,i) of the Tutte polynomial of a binary matroid. *J. Algebr. Comb.*, 39(1):141–152, February 2014. doi:10.1007/s10801-013-0442-0.

[PHOW20] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. Simulating large quantum circuits on a small quantum computer. *Phys. Rev. Lett.*, 125:150504, Oct 2020. doi:10.1103/PhysRevLett.125.150504.

[PMS$^+$14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.*, 5(1):4213, September 2014. doi:10.1038/ncomms5213.

[Pre12] John Preskill. Quantum computing and the entanglement frontier, November 2012. URL `http://arxiv.org/abs/1203.5813`. arXiv: 1203.5813.

[Pre18] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. doi:10.22331/q-2018-08-06-79. arXiv: 1801.00862.

[PSSO21] Michael A Perlin, Zain H Saleem, Martin Suchara, and James C Osborn. Quantum circuit cutting with maximum-likelihood tomography. *npj Quantum Inf.*, 7(1):1–8, 2021. doi:https://doi.org/10.1038/s41534-021-00390-6.

[RB01] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001. doi:10.1103/PhysRevLett.86.5188.

[RUV13] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013, arXiv:1209.0449. doi:10.1038/nature12035.

[SB09] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proc. R. Soc. A*, 465(2105):1413–1439, may 2009. doi:10.1098/rspa.2008.0443.

[She10] Dan Shepherd. Binary Matroids and Quantum Probability Distributions. *arXiv:1005.1744 [quant-ph]*, May 2010. URL `http://arxiv.org/abs/1005.1744`. arXiv: 1005.1744.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, 1994. IEEE Comput. Soc. Press. doi:10.1109/SFCS.1994.365700.

[Sno20] Ryan Snoyman. A Proof of Quantumness, 2020. Honor thesis, UNSW.

[SQC$^+$21] Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C. Benjamin, and Ying Li. Learning-based quantum error mitigation. *arXiv:2005.07601 [quant-ph]*, March 2021. URL http://arxiv.org/abs/2005.07601. arXiv: 2005.07601.

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, October 1976. doi:10.1016/0304-3975(76)90061-X.

[SZM$^+$21] Kai Sun, Zi-Jian Zhang, Fei Meng, Bin Cheng, Zhu Cao, Jin-Shi Xu, Man-Hong Yung, Chuan-Feng Li, and Guang-Can Guo. Experimental verification of group non-membership in optical circuits. *Photon. Res.*, 9(9):1745, September 2021. doi:10.1364/PRJ.427897.

[Tak21] Ryuji Takagi. Optimal resource cost for error mitigation. *Phys. Rev. Research*, 3(3):033178, August 2021. doi:10.1103/PhysRevResearch.3.033178.

[TBG17] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119(18):180509, November 2017. doi:10.1103/PhysRevLett.119.180509. arXiv: 1612.02058.

[TD02] Barbara M. Terhal and David P. DiVincenzo. Adaptive Quantum Computation, Constant Depth Quantum Circuits and Arthur-Merlin Games, May 2002. URL http://arxiv.org/abs/quant-ph/0205133. arXiv: quant-ph/0205133.

[TEMG21] Ryuji Takagi, Suguru Endo, Shintaro Minagawa, and Mile Gu. Fundamental limits of quantum error mitigation, March 2021. URL http://arxiv.org/abs/2109.04457. arXiv:2109.04457 [quant-ph].

[TG05] Géza Tóth and Otfried Gühne. Entanglement detection in the stabilizer formalism. *Phys. Rev. A*, 72:022340, Aug 2005. doi:10.1103/PhysRevA.72.022340.

[TTL+21] Wei Tang, Teague Tomesh, Jeffrey Larson, Martin Suchara, and Margaret Martonosi. CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 473–486, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3445814.3446758. arXiv: 2012.02333.

[Ver98] Dirk Vertigan. Bicycle Dimension and Special Points of the Tutte Polynomial. *Journal of Combinatorial Theory, Series B*, 74(2):378–396, November 1998. doi:10.1006/jctb.1998.1860.

[Wat00] J. Watrous. Succinct Quantum Proofs for Properties of Finite Groups. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 537–546, Redondo Beach, CA, USA, 2000. IEEE Comput. Soc. doi:10.1109/SFCS.2000.892141.

[WBC+21] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei

Pan. Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Phys. Rev. Lett.*, 127(18):180501, October 2021. doi:10.1103/PhysRevLett.127.180501. arXiv:2106.14734.

[WCCY22] Fan Wang, Bin Cheng, Zi-Wei Cui, and Man-Hong Yung. Quantum Computing by Quantum Walk on Quantum Slide, May 2022. URL `http://arxiv.org/abs/2211.08659`. arXiv:2211.08659.

[WCZ+20] Bujiao Wu, Bin Cheng, Jialin Zhang, Man-Hong Yung, and Xiaoming Sun. Speedup in Classical Simulation of Gaussian Boson Sampling. *Sci. Bull.*, 65(10):832–841, 2020. doi:10.1016/j.scib.2020.02.012.

[XCL+23] Lei Xie, Bin Cheng, Xiaodie Lin, Zhenyu Chen, Zhaohui Wei, and Zhengfeng Ji. A Machine Learning Approach to Quantum Error Mitigation, 2023.

[Yao93] A. Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, Palo Alto, CA, USA, 1993. IEEE. doi:10.1109/SFCS.1993.366852.

[YC20] Man-Hong Yung and Bin Cheng. Anti-Forging Quantum Data: Cryptographic Verification of Quantum Cloud Computing. *arXiv:2005.01510*, May 2020. URL `http://arxiv.org/abs/2005.01510`. arXiv: 2005.01510.

[YCM+14] M.-H. Yung, J. Casanova, A. Mezzacapo, J. McClean, L. Lamata, A. Aspuru-Guzik, and E. Solano. From transistor to trapped-ion computers for quantum chemistry. *Sci Rep*, 4(1):3589, January 2014. doi:10.1038/srep03589.

[YCZ+23] Chong Ying, Bin Cheng, Youwei Zhao, He-Liang Huang, Yu-Ning Zhang, Ming Gong, Yulin Wu, Shiyu Wang, Futian Liang, Jin Lin, Yu Xu, Hui Deng, Hao Rong, Cheng-Zhi Peng, Man-Hong Yung, Xiaobo Zhu, and

Jian-Wei Pan. Experimental Simulation of Larger Quantum Circuits with Fewer Superconducting Qubits. *Phys. Rev. Lett.*, 130(11):110601, March 2023. doi:10.1103/PhysRevLett.130.110601.

[YSL+21] Xiao Yuan, Jinzhao Sun, Junyu Liu, Qi Zhao, and You Zhou. Quantum simulation with hybrid tensor networks. *Phys. Rev. Lett.*, 127:040501, Jul 2021. doi:10.1103/PhysRevLett.127.040501.

[ZCC+21] Qingling Zhu, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yulin Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Science Bulletin*, page S2095927321006733, October 2021. doi:10.1016/j.scib.2021.10.017.

[ZKML+22] Daiwei Zhu, Gregory D. Kahanamoku-Meyer, Laura Lewis, Crystal Noel, Or Katz, Bahaa Harraz, Qingfeng Wang, Andrew Risinger, Lei Feng, Debopriyo Biswas, Laird Egan, Alexandru Gheorghiu, Yunseong Nam, Thomas Vidick, Umesh Vazirani, Norman Y. Yao, Marko Cetina, and Christopher Monroe. Interactive Protocols for Classically-Verifiable Quantum Advantage, June 2022. URL `http://arxiv.org/abs/2112.05156`. arXiv:2112.05156.

[ZWD+20] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, December 2020. doi:10.1126/science.abe8770.