

Hybrid Techniques for Simulating Quantum Circuits using the Heisenberg Representation

by

Héctor J. García-Ramírez

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2014

Doctoral Committee:

Professor Igor L. Markov, Chair
Professor John P. Hayes
Associate Professor Yaoyun Shi
Professor Kim Winick

© Héctor J. García-Ramírez 2014

All Rights Reserved

Dedicado principalmente a mi amada esposa, *Myriam Estevez*. Vida mía, gracias por confiar en mi y apoyarme en esta gran aventura. Los frutos de mis esfuerzos van dedicados a mis tres hijos: *José David*, *Gael Ernesto* y *Luis Eduardo*. Ustedes han sido el compás de mi destino y la verdadera manifestación de mi orgullo y amor.

ACKNOWLEDGEMENTS

First and foremost, I must thank my beautiful and lovely wife, *Myriam Estevez*. Without her love and support, none of this would have been possible so she deserves as much credit as I do for the completion of this PhD dissertation. I am forever grateful to my three sons, *José David*, *Gael Ernesto* and *Luis Eduardo*, who drove me crazy just enough to help me keep my sanity during these past few years. Special thanks to my advisor Professor Igor L. Markov for providing me with excellent research ideas. I would like to acknowledge the work and effort from the members of my dissertation committee. Thanks to all my Michigan friends (too many to list here, but you know who you are) for your constant emotional and academic help and support. Finally, I must thank my mother, *Elsie M. Ramírez*, for her unyielding encouragement and confidence in my success. You are an exemplary human being and I am very proud to call you mother.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	vii
LIST OF TABLES	xiii
LIST OF ALGORITHMS	xv
ABSTRACT	xvi
CHAPTER	
I. Introduction	1
1.1 Toward Scalable Simulation of Quantum Circuits	2
1.2 Quantum States and Circuits	5
1.3 Quantum Measurements	7
1.4 Quantum Error-correcting Codes and Fault-tolerant Quantum Computation	9
1.5 The Pauli operator basis	12
1.6 Simulation of Quantum Circuits	17
1.7 Objectives of the Dissertation	18
1.8 Dissertation Outline	21
II. Simulation Techniques and the Heisenberg Representation	23
2.1 Classification of Simulation Techniques	24
2.1.1 Strong versus weak	24
2.1.2 Stateless versus direct	25
2.2 The Stabilizer Formalism	27
2.2.1 Stabilizer gates and circuits	29
2.2.2 Stabilizer states	31
2.2.3 Canonical forms	38

2.3	Algorithms for Simulation and Equivalence-checking of Stabilizer Circuits	52
2.3.1	The tableau technique	52
2.3.2	Equivalence checking	56
2.3.3	Global-phase maintenance	56
2.4	Stateless Simulation of Stabilizer Circuits	58
2.5	Summary	62
III. Metric Geometry of Stabilizer States		64
3.1	Geometric Properties of Stabilizer States	66
3.1.1	Inner products and k -neighbor stabilizer states	68
3.1.2	Exterior products and stabilizer bivectors	76
3.1.3	Linear dependence of stabilizer states	79
3.2	The Embedding of Stabilizer Geometry in Hilbert Space	81
3.2.1	Approximating an arbitrary quantum state with one stabilizer state	82
3.2.2	Approximating arbitrary states with superpositions of stabilizer states	83
3.3	Summary	87
IV. Computational Geometry of Stabilizer States		89
4.1	Synthesis of Canonical Stabilizer Circuits	89
4.2	An Inner-product Algorithm	93
4.3	Orthogonalization of Stabilizer States	95
4.4	Computation of Stabilizer Bivectors	97
4.5	Applications of Geometric Algorithms	100
4.5.1	Mixed stabilizer states	100
4.5.2	Simulation of quantum systems	101
4.6	Empirical Studies	102
4.7	Summary	105
V. Engineering Stabilizer-based Simulation of Generic Quantum Circuits		106
5.1	Stabilizer Frames	106
5.1.1	Frame Operations	108
5.1.2	Frame-based Simulation of Quantum Circuits	110
5.1.3	Frame measurements	113
5.2	Multiframe Simulation	113
5.2.1	Orthogonality of Multiframe	116
5.2.2	Parallel Frame-based Simulation	117
5.3	Empirical Validation	119
5.3.1	Stabilizer circuits	119

5.3.2	Quantum ripple-carry adders	120
5.3.3	Quantum Fourier transform (QFT) circuits	123
5.3.4	Fault-tolerant (FT) circuits	125
5.4	p -blocked Multiframes	129
5.5	Summary	132
VI. Stateless Simulation of Generic Quantum Circuits		135
6.1	Pauli Expansions of Linear Operators	135
6.2	Stateless Simulation of Generic Quantum Circuits	137
6.2.1	Compact Representation using Decision Diagrams	139
6.2.2	Empirical Validation	140
6.3	Summary	142
VII. Graph-based Techniques and Matrix Product States		144
7.1	Matrix Product State (MPS) Simulation	145
7.2	Quantum Multivalued Decision Diagrams (QMDDs)	148
7.3	MPS-based Decision Diagrams	150
7.4	Summary	154
VIII. Conclusions		156
8.1	Summary of Contributions	157
8.2	Open Challenges	162
8.3	Future Directions	163
8.3.1	Frame-based simulation using Pauli expansions	164
8.3.2	Stabilizer-based matrix product states	164
APPENDIX		166
BIBLIOGRAPHY		170

LIST OF FIGURES

Figure

1.1	A software architecture for synthesizing, optimizing and simulating quantum circuits.	4
1.2	Graphical representation of the EPR pair circuit.	7
1.3	The one-qubit Pauli operators.	10
1.4	Transversal implementation of a stabilizer circuit acting on 3-qubit QECC registers.	11
1.5	The stabilizer gates: Hadamard (H), Phase (P) and controlled-NOT ($CNOT$). Note that $H^\dagger = H$ and $CNOT^\dagger = CNOT$. Furthermore $P^4 = I$, thus $P^\dagger = P^3$	17
2.1	General quantum circuit for decision problems.	26
2.2	Implementation of the (a) Controlled-Z ($CPHASE$) and (b) Controlled- Y operations using Pauli and Clifford gates. These gates can be simulated directly on the stabilizer or using the equivalences shown here.	31
2.3	Representation of the states generated by the EPR-pair circuit using stabilizer matrices. The phase array is omitted for simplicity. The rows of the matrices $\{R_1, R_2\}$ denote the stabilizer generators of the corresponding two-qubit state.	31
2.4	Canonical (row-reduced echelon) form for stabilizer matrices. The X -block contains a <i>minimal</i> set of rows with X/Y literals. The rows with Z literals only appear in the Z -block. Each block is arranged so that the leading non- I literal of each row is strictly to the right of the leading non- I literal in the row above. The number of Pauli (non- I) literals in each block is minimal.	46

2.5	Simulation of EPR-pair circuit using the tableau technique. We omit phase vectors for clarity.	54
2.6	Classical simulation complexities for sets of Clifford computational tasks [40]. IN(BITS) and IN(PROD) refer to allowing computational basis states and general product states as inputs. OUT(1) and OUT(MANY) refer to having single bit and multi-bit outputs. NON-ADAPT and ADAPT refer to circuits with intermediate measurements and gates conditioned on measurement outcomes (ADAPT). WEAK and STRONG refer to two notions of classical simulation as defined in Section 2.1.1. Cl-P denotes that classical efficient simulation is possible, QC-hard denotes that universal quantum computation is possible, and #P-hard asserts that classical simulation could be used to solve arbitrary problems in the classical class #P (and hence NP too).	63
3.1	The angle between any stabilizer state and its nearest neighbors is 45° and the distance is $\sqrt{2} - \sqrt{2}$. Here, $ s_1\rangle$ is a nearest neighbor of both $ 00\rangle$ and $ 10\rangle$. Similarly, $ s_2\rangle$ is a nearest neighbor of $ 00\rangle$ and $ 01\rangle$. The angle between these two nearest neighbors of $ 00\rangle$ is 60° . Consider the linearly-dependent triplets $\{ 00\rangle, 10\rangle, s_1\rangle\}$ and $\{ 00\rangle, 01\rangle, s_2\rangle\}$. Each set contains two pairs of nearest neighbors and one pair of orthogonal states.	65
3.2	(a) For an n -qubit stabilizer state $ \psi\rangle$, the fraction ($\approx 2/3$) of all stabilizer states that are oblique to $ \psi\rangle$ is dominated by its $(n - k)$ -neighbors, where $k \in \{0, 1, 2, 3, 4\}$. (b) The stabilizer states orthogonal to $ \psi\rangle$ together with its $(n - k)$ -neighbors ($k < 3$) account for $\approx 99\%$ of all states.	77
3.3	A ball \mathbf{B} with radius $\sqrt{2}$ centered on the unit sphere covers half of the unit sphere. Every such ball contains at least one stabilizer state (in most cases, half of all stabilizer states).	87
4.1	Template structure for the basis-normalization circuit synthesized by Algorithm 4.1.1. The input is an arbitrary stabilizer state $ \psi\rangle$ while the output is a basis state $ b_1 b_2 \dots b_n \in \{0, 1\}^n\rangle$	91
4.2	(a) Average runtime for Algorithm 4.2.1 to compute the inner product between two random n -qubit stabilizer states. The stabilizer matrices that represent the input states are generated by applying $\beta n \log_2 n$ unitary stabilizer gates to $ 0^{\otimes n}\rangle$. (b) Average number of gates in the circuits produced by Algorithm 4.1.1.	103

4.3	Average runtime for Algorithm 4.2.1 to compute the inner product between (a) $ 0^{\otimes n}\rangle$ and random stabilizer state $ \phi\rangle$ and (b) the n -qubit GHZ state and random stabilizer state $ \phi\rangle$	103
4.4	(a) Runtime and (b) circuit-size comparisons between Algorithm 4.1.1 and the circuit synthesis portion of the Audenaert-Plenio inner-product algorithm. On average, Algorithm 4.1.1 runs roughly twice as fast and produces canonical circuits that contain less than half as many gates. Furthermore, the Audenaert-Plenio circuits are not canonical.	104
5.1	Example of a stabilizer frame that represents $ \psi\rangle$. Observe that while $ \psi\rangle$ is composed of four computational-basis amplitudes, its frame representation has only two phase vectors. For $ a_1 ^2 = a_2 ^2$, one can manipulate \mathcal{F} to reduce its size. We discuss this technique in Section 5.1.2.	108
5.2	Simulation of $TOF_{c_1 c_2 t} \Psi\rangle$ using a stabilizer-state superposition (Equation 5.3). Here, $c_1 = 1$, $c_1 = 2$ and $t = 3$. Amplitudes are omitted for clarity and the (\pm) -phase vectors are shown as columns prefixed to their corresponding matrices. The X gate is applied to the third qubit of the $ \Psi^{c_1 c_2 = 11}\rangle$ cofactor.	111
5.3	Example of how a multiframe representation is derived from a single-frame representation.	115
5.4	Overall simulation flow for Quipu	118
5.5	Our C++11 template function for executing the frame operations (Func f) described in Section 5.1.2 in parallel. The function accepts a range of vector elements defined by iterators Iter begin and Iter end . Params... p is the variadic template argument that defines the parameters of Func f . The number of threads allowed is defined by MTHREAD . The function returns a vector of std::futures , which can be used to access the result of the asynchronous operations. . .	119
5.6	Average time needed by Quipu and CHP to simulate an n -qubit stabilizer circuit with $\beta n \log n$ gates and n measurements. Quipu is asymptotically as fast as CHP but is not limited to stabilizer circuits.	121
5.7	Ripple-carry (Cuccaro) adder for 3-bit numbers $a = a_0 a_1 a_2$ and $b = b_0 b_1 b_2$ [25, Figure 6]. The third qubit from the top is an ancilla and the z qubit is the carry. The b -register is overwritten with the result $s_0 s_1 s_2$	121

5.8	Average runtime and memory needed by Quipu and QuIDDP to simulate n -bit Cuccaro adders after a superposition of all computational-basis states is obtained using a block of Hadamard gates (Figure 5.7). The quadratic function $f(x) = 0.5248x^2 - 15.815x + 123.86$ fits Quipu 's curve with $R^2 = .9986$	122
5.9	The three-qubit QFT circuit. In general, The first qubit requires one Hadamard gate, the next qubit requires a Hadamard and a controlled- $R(\alpha)$ gate, and each following qubit requires an additional controlled- $R(\alpha)$ gate. Summing up the number of gates gives $O(n^2)$ for an n -qubit QFT circuit.	124
5.10	Average runtime and memory needed by Quipu (single-threaded and multi-threaded) and QuIDDP to simulate n -qubit QFT circuits, which contain $n(n+1)/2$ gates. We used the $ 11\dots 1\rangle$ input state for all benchmarks.	124
5.11	Fault-tolerant implementation of a Toffoli gate. Each line represents a 5-qubit register and each gate is applied transversally. The state $ cat\rangle = (0^{\otimes 5}\rangle + 1^{\otimes 5}\rangle)/\sqrt{2}$ is obtained using a stabilizer subcircuit (not shown). The arrows point to the set of gates that is applied if the measurement outcome is 1; no action is taken otherwise. Controlled- Z gates are implemented as $H_j CNOT_{i,j} H_j$ with control i and target j . Z gates are implemented as P^2	126
5.12	Adder circuits from our benchmarks. We used the 5-qubit DiVincenzo/Shor QECC and implemented Toffoli gates using the FT architecture from Figure 5.11.	126
5.13	Mod-exp with $M = 15$ implemented as $(2, 4)$ -LUTs [44] for several coprime base values. Negative controls are shown with hollow circles. We apply Hadamards to each x -qubit to generate a superposition of all the input values for x . Our benchmarks implement these computations using the 3-qubit bit-flip code [51, Ch. 10] and the FT-Toffoli architecture from Figure 5.11.	127
5.14	(a) Stabilizer frame (single-term multiframe) representation for $U 1111\rangle$, where U is the 4-qubit QFT circuit. (b) p -blocked multiframe representation for $U 1111\rangle$. Observe that $\mathcal{V} = \mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \mathcal{V}_3 \otimes \mathcal{V}_4$	130
5.15	Average runtime and memory for Quipu to simulate n -qubit QFT circuits on input state $ 11\dots 1\rangle$ using p -blocked multiframe. The poly-fit functions for runtime and memory have $R^2 = .9886$ and $R^2 = .9964$, respectively.	132

6.1	MDD for addition operator ADD_2 . The circular (internal) nodes represent Pauli literals and the edges represent possible values for such literals. The square (terminal) nodes represent the possible phases $(\pm 1, \pm i)$ of each term in ADD_2	141
6.2	Average time needed by Quipu (stateless) and CHP to simulate an n -qubit stabilizer circuit with $2n + 1$ gates and a single measurement (circuit structure from Figure 2.1). Quipu runs in linear time for such instances while CHP takes quadratic time.	142
6.3	(a) Average time needed by Quipu (stateless) and Quipu (direct) to simulate n -bit adders (Section 5.3) after a superposition of all computational-basis states is obtained. A single measurement is performed on qubit $2n + 2$, which is the highest carry bit. The stateless approach takes exponential time and scales to around 18-bit instances only. (b) Comparison of the number of terms in the direct frame-based approach (size of multiframe) and the stateless approach (size of the Pauli-string list).	143
7.1	Average runtime and peak memory for ZKCM to simulate n -qubit QFT circuits on input state $ 11 \dots 1\rangle$. The maximum Schmidt rank χ observed for each benchmark is 1. $R^2 = .999$ for polynomial fit $f(x)$ in both plots.	147
7.2	Several representations for the controlled- V gate on three qubits, where $V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$. The second and third qubits from the top are the control and target, respectively. The first qubit is not modified. The QMDD exploits the structure of the matrix and thus provides a more compact representation. Note that the edge weights are normalized as described in Definition VII.2.	150
7.3	General form for an MPMDD . Level l of the diagram (dotted rectangles) corresponds to a tensor $A^{[l]}$ in Equation 7.3. The degree (number of outgoing edges) of each internal node in an MPMDD is equal to twice the local Schmidt rank (χ_A in Equation 7.2). Therefore, the width of the graph is twice the Schmidt rank χ for the state. The weights of nodes and edges depends on the particular variable assignment (Definition VII.4-iv).	152
7.4	Two graph-based representations for the separable state $ \psi\rangle = (00\rangle + 01\rangle + 10\rangle + 11\rangle)/2$. Here, $v_1^{[1]} = v_1^{[2]} = 1.0$ and $A_1^{[1]0} = A_1^{[1]1} = A_1^{[2]0} = A_1^{[2]1} = 1/\sqrt{2}$	153

7.5 Two graph-based representations for the entangled state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$. The tensor (A) and vector (v) values for the MPMDD are listed in Example VII.1. 154

LIST OF TABLES

Table

1.1	Graphical representation of basic quantum gates.	6
1.2	Multiplication table for Pauli matrices. The shaded cells indicate anticommuting products.	13
2.1	(a) Transformation properties of the Pauli-group elements under conjugation by the Clifford operators [51]. In the <i>CNOT</i> case, subscript 1 indicates the control and 2 the target. (b) Pauli matrices expressed as Clifford sequences.	29
2.2	Clifford sequences for generating an arbitrary permutation of the Pauli operators. Subscripts indicate qubit indices. Given two different generators, such as X_1Y_2 one can apply one qubit rotation(s) to obtain homogenous generators, such as Y_1Y_2 , and then apply a two-qubit sequence. A pure <i>YZ</i> swap is achieved by conjugating β and γ . Conjugating the pure <i>YZ</i> swap with γ achieves any pure swap, e.g., $\beta\gamma\alpha$ achieves an <i>XY</i> swap.	30
2.3	One-qubit stabilizer states and their Pauli-matrix stabilizer. Short-hand notation lists only the normalized amplitudes of the basis states.	33

2.4	Sixty two-qubit stabilizer states and their corresponding Pauli generators. Shorthand notation represents a stabilizer state as $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ where α_i are the normalized amplitudes of the basis states. The basis states are emphasized in bold. The first column lists states whose generators do not include an upfront minus sign, and other columns introduce the signs. A sign change creates an orthogonal vector. Therefore, each row of the table gives an orthogonal basis. The cells in dark grey indicate stabilizer states with four non-zero basis amplitudes, i.e., $\alpha_i \neq 0 \forall i$. The \angle column indicates the angle between that state and $ 00\rangle$, which has twelve nearest-neighbor states (light gray) and 15 orthogonal states (\perp). All three-qubit stabilizer states are listed in Appendix A.	34
2.5	Several unbiased states that are <i>not</i> stabilizer states. The three-qubit state is not a stabilizer state because its $ 0\rangle$ -cofactor on the third qubit is not a stabilizer state.	42
2.6	One- and two-qubit stabilizer states that generate non-trivial global phases when a stabilizer gate is applied. For simplicity, the $2^{-n/2}$ factors were omitted in columns two and three. The third column shows the normalized output state obtained with the stabilizer formalism (Definition II.14).	45
2.7	Addition table for \mathbb{Z}_2^2 representation of Pauli operators.	53
2.8	Comparison between direct (stabilizer formalism) and stateless simulation for n -qubit Clifford circuit with g gates and m ancilla qubits (used for multi-qubit measurements in stateless simulation).	60
3.1	Distribution of inner products (angles) between any one n -qubit stabilizer state and all other stabilizer states for $n \in \{1, \dots, 7\}$. The last column indicates the ratio of orthogonal (\perp) states.	74
5.1	Average time and memory needed by <code>Quipu</code> and <code>QPLite</code> to simulate our benchmark set of quantum FT circuits. The second column indicates the QECC used to encode k logical qubits into n physical qubits. We used the 3-qubit bit-flip code for larger benchmarks and the 5-qubit DiVincenzo/Shor code [16] for smaller ones (*). The third column shows the total number of qubits including ancillas required to implement FT-Toffoli gates. We used the $ 00\dots 0\rangle$ input state for all benchmarks. The multithreaded version of <code>Quipu</code> exhibited similar runtime and memory requirements for these benchmarks since the total number of states observed is relatively small.	128

LIST OF ALGORITHMS

Algorithm

2.2.1 Canonical form reduction for stabilizer matrices	47
2.2.2 Computation of 2^n basis amplitudes for a stabilizer state	49
4.1.1 Synthesis of a basis-normalization circuit	92
4.2.1 Inner product for stabilizer states	95
4.3.1 Orthogonalization procedure for linear combinations of stabilizer states	98
4.4.1 Computation of stabilizer bivectors	99
5.1.1 Frame-based simulation of the Toffoli gate	111
5.2.1 Frame coalescing	114
5.2.2 Frame coalescing (extended)	115
5.2.3 Simulation-flow steps	116
5.4.1 p -blocked multiframe simulation	134
A. The 1080 three-qubit stabilizer states	167

ABSTRACT

Hybrid Techniques for Simulating Quantum Circuits using the Heisenberg Representation

by

Héctor J. García-Ramírez

Chair: Igor L. Markov

Simulation of quantum information processing remains a major challenge with important applications in quantum computer science and engineering. Generic quantum-circuit simulation appears intractable for conventional computers and may be unnecessary because useful quantum circuits exhibit significant structure that can be exploited during simulation. For example, Gottesman and Knill identified an important subclass, called *stabilizer circuits*, which can be simulated efficiently using the Heisenberg representation for quantum computers. Stabilizer circuits are exclusively composed of *stabilizer gates* – Hadamard, Phase and CNOT – followed by one-qubit measurements in the computational basis. Such circuits are applied to a computational-basis state and produce so-called *stabilizer states*. Aaronson and Gottesman generalized stabilizer-circuit simulation to additionally handle a small number of non-stabilizer gates. We design new, more efficient data structures and algorithms for such beyond-stabilizer simulation using superpositions of stabilizer states. One such data structure, a *stabilizer frame*, offers more compact storage than previous approaches but require additional algorithms to maintain the global phases

of each state in the superposition. To explore the advantages and limitations of our technique, we analyze the geometric structure of stabilizer states and their embedding in Hilbert space. Our analysis includes results on the computational geometry of stabilizer states such as efficient algorithms for computing distances, angles and volumes between them. The main advantages of using stabilizer-state superpositions to simulate quantum circuits are: (i) stabilizer subcircuits are simulated with high efficiency, (ii) superpositions can be restructured and compressed on the fly during simulation to reduce resource requirements, and (iii) operations performed on such superpositions lend themselves to distributed or asynchronous processing. Our software implementation, called **Quipu**, simulates certain quantum arithmetic circuits (e.g., reversible ripple-carry adders) and quantum Fourier transform circuits in polynomial time and space for specific input states. On such instances, known linear-algebraic simulation techniques, such as the (state-of-the-art) BDD-based simulator **QuIDDPro**, take exponential time. We simulate quantum fault-tolerant circuits using **Quipu**, and the results indicate that our stabilizer-based technique empirically outperforms **QuIDDPro** in all cases. While previous structure-aware simulations of quantum circuits were difficult to parallelize, we demonstrate a parallel version of **Quipu** that achieves a nontrivial speedup.

CHAPTER I

Introduction

Among the principal questions studied in quantum information processing are the computational complexity and achievable empirical performance of quantum-mechanical simulation on conventional computers [51]. The apparent intractability of generic quantum simulation suggested the idea of using quantum phenomena to accelerate conventional computation. The key insight is to replace the familiar 0 and 1 bits of conventional computing with information units called qubits (quantum bits) that capture quantum states of elementary particles or atomic nuclei. By operating on qubits, a *quantum computer* can, in principle, process exponentially more data than a classical computer in a similar number of steps. Quantum information processing has been demonstrated with a variety of physical technologies (NMR, ion traps, Josephson junctions in superconductors, optics) and used in recently developed commercial products. High-speed quantum communication systems have been built by the National Institute of Standards and Technology, BBN Technologies and start-ups in the United States and Europe. Some of these systems are currently available commercially, and others are operated for research purposes and as testbeds for network protocol development. In the mid-nineties, several researches discovered quantum algorithms to carry out computation more efficiently than conventional computers. One example is Shor's number-factoring algorithm [62], which finds the prime fac-

tors of an integer exponentially faster than the most efficient known classical factoring algorithm. These developments have fueled research efforts to build and program scalable quantum computers. In order to scale quantum information processing, including computation and communication, to large and complex systems, quantum device operation is compactly captured by the abstract formalism of *quantum circuits* [51]. These circuits consist of interconnected quantum gates that act on qubits. They can be composed in a hierarchical manner, using design techniques similar to those used in digital logic design. However, as quantum circuits offer a much broader range of information-processing possibilities, their design and evaluation entail a dramatic increase of complexity, requiring new levels of sophistication in design algorithms and tools [53, 64]. A particularly important class of design tools performs simulation of quantum circuits on conventional workstations, i.e., these tools produce representative outputs of ideal quantum circuits on particular inputs, but without requiring quantum hardware. The most serious fundamental obstacle to practical quantum information processing known today is the inherent instability of qubits. This obstacle is traditionally addressed by quantum error-correction techniques [17, 33, 51, 56], which are generally available but require significant overhead and require adaptation to individual quantum circuits.

1.1 Toward Scalable Simulation of Quantum Circuits

Quantum simulation tools typically consist of a front-end and a back-end [64, 67]. The front-end facilitates the development of quantum software and the back-end acts as a temporary replacement of (hardware) quantum processing units to run such software. Once quantum hardware is available, it can be used in conjunction with pre-existing front-end to run the accumulated software with increased efficiency. Figure 1.1 shows an ideal design flow for a comprehensive software architecture for synthesizing and simulating quantum circuits. The primary component of the initial

phase in the design flow consists of a compiler for quantum computers (q-compiler). One important job of the compiler is to access the functional modules contained in a separate library – e.g., quantum Fourier transform (QFT), quantum walks, quantum search, etc. – and integrate the functionality of these modules into high-level programs as specified. Thus, the details of such reusable modules are abstracted away from the high-level quantum programmer. The output of the compiler is an intermediate, technology-independent circuit specification that is manipulated in later parts of the design flow.

After generating quantum circuits from a high-level program, and after several possible optimization steps, it may be useful to simulate such a circuit in order to: (i) characterize the effect of various errors in practical quantum circuits, (ii) test error correction techniques or (iii) verify the correctness of synthesized quantum circuits. Researchers have developed many different approaches to quantum computer simulation, each focusing on a particular set of quantum circuits and requiring data structures and algorithms that are dependent on the techniques used [51, 67]. Such diversity makes it more challenging to develop a comprehensive software architecture for quantum computing design tools. Therefore, it is of particular importance to *design data structures and algorithms that incorporate a variety of modeling techniques and are flexible enough to exploit the performance speedups offered by different simulation approaches*. To this end, we have completed a large body of work on the design of new hybrid simulation techniques that exploit the Heisenberg representation for quantum computers [33, 51]. This work includes a thorough analysis of the geometry of *stabilizer states* – the class of quantum states that admits a compact representation on conventional computers via the Heisenberg representation [30, 31]. We leveraged our theoretical analysis to develop several data structures including: *stabilizer frames, multiframe and p-blocked multiframe*. We describe the design of practical software methods that implement these data structures and algorithms to facilitate simulation

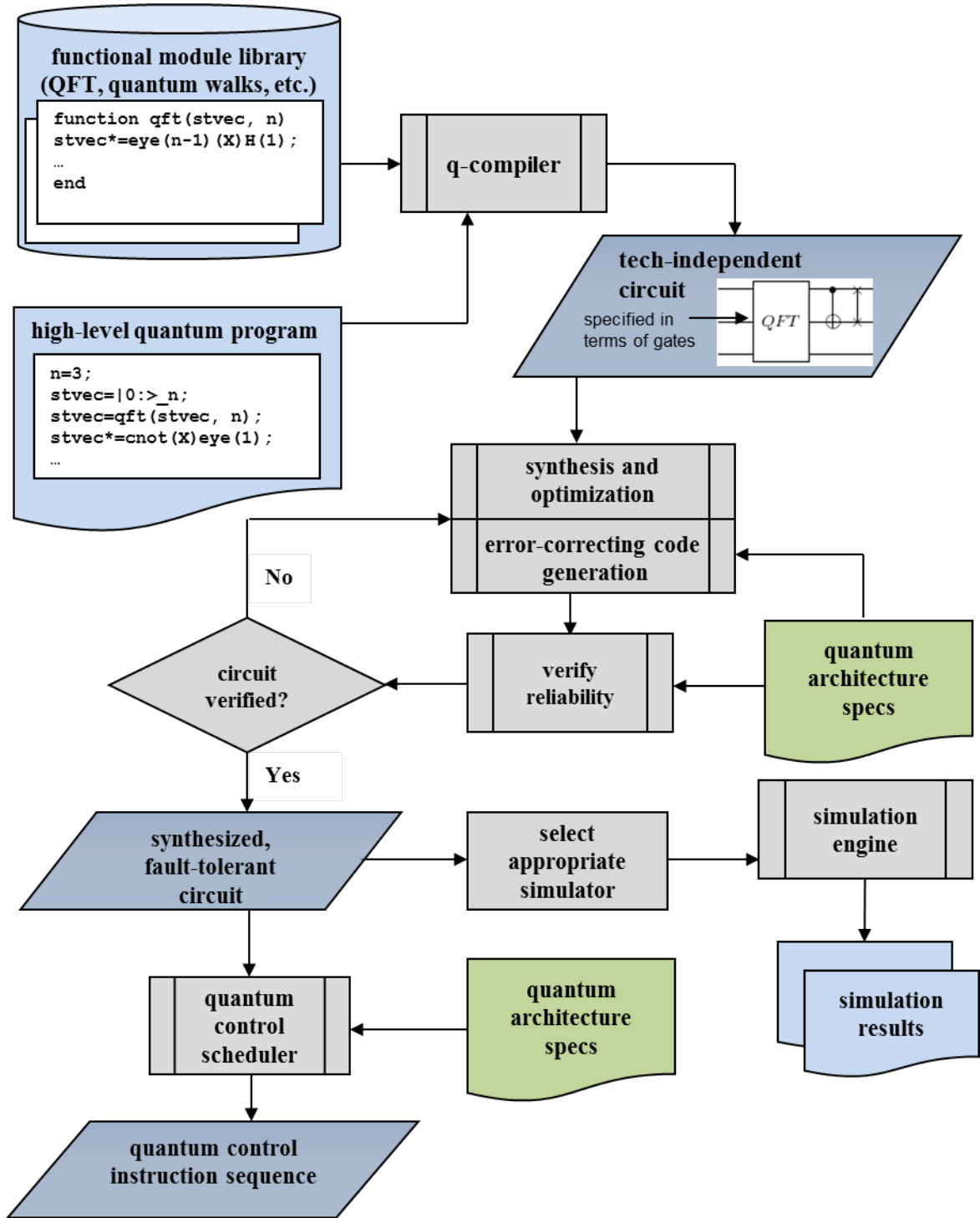


Figure 1.1: A software architecture for synthesizing, optimizing and simulating quantum circuits.

of larger sets of quantum circuits on conventional computers [28, 29]. However, before we take a closer look at the Heisenberg representation for quantum computers in Chapter II, it is instructive to first review background information on quantum computation.

1.2 Quantum States and Circuits

Quantum information processes, including quantum algorithms, are often modeled using *quantum circuits*, which are sequences of *gate operations* that act on some register of *qubits* – the basic unit of information in a quantum system. The quantum state $|\psi\rangle$ of a single qubit is described by a two-dimensional complex-valued vector. In contrast to classical bits, qubits can be in a *superposition* of the 0 and 1 states. Formally, $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, where $|0\rangle = (1, 0)^\top$ and $|1\rangle = (0, 1)^\top$ are the two-dimensional *computational-basis states* and α_i are *probability amplitudes* that satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$. An n -qubit register is the tensor product of n single qubits and thus is modeled by a complex vector $|\psi^n\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle = \sum_{i=0}^{2^n-1} \alpha_i |b_i\rangle$, where each b_i is a binary string representing the value i of each basis state. Furthermore, $|\psi^n\rangle$ satisfies $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. Each gate operation or *quantum gate* is a *unitary matrix* that operates on a small subset of the qubits in a register. For example, the quantum analogue of a *NOT* gate is the operator $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$,

$$\alpha_0 |00\rangle + \alpha_1 |10\rangle \xrightarrow{X \otimes I} \alpha_0 |10\rangle + \alpha_1 |00\rangle$$

Similarly, the two-qubit *CNOT* operator flips the second qubit (target) iff the first qubit (control) is set to 1, e.g.,

$$\alpha_0 |00\rangle + \alpha_1 |10\rangle \xrightarrow{CNOT} \alpha_0 |00\rangle + \alpha_1 |11\rangle$$

Another operator of particular importance is the Hadamard (H), which is frequently used to put a qubit in a superposition of computational-basis states, e.g.,

$$\alpha_0 |00\rangle + \alpha_1 |10\rangle \xrightarrow{I \otimes H} \frac{\alpha_0(|00\rangle + |01\rangle) + \alpha_1(|10\rangle + |11\rangle)}{\sqrt{2}}$$





Note that the H gate generates *unbiased* superpositions in the sense that the squares of the absolute value of the amplitudes are equal.

Definition I.1. An arbitrary pure state $|\psi\rangle$ with computational-basis decomposition $\sum_{k=0}^n \lambda_k |k\rangle$ is said to be *unbiased* if for all $\lambda_i \neq 0$ and $\lambda_j \neq 0$, $|\lambda_i|^2 = |\lambda_j|^2$. Otherwise, the state is *biased*.

Graphical representation

Just like conventional digital circuits, quantum circuits are commonly represented by diagrams [51, 67]. Table 1.1 shows what those diagrams look like for the gates described in Section 1.2. The last gate shown in the table is called the Toffoli (TOFF) gate, which is a 3-qubit gate that maps (a, b, c) to $(a, b, c \oplus (ab))$. It is also known as the “controlled-controlled-NOT” gate. The Toffoli gate along with single-qubit gates (e.g., Hadamard) can be used for universal quantum computation. The horizontal lines seen in the drawings represent the passage through time of the qubit system and are

Table 1.1: Graphical representation of basic quantum gates.

GATE	GRAPHICAL SYMBOL
Hadamard	
NOT	
CNOT	
TOFF	

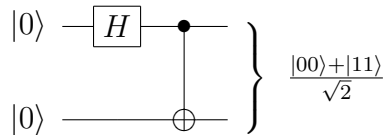


Figure 1.2: Graphical representation of the EPR pair circuit.

read from left to right. Figure 1.2 shows a graphical representation for the quantum circuit that generates an entangled EPR pair. Note that wires are usually labeled from top to bottom. Vertical lines indicate the control and target qubits of a gate. The “•” symbol indicates that the gate is activated when the state of the controlling qubit is $|1\rangle$. If a “o” (not shown in the figure) symbol is used instead of a “•” the gate is activated if the state of the controlling qubit is $|0\rangle$. The “ \oplus ” symbol indicates which target qubits are negated when the gate is activated. Lastly, other gates like the Hadamard and measurement gates are depicted using a labeled box that covers the wires on which they act.

1.3 Quantum Measurements

The dynamics involved in observing a quantum state are described by non-unitary *measurement operators* [51, Section 2.2.3]. There are different types of quantum measurements, but the type most pertinent to this work comprises *projective measurements in the computational basis*, i.e., measurements with respect to the $|0\rangle$ or $|1\rangle$ basis states. The corresponding single-qubit measurement operators are $P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $P_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, respectively. The probability $p(x)$ of obtaining outcome $x \in \{0, 1\}$ on the j^{th} qubit of state $|\psi\rangle$ is given by the inner product $\langle\psi|P_x^j|\psi\rangle$, where $\langle\psi|$ is the conjugate transpose of $|\psi\rangle$. For example, the probability of obtaining $|1\rangle$ upon measuring state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ is

$$p(1) = (\alpha_0^*, \alpha_1^*)P_1(\alpha_0, \alpha_1)^\top = (\alpha_0, \alpha_1)(0, \alpha_1)^\top = |\alpha_1|^2$$

Post-measurement states are computed as, $P_x^j |\psi\rangle / \sqrt{p(x)}$ where $x \in \{0, 1\}$ is the outcome obtained after measuring qubit j . In the above example, $P_1 |\psi\rangle / \sqrt{p(1)} = (0, \alpha_1)^\top / |\alpha_1| = (0, 1)^\top$.

Cofactors of quantum states. When dealing with single-qubit computational-basis measurements, we call the post-measurement states *cofactors*. Such states are separable states of the form $|0\rangle |\psi_0\rangle$ and $|1\rangle |\psi_1\rangle$. We denote the $|0\rangle$ - and $|1\rangle$ -cofactor by $|\psi^{j=0}\rangle$ and $|\psi^{j=1}\rangle$, respectively, where j is the index of the measured qubit. The states $|\alpha_0\rangle$ and $|\alpha_1\rangle$ are called *reduced cofactors*. One can also consider *iterated cofactors*, such as *double cofactors* $|\psi^{qr=00}\rangle \dots |\psi^{qr=11}\rangle$. Cofactoring with respect to all qubits produces the individual amplitudes of the computational-basis states.

Definition I.2. Let $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle$ and $x \in \{0, 1\}$. Furthermore, let $C^{j=x}$ be the set of computational-basis states in $|\psi\rangle$ with the j^{th} qubit set to x and $\alpha_k \neq 0$. The *support* of $|\psi^{j=x}\rangle$, denoted by $|\psi^{j=x}|$, is $|C^{j=x}|$.

(i) The *support* of $|\psi^{j=x}\rangle$, denoted by $|\psi^{j=x}|$, is $|C^{j=x}|$.

(ii) Projections onto the cofactors are denoted by $Cof_{j=x} : |\psi\rangle \mapsto |\psi^{j=x}\rangle, x \in \{0, 1\}$.

Observation I.3. The $|0\rangle$ -cofactor of $|\psi\rangle$, with respect to qubit j , is $|\psi^{j=0}\rangle = \sum_{|k\rangle \in C^{j=0}} \alpha_k |k\rangle$. The $|1\rangle$ -cofactor of $|\psi\rangle$, with respect to j , is $|\psi^{j=1}\rangle = \sum_{|k\rangle \in C^{j=1}} \alpha_k |k\rangle$.

In Section 1.4, we will see that quantum measurements are often performed as an intermediate step in a quantum circuit. Furthermore, the measurement results are used to conditionally control subsequent quantum gates. The following two important theorems are worth bearing in mind about quantum circuits.

Theorem I.4 (Principle of deferred measurements [51]). *Measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit; if the measurement results are used at any stage of the circuit then the classically controlled operations can be replaced by conditional quantum operations.*

Theorem I.5 (Principle of implicit measurements [51]). *Without loss of generality, any unterminated quantum wires (qubits which are not measured) at the end of a quantum circuit may be assumed to be measured.*

1.4 Quantum Error-correcting Codes and Fault-tolerant Quantum Computation

Quantum information involves manipulating the quantum states of particles that are in coherent quantum superpositions. Quantum states are intrinsically delicate in the sense that they are prone to environmental noise and decay easily – a phenomenon called *decoherence* [51]. Therefore, reliable quantum computers will require methods for protecting quantum information and preventing the environment from interacting with the data. The theory of quantum error-correcting codes (QECC) shares some similarities with its classical counterpart, but also exhibits important differences. On conventional computers, error correction works by storing the information in a redundant way. However, as the following theorem shows, one cannot make a “backup copy” of a qubit.

Theorem I.6 (No-cloning [33, 51]). *There is no quantum operation that takes a quantum state $|\psi\rangle$ to $|\psi\rangle \otimes |\psi\rangle$ for all states $|\psi\rangle$.*

Instead of using multiple copies of qubits, a QECC distributes the information of a single “logical” qubit over a block of physical qubits (QECC register) so that a small number of errors in the qubits of any block has little or no effect on the encoded qubits [7, 17, 33, 61]. A quantum code using n qubits to encode k qubits is written as an $[[n, k]]$ code (the double brackets distinguish it from a classical code). Most well-known quantum codes express the evolution of an encoded qubit as a linear combination of four possibilities: (i) no error occurs, (ii) bit flip $|0\rangle \longleftrightarrow |1\rangle$ occurs, (iii) relative phase flip $|0\rangle + |1\rangle \longleftrightarrow |0\rangle - |1\rangle$ occurs, and (iv) combined bit and

phase flip error occurs. Such errors can be described as Pauli operations acting on the encoded qubits. The one-qubit Pauli matrices are shown in Figure 1.3 along with the identity matrix, which corresponds to the case where no error occurs. In Section 1.5, we review specific properties of multi-qubit Pauli operators and show that such operators form an *error basis* that facilitates detection and correction of arbitrary qubit errors.

Once the quantum information is properly encoded, one can diagnose which of the four error types (Figure 1.3) occurred by making a suitable measurement. Then, the error is corrected by applying one of the four Pauli transformations.

Example I.7. Consider a variant of the $[[9, 1]]$ code from [61]. We encode a qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as a tensor product of nine qubits $|\bar{\psi}\rangle = \alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$, where

$$|\bar{0}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$$

$$|\bar{1}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)$$

The three sets of qubits that encode each logical qubit are designed to detect bit-flip errors. To diagnose such errors, we first measure the parity of the first two qubits and the parity of the second and third qubits in each set of three. Then, we take the majority within each set and correct the error, e. g., $|010\rangle \pm |101\rangle \rightarrow |000\rangle \pm |111\rangle$.

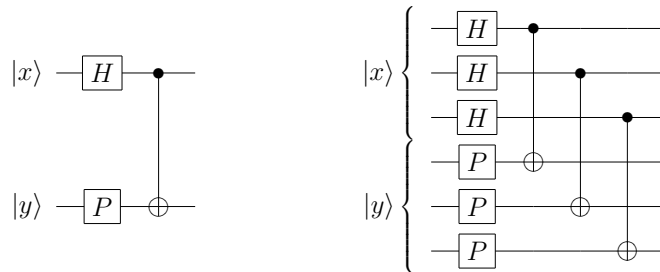
PAULI OPERATOR	ACTION ($b \in \{0, 1\}$)	ERROR TYPE
$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$I b\rangle = b\rangle$	None
$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$X b\rangle = b \oplus 1\rangle$	Bit flip
$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$Z b\rangle = (-1)^b b\rangle$	Phase flip
$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$Y b\rangle = i(-1)^b b \oplus 1\rangle$	Bit-phase flip

Figure 1.3: The one-qubit Pauli operators.

The first qubit of each set can be used to diagnose a phase-flip error – one takes the majority of the three signs, e. g., $(|000\rangle+|111\rangle)\otimes(|000\rangle-|111\rangle)\otimes(|000\rangle+|111\rangle) \rightarrow |\bar{0}\rangle$. Since these two error correction steps are independent, the code also works if there is both a bit-flip error and a phase-flip error.

Observe that, in order for a correction procedure to be effective, one needs to design measurements in such a way that the superposition used in the code is not destroyed. Such measurement procedures often require ancilla qubits that are discarded after errors are corrected.

The code used in Example I.7 facilitates prolonged storage of quantum information and error-correction procedures. However, it is also important to understand: (i) how to perform operations on an encoded state without disrupting the coding scheme, and (ii) how to safely correct errors when the gates themselves are noisy. Quantum circuits that implement procedures to perform such tasks are said to be *fault-tolerant* (FT). FT operations limit the propagation errors from one qubit in a QECC register to another qubit in the same register, and a single faulty gate damages at most one qubit in each register. To accomplish this, FT procedures make use of *transversal gates* whenever possible. In a transversal operation, the i^{th} qubit in each QECC register interacts only with the i^{th} qubit of other QECC registers. Figure 1.4 shows a transversal implementation of a stabilizer circuit. Universal quantum computations using only



(a) Logical operation (b) Transversal implementation

Figure 1.4: Transversal implementation of a stabilizer circuit acting on 3-qubit QECC registers.

transversal gates does not appear to be possible. Therefore, researchers have designed quantum FT architectures that require ancilla QECC registers, FT measurement schemes, and correction procedures conditioned on measurement outcomes. Such architectures implement protocols that carry out reliable computation in the presence of individual circuit-component failures. In Section 5.3, we describe and show several examples of quantum FT circuits that implement half-adders, full-adders and modular exponentiation. For a comprehensive review of QECC and quantum FT architectures, we refer the reader to the work in [33, 51, 56, 61].

1.5 The Pauli operator basis

As discussed in Section 1.4, the Pauli matrices (Figure 1.3) play a large role in quantum error correction because they define the set of possible one-qubit errors – bit-flip, phase-flip, and combined bit and phase flip. In this section, we introduce key properties of *n-qubit Pauli operators* – *n*-fold tensor products of Pauli matrices – and prove that the Pauli operators form an orthonormal error basis allowing QECC to correct not only single-qubit errors but any *linear combination* of such errors.

Definition I.8. The *Pauli group* \mathcal{P}_n on *n* qubits consists of the *n*-fold tensor product of Pauli matrices, $P = \pm i^k P_1 \otimes \cdots \otimes P_n$ such that $P_j \in \{I, X, Y, Z\}$. For brevity, the tensor-product symbol is often omitted so that P is denoted by the string of *I*, *X*, *Y* and *Z* characters. Each such character is called a *literal*.

Table 1.2 shows the multiplication table for \mathcal{P}_1 . One can check that the Pauli operators with phases are closed under multiplication, and their inverses are also Pauli operators. Therefore \mathcal{P}_n is a group. The string representation for Pauli operators in Definition I.8 allows us to compute the product of Pauli operators without explicitly computing the tensor products,¹ e.g., $(-IIXI)(iIYII) = -iIYXI$. Since

¹This holds true due to the following identity: $(A \otimes B)(C \otimes D) = (AC \otimes BD)$.

	I	X	Y	Z
I	I	X	Y	Z
X	X	I	iZ	$-iY$
Y	Y	$-iZ$	I	iX
Z	Z	iY	$-iX$	I

Table 1.2: Multiplication table for Pauli matrices. The shaded cells indicate anticommuting products.

$|\mathcal{P}_n| = 4^{n+1}$, \mathcal{P}_n can have at most $\log_2 |\mathcal{P}_n| = \log_2 4^{n+1} = 2(n+1)$ irredundant generators.

Lemma I.9. *Let $P = i^k P_1 \dots P_n \in \mathcal{P}_n$ and $Q = i^l Q_1 \dots Q_n \in \mathcal{P}_n$,*

(i) $P^2 = i^{2k} I$.

(ii) P and Q either commute ($PQ = QP$) or anticommute ($PQ = -QP$).

(iii) For a given P , half of the possible Q that can be selected will commute with it while the other half will anticommute.

Proof. (i) Table 1.2 shows that $\forall j P_j^2 = I$. Therefore, $P^2 = (i^k)^2 I \dots I = i^{2k} I$.

(ii) The case of single Pauli matrices can be checked using Table 1.2. Therefore, in the general case, each pair P_j and Q_j , $j \in \{1, \dots, n\}$ either commutes or anticommutes. Since the tensor components share the same global phase, P commutes with Q iff the number of anticommuting components is even; otherwise P anticommutes with Q .

(iii) Since commutation and anticommutation are determined by the parity of the number of anticommuting components, the commuting and anticommuting cases will occur as often.

□

Lemma I.10. *Let $P = P_1 \dots P_n$ be a Pauli-matrix tensor product such that $P \neq I^{\otimes n}$.*

(i) P has two eigenvalues ± 1 , each with a 2^{n-1} -dimensional eigenspace.

(ii) The 1- and (-1) -eigenvectors of P are orthogonal.

(iii) $\frac{I \pm P}{2}$ is a projector onto the ± 1 eigenspace.

Proof. (i) Consider the following two properties.

- Table 1.2 shows that $\forall j (P_j)^2 = I$. If $P_j \neq I$ then $\text{tr}(P_j) = 0$ and the two eigenvalues of P_j are ± 1 . Thus, each eigenvector of P_j has either $+1$ or -1 eigenvalue.
- Let the eigenvectors of P_i and P_k be $|\psi_i\rangle$ and $|\psi_k\rangle$ with corresponding eigenvalues λ_i and λ_k . Then the eigenvalue of $P_i P_k$ is equal to $\lambda_i \cdot \lambda_k$, which is the eigenvalue of $|\psi_i\rangle \otimes |\psi_k\rangle$.

Let $|\psi_{1,\dots,j}\rangle = |\psi_1\rangle \otimes \dots \otimes |\psi_j\rangle$ be an eigenvector of $P_1 \dots P_j$. By the first property, we define the *parity* of $|\psi_{1,\dots,j}\rangle$ as odd (even) if the number of -1 eigenvalues of its tensor components is odd (even). By the second property, the eigenvalue of $P_1 \dots P_j$ is given by the parity of $|\psi_{1,\dots,j}\rangle$.

Consider the base case $j = 1$. Since $P_1 \neq I$, the even parity of its eigenvector $|\psi_1\rangle$ occurs as often as its odd parity. Thus, each ± 1 eigenspace of P_1 has dimension $\frac{2^1}{2} = 1$. Assume that the parity of $|\psi_{1,\dots,j}\rangle$ is even (odd). Two cases need to be considered.

- $|\psi_{j+1}\rangle$ is an eigenvector of $P_{j+1} = I$ and $P_1 \dots P_j \neq I^{\otimes j}$. Then the parity of $|\psi_{1,\dots,j+1}\rangle$ remains even (odd).
- $|\psi_{j+1}\rangle$ is an eigenvector of $P_{j+1} \neq I$. If the eigenvalue of P_{j+1} is $+1$, then the parity of $|\psi_{1,\dots,j+1}\rangle$ remains even (odd). If the eigenvalue of P_{j+1} is -1 , then the parity of $|\psi_{1,\dots,j+1}\rangle$ is odd (even).

Since the even parity of $|\psi_{1,\dots,n}\rangle$ occurs as often as its odd parity, the dimension of each ± 1 eigenspace of P is $\frac{2^n}{2} = 2^{n-1}$.

(ii) Since P is self-adjoint, the 1- and (-1) -eigenvectors are orthogonal.

(iii) Note that $\left(\frac{I \pm P}{2}\right)^2 = \frac{I \pm P}{2}$ because $P^2 = I$. Consider an arbitrary vector $|\psi\rangle$.

Then $P\frac{I+P}{2}|\psi\rangle = \frac{P+I}{2}|\psi\rangle$, and $P\frac{I-P}{2}|\psi\rangle = \frac{P-I}{2}|\psi\rangle = -\frac{I-P}{2}|\psi\rangle$.

□

Definition I.11. Let A be an $n \times n$ square matrix. The *trace* of A is defined as the sum of its diagonal entries,

$$\mathrm{tr}(A) = \sum_i^n A_{ii} \quad (1.1)$$

Definition I.12. Let $L_{\mathcal{H}}$ be the set of linear operators on Hilbert space \mathcal{H} . Clearly $L_{\mathcal{H}}$ is a vector space. The function $\langle \cdot, \cdot \rangle$ on $L_{\mathcal{H}} \times L_{\mathcal{H}}$ is defined by

$$\langle A^\dagger, B \rangle := \frac{1}{2} \mathrm{tr}(A^\dagger B) \quad A, B \in L_{\mathcal{H}} \quad (1.2)$$

is called the *Hilbert-Schmidt inner product*.

It is not hard to show that the operator function from Definition I.12 satisfies the three axioms of an inner product.

Corollary I.13. *The set of linear operators on a complex \mathcal{H} is a Hilbert space under the Hilbert-Schmidt inner product.*

Theorem I.14. *The n -qubit Pauli group \mathcal{P}_n forms an orthonormal basis for the set of linear operators $L_{\mathcal{H}_N}$ on a Hilbert space \mathcal{H}_N , where $N = 2^n$.*

Proof. Recall that $L_{\mathcal{H}_N}$ is a Hilbert space (Corollary I.13). For $L_{\mathcal{H}_N}$ to fully span \mathcal{H}_N , it must be able to map any vector in \mathcal{H}_N into itself and into any other vector in \mathcal{H}_N . Thus there must be $N^2 = 4^n$ different linear operators. Since \mathcal{P}_n is a group of linear operators and $|\mathcal{P}_n| = 4^n$, it suffices to show that (i) for all $P \in \mathcal{P}_n$, $\langle P^\dagger, P \rangle = 1$ under the Hilbert-Schmidt inner product (Definition I.12), and (ii) for all $P, Q \in \mathcal{P}_n$, $P \neq Q$, $\langle P^\dagger, Q \rangle = 0$. Since Pauli operators are Hermitian, we can effectively ignore the Hermitian conjugate in the definition of the inner product.

(i) Clearly, the identity has norm 1 under the Hilbert-Schmidt inner product:

$\langle I, I \rangle = \frac{1}{2} \text{tr}(I \times I) = \frac{1}{2}(1 + 1) = 1$. By Theorem I.9, $P^2 = I$ for all $P \in \mathcal{P}$ thus we have: $\langle P, P \rangle = \frac{1}{2} \text{tr}(P \times P) = \frac{1}{2} \text{tr}(I) = 1$. Therefore, all Pauli operators have norm 1 under the Hilbert-Schmidt inner product.

(ii) By Definition I.8, the product of any two Pauli operators, up to a factor of $\pm i$, is another Pauli operator. Thus, let P, Q and L be distinct Pauli operators, we have: $\langle P, Q \rangle = \frac{1}{2} \text{tr}(P \times Q) = \frac{1}{2} \text{tr}(\pm iL) = \frac{\pm i}{2} \text{tr}(L)$. However, since any Pauli operator has zero trace, $\langle P, Q \rangle = 0$.

Therefore, \mathcal{P}_n forms an orthonormal basis over $L_{\mathcal{H}_N}$. □

Theorem I.14 implies that \mathcal{P}_n forms an *error basis* and thus any code that can correct errors in a subgroup of \mathcal{P}_n (e.g., I, X, Y , and Z in the one-qubit case) can then also correct an arbitrary linear combination of such errors. Theorem I.14 also implies that one can decompose an arbitrary operator into a linear combination of Pauli operators.

Definition I.15. The *Pauli expansion* of a linear operator U is $U_P = \sum_i \alpha_i P_i$ where $P_i \in \mathcal{P}$. If U is unitary, $|U| = \sum_i |\alpha_i|^2 = 1$.

Proposition I.16. *The Pauli expansion of a linear operator is unique.*

Proof. From the proof of Theorem I.14 we know that the terms in a Pauli expansion is are orthogonal. Thus, the expansion is unique. □

Definition I.17. Let U be linear operator acting on an n -qubit Hilbert space. We define the *Pauli expansion operator function* as

$$f_p(U) = \sum_{\forall P \in \mathcal{P}_n} \langle P, U \rangle P = \frac{1}{2^n} \text{tr}(P \times U) \tag{1.3}$$

Pauli expansions of quantum operators have useful applications in the context of quantum-circuit simulation. Such applications along with related software methods are discussed in Chapters II, V and VI.

1.6 Simulation of Quantum Circuits

Building on the background information about the quantum-circuit model and the properties of Pauli operators, we now turn to known techniques for efficient classical simulation of quantum circuits. As mentioned in Section 1.1, generic quantum-circuit simulation appears intractable for conventional computers. However, it may be unnecessary because useful quantum circuits exhibit significant structure that can be exploited during simulation. For example, Gottesman [34] and Knill identified an important subclass, called *stabilizer circuits*, which can be simulated efficiently using the *Heisenberg representation for quantum computers*. Stabilizer circuits are exclusively composed of *stabilizer gates* – Hadamard, Phase and controlled-NOT (Figure 1.5) – followed by one-qubit measurements in the computational basis. Such circuits are applied to a computational-basis state (usually $|00\dots 0\rangle$) and produce so-called *stabilizer states*. In the context of quantum-circuit simulation, the Heisenberg model is also known as the *stabilizer formalism* because it describes the evolution of *Pauli stabilizer² operators* rather than quantum states.

Only a polynomial number of such stabilizer operators are maintained during

²An operator U is a stabilizer for state $|\psi\rangle$ iff $U|\psi\rangle = |\psi\rangle$.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 1.5: The stabilizer gates: Hadamard (H), Phase (P) and controlled-NOT ($CNOT$). Note that $H^\dagger = H$ and $CNOT^\dagger = CNOT$. Furthermore $P^4 = I$, thus $P^\dagger = P^3$.

simulation and they uniquely represent stabilizer states up to an unobservable global phase. Because of their extensive applications in QECC, stabilizer circuits have been studied extensively. Equation 2.2 shows that the number of n -qubit stabilizer states grows as $2^{n^2/2}$, therefore, describing a generic stabilizer state requires at least $n^2/2$ bits. Thus, any simulation technique dealing directly with stabilizer states will use $\Omega(n^2)$ memory. Aaronson and Gottesman [2] proposed an improved technique that uses a bit-vector representation to simulate stabilizer circuits. This simulation technique was implemented by Aaronson in his software tool **CHP** [2], which simulates each stabilizer gate in $\Theta(n)$ time and $O(1)$ additional space. However, the overall execution time of **CHP** is dominated by the number of measurement gates, which require $O(n^2)$ time to simulate. In this work, we provide a comprehensive review of the stabilizer formalism, restating major results from published literature, with proofs, emphasizing efficient computation.

Anders and Briegel [5] designed a graph-based data structure that allows their software tool **GraphSim** to simulate measurement gates in $O(n \log^2 n)$ time. This improvement applies to circuits where the number of entangled qubits is $O(\log n)$. Each qubit is represented by a vertex in the graph, and there is an edge between every interacting pair of qubits. Any quantum state that can be represented using this graph-based approach is called a *graph state*. It has been shown that any stabilizer state is equivalent to a graph state under local stabilizer operations. For circuits that produce maximally-entangled states (worst case), the graph-based data structure from [5] does not offer an asymptotic efficiency improvement over **CHP**.

1.7 Objectives of the Dissertation

This dissertation seeks to expand the existing body of research in computational techniques including new data structures and algorithms for efficient simulation of quantum circuits. To this end, our work considers four main objectives which are

presented here along with an outline of key contributions.

(1) Study of geometric properties of stabilizer states. One of the objectives of our work is to provide a comprehensive analysis of the geometric structure of stabilizer states and their embedding in Hilbert space. Our analysis includes important results on the computational geometry of stabilizer states such as efficient algorithms for computing distances, angles and volumes between them. This line of research allowed us to identify efficient techniques for representing and manipulating new classes of quantum states, rule out some techniques as inefficient, and quantify entanglement of relevant states.

(2) Stabilizer-based simulation of generic quantum circuits. Another objective of our work is to generalize the stabilizer formalism to admit simulation of *non-stabilizer gates* such as Toffoli gates (Section 1.2). This line of research was first outlined in [2], where the authors describe a stabilizer-based representation that stores an arbitrary quantum state as a sum of density-matrix³ terms. In contrast, we *store arbitrary states as superpositions⁴ of pure stabilizer states*. Such superpositions are stored more compactly than the approach from [2], although we do not handle mixed stabilizer states. The key obstacle to the more efficient pure-state approach has been the need to maintain the global phase of each stabilizer state in a superposition, where such phases become relative. We develop a new algorithm to overcome this obstacle. The main advantages of using stabilizer-state superpositions to simulate quantum circuits are:

(i) *Stabilizer subcircuits are simulated with high efficiency.*

(ii) *Superpositions can be restructured and compressed on the fly during simulation*

³Density matrices are self-adjoint positive-semidefinite matrices of trace 1.0, that describe the statistical state of a quantum system [51].

⁴A superposition is a norm-1 linear combination of terms.

to reduce resource requirements.

(iii) *Operations performed on such superpositions can be computed in parallel and lend themselves to distributed or asynchronous processing.*

To complete our study of known stabilizer-based simulation techniques, we describe and analyze the *stateless* approach from [39], which simulates stabilizer circuits efficiently without maintaining a state representation. We explore Pauli decompositions of several non-stabilizer operators and design a technique to represent them compactly using *Pauli multivalued decision diagrams*. We use these Pauli expansions to generalize the technique from [39] and develop stateless simulation of generic quantum circuits. We discuss the advantages and limitations of such an approach, and compare it to our techniques based on superpositions of stabilizer states.

(3) Compact representation of new classes of quantum states. Vidal [68] established a necessary condition for a quantum algorithm to defy efficient classical simulation. This condition demands that the amount of entanglement generated by the algorithm on n qubits grow faster than $\log(n)$. In other words, if an algorithm does not generate sufficiently high entanglement, it can be simulated efficiently. Since stabilizer states can be maximally entangled, entangled states are not sufficient to prevent efficient simulation. Therefore, a key objective of our work is to: (i) facilitate simulation of new classes of entangled states and circuits that generate them, and (ii) identify classes of states that cannot be simulated using our stabilizer-based approach and may therefore suggest new types of quantum speedups.

Our stabilizer-based technique simulates certain quantum arithmetic and quantum Fourier transform circuits in polynomial time and space for input states consisting of equal superpositions of computational-basis states. Therefore, we demonstrate that the quantum states generated by such circuits are not sufficient to provide computational speedups. By comparison, simulating these same circuits using well-known

generic techniques takes exponential time.

(4) Fast simulation of quantum fault-tolerant quantum circuits. As mentioned in Section 1.1, quantum circuits require functional simulation to determine the best fault-tolerant (FT) design choices given limited resources. In particular, *high-performance simulation* is a key component in quantum design flows [64] that facilitate analysis of trade-offs between performance and accuracy. Since the state-vector representation outlined in Section 1.2 requires excessive computational resources in general, simulation-based reliability studies (e.g. fault-injection analysis) of quantum FT architectures using general-purpose simulators has been limited to small quantum circuits [24]. Therefore, designing fast simulation methods that target quantum FT circuits facilitates robust reliability analysis of larger quantum circuits. Quantum FT circuits are dominated by stabilizer subcircuits and contain a relatively small number of non-stabilizer components. Since our simulation techniques exploit speedups offered by the stabilizer formalism, they facilitate parallel simulation of such circuits. We simulate various quantum FT circuits, and the results demonstrate that our techniques lead to orders-of-magnitude improvement in runtime and memory as compared to state-of-the-art simulators.

1.8 Dissertation Outline

This dissertation is structured as follows. In Chapter II we review the stabilizer formalism following [13, 14, 33, 34, 51] in detail, and describe techniques for simulating stabilizer circuits and equivalence checking [2, 5]. We also define the notion of *stateless simulation* for stabilizer circuits as discussed in [8, 11, 39].

Our findings related to the geometric structure of stabilizer states and their embedding in Hilbert space are described in Chapter III. In particular, we characterize the nearest-neighbor structure of stabilizer states and explore the approximation of

non-stabilizer states by single stabilizer states and short superpositions of stabilizer states. In Chapter IV, we exploit our analysis of stabilizer-state geometry to design algorithms for the following fundamental tasks: (i) synthesizing new *canonical stabilizer circuits* that reduce the number of gates required for QECC encoding procedures, (ii) computing the inner product between stabilizer states, and (iii) computing stabilizer bivectors.

In Chapter V, we describe our proposed techniques for representing, manipulating and compressing superpositions of stabilizer states. This includes our primary data structure called *stabilizer frames*, which offer more compact storage than previous approaches but require more sophisticated bookkeeping. We use several quantum circuits as benchmarks to evaluate the performance of our technique and the results indicate that stabilizer frames enable efficient simulation of certain instances of quantum circuits that act on input computational-basis states. On such instances, known linear-algebraic simulation techniques, such as the (state-of-the-art) BDD-based simulator **QuIDDP**, take exponential time. We also describe significant extensions that yield improved performance on additional benchmarks.

Chapter VI describes our technique for generalizing the stateless approach from [8, 11, 39] to admit simulation of generic quantum circuits. We perform comparisons and show that this technique is outperformed by stabilizer frames for practical benchmarks. In Chapter VII, we describe matrix product states and derive a graph-based representation for such states. The results of our work are summarized in Chapter VIII, which also discusses open questions and directions for further research.

CHAPTER II

Simulation Techniques and the Heisenberg Representation

In general, to simulate a quantum circuit C , we first initialize the quantum system to some desired state $|\psi\rangle$ (usually a basis state). $|\psi\rangle$ can be represented using a fixed-size data structure (e.g., an array of 2^n complex numbers) or a variable-size data structure (e.g., an Algebraic Decision Diagram). We then track the evolution of $|\psi\rangle$ via its internal representation as the gates in C are applied until one obtains the output state $C|\psi\rangle$ [2, 51, 67]. Therefore, most quantum-circuit simulators such as QCL, QuIDDPPro and the parallel simulators proposed in [52] and [22] support some form of the linear-algebraic operations described above. The drawback of such simulators is that their runtime grows exponentially in the number of qubits [67]. This holds true not only in the worst case but also in many practical applications involving fault-tolerant circuits. For example, although QuIDDPPro uses QuIDD's to store some state vectors more compactly, it takes an average of 18 minutes to simulate 50-qubit circuits that are representative of QECC implementations (Section 1.4).

Consequently, researchers have focused on finding non-trivial classes of quantum circuits that can be simulated efficiently on conventional computers. One example is the class of *stabilizer circuits*, which has important applications in the design of fault-tolerant quantum architectures. Gottesman [34] developed a simulation method

for such circuits involving the *Heisenberg representation* often used by physicists to describe atomic phenomena. *In this model, one keeps track of the symmetries of an object rather than represent the object explicitly.* In the context of quantum-circuit simulation, this model represents quantum states by their symmetries, rather than complex vectors. The symmetries are operators for which these states are 1-eigenvectors. Algebraically, symmetries form *group* structures, which can be specified compactly by group generators. Taking advantage of this fact, Gottesman showed that his technique requires only polynomial memory and runtime when simulating stabilizer circuits [33]. We discuss this technique in Section 2.2. However, the gates that comprise a stabilizer circuit do not form a universal set for quantum computing. Shi [60] demonstrated that to make the set of stabilizer gates universal one can add any single-qubit gate type that does not preserve the computational basis (e.g., the T gate). In Chapter V, we describe *simulation techniques that admit a universal gate library while taking advantage of the speed-ups offered by the stabilizer formalism.*

2.1 Classification of Simulation Techniques

In this section, we describe several properties of known simulation techniques. Such properties can be used to categorize distinct simulation approaches in order to facilitate performance comparisons and resource trade-offs.

2.1.1 Strong versus weak

The work in [14, 40] describes two notions of classical simulation for quantum computation. *Weak simulation* refers to a set of classical randomized computational tasks which, given a description of quantum process Q as input, outputs a *sample* of the output distribution $p(\mathbf{y}) = p(y_{j_1}, \dots, y_{j_l})$ of Q . The string \mathbf{y} denotes the output obtained from a final measurement on a specified set $1 \leq j_1 < \dots < j_l \leq n$ for l lines in an n -qubit circuit. In contrast, a *strong simulation* approach is a classical

computation whose input is a description of Q and bit values for the subset of its output lines. The output is the value of the corresponding marginal probability of $p(\mathbf{y})$. Therefore, such an approach yields a classical computation of any desired output probability or marginal probability of Q [40]. Strong simulation is preferable over a weak approach because it facilitates studying intermediate states generated by a quantum algorithm, e.g., estimates of the amount of quantum entanglement in such states. In the rest of this document, the term simulation refers to the strong sense unless otherwise noted.

2.1.2 Stateless versus direct

Thus far, our notion of simulation has been concerned with maintaining a description of some quantum process Q . For example, we can maintain a full representation of the quantum state $|\psi\rangle$, and then record the updates to $|\psi\rangle$ as defined by circuit \mathbf{C} . Therefore, we regard $\mathbf{C}|\psi\rangle$ as the output of a specific computation performed on input state $|\psi\rangle$. If \mathbf{C} can be *simulated in polynomial time* on a classical computer, the computation defined by \mathbf{C} can also be simulated efficiently. We now consider a different notion of quantum simulation based on the techniques outlined in [8, 11, 39].

Definition II.1 (Stateless simulation). The simulation of an n -qubit quantum circuit \mathbf{C} on state $|\psi\rangle$ is considered *stateless* if it computes the probabilities of measurement outcomes to d digits without directly computing $\mathbf{C}|\psi\rangle$. Such an approach is termed efficient if the output probabilities are computed in $\text{poly}(n, d)$ time.

Observe that stateless simulation does not maintain a quantum-state representation during simulation. This is in contrast to a *direct simulation* approach which does maintain a description of the quantum state. Although stateless simulation seems counter-intuitive (How does one determine measurement probabilities without knowing the state of the system?), it emulates how a user would interface with an actual quantum computer since a user would not have direct access to the quantum state.

To see how stateless simulation works, consider an arbitrary *decision problem*, i.e., a problem where the answer is either “yes” or “no.” Many interesting problems are either explicitly decision problems (e.g., satisfiability) or can be recast as equivalent decision problems (e.g., travelling salesman, factoring). In particular, it is possible to simulate the quantum computation of such problems by maintaining only a single matrix element of the circuit that defines the computation. Let \mathbf{C}_x be a quantum circuit that computes $f(x)$ for an instance of a decision problem. Without loss of generality assume that the output is indicated by the value of the first qubit $|q_1 := f(x)\rangle$, i.e., if $|q_1 = 0\rangle$, the answer is “no” and “yes” otherwise. The state of the remaining output qubits can be safely ignored. We make use of an ancilla qubit $|q_a\rangle$ appended to the input register and initialized to $|0\rangle$. We apply a CNOT with control $|q_1\rangle$ and target $|q_a\rangle$ so that the answer $f(x)$ is copied into $|q_a\rangle$. Then, we apply the inverse operation \mathbf{C}_x^\dagger as shown in Figure 2.1 to obtain the output state $|0\rangle_{q_1} |0\rangle_{q_2} \dots |0\rangle |f(x)\rangle_{q_a}$. Therefore, any quantum circuit for an instance of a decision problem outputs one of two possible states: $|00\dots 0\rangle |0\rangle$ or $|00\dots 0\rangle |1\rangle$. This implies that estimating the single matrix element $m = \langle 00\dots 0 | \mathbf{C}_x | 00\dots 0 \rangle$ is sufficient to simulate the computation since this gives the output probability. In the stateless simulation model one seeks to estimate m without explicitly computing $\mathbf{C}_x | 00\dots 0 \rangle$. If m can be computed in poly-time and space on a classical computer, the simulation is considered efficient in the sense of Definition II.1. However, in general, estimating m on a classical computer to some constant precision requires an exponential number of trials [11]. Nonetheless,

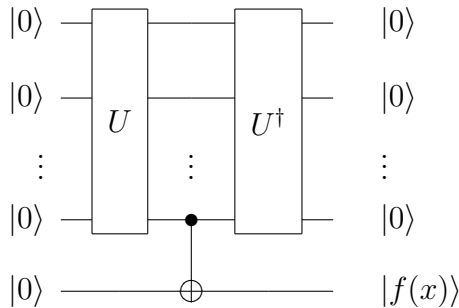


Figure 2.1: General quantum circuit for decision problems.

Section 2.2 describes a stateless approach that simulates Clifford (stabilizer) circuits efficiently on classical computers.

2.2 The Stabilizer Formalism

A unitary operator U is said to *stabilize* a state $|\psi\rangle$ if $U|\psi\rangle = |\psi\rangle$. We will mostly be interested in operators U derived from the Pauli matrices shown in Figure 1.3. For example, the Pauli matrix Z stabilizes the state $|0\rangle$ since $Z|0\rangle = |0\rangle$, while the Pauli matrix X *destabilizes* it because $X|0\rangle = |1\rangle$. Furthermore, observe that I stabilizes all states and $-I$ does not stabilize any state. Another example is the entangled state $(|00\rangle + |11\rangle)/\sqrt{2}$, which is stabilized by the Pauli operators XX , $-YY$ and ZZ .

The stabilizer formalism allows one to represent some quantum states compactly by keeping track of the Pauli operators that stabilize them, rather than use an exponential set of complex amplitudes. Such compact representations are based on the fact that the Pauli matrices along with the identity I form a group.

Definition II.2. The *stabilizer group* of a pure state $|\psi\rangle$, denoted by $S(|\psi\rangle)$, is the group of Pauli operators (subgroup of \mathcal{P}_n) that stabilize $|\psi\rangle$.

In this document, all quantum states are assumed to be pure unless indicated otherwise.

Theorem II.3. For an n -qubit pure state $|\psi\rangle$, there exists a $k \leq n$ such that $S(|\psi\rangle) \cong \mathbb{Z}_2^k$. If $k = n$, $|\psi\rangle$ is specified uniquely by $S(|\psi\rangle)$ and is called a *stabilizer state*.

Proof. (i) To prove that $S(|\psi\rangle)$ is commutative, let $P, Q \in S(|\psi\rangle)$ such that $PQ|\psi\rangle = |\psi\rangle$. If P and Q anticommute, $-QP|\psi\rangle = -Q(P|\psi\rangle) = -Q|\psi\rangle = -|\psi\rangle \neq |\psi\rangle$. Thus, P and Q cannot both be elements of $S(|\psi\rangle)$.

(ii) To prove that every element of $S(|\psi\rangle)$ is of degree 2, let $P \in S(|\psi\rangle)$ such that $P|\psi\rangle = |\psi\rangle$. Observe that $P^2 = i^l I$ for $l \in \{0, 1, 2, 3\}$. Since $P^2|\psi\rangle = P(P|\psi\rangle) = P|\psi\rangle = |\psi\rangle$, we obtain $i^l = 1$ and $P^2 = I$.

(iii) From group theory, a finite Abelian group with $a^2 = a$ for every element must be $\cong \mathbb{Z}_2^k$.

(iv) We now prove that $k \leq n$. First note that each independent generator $P \in S(|\psi\rangle)$ imposes the linear constraint $P|\psi\rangle = |\psi\rangle$ on the 2^n -dimensional vector space. The subspace of vectors that satisfy such a constraint has dimension 2^{n-1} , or half the space. Let $gen(|\psi\rangle)$ be the set of generators for $S(|\psi\rangle)$. We add independent generators to $gen(|\psi\rangle)$ one by one and impose their linear constraints, to limit $|\psi\rangle$ to the shared 1-eigenvector. Thus the size of $gen(|\psi\rangle)$ is at most n . In the case $|gen(|\psi\rangle)| = n$, the n independent generators reduce the subspace of possible states to dimension one. Thus, $|\psi\rangle$ is uniquely specified. \square

The proof of Theorem II.3 shows that $S(|\psi\rangle)$ is specified by only $\log_2 2^n = n$ *irredundant stabilizer generators*.

Definition II.4. A *stabilizer matrix* \mathcal{M} for n -qubit stabilizer state $|\psi\rangle$ is a matrix of Pauli literals, where the rows represent the generators R_1, \dots, R_n of $S(|\psi\rangle)$. Since each R_i is a string of n Pauli literals, the size of the matrix is $n \times n$.

In the rest of this document, we use the terms *stabilizer matrix* and *generator set* interchangeably. The fact that $R_i \in S(|\psi\rangle)$ implies that the leading phase of R_i can only be ± 1 and not $\pm i$.¹ Therefore, we store the phases of each R_i separately using a binary vector of size n .

The storage cost for \mathcal{M} is $\Theta(n^2)$, which is an *exponential improvement* over the $O(2^n)$ cost often encountered in vector-based representations.

Example II.5. The state $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is uniquely specified by any of the following stabilizer matrices: $\mathcal{M}_1 = \begin{smallmatrix} + \\ + \end{smallmatrix} \begin{bmatrix} XX \\ ZZ \end{bmatrix}$, $\mathcal{M}_2 = \begin{smallmatrix} + \\ - \end{smallmatrix} \begin{bmatrix} XX \\ YY \end{bmatrix}$, $\mathcal{M}_3 = \begin{smallmatrix} - \\ + \end{smallmatrix} \begin{bmatrix} YY \\ ZZ \end{bmatrix}$.

¹Suppose the phase of R_i is $\pm i$, then $R_i^2 = -I \in S(|\psi\rangle)$ which is not possible since $-I$ does not stabilize any state.

Theorem II.3 suggests that Pauli literals can be represented using only two bits, e.g., $00 = I$, $01 = Z$, $10 = X$ and $11 = Y$. Therefore, a stabilizer matrix can be encoded using an $n \times 2n$ binary matrix or *tableau*. The advantage of this approach is that this literal-to-bits mapping induces an isomorphism $\mathbb{Z}_2^{2n} \rightarrow \mathcal{P}_n$ because vector addition in \mathbb{Z}_2^2 is equivalent to multiplication of Pauli operators up to a global phase. The tableau implementation of the stabilizer formalism is covered in Section 2.3.1.

2.2.1 Stabilizer gates and circuits

Proposition II.6. Consider n -qubit state $|\psi\rangle$ stabilized by the Pauli operator P , and $U \in \mathbb{U}(2^n)$. Then the state $U|\psi\rangle$ is stabilized by UPU^\dagger .

Proof. Since $P|\psi\rangle = |\psi\rangle$, we have $(UPU^\dagger)U|\psi\rangle = UP|\psi\rangle = U|\psi\rangle$. \square

Definition II.7. Let $P \in \mathcal{P}_n$. The *Clifford group* \mathcal{C}_n on n qubits consists of the operators $U \in \mathbb{U}(2^n)$ such that $UPU^\dagger \in \mathcal{P}_n$.

The group \mathcal{C}_n can be generated by *CNOT*, *H* and *P* gates [7], which all conjugate the Pauli group \mathcal{P}_n into itself (Table 2.1a). In fact, any Pauli matrix can be expressed as a sequence of Clifford operations as shown on Table 2.1b. Furthermore, Table 2.2 suggests that any permutation and sign change of Pauli operators can also be achieved

OPERATION	INPUT	OUTPUT	OPERATION	INPUT	OUTPUT
<i>H</i>	<i>X</i>	<i>Z</i>	<i>CNOT</i>	$I_1 X_2$	$I_1 X_2$
	<i>Y</i>	$-Y$		$X_1 I_2$	$X_1 X_2$
	<i>Z</i>	<i>X</i>		$I_1 Y_2$	$Z_1 Y_2$
<i>P</i>	<i>X</i>	<i>Y</i>		$Y_1 I_2$	$Y_1 X_2$
	<i>Y</i>	$-X$		$I_1 Z_2$	$Z_1 Z_2$
	<i>Z</i>	<i>Z</i>		$Z_1 I_2$	$Z_1 I_2$

CLIFFORD SEQUENCE	PAULI MATRIX
HP^2H	<i>X</i>
PHP^2HP^3	<i>Y</i>
P^2	<i>Z</i>

(a)
(b)

Table 2.1: (a) Transformation properties of the Pauli-group elements under conjugation by the Clifford operators [51]. In the *CNOT* case, subscript 1 indicates the control and 2 the target. (b) Pauli matrices expressed as Clifford sequences.

ONE-QUBIT CLIFFORD SEQUENCES	PAULI MATRICES			INTERPRETATION
Identity	X	Y	Z	No action
$\alpha : PH$	Z	X	Y	$X \rightarrow Y \rightarrow Z$ rotation
$\beta : PHP^3$	$-X$	Z	Y	YZ swap with X sign flip

TWO-QUBIT CLIFFORD SEQUENCE	PAULI MATRICES			INTERPRETATION
Identity	X_1X_2	Y_1Y_2	Z_1Z_2	No action
$\gamma_{1,2} : P_1H_1C_{1,2}P_2H_2P_2^3C_{2,1}P_2^2H_2P_1H_1$	$-X_1X_2$	Y_1Y_2	Z_1Z_2	X sign flip

Table 2.2: Clifford sequences for generating an arbitrary permutation of the Pauli operators. Subscripts indicate qubit indices. Given two different generators, such as X_1Y_2 one can apply one qubit rotation(s) to obtain homogenous generators, such as Y_1Y_2 , and then apply a two-qubit sequence. A pure YZ swap is achieved by conjugating β and γ . Conjugating the pure YZ swap with γ achieves any pure swap, e.g., $\beta\gamma\alpha$ achieves an XY swap.

by applying the proper sequence of Clifford gates. For example, let Q be an n -tensor Pauli operator, then $\gamma_{i,j}\beta_jQ$ achieves a pure Y - Z swap on the j^{th} Pauli matrix of Q . All other pure swaps can be achieved by combining the Y - Z swap ($\gamma\beta$) with rotations (α). Using pure swaps we can generate any permutation of Pauli operators. We can also combine the X sign change (γ) with permutations to obtain other sign changes. Thus, an arbitrary set of sign changes is also possible.

The $CNOT$, P and H gates are commonly called *stabilizer gates* because their action on stabilizer states can be directly simulated via the stabilizer formalism (up to a global phase) as shown in Example II.8. One can also simulate the Controlled- Y and Controlled- Z ($CPHASE$) operations since these gates can be implemented by conjugating Pauli matrices and Clifford sequences as shown in Figure 2.2.

Example II.8. Figure 2.3 shows the circuit for generating an EPR pair and the stabilizer-matrix updates after simulating each gate.

Theorem II.9. [7, Section 2] *The number of elements in \mathcal{C}_n is*

$$|\mathcal{C}_n| = 2^{n^2+2n+3} \prod_{j=1}^n (4^j - 1) \quad (2.1)$$

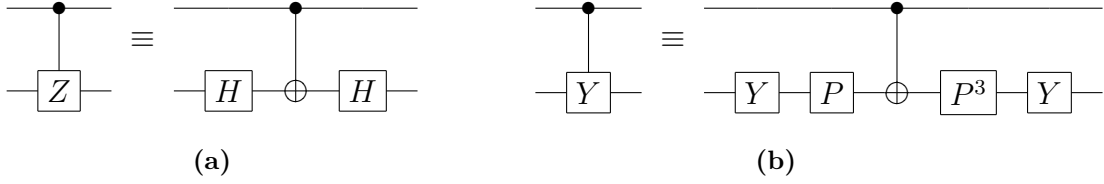


Figure 2.2: Implementation of the (a) Controlled-Z (*CPHASE*) and (b) Controlled-Y operations using Pauli and Clifford gates. These gates can be simulated directly on the stabilizer or using the equivalences shown here.

Definition II.10. A *stabilizer* or *Clifford circuit* is an array of quantum gates composed exclusively of *CNOT*, *P*, *H* and one-qubit computational basis measurements (*Z* measurements). A *unitary stabilizer circuit* is a stabilizer circuit that does not include measurement gates. A *basis-preserving stabilizer circuit* is a unitary stabilizer circuit composed exclusively of *CNOT*, *CPHASE* and *P* gates.

2.2.2 Stabilizer states

As shown in Theorem II.3, an n -qubit stabilizer state can be represented by specifying n irredundant n -tensor Pauli generators of its stabilizer group. The following theorem-definition specifies properties of stabilizer states relevant to quantum-circuit simulation.

Theorem-definition II.11. Any n -qubit stabilizer state $|\psi\rangle$ can be obtained by applying a stabilizer circuit to the state $|0\rangle^{\otimes n}$. This class of states is equivalently characterized by each of the following properties [2]:

$$\begin{array}{c}
 \begin{array}{c}
 \boxed{H} \text{---} \bullet \\
 \downarrow \quad \downarrow \quad \downarrow \\
 \downarrow \quad \oplus \quad \downarrow \\
 |\psi\rangle \quad |\psi'\rangle \quad |\psi''\rangle
 \end{array}
 \quad \parallel \quad
 \begin{array}{l}
 |\psi\rangle = |00\rangle \equiv \begin{array}{l} R_1 \begin{bmatrix} Z & I \\ I & Z \end{bmatrix} \\ R_2 \begin{bmatrix} Z & I \\ I & Z \end{bmatrix} \end{array} \\
 |\psi'\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}} \equiv \begin{array}{l} R_1 \begin{bmatrix} X & I \\ I & Z \end{bmatrix} \\ R_2 \begin{bmatrix} X & I \\ I & Z \end{bmatrix} \end{array} \\
 |\psi''\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \equiv \begin{array}{l} R_1 \begin{bmatrix} X & X \\ Z & Z \end{bmatrix} \\ R_2 \begin{bmatrix} X & X \\ Z & Z \end{bmatrix} \end{array}
 \end{array}
 \end{array}$$

Figure 2.3: Representation of the states generated by the EPR-pair circuit using stabilizer matrices. The phase array is omitted for simplicity. The rows of the matrices $\{R_1, R_2\}$ denote the stabilizer generators of the corresponding two-qubit state.

(i) $|\psi\rangle$ can be obtained from $|0\rangle^{\otimes n}$ by *CNOT*, *H* and *P* gates only without measurement gates.

(ii) $|\psi\rangle$ is stabilized by exactly 2^n Pauli operators.

(iii) $S(|\psi\rangle) \cong \mathbb{Z}_2^n$.

(iv) $|\psi\rangle$ is uniquely determined by $S(|\psi\rangle)$ according to Definition II.2.

Proof. The fact that the addition or removal of measurement gates does not affect the set of stabilizer states is demonstrated on the next page. Therefore, we only need to show the equivalence of definitions (i), (ii), (iii) and (iv).

To show that (i) \Rightarrow (ii), note that the $|0\rangle^{\otimes n}$ state is stabilized by any and all n -tensor products of *I* and *Z*. Furthermore, when a *CNOT*, *H* or *P* gate is applied, the stabilizers change according to Table 2.1a. Verifying that this table defines a group isomorphism also proves (i) \Rightarrow (iii).

To prove that (ii) \Rightarrow (iii), note that the stabilizers form the group $S(|\psi\rangle)$. Since the size of $S(|\psi\rangle)$ is 2^n , the group must be \mathbb{Z}_2^n .

To prove that (iii) \Rightarrow (iv) one uses an argument similar to that used in the proof of Theorem II.3-iv. Namely, each generator imposes a linear constraint on any given stabilizer state, and reduces the dimension of the possible of states by a factor of two. Therefore, n independent generators reduce the subspace of possible states to dimension one.

To prove that (iv) \Rightarrow (i), the work in [2] represents the generators using a tableau, and then shows how to construct a unitary stabilizer circuit from the tableau. We refer the reader to [2, Theorem 8] for details of the proof. The tableau approach to stabilizer simulation is discussed in Section 2.3.1. \square

Given Theorem-definition II.11-iv, we can track the evolution of $|\psi\rangle$, under the action of any given unitary stabilizer circuit \mathbf{C} , by conjugating each stabilizer gate in \mathbf{C} with the generators of $S(|\psi\rangle)$.

Corollary II.12. Any n -qubit stabilizer state $|\psi\rangle$ can be transformed by unitary stabilizer gates into the $|0\rangle^{\otimes n}$ state.

Proof. Since every stabilizer state can be produced by applying some unitary stabilizer circuit \mathbf{C} to the $|0\rangle^{\otimes n}$ state, it suffices to reverse \mathbf{C} to perform the inverse transformation. To reverse a stabilizer circuit, reverse the order of gates and replace every P gate with P^3 . \square

Tables 2.3 and 2.4 list all one- and two-qubit stabilizer states, respectively.

Proposition II.13. [2, Proposition 2] The number of n -qubit pure stabilizer states is given by

$$N(n) = 2^n \prod_{k=0}^{n-1} (2^{n-k} + 1) = 2^{(.5+o(1))n^2} \quad (2.2)$$

Proof. Let $G(n)$ be the total number of generating sets and $A(n)$ the number of equivalent generating sets for stabilizer S . Then $N(n) = G(n)/A(n)$. To calculate $G(n)$, recall from Section 1.5 that $|\mathcal{P}_n| = 4^n$. Thus there are $4^n - 1$ choices for the first generator g_1 since it can be anything but I . g_2 cannot be g_1 or I and, by Theorem II.3-*i*, it must commute with g_1 . According to Lemma I.9-*iii*, half of the possible generators will commute, which gives $4^n/2 - 2$ choices for g_2 . In general, g_k must commute with $g_i, i \in \{1, \dots, k-1\}$ and cannot be in the group generated by them. Therefore, the number of choices for g_k is $4^n/2^k - 2^k$. Since each of the n generators has a ± 1 multiplicative factor, the number of sign configurations is 2^n . Thus $G(n) = 2^n \prod_{k=0}^{n-1} \left(\frac{4^n}{2^k} - 2^k\right)$. To calculate $A(n)$, note that given S , there are

STABILIZER STATE	SHORTHAND NOTATION	PAULI GENERATOR	STABILIZER STATE	SHORTHAND NOTATION	PAULI GENERATOR
$(0\rangle + 1\rangle)/\sqrt{2}$	$1, 1$	X	$(0\rangle - 1\rangle)/\sqrt{2}$	$1, -1$	$-X$
$(0\rangle + i 1\rangle)/\sqrt{2}$	$1, i$	Y	$(0\rangle - i 1\rangle)/\sqrt{2}$	$1, -i$	$-Y$
$ 0\rangle$	$\mathbf{1}, \mathbf{0}$	\mathbf{Z}	$ 1\rangle$	$\mathbf{0}, \mathbf{1}$	$-\mathbf{Z}$

Table 2.3: One-qubit stabilizer states and their Pauli-matrix stabilizer. Shorthand notation lists only the normalized amplitudes of the basis states.

	STATE	GEN'TORS	\angle	STATE	GEN'TORS	\angle	STATE	GEN'TORS	\angle	STATE	GEN'TORS	\angle
SEPARABLE	1, 1, 1, 1	IX, XI	$\pi/3$	1, -1, 1, -1	-IX, XI	$\pi/3$	1, 1, -1, -1	IX, -XI	$\pi/3$	1, -1, -1, 1	-IX, -XI	$\pi/3$
	1, 1, 1, i	IX, YI	$\pi/3$	1, -1, i, -i	-IX, YI	$\pi/3$	1, 1, -i, -i	IX, -YI	$\pi/3$	1, -1, -i, i	-IX, -YI	$\pi/3$
	1, 1, 0, 0	IX, ZI	$\pi/4$	1, -1, 0, 0	-IX, ZI	$\pi/4$	0, 0, 1, 1	IX, -ZI	\perp	0, 0, 1, -1	-IX, -ZI	\perp
	1, i, 1, i	IY, XI	$\pi/3$	1, -i, 1, -i	-IY, XI	$\pi/3$	1, i, -1, -i	IY, -XI	$\pi/3$	1, -i, -1, i	-IY, -XI	$\pi/3$
	1, i, 1, -1	IY, YI	$\pi/3$	1, -i, i, 1	-IY, YI	$\pi/3$	1, i, -i, 1	IY, -YI	$\pi/3$	1, -i, -i, -1	-IY, -YI	$\pi/3$
	1, i, 0, 0	IY, ZI	$\pi/4$	1, -i, 0, 0	-IY, ZI	$\pi/4$	0, 0, 1, i	IY, -ZI	\perp	0, 0, 1, -i	-IY, -ZI	\perp
	1, 0, 1, 0	IZ, XI	$\pi/4$	0, 1, 0, 1	-IZ, XI	\perp	1, 0, -1, 0	IZ, -XI	$\pi/4$	0, 1, 0, -1	-IZ, -XI	\perp
	1, 0, i, 0	IZ, YI	$\pi/4$	0, 1, 0, i	-IZ, YI	\perp	1, 0, -i, 0	IZ, -YI	$\pi/4$	0, 1, 0, -i	-IZ, -YI	\perp
	1, 0, 0, 0	IZ, ZI	0	0, 1, 0, 0	-IZ, ZI	\perp	0, 0, 1, 0	IZ, -ZI	\perp	0, 0, 0, 1	-IZ, -ZI	\perp
	ENTANGLED	0, 1, 1, 0	XX, YY	\perp	1, 0, 0, -1	-XX, YY	$\pi/4$	1, 0, 0, 1	XX, -YY	$\pi/4$	0, 1, -1, 0	-XX, -YY
1, 0, 0, i		XY, YX	$\pi/4$	0, 1, i, 0	-XY, YX	\perp	0, 1, -i, 0	XY, -YX	\perp	1, 0, 0, -i	-XY, -YX	$\pi/4$
1, 1, 1, -1		XZ, ZX	$\pi/3$	1, 1, -1, 1	-XZ, ZX	$\pi/3$	1, -1, 1, 1	XZ, -ZX	$\pi/3$	1, -1, -1, -1	-XZ, -ZX	$\pi/3$
1, i, 1, -i		XZ, ZY	$\pi/3$	1, i, -1, i	-XZ, ZY	$\pi/3$	1, -i, 1, i	XZ, -ZY	$\pi/3$	1, -i, -1, -i	-XZ, -ZY	$\pi/3$
1, 1, i, -i		YZ, ZX	$\pi/3$	1, 1, -i, i	-YZ, ZX	$\pi/3$	1, -1, i, i	YZ, -ZX	$\pi/3$	1, -1, -i, -i	-YZ, -ZX	$\pi/3$
1, i, i, 1		YZ, ZY	$\pi/3$	1, i, -i, -1	-YZ, ZY	$\pi/3$	1, -i, i, -1	YZ, -ZY	$\pi/3$	1, -i, -i, 1	-YZ, -ZY	$\pi/3$

Table 2.4: Sixty two-qubit stabilizer states and their corresponding Pauli generators. Shorthand notation represents a stabilizer state as $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ where α_i are the normalized amplitudes of the basis states. The basis states are emphasized in bold. The first column lists states whose generators do not include an upfront minus sign, and other columns introduce the signs. A sign change creates an orthogonal vector. Therefore, each row of the table gives an orthogonal basis. The cells in dark grey indicate stabilizer states with four non-zero basis amplitudes, i.e., $\alpha_i \neq 0 \forall i$. The \angle column indicates the angle between that state and $|00\rangle$, which has twelve nearest-neighbor states (light grey) and 15 orthogonal states (\perp). All three-qubit stabilizer states are listed in Appendix A.

$2^n - 2^k$ choices for g_k . Thus $A(n) = \prod_{k=0}^{n-1} (2^n - 2^k)$. Finally, we have

$$N(n) = \frac{G(n)}{A(n)} = 2^n \prod_{k=0}^{n-1} \left(\frac{4^n - 2^k}{2^n - 2^k} \right) = 2^n \prod_{k=0}^{n-1} (2^{n-k} + 1) \quad (2.3)$$

□

An alternate interpretation of Equation 2.2 is given by the simple recurrence relation $\mathcal{N}(n) = 2(2^n + 1)\mathcal{N}(n-1)$ with base case $\mathcal{N}(1) = 6$. For example, for $n = 2$ the number of stabilizer states is 60, and for $n = 3$ it is 1080. This recurrence relation stems from the fact that there are $2^n + 1$ ways of combining the generators of the $\mathcal{N}(n-1)$ states with additional Pauli matrices to form valid n -qubit generators. The factor of 2 accounts for the increase in the number of possible sign configurations. Table 2.4 and Appendix A list all two-qubit and three-qubit stabilizer states, respectively.

Definition II.14. The *lexicographic ordering* of stabilizer states is defined as follows:

- We order generator-based representations of stabilizer states, which are phase-invariant. When listing the amplitudes for such states, we normalize the global phase so that the first non-zero amplitude is 1.0 (i.e., this normalization does not affect the ordering defined in terms of generators).
- We write the tensor products of n Pauli matrices as strings and order them lexicographically (for the leading phase, we assume $\alpha \prec -\alpha$).
- For a given stabilizer state, we consider different generating sets (each ordered lexicographically) — of those we choose the lexicographically smallest.
- When dealing with multiple stabilizer states, we order them lexicographically in terms of their lexicographically smallest generating set.

This ordering is illustrated in Table 2.4 for all two-qubit stabilizer states. For reference, the list of all three-qubit stabilizer states is shown in Appendix A.

Proposition II.15. *The tensor product $|\psi\rangle \otimes |\varphi\rangle$ of two stabilizer states $|\psi\rangle, |\varphi\rangle$ is a stabilizer state.*

Proof. Consider two Pauli operators, P and Q , which stabilize the n -qubit state $|\psi\rangle$ and the m -qubit state $|\varphi\rangle$, respectively. The $(n + m)$ -qubit state $|\psi\rangle \otimes |\varphi\rangle$ is stabilized by $P \otimes Q = PQ$ since $PQ |\psi\rangle |\varphi\rangle = P |\psi\rangle Q |\varphi\rangle = |\psi\rangle |\varphi\rangle$. Let $\{P_1, \dots, P_n\}$ and $\{Q_1, \dots, Q_m\}$ be sets of generators for $S(|\psi\rangle)$ and $S(|\varphi\rangle)$, respectively. We create an $(n + m)$ -element set of Pauli operators as follows, $P_j \otimes I^{\otimes m}$, $j \in \{1, \dots, n\}$ and $I^{\otimes n} \otimes Q_k$, $k \in \{1, \dots, m\}$. This is a generator set for $S(|\psi\rangle |\varphi\rangle)$ because each of the operators in the set stabilizes $|\psi\rangle |\varphi\rangle$ and the set generates all tensor products $P_j Q_k$.² □

²This holds true due to the following identity: $(A \otimes B)(C \otimes D) = (AC \otimes BD)$.

Table 2.4 illustrates tensor products of stabilizer states from Table 2.3, as well as entangled stabilizer states.

Corollary II.16. *Each computational-basis state is a stabilizer state.*

Observation II.17. Consider a stabilizer state $|\psi\rangle$ represented by a set of generators of its stabilizer group $S(|\psi\rangle)$. Recall from the proof of Theorem II.3 that, since $S(|\psi\rangle) \cong \mathbb{Z}_2^n$, each generator imposes a linear constraint on $|\psi\rangle$. Therefore, the set of generators can be viewed as a system of linear equations whose solution yields the 2^k (for some k between 0 and n) non-zero computational basis amplitudes that make up $|\psi\rangle$. Thus, one needs to perform Gaussian elimination to obtain such basis amplitudes from a generator set.

Computational basis measurements. In addition to unitary stabilizer gates, stabilizer circuits contain one-qubit measurements in the computational basis (Z measurements). Here we explain how the generating set of an n -qubit stabilizer state $|\psi\rangle$ is affected by such non-unitary operations. Lemma I.10 shows how to associate a projection with a Pauli operator, e.g., to apply a Z measurement on the k^{th} qubit of $|\psi\rangle$ we set $P = I_1 \dots Z_k \dots I_n$. Upon measurement, the evolution of the generating set is dependent on whether P commutes with all the generators of $S(|\psi\rangle)$.

(i) If P commutes with all generators, then either P or $-P$ is an element of $S(|\psi\rangle)$.

To see this, note that for each generator Q_i of $S(|\psi\rangle)$, $Q_i P |\psi\rangle = P Q_i |\psi\rangle = P |\psi\rangle$. Thus, $P |\psi\rangle$ is stabilized by $S(|\psi\rangle)$. Since the eigenvalues of P are ± 1 (Lemma I.10-*i*), $P |\psi\rangle = \pm |\psi\rangle$. Therefore, either $P \in S(|\psi\rangle)$ or $-P \in S(|\psi\rangle)$, and the measurement is deterministic with outcomes $+1$ and -1 , respectively. Since the measurement does not affect the state of the system, the generating set of $S(|\psi\rangle)$ remains unchanged.

(ii) If P does not commute with at least one generator, we can assume without loss of generality that it only anticommutes with one generator Q (if P anticommutes

with another generator R , then we can replace R with QR , which does commute with P .³) In this case, the measurement does affect the state of the system, and the ± 1 outcomes are random with equal probability. To see this, note that the projectors for the measurement outcomes ± 1 are given by $\frac{I \pm P}{2}$, respectively. Let $p(a)$ denote the probability of outcome a after measuring $|\psi\rangle$,

$$p(+1) = \text{tr} \left(\frac{I + P}{2} Q |\psi\rangle \langle \psi| \right) = \text{tr} \left(Q \frac{I - P}{2} |\psi\rangle \langle \psi| \right) \quad (2.4)$$

since P and Q anticommute. Using the fact that $Q = Q^\dagger$ we have,

$$p(+1) = \text{tr} \left(\frac{I - P}{2} |\psi\rangle \langle \psi| Q^\dagger \right) = \text{tr} \left(\frac{I - P}{2} |\psi\rangle \langle \psi| \right) = p(-1) \quad (2.5)$$

Since $p(+1) + p(-1) = 1$, the probability of each outcome is $1/2$. After the projector is applied to $|\psi\rangle$, the generating set of $S(|\psi\rangle)$ is modified as follows. If the outcome of the measurement is $+1$, then Q is replaced by P ; otherwise Q is replaced by $-P$.

Example II.18. Consider the stabilizer state $|\psi\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}}$, which has generators $\{IZ, XI\}$. To measure the first qubit, let $P = ZI$. Assume that the measurement outcome is $+1$, then the new state is $|\psi'\rangle = \frac{I+P}{\sqrt{2}} |\psi\rangle = |00\rangle$, which has generators $\{IZ, ZI\}$. Similarly, if the measurement outcome is -1 , then $|\psi'\rangle = \frac{I-P}{\sqrt{2}} |\psi\rangle = |10\rangle$, which has generators $\{IZ, -ZI\}$.

The generators of the reduced cofactors (Section 1.3) can be obtained from the generators of the respective cofactors by dropping the generator added during measurement and removing from each remaining generator the Z symbol at the j^{th} position.

³The above procedure works for any projective measurement. For single-qubit Z measurements, the commutativity check can be simplified by looking at the k^{th} operators only, which takes $O(1)$ time per generator. Thus, it is possible to check commutativity in linear time. However, replacing a generator by a product of two generators still takes linear time, therefore the overall runtime is $\Theta(n^2)$ for n generators.

2.2.3 Canonical forms

In this section, we characterize the amplitudes of stabilizer states and describe *canonical forms* for unitary stabilizer circuits and stabilizer states. We use these canonical forms to derive several general properties that are useful for circuit simulation and equivalence-checking.

Reversible circuits, made of *NOT*, *CNOT* and Toffoli gates, preserve unbiased states. This is due to the fact that such gates only permute amplitudes of a quantum state. In particular, the *modular exponentiation* circuit that plays a key role in Shor's factoring algorithm [62] is a reversible circuit. In this algorithm, the modular exponentiation circuit is applied after a chain of *H* gates. However, as we demonstrate below, *H* gates do not introduce bias when applied to stabilizer states. They may change the support of a stabilizer state, which we track below through parameter k , such that 2^k captures the number of basis vectors supporting a given stabilizer state.

The \mathbb{Z}_2 -form for stabilizer states. The following theorem shows that any stabilizer state can be specified using linear transformations and bilinear forms over the two-element field.

Theorem II.19. *Each n -qubit stabilizer state $|\psi\rangle$ is of the form [14, Equation 2]*

$$|\psi\rangle = \frac{1}{2^{k/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^k} i^{l(\mathbf{y})} (-1)^{q(\mathbf{y})} |\mathbf{y} = R\mathbf{x} + \mathbf{t}\rangle \quad (2.6)$$

where $k \leq n$, R is an $n \times k$ matrix with full column rank k and $\mathbf{t} \in \mathbb{Z}_2^n$. Furthermore, l maps the n -bit string $\mathbf{y} = y_1 \dots y_n$ to $l(\mathbf{y}) = d^T \mathbf{y}$ for $d \in \mathbb{Z}_2^n$ and $q(\mathbf{y}) = \sum c_{ij} y_i y_j + c_i y_i$, for $c_{ij}, c_i \in \{0, 1\}$.

Proof. According to Definition II.11, a stabilizer state $|\psi\rangle$ is constructed by applying *CNOT*, *H* or *P* gates to the initial state $|0\rangle^{\otimes n}$. The initial state is described by Equation 2.6 with $R = 0$ and $t = 0$. Assume that after applying m stabilizer gates,

$|\psi\rangle$ is given by Equation 2.6. We now show that when the $(m+1)^{th}$ gate is applied, $|\psi\rangle$ retains this form.

- (i) Consider the action of the Phase gate on qubit k , $P_k |y_1 \dots y_n\rangle = i^{y_k} |y_1 \dots y_n\rangle$.

After P_k is applied, the resulting state is

$$|\psi'\rangle = \frac{1}{2^{k/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^k} i^{l(\mathbf{y})} i^{y_k} (-1)^{q(\mathbf{y})} |\mathbf{y} = R\mathbf{x} + \mathbf{t}\rangle \quad (2.7)$$

which is of the form given by Equation 2.6 since $i^{l(\mathbf{y})} i^{y_k} = (-1)^{l(\mathbf{y}) \cdot y_k} i^{l(\mathbf{y}) \oplus y_k}$.

- (ii) Note that $CNOT_{j,l} |y_1 \dots y_n\rangle = |y_1 \dots y_j \dots (y_l \oplus y_j) \dots y_n\rangle$, which is a permutation of basis amplitudes. Therefore, $CNOT_{j,l} R = R'$ and $CNOT_{j,l} \mathbf{t} = \mathbf{t}'$. The resulting state is

$$|\psi'\rangle = \frac{1}{2^{k/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^k} i^{l(\mathbf{y})} (-1)^{q(\mathbf{y})} |\mathbf{y} = R'\mathbf{x} + \mathbf{t}'\rangle \quad (2.8)$$

which is of the form given by Equation 2.6.

- (iii) Without loss of generality, assume the Hadamard gate is applied to the first qubit: $H_1 |y_1 \dots y_n\rangle = |0 \cdot y_2 \dots y_n\rangle + (-1)^{y_1} |1 \cdot y_2 \dots y_n\rangle$. Let r be the first row of R , let \bar{R} be the $(n-1) \times k$ matrix obtained by removing r from R , and $\bar{\mathbf{t}} = (t_2, \dots, t_n)$. The resulting state is

$$|\psi'\rangle = \frac{1}{2^{k/2}} \sum_{y_1 \in \{0,1\}} \sum_{\mathbf{x} \in \mathbb{Z}_2^k} i^{l(\mathbf{y})} (-1)^{q(\mathbf{y}) + y_1 \cdot (r^T \mathbf{y}) + y_1 t_1} |\mathbf{y} = y_1, \bar{R}\mathbf{x} + \bar{\mathbf{t}}\rangle \quad (2.9)$$

where \mathbf{y} now denotes the concatenation of y_1 with the $(n-1)$ -bit string obtained from $\bar{R}\mathbf{x} + \bar{\mathbf{t}}$. Since R has full column rank k , \bar{R} has either full rank k or rank $k-1$. If \bar{R} has full rank, then Equation 2.9 is of the same form as Equation 2.6. If \bar{R} is of rank $k-1$ then we must transform \bar{R} to have full rank. Let $c^i, i \in \{1, \dots, k\}$, denote the i^{th} column of \bar{R} , i.e., $\bar{R} = [c^1 | c^2 | \dots | c^k]$. Without loss of generality, assume that $c^1 = \sum_{i=2}^k a_i c^i$ and $c^i, i \in \{2, \dots, k\}$ are linearly independent. We use the vector $\mathbf{a} = (a_2, \dots, a_k)$ to create the following $k \times k$ invertible matrix

$$Q = \begin{bmatrix} 1 & & & \\ a_2 & 1 & & \\ \vdots & & \ddots & \\ a_k & & & 1 \end{bmatrix}$$

such that $\bar{R}Q = [0 \mid c^2 \mid \dots \mid c^k]$. Since \mathbf{a} is the solution to a system of linear equations over \mathbb{Z}_2 , it can be efficiently computed. Let $\mathbf{u} = (u_1, \dots, u_k)$ such that $\mathbf{x} = Q\mathbf{u}$, and let $q'(\mathbf{y}) = q(\mathbf{y}) + y_1 \cdot (r^T \mathbf{y}) + y_1 t_1$. We rewrite Equation 2.9 as follows

$$|\psi'\rangle = \frac{1}{2^{k/2}} \sum_{y_1 \in \{0,1\}} \sum_{\mathbf{u} \in \mathbb{Z}_2^k} i^{l(\mathbf{y})} (-1)^{q'(\mathbf{y})} |\mathbf{y} = y_1, \bar{R}Q\mathbf{u} + \bar{\mathbf{t}}\rangle \quad (2.10)$$

Since the first column of $\bar{R}Q$ is zero, $\bar{R}Q\mathbf{u}$ does not depend on u_1 . Thus, $\bar{R}Q\mathbf{u} = [c^2 \mid \dots \mid c^k]\bar{\mathbf{u}}$, where $\bar{\mathbf{u}} = (u_2, \dots, u_k)$. This implies that we can rewrite Equation 2.10 using $\bar{\mathbf{u}}$.

First, we find l' , q'' and l^* from l and q' on \mathbb{Z}_2^{k-1} such that

$$\frac{1}{2^{k/2}} \sum_{u_1 \in \{0,1\}} i^{l(\mathbf{y}_{\mathbf{u}})} (-1)^{q'(\mathbf{y}_{\mathbf{u}})} = \frac{1}{2^{(k-1)/2}} i^{l'(\mathbf{y}_{\bar{\mathbf{u}}})} (-1)^{q''(\mathbf{y}_{\bar{\mathbf{u}}})} \delta_{l^*(\mathbf{y}_{\bar{\mathbf{u}}}), 0} \quad (2.11)$$

where $\mathbf{y}_{\mathbf{u}} = y_1, \bar{R}Q\mathbf{u} + \bar{\mathbf{t}}$ and $\mathbf{y}_{\bar{\mathbf{u}}} = y_1, \bar{R}Q\bar{\mathbf{u}} + \bar{\mathbf{t}}$. This can be performed efficiently since these are operations on \mathbb{Z}_2 .

Second, let R'' be the $(n-1) \times (k-1)$ matrix with full rank $k-1$ obtained by removing the first column of $\bar{R}Q$. Making the proper substitutions, we rewrite Equation 2.10 as

$$|\psi'\rangle = \frac{1}{2^{(k-1)/2}} \sum_{y_1 \in \{0,1\}} \sum_{\bar{\mathbf{u}} \in \mathbb{Z}_2^{k-1}} i^{l'(\mathbf{y})} (-1)^{q''(\mathbf{y})} |\mathbf{y} = y_1, R''\bar{\mathbf{u}} + \bar{\mathbf{t}}\rangle \quad (2.12)$$

which is of the form given by Equation 2.6.

□

Example II.20. The EPR pair $(|00\rangle - i|11\rangle)/\sqrt{2}$ can be obtained from Equation 2.6 by setting the following parameters.

$$R = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{x}_1 = \begin{bmatrix} 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 1 \end{bmatrix} \quad l(\mathbf{y}) = q(\mathbf{y}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \mathbf{y}$$

Corollary II.21. *The basis amplitudes of a stabilizer state $|s\rangle$ have the following properties:*

- (i) *The number of non-zero amplitudes (support) in $|s\rangle$ is a power of two.*
- (ii) *State $|s\rangle$ is unbiased, and every non-zero amplitude is $\frac{\pm 1}{\sqrt{|s|}}$ or $\frac{\pm i}{\sqrt{|s|}}$, where $|s|$ is the support of $|s\rangle$.*
- (iii) *The number of imaginary amplitudes in $|s\rangle$ is either zero or half the number of non-zero amplitudes.⁴*
- (iv) *The number of negative amplitudes in $|s\rangle$ is either zero or a power of two.⁵*
- (v) *For each qubit q of $|s\rangle$, the number of basis states with $q = 0$ in the $|0\rangle$ - and $|1\rangle$ -cofactors is either zero or a power of two. Furthermore, if both $|0\rangle$ - and $|1\rangle$ -cofactors with respect to a given qubit are nontrivial, the supports of the cofactors must be equal and the norms of the cofactors must be equal.*

Corollary II.21 allows one to quickly distinguish non-stabilizer states, e.g., those in Table 2.5.

⁴Assuming the amplitudes are normalized as in Definition II.14, otherwise all amplitudes can be complex.

⁵For normalized amplitudes, this might require multiplying by a -1 global phase.

UNBIASED STATE	SHORTHAND NOTATION
$(00\rangle + 01\rangle + 10\rangle)/\sqrt{3}$	1, 1, 1, 0
$(00\rangle + 01\rangle + 10\rangle + i 11\rangle)/2$	1, 1, 1, i
$(00\rangle + i 01\rangle + i 10\rangle + i 11\rangle)/2$	1, i , i , i
$(000\rangle + 010\rangle + 100\rangle + 111\rangle)/2$	1, 0, 1, 0, 1, 0, 0, 1

Table 2.5: Several unbiased states that are *not* stabilizer states. The three-qubit state is not a stabilizer state because its $|0\rangle$ -cofactor on the third qubit is not a stabilizer state.

2.2.3.1 The global phase of a stabilizer state

In quantum mechanics, the states $e^{i\theta}|\psi\rangle$ and $|\psi\rangle$ are considered phase-equivalent, and the statistics of measurement are the same. Since global phases are unobservable, they can be safely ignored during simulation. Since both the \mathbb{Z}_2 formulation and the stabilizer-matrix representation simulate stabilizer gates via their action-by-conjugation, *relative phases* are accurately maintained but *global phases* are not.

Example II.22. (a) Suppose we have the stabilizer state $|1\rangle$. In the state-vector representation, $P|1\rangle = i|1\rangle$. According to Table 2.3, the state $|1\rangle$ is stabilized by $-Z$. Applying P leaves the stabilizer unchanged since $P(-Z)P^\dagger = -Z$ (Table 2.1a). Since $|1\rangle$ and $i|1\rangle$ are equal up to a global phase, the stabilizer formalism safely ignores the factor $e^{i\pi/2} = i$.

(b) Suppose we have the stabilizer state $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. In the state-vector representation, $P|\psi\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$. According to Table 2.3, the state $|\psi\rangle$ is stabilized by X . Applying P to the stabilizer gives $P(X)P^\dagger = Y$, which stabilizes $P|\psi\rangle$ and reflects the change in relative phase.

Lemma II.23. *Consider a stabilizer gate $U \in \{P, CNOT, H\}$ acting on an n -qubit stabilizer state $|\psi\rangle$. Without loss of generality, we assume that the global phase of $|\psi\rangle$ is 1. Then the posterior state $U|\psi\rangle$ has global phase $\alpha \in \{\pm 1, \pm i, \pm \frac{1+i}{\sqrt{2}}\}$.*

Proof. We consider the effect of $U \in \{P, CNOT, H\}$ on the cofactors of $|\psi\rangle$ with respect to the qubits on which the gate acts, and show the global phases generated

in each case. Let $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle$, where k denotes the integer representation of each computational basis. Let $\delta(\psi)$ denote the smallest k such that $\alpha_k \neq 0$, e.g., if $|\psi\rangle \propto |01\rangle + |10\rangle$, $\delta(\psi) = 01$ since $01 < 10$. By Definition II.14, $|\psi\rangle$ and $U|\psi\rangle$ are normalized such that $|\delta(\psi)\rangle$ and $|\delta(U\psi)\rangle$ have amplitude 1. Furthermore, in any cofactoring of $|\psi\rangle$, $|\delta(\psi)\rangle$ must be an element of exactly one of the cofactors.

(i) Let $|\psi\rangle \propto \alpha_1 |\psi_{j=0}\rangle + \alpha_2 |\psi_{j=1}\rangle$, where $\alpha_i \in \{\pm 1, \pm i\}$ according to Theorem II.19. Consider the action of P on qubit j , $P_j |\psi\rangle \propto \alpha_1 |\psi_{j=0}\rangle + i\alpha_2 |\psi_{j=1}\rangle$. If $|\delta(\psi)\rangle \in |\psi_{j=0}\rangle$, $\alpha_1 = 1$ and the global phase generated is 1 since the relative i phase is absorbed into the stabilizer. If $|\delta(\psi)\rangle \in |\psi_{j=1}\rangle$, $\alpha_2 = 1$ and the global phase generated is i since $P_j |\psi\rangle$ is normalized such that the amplitude of $|\delta(\psi)\rangle$ is 1.

(ii) Let $|\psi\rangle \propto \alpha_1 |\psi_{i=0,j=0}\rangle + \alpha_2 |\psi_{i=0,j=1}\rangle + \alpha_3 |\psi_{i=1,j=0}\rangle + \alpha_4 |\psi_{i=1,j=1}\rangle$, where $\alpha_i \in \{\pm 1, \pm i\}$. Consider the action of $CNOT$ on control qubit i and target j , $CNOT_{i,j} |\psi\rangle \propto \alpha_1 |\psi_{i=0,j=0}\rangle + \alpha_2 |\psi_{i=0,j=1}\rangle + \alpha_4 |\psi_{i=1,j=0}\rangle + \alpha_3 |\psi_{i=1,j=1}\rangle$. If $|\delta(\psi)\rangle \in \{|\psi_{i=0,j=0}\rangle, |\psi_{i=0,j=1}\rangle\}$, the global phase is 1 since no action is performed by $CNOT$ on these cofactors. If $|\delta(\psi)\rangle \in |\psi_{i=1,j=0}\rangle$, we know $\alpha_3 = 1$ and the global phase generated is α_4 since the basis amplitudes get swapped. Conversely, if $|\delta(\psi)\rangle \in |\psi_{i=1,j=1}\rangle$, $\alpha_4 = 1$ and the global phase is α_3 .

(iii) Observe that when H is applied to qubit j , the support of $|\psi\rangle$ may increase (decrease) or remain the same. We consider these two cases separately.

Case 1 Let $|\psi\rangle \propto \alpha_1 |\psi_{j=0}\rangle + \alpha_2 |\psi_{j=1}\rangle$, where $\alpha_i \in \{\pm 1, \pm i\}$. Suppose $H_j |\psi\rangle \propto \alpha_1 |\psi'_{j=0}\rangle + \alpha_2 |\psi'_{j=1}\rangle$, where $|\psi'_{j=0}\rangle > |\psi_{j=0}\rangle$ and $|\psi'_{j=1}\rangle > |\psi_{j=1}\rangle$, i.e., the support increased. If $|\delta(\psi)\rangle \in |\psi_{j=0}\rangle$ and $|\delta(H_j\psi)\rangle \in |\psi'_{j=1}\rangle$, the global phase generated is α_2 . If $|\delta(\psi)\rangle \in |\psi_{j=1}\rangle$ and $|\delta(H_j\psi)\rangle \in |\psi'_{j=0}\rangle$, the global phase is α_1 . Otherwise, the global phase is 1. The case when the support decreases, namely, when $|\psi'_{j=0}\rangle < |\psi_{j=0}\rangle$ and $|\psi'_{j=1}\rangle < |\psi_{j=1}\rangle$, follows in the

same way. We now consider the special cases when either $|\psi\rangle$ or $H_j |\psi\rangle$ has an empty cofactor.

Case 1-a Let $|\psi\rangle = \alpha |\psi_{j=a}\rangle$, where $a \in \{0, 1\}$ and $\alpha = 1$ by Definition II.14.

Then $H_j |\psi\rangle \propto |\psi_{j=a}\rangle \pm |\psi_{j=\bar{a}}\rangle$ and the support of $|\psi\rangle$ increases. Since the stabilizer absorbs the relative ± 1 phases, only the trivial global phase is generated.

Case 1-b Let $|\psi\rangle \propto |\psi_{j=0}\rangle + \alpha |\psi_{j=1}\rangle$, where $\alpha = \pm 1$. Suppose $H_j |\psi\rangle \propto |\psi_{j=a}\rangle$, where $a \in \{0, 1\}$. In this case, the support of $|\psi\rangle$ decreases since the relative ± 1 phases cancel out. If $|\delta(\psi)\rangle \in |\psi_{j=0}\rangle$ and $H_j |\psi\rangle \propto |\psi_{j=1}\rangle$, then the global phase is α . Otherwise, the global phase is 1.

Case 2 Without loss of generality, let $|\psi\rangle \propto \alpha_1 |\psi_{j=0}\rangle + \alpha_2 |\psi_{j=1}\rangle$, where $\alpha_1 = \pm 1$ and $\alpha_2 = \pm i$. Then $H_j |\psi\rangle \propto \frac{\alpha_1}{\sqrt{2}} |\psi_{j=0}\rangle + \frac{\alpha_1}{\sqrt{2}} |\psi_{j=1}\rangle + \frac{\alpha_2}{\sqrt{2}} |\psi_{j=0}\rangle - \frac{\alpha_2}{\sqrt{2}} |\psi_{j=1}\rangle = \frac{\alpha_1 + \alpha_2}{\sqrt{2}} |\psi_{j=0}\rangle + \frac{\alpha_1 - \alpha_2}{\sqrt{2}} |\psi_{j=1}\rangle = \frac{\alpha_1 + \alpha_2}{\sqrt{2}} (|\psi_{j=0}\rangle \mp \alpha_2 |\psi_{j=1}\rangle)$. Thus, the global phase generated is $\pm \frac{1 \pm i}{\sqrt{2}}$.

Table 2.6 shows the cases in which a non-trivial global phase was generated for one- and two-qubit stabilizer states.⁶ □

Although Lemma II.23 tells us the possible values of the global phase (Table 2.6), calculating the global phase directly from a stabilizer generator set is not straightforward. In Section 2.3.3, we describe an algorithm based on the cofactoring approach used to prove Lemma II.23. It computes the global phase by comparing a subset of the basis amplitudes of a stabilizer state before and after a gate is applied.

2.2.3.2 Canonical stabilizer generators

Although stabilizer states are uniquely determined by their stabilizer group, the set of generators may be selected in different ways. For example, the state $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is uniquely specified by any of the following:

⁶We double-checked the completeness of this table by applying stabilizer gates to all two- and three-qubit stabilizer states.

STABILIZER GATE	INPUT STATE	NORMALIZED OUTPUT STATE	GLOBAL PHASE
Phase	$ 1\rangle$	$ 1\rangle$	i
CNOT (first qubit is the control)	$ 10\rangle - 11\rangle$	$ 10\rangle - 11\rangle$	-1
	$ 01\rangle - 11\rangle$	$ 10\rangle - 01\rangle$	-1
	$ 10\rangle + i 11\rangle$	$ 10\rangle - i 11\rangle$	$-i$
	$ 10\rangle - i 11\rangle$	$ 10\rangle + i 11\rangle$	i
	$ 00\rangle - i 10\rangle$	$ 00\rangle - i 11\rangle$	i
	$ 01\rangle - i 11\rangle$	$ 10\rangle + i 01\rangle$	$-i$
Hadamard	$ 0\rangle + i 1\rangle$	$ 0\rangle - i 1\rangle$	$\frac{1+i}{\sqrt{2}}$
	$ 0\rangle - i 1\rangle$	$ 0\rangle + i 1\rangle$	$\frac{1-i}{\sqrt{2}}$
Hadamard (applied to 2^{nd} qubit)	$ 10\rangle - 01\rangle$	$ 00\rangle - 01\rangle - 10\rangle - 11\rangle$	-1
	$ 10\rangle + i 01\rangle$	$ 00\rangle - 01\rangle - i 10\rangle - i 11\rangle$	i
	$ 10\rangle - i 01\rangle$	$ 00\rangle - 01\rangle + i 10\rangle + i 11\rangle$	$-i$

Table 2.6: One- and two-qubit stabilizer states that generate non-trivial global phases when a stabilizer gate is applied. For simplicity, the $2^{-n/2}$ factors were omitted in columns two and three. The third column shows the normalized output state obtained with the stabilizer formalism (Definition II.14).

$$\mathcal{M}_1 = \begin{vmatrix} XX \\ ZZ \end{vmatrix} \quad \mathcal{M}_2 = \begin{vmatrix} XX \\ -YY \end{vmatrix} \quad \mathcal{M}_3 = \begin{vmatrix} -YY \\ ZZ \end{vmatrix}$$

One obtains \mathcal{M}_2 from \mathcal{M}_1 by left-multiplying the second row by the first. Similarly, one can also obtain \mathcal{M}_3 from \mathcal{M}_1 or \mathcal{M}_2 via row multiplication. Observe that multiplying any row by itself yields II , which stabilizes $|\psi\rangle$. However, II cannot be used as a stabilizer generator because it is redundant and carries no information about the structure of $|\psi\rangle$. This also holds true in general for \mathcal{M} of any size. Any stabilizer matrix can be rearranged by applying sequences of elementary row operations in order to obtain a particular matrix structure. Such operations do not modify the stabilizer state. The elementary row operations that can be performed on a stabilizer matrix are transposition, which swaps two rows of the matrix, and multiplication, which left-multiplies one row with another. Such operations allow one to rearrange the stabilizer matrix in a series of steps that resemble Gauss-Jordan elimination.⁷ Given an $n \times n$ stabilizer matrix, row transpositions are performed in constant time⁸ while

⁷Since Gaussian elimination essentially inverts the $n \times n$ matrix, this could be sped up to $O(n^{2.376})$ time by using fast matrix inversion algorithms. However, $O(n^3)$ -time Gaussian elimination seems more practical.

⁸Storing pointers to rows facilitates $O(1)$ -time row transpositions — one simply swaps relevant pointers.

row multiplications require $\Theta(n)$ time. Algorithm 2.2.1 rearranges a stabilizer matrix into a *row-reduced echelon form* that contains: (i) a *minimum set* of generators with X and Y literals appearing at the top, and (ii) generators containing a *minimum set* of Z literals only appearing at the bottom of the matrix. This particular stabilizer-matrix structure, shown in Figure 2.4, defines a canonical representation for stabilizer states [18, 34]. Several row-echelon (standard) forms for stabilizer generators along with relevant algorithms to obtain them have been introduced in the literature [6, 34, 51]. However, such standard forms are not always canonical, e.g, the row-echelon form described in [6] does not guarantee a minimum set of Z literals. Since most of our algorithms manipulate canonical stabilizer matrices, we will describe in detail our Gaussian-elimination procedure for obtaining the canonical structure depicted in Figure 2.4. The algorithm iteratively determines which row operations to apply based on the Pauli (non- I) literals contained in the first row and column of an increasingly smaller submatrix of the full stabilizer matrix. Initially, the submatrix considered is the full stabilizer matrix. After the proper row operations are applied, the dimensions of the submatrix decrease by one until the size of the submatrix reaches one. The algorithm performs this process twice, once to position the rows with $X(Y)$ literals at the top, and then again to position the remaining rows containing Z literals only at the bottom. Let $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\}$ be

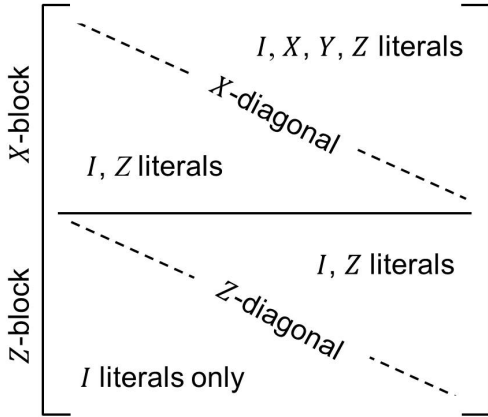


Figure 2.4: Canonical (row-reduced echelon) form for stabilizer matrices. The X -block contains a *minimal* set of rows with X/Y literals. The rows with Z literals only appear in the Z -block. Each block is arranged so that the leading non- I literal of each row is strictly to the right of the leading non- I literal in the row above. The number of Pauli (non- I) literals in each block is minimal.

the index of the first row and first column, respectively, of submatrix \mathcal{A} . The steps to construct the upper-triangular portion of the row-echelon form shown in Figure 2.4 are as follows.

1. Let k be a row in \mathcal{A} whose j^{th} literal is $X(Y)$. Swap rows k and i such that k is the first row of \mathcal{A} . Decrease the height of \mathcal{A} by one (i.e., increase i).
2. For each row $m \in \{0, \dots, n\}, m \neq i$ that has an $X(Y)$ in column j , use row multiplication to set the j^{th} literal in row m to I or Z .
3. Decrease the width of \mathcal{A} by one (i.e., increase j).

Algorithm 2.2.1 Canonical form reduction for stabilizer matrices

Require: Stabilizer matrix \mathcal{M} for $S(|\psi\rangle)$ with rows R_1, \dots, R_n

Ensure: \mathcal{M} is reduced to row-echelon form

\Rightarrow ROWSWAP(\mathcal{M}, i, j) swaps rows R_i and R_j of \mathcal{M}

\Rightarrow ROWMULT(\mathcal{M}, i, j) left-multiplies rows R_i and R_j , returns updated R_i

```

1:  $i \leftarrow 1$ 
2: for  $j \in \{1, \dots, n\}$  do                                      $\triangleright$  Setup  $X$  block
3:    $k \leftarrow$  index of row  $R_{k \in \{i, \dots, n\}}$  with  $j^{\text{th}}$  literal set to  $X(Y)$ 
4:   if  $k$  exists then
5:     ROWSWAP( $\mathcal{M}, i, k$ )
6:     for  $m \in \{0, \dots, n\}$  do
7:       if  $j^{\text{th}}$  literal of  $R_m$  is  $X$  or  $Y$  and  $m \neq i$  then
8:          $R_m =$  ROWMULT( $\mathcal{M}, R_i, R_m$ )                              $\triangleright$  Gauss-Jordan elimination step
9:       end if
10:    end for
11:     $i \leftarrow i + 1$ 
12:  end if
13: end for
14: for  $j \in \{1, \dots, n\}$  do                                      $\triangleright$  Setup  $Z$  block
15:    $k \leftarrow$  index of row  $R_{k \in \{i, \dots, n\}}$  with  $j^{\text{th}}$  literal set to  $Z$ 
16:   if  $k$  exists then
17:     ROWSWAP( $\mathcal{M}, i, k$ )
18:     for  $m \in \{0, \dots, n\}$  do
19:       if  $j^{\text{th}}$  literal of  $R_m$  is  $Z$  or  $Y$  and  $m \neq i$  then
20:          $R_m =$  ROWMULT( $\mathcal{M}, R_i, R_m$ )                              $\triangleright$  Gauss-Jordan elimination step
21:       end if
22:     end for
23:      $i \leftarrow i + 1$ 
24:   end if
25: end for

```

To bring the matrix to its lower-triangular form, one executes the same procedure with the following difference: (i) step 1 looks for rows that have a Z literal (instead of X or Y) in column j , and (ii) step 2 looks for rows that have Z or Y literals (instead of X or Y) in column j . Observe that Algorithm 2.2.1 ensures that the columns in \mathcal{M} have at most two distinct types of non- I literals. Since Algorithm 2.2.1 inspects all n^2 entries in the matrix and performs a constant number of row multiplications each time, its runtime is $O(n^3)$.

Definition II.24. Two stabilizer matrices are *similar* if they contain the same Pauli generators with different ± 1 phases, i.e., the matrices are equivalent up to a phase-vector permutation. Otherwise, the matrices are called *dissimilar*.

Observation II.25. The total number of dissimilar n -qubit canonical stabilizer matrices is $\mathcal{N}(n)/2^n$.

Once a stabilizer matrix is in canonical form, it is possible to obtain the 2^n basis amplitudes of the corresponding stabilizer state. Given stabilizer matrix \mathcal{M} as input, Algorithm 2.2.2 implements the process described in Observation II.17. The algorithm requires exponential resources and only works when the number of non-zero amplitudes is small. This can be predetermined by looking at the return value (the logarithm of the support of the stabilizer state) obtained when Algorithm 2.2.1 is initially applied to \mathcal{M} . Then, we use the Z -only rows in \mathcal{M} to find an operator P such that the basis state $P|00\dots 0\rangle$ occurs with non-zero amplitude in $|\psi\rangle$. Next, we create a new stabilizer matrix \mathcal{M}' whose rows consist of the rows in \mathcal{M} that contain X and Y literals and find the group of operators \mathbf{Q} generated by \mathcal{M}' . Note that $|\mathbf{Q}| = 2^k \leq 2^n$, where k is the number of rows in \mathcal{M}' . Finally, each operator $Q_i \in \mathbf{Q}$ is multiplied by P to obtain the basis state $(Q_i \times P)|00\dots 0\rangle = U_i|00\dots 0\rangle$ and its corresponding non-zero amplitude. Each basis-amplitude pair is obtained by iterating over the literals of U_i and observing the state of the qubit stabilized by that literal

Algorithm 2.2.2 Computation of 2^n basis amplitudes for a stabilizer state

Require: Stabilizer matrix \mathcal{M} for $S(|\psi\rangle)$ with rows R_1, \dots, R_n

Ensure: Vector $\mathbf{a} = \{a_1, \dots, a_{2^n}\}$ of basis amplitudes for $|\psi\rangle$

\Rightarrow GROUP(\mathcal{M}) returns the group $\mathbf{Q} = \{Q_1, \dots, Q_{2^k}\}$ generated by the first k rows of \mathcal{M}

```

1:  $\mathbf{a} \leftarrow \{a_1 \leftarrow 0, \dots, a_{2^n} \leftarrow 0\}$ 
2:  $P \leftarrow I^{\otimes n}$ 
3: GAUSS( $\mathcal{M}$ ) ▷ Algorithm 2.2.1
4: for each row  $R_i \in \mathcal{M}$  with only  $Z$  and  $I$  literals do ▷ Find  $P|00\dots 0\rangle$  for  $P \in \mathcal{P}_n$ 
5:   for each  $Z$  literal at position  $j$  in  $R_i$  do
6:     if phase of  $R_i$  is negative and the  $j^{\text{th}}$  literal of  $P$  is  $I$  then
7:       set the  $j^{\text{th}}$  literal of  $P$  to  $X$ 
8:     end if
9:   end for
10: end for
11:  $\mathcal{M}' \leftarrow \emptyset$  ▷ Initialize a new stabilizer matrix
12: for each row  $R_i \in \mathcal{M}$  with  $X$  or  $Y$  literals do
13:    $\mathcal{M}' \leftarrow \mathcal{M}' \cup R_i$ 
14: end for
15:  $\mathbf{Q} \leftarrow \text{GROUP}(\mathcal{M}')$  ▷ Note  $|\mathbf{Q}| = 2^k, k \leq n$ 
16: for each operator  $Q_i \in \mathbf{Q}$  do
17:    $U_i \leftarrow Q_i \times P$ 
18:    $b \leftarrow 0$ 
19:    $s \leftarrow$  leading  $(\pm 1)$ -phase of  $U_i$ 
20:   for each literal  $l \in U_i$  at position  $j$  do ▷ Set basis state and amplitude directly from  $U_i$ 
21:     if  $l = X$  or  $l = Y$  then
22:        $b \leftarrow b + 2^{j-1}$ 
23:       if  $l = Y$  then
24:          $s \leftarrow s \cdot i$ 
25:       end if
26:     end if
27:   end for
28:    $a_b = s / \sqrt{2^{|\mathbf{Q}|}}$  ▷ Store the basis-amplitude pair
29: end for
30: return  $\mathbf{a}$ 

```

while keeping track of the phase. Note that Algorithm 2.2.2 will generate a *normalized* set of basis amplitudes where the first non-zero amplitude is 1.0 (Definition II.14).

Example II.26. Suppose we have the following stabilizer matrix where the \pm column indicates the phase of the generator row.

$$\mathcal{M} \equiv \begin{array}{l} R_1 \begin{bmatrix} I & I & X \end{bmatrix} + \\ R_2 \begin{bmatrix} I & Z & I \end{bmatrix} - \\ R_3 \begin{bmatrix} I & I & Z \end{bmatrix} - \end{array}$$

Following Algorithm 2.2.2, we use R_2 and R_3 to create $P = IXX$ (lines 4 –

10) such that $P|000\rangle = IXX|000\rangle = |011\rangle$, and R_1 to create $\mathcal{M}' \equiv \begin{bmatrix} I & I & X \end{bmatrix}_+$. In this case, $\mathbf{Q} = \{Q_1 = III, Q_2 = IIX\}$ and the two basis states are given by $Q_1P|000\rangle = Q_1|011\rangle = |011\rangle$ and $Q_2P|000\rangle = Q_2|011\rangle = |010\rangle$.

2.2.3.3 Canonical stabilizer circuits

We now describe the canonical forms for stabilizer circuits proposed in [2] and [14]. Such forms are useful for synthesizing stabilizer circuits that minimize the number of gates and qubits required to produce a particular computation.

Definition II.27. Given a finite sequence of quantum gates, a *circuit template* describes a segmentation of the circuit into blocks where each block uses only one gate type. The blocks must match the sequence and be concatenated in that order. For example, a circuit satisfying the *H-C-P* template starts with a block of Hadamard (*H*) gates, followed by a block of CNOT (*C*) gates, followed by a block of Phase (*P*) gates.

Definition II.28. A circuit with a *template structure* consisting entirely of stabilizer-gate blocks is called a *canonical stabilizer circuit*.

Definition II.29. Two unitary stabilizer circuits \mathbf{C}_1 and \mathbf{C}_2 are *equivalent* if $\forall |\psi\rangle$, $\mathbf{C}_1|\psi\rangle = \mathbf{C}_2|\psi\rangle$.

Theorem II.30. Any stabilizer state $|\psi\rangle$ can be produced by an *H-C-X-P-CZ* canonical circuit, where the *X* and *CZ* blocks consist of *NOT* and *Controlled-Z* gates.

Proof. Consider the \mathbb{Z}_2 -form for $|\psi\rangle$ given by Equation 2.6. We construct a circuit \mathbf{C} that outputs $|\psi\rangle$ given the initial state $|0\rangle^{\otimes n}$. First, apply h Hadamard gates to yield $\frac{1}{2^{h/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^h} |\mathbf{x}\rangle |0\rangle^{n-h}$. Second, apply *CNOT* and *NOT* gates to yield $\frac{1}{2^{h/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^h} |R\mathbf{x} + \mathbf{t}\rangle$. Finally, apply *P* and gates to get

$$|\psi\rangle = \mathbf{C}|0\rangle^{\otimes n} = \frac{1}{2^{h/2}} \sum_{\mathbf{x} \in \mathbb{Z}_2^h} i^{l(\mathbf{y})} (-1)^{q(\mathbf{y})} |\mathbf{y} = R\mathbf{x} + \mathbf{t}\rangle \quad (2.13)$$

□

Corollary II.31. [14, Theorem 2] *For any stabilizer circuit \mathbf{C} there is an H - C - X - P - CZ canonical circuit \mathbf{C}' such that $\mathbf{C}_1 |0\rangle^{\otimes n} = \mathbf{C}_2 |0\rangle^{\otimes n}$.*

Using Corollary II.31, one can show that any unitary stabilizer circuit has an equivalent circuit with template structure B_1 - H - B_2 , where B_i are basis-preserving circuit blocks [14, Theorem 3].

Note that the H - C - X - P - CZ canonical form \mathbf{C}' only applies when the initial state is $|0\rangle^{\otimes n}$ since the original circuit and \mathbf{C}' might not be equal as $2^n \times 2^n$ matrices, e.g., the single-qubit circuit $\mathbf{C} = HPH$ has no equivalent \mathbf{C}' such that $\mathbf{C} = \mathbf{C}'$. For arbitrary initial states, the work in [2] establishes a 7-block⁹ canonical-circuit template with the sequence H - C - P - C - P - C - H . In Section 4.1, we describe an algorithm for synthesizing canonical circuits with the block sequence H - C - CZ - P - H . Our circuits are therefore close to the smallest known circuits proposed in [14].

Corollary II.32. [2, Corollary 9] *For any n -qubit unitary stabilizer circuit \mathbf{C} there is an equivalent circuit \mathbf{C}' with only $O(n^2/\log n)$ gates.*

Proof. As stated above, we can transform \mathbf{C} into the 7-round canonical circuit \mathbf{C}' proposed in [2]. Clearly, the H and P segments of \mathbf{C}' each have $O(n)$ gates. Using the result by Patel, Markov and Hayes [54], the $CNOT$ segments can be minimized to have only $O(n^2/\log n)$ gates. □

A Shannon counting argument shows that the result in Corollary II.32 is optimal [2].

⁹Theorem 8 in [2] actually describes an 11-step procedure to obtain such circuits. However, the last four steps are used to reduce destabilizer rows, which we do not consider here.

2.3 Algorithms for Simulation and Equivalence-checking of Stabilizer Circuits

In the previous section, we described how the stabilizer formalism allows us to represent n -qubit stabilizer states with only $\Theta(n^2)$ storage cost. We now describe the *check-matrix* or *tableau* technique for simulating stabilizer circuits, which also uses $\Theta(n^2)$ space, but asymptotically improves the simulation of measurement [2] from $O(n^3)$ time to $O(n^2)$ time. This technique can also be used to determine whether two unitary stabilizer circuits are equivalent. Furthermore, we describe a tableau-based algorithm to compute the non-trivial global phases that arise during simulation of stabilizer circuits.

2.3.1 The tableau technique

In this approach, the generator set of a stabilizer state is represented using binary vectors by encoding each Pauli literal using two bits.

Definition II.33. A *tableau* for the n -qubit stabilizer state $|\psi\rangle$ is an $n \times 2n$ binary matrix that encodes the stabilizer matrix of $|\psi\rangle$. The j^{th} tensor factor (Pauli literal) of generator $g_i \in \{g_1, \dots, g_n\}$ in the stabilizer matrix is determined by the bits $x_{ij}z_{ij}$ as follows: $00 = I$, $01 = Z$, $10 = X$ and $11 = Y$. Tableaux have the following general structure.

$$\left[\begin{array}{ccc|ccc} x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} \end{array} \right] \quad (2.14)$$

As in the stabilizer-matrix representation, the tableau technique stores the phase array separately. The advantage of this approach is that the literal-to-bits mapping from Definition II.33 induces an isomorphism $\mathbb{Z}_2^{2n} \rightarrow \mathcal{P}_n$ because vector addition in \mathbb{Z}_2^2 is equivalent to multiplication of Pauli operators up to a global phase. Table 2.7

	$I(00)$	$X(10)$	$Y(11)$	$Z(01)$
$I(00)$	$I(00)$	$X(10)$	$Y(11)$	$Z(01)$
$X(10)$	$X(10)$	$I(00)$	$iZ(01)$	$-iY(11)$
$Y(11)$	$Y(11)$	$-iZ(01)$	$I(00)$	$iX(10)$
$Z(01)$	$Z(01)$	$iY(11)$	$-iX(10)$	$I(00)$

Table 2.7: Addition table for \mathbb{Z}_2^2 representation of Pauli operators.

shows the equivalence between binary addition and multiplication for one-qubit Pauli operators.

Upon application of stabilizer gates, the evolution of a generator set represented by a tableau is recorded as follows. When a unitary stabilizer gate is applied, update the proper entries in each row of the tableau and each entry in the phase array according to Table 2.7. Since there is a constant number of such updates per row, the runtime for applying a unitary stabilizer gate is $\Theta(n)$. Example II.34 illustrates how the tableau evolves while simulating the EPR-pair circuit. On the other hand, updating the tableau to simulate a measurement on qubit j is not as efficient. As discussed in Section 2.2.1, the outcome can be random or deterministic. The outcome is random if the j^{th} Pauli matrix of the measurement operator anticommutes with the j^{th} Pauli matrix of some generator(s); otherwise the outcome is deterministic. In the random case, updating the tableau after the measurement takes $O(n^2)$ time since we need to update the generators (rows of the tableau).¹⁰ If the outcome is deterministic, then we apply Algorithm 2.2.1 to rearrange the tableau such that (i) the number of X and Y literals is minimized, and (ii) the rows with only Z literals are contained in the lower-triangular section of the tableau. To determine the outcome of the measurement, we multiply the rows in the lower triangle that have a Z in their j^{th} position and return the resulting phase factor, which is either $+1$ or -1 . Since the runtime bottleneck of this procedure is Gaussian elimination, the procedure has $O(n^3)$ runtime complexity.¹¹

¹⁰Whether generators can be updated in $O(n)$ time is an open question.

¹¹Alternative algorithms with lower complexity exist, but Gaussian elimination typically runs faster than n^3 in practice and remains a reasonable implementation choice.

Example II.34. The EPR-pair circuit from Example II.8 can be simulated using the tableau (check-matrix) technique as shown in Figure 2.5.

Definition II.35. The *destabilizer generators* $\{D_1, \dots, D_n\}$ of stabilizer state $|\psi\rangle$ are the set of Pauli operators such that $D_i |\psi\rangle \neq |\psi\rangle$, which, together with the n stabilizer generators of $S(|\psi\rangle)$, generate \mathcal{P}_n .

Aaronson and Gottesman [2] improved the runtime of deterministic measurements by doubling the size of the tableaux to include n destabilizer generators in addition to the n stabilizer generators. Thus, the cost is a factor of two increase in the size of the tableau. Let R_i represent the i^{th} row of the tableau. $R_i, i \in \{1, \dots, n\}$ represent the destabilizer generators and $R_i, i \in \{n+1, \dots, 2n\}$ represent the stabilizer generators.

Proposition II.36. [2, Proposition 3] *The following are invariant properties of the tableau:*

- (i) R_{n+1}, \dots, R_{2n} generate $S(|\psi\rangle)$, and R_1, \dots, R_{2n} generate \mathcal{P}_n by Definition II.35.
- (ii) R_1, \dots, R_n commute.
- (iii) For all $h \in \{1, \dots, n\}$, R_h anticommutes with R_{h+n} .
- (iv) For all $i, h \in \{1, \dots, n\}$ such that $i \neq h$, R_i commutes with R_{h+n} .

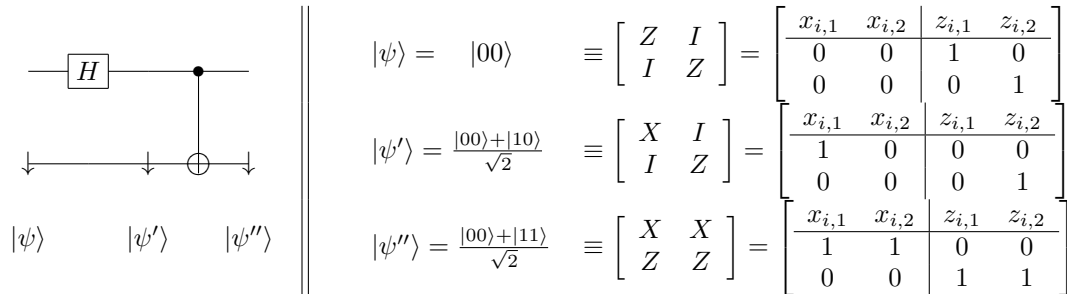


Figure 2.5: Simulation of EPR-pair circuit using the tableau technique. We omit phase vectors for clarity.

The measurement algorithm proposed in [2] uses a subroutine called `rowsum`(h, i), which sets generator h equal to $i + h$ and keeps track of the phase factor when multiplying the Pauli matrices. Since `rowsum` looks at each Pauli matrix in rows i and h , the runtime is on the order of $\Theta(n)$. See [2] for implementation details. Suppose that a measurement of qubit j yields a deterministic outcome. The measurement operator Z_j must commute with all the stabilizer generators

$$\sum_{h=1}^n c_h R_{h+n} = \pm Z_j \quad (2.15)$$

where $c_h \in \{0, 1\}$ and the sum over R_{h+n} are defined by the `rowsum` procedure. To determine the phase factor representing the outcome, one needs to find vector $\mathbf{c} = c_1, \dots, c_n$ such that the appropriate R_{h+n} are summed. Observe that R_h commutes with R_i if the *symplectic inner product*

$$R_h \cdot R_i = x_{h1}z_{i1} \otimes \cdots \otimes x_{hn}z_{in} \otimes x_{i1}z_{h1} \otimes \cdots \otimes x_{in}z_{hn}. \quad (2.16)$$

equals 0, and anticommutes if it equals 1. Using this fact, we can use the destabilizer generators to find \mathbf{c} since for all $i \in \{1, \dots, n\}$

$$c_i \equiv \sum_{h=1}^n c_h (R_i \cdot R_{h+n}) \equiv R_i \cdot \sum_{h=1}^n c_h R_{h+n} \equiv R_i \cdot Z_j \pmod{2} \quad (2.17)$$

by Proposition II.36. Thus, we learn whether $c_i = 1$ by checking if the destabilizer generator R_i anticommutes with Z_j . Since the tableau provides constant-time access to the destabilizer generators, determining \mathbf{c} takes $\Theta(n)$ time. In the worst case, $c_i = 1 \forall i \in \{1, \dots, n\}$, and there will be n calls to `rowsum`. Since each call to `rowsum` takes $\Theta(n)$ time, the overall runtime of the measurement algorithm is $O(n^2)$.

Observation II.37. Given two stabilizer circuits C_1 and C_2 , the concatenated circuits C_1C_2 and C_2C_1 are also simulatable.

Observation II.37 follows from the tableau-based implementation of the Gottesman-Knill theorem. While most simulation techniques also possess this property, some techniques such as tensor-network contraction [45] do not.

2.3.2 Equivalence checking

Using the tableau approach we can directly determine whether two unitary stabilizer circuits are equivalent, as shown next.

Lemma II.38. [2, Lemma 5] *Let τ_1 and τ_2 be the final tableaux obtained by applying the unitary stabilizer circuits \mathbf{C}_1 and \mathbf{C}_2 to the initial state $|0\rangle^{\otimes n}$. Then \mathbf{C}_1 and \mathbf{C}_2 are equivalent iff $\tau_1 = \tau_2$.*

Proof. Clearly, if \mathbf{C}_1 and \mathbf{C}_2 are equivalent, then $\tau_1 = \tau_2$. It remains to show that, if $\tau_1 = \tau_2$, then \mathbf{C}_1 and \mathbf{C}_2 are equivalent. By Proposition II.36-*i*, the rows of the initial tableau generate \mathcal{P}_n . Since each unitary stabilizer circuit acts linearly on the Pauli generators, the basis is preserved. Therefore, the linear transformations given by \mathbf{C}_1 and \mathbf{C}_2 are equivalent. \square

2.3.3 Global-phase maintenance

As discussed in Section 2.2.3, the stabilizer formalism (and its tableau-based implementation) does not maintain global phases since U and $e^{i\theta}U$ have the same action by conjugation. Maintaining such phases is important in order to compute complex-valued inner products of stabilizer states (Section 4.2). Furthermore, let $|\Psi\rangle = \sum_{j=1}^k a_j |\psi_j\rangle$, where each $|\psi_j\rangle$ is a stabilizer state. Then,

$$U |\Psi\rangle = \sum_{j=1}^k a_j U |\psi_j\rangle = \sum_{j=1}^k a_j e^{i\theta_j} |\varphi_j\rangle \quad (2.18)$$

The global phase $e^{i\theta_j}$ of each $|\varphi_j\rangle$ becomes relative with respect to $U |\Psi\rangle$. Therefore, such phases need to be computed explicitly in order to maintain a consistent

representation based on linear combinations of stabilizer states (Section 5.1).

During simulation, the global phase of a stabilizer state can be maintained separately from the stabilizer tableau. Let U be a stabilizer gate applied to the state $|\psi\rangle$ represented by stabilizer matrix \mathcal{M} . The following process computes the global phase of $U|\psi\rangle$:

1. Use Gaussian elimination to obtain a basis state $|b\rangle$ from \mathcal{M} (Observation II.17) and store its non-zero amplitude α . If U is the Hadamard gate, it may be necessary to sample a sum of two non-zero (one real, one imaginary) basis amplitudes (see Example II.39).
2. Compute $\alpha U|b\rangle = \alpha'|b'\rangle$ directly using the state-vector representation.
3. Obtain $|b'\rangle$ from $U\mathcal{M}U^\dagger$ and store its non-zero amplitude β .
4. The phase factor generated is α'/β .

Example II.39. Suppose $|\psi\rangle = |00\rangle + |01\rangle - i|10\rangle - i|11\rangle$, where we omit the normalization factor for clarity. A generator set for $|\psi\rangle$ is $\mathcal{M} = \{-YI, IX\}$. We will compute the global-phase factor generated when a Hadamard gate is applied to the first qubit. Following Step 1, we obtain basis states $|00\rangle$ and $i|10\rangle$ from \mathcal{M} , and set $\alpha = 1$ (the amplitude of $|00\rangle$). Next, we compute $H_1(|00\rangle + i|10\rangle) = \frac{1-i}{\sqrt{2}}|00\rangle + \frac{1+i}{\sqrt{2}}|10\rangle$ and set $\alpha' = \frac{1-i}{\sqrt{2}}$ (Step 2). According to Step 3, we obtain the $|00\rangle$ amplitude from $H_1\mathcal{M}H_1^\dagger = \{YI, IX\}$, which gives $\beta = 1$. The global-phase factor is $\alpha'/\beta = \frac{1-i}{\sqrt{2}}$. One can obtain the factor generated when CNOT, Phase and measurements gates are applied in a similar fashion. However, for such gates, only one basis-state amplitude needs to be sampled in Step 1.

2.4 Stateless Simulation of Stabilizer Circuits

In Section 2.2, we discussed how the stabilizer formalism facilitates the compact representation of certain quantum states using only $\Omega(n^2)$ bits on classical computers. We will show in Chapter III, that such a compact representation is useful for the analysis of specific geometric properties of stabilizer states and error correcting codes. Furthermore, in Chapter V, we describe algorithms that take advantage of this compact representation to directly simulate generic quantum circuits. However, in the context of *stateless simulation* (Section 2.1.2), maintaining such state representations is not necessary, making the simulation task simpler as compared to direct simulation. We now describe how the stateless model works in the context of stabilizer (Clifford) circuit simulation and compare it to direct simulation using stabilizer matrices (Section 2.2).

Theorem II.40. [39] *Consider a unitary Clifford circuit \mathbf{C} such that: (i) the input state $|\psi\rangle$ is a product (separable) state and (ii) the output is a final Z -measurement on any single qubit. Then the computation may be simulated in linear time on a classical computer in the sense of Definition II.1.*

Proof. The case where $|\psi\rangle$ is a stabilizer state is entailed by the Gottesman-Knill theorem. We now show this holds in general for any product state. Let \mathbf{C} be a stabilizer circuit on starting *product state* $|\psi_0\rangle$. Suppose we want to measure qubit k after \mathbf{C} is applied. Recall from Section 2.2.2 that Z -measurements are given by the projectors associated with Pauli operators of the form $I \cdots Z_k \cdots I$, where k is the qubit we seek to measure. Let $\mathbf{E}(Z_k^f)$ denote the expectation value of Z_k in the final state $\mathbf{C}|\psi_0\rangle$,

$$\mathbf{E}(Z_k^f) = \langle \psi_0 | \mathbf{C}^\dagger Z_k \mathbf{C} | \psi_0 \rangle \quad (2.19)$$

Since $Z_k \in \mathcal{P}_n$ and \mathbf{C} is a Clifford circuit, we have $P = \mathbf{C}^\dagger Z_k \mathbf{C}$ where P is a Pauli string, i.e., $P = P_1 \otimes \cdots \otimes P_n \in \mathcal{P}_n$. By Theorem I.10, the eigenvalues of P are ± 1

and therefore $\mathbf{E}(Z_k^f) = p_0 - p_1$, where p_0 and p_1 are the probabilities that the output is 0 and 1, respectively. Using Equation 2.19 and the relation $p_0 + p_1 = 1$, one can derive both p_0 and p_1 . Since $|\psi_0\rangle = |s_1\rangle \otimes |s_2\rangle \cdots |s_n\rangle$,

$$\mathbf{E}(Z_k^f) = \prod_{j=1}^n \langle s_j | P_j | s_j \rangle \quad (2.20)$$

which can be computed on a classical computer in $O(n)$ time. Since P can also be computed in time linear in the size of the circuit (Section 2.2), this approach leads to linear-time classical simulation algorithm. \square

As compared to the stabilizer formalism, the simulation approach from Theorem II.40 has several key different properties:

- (i) The input state allowed is any general product (separable) state. In stabilizer-based simulation only computational-basis input states are allowed.
- (ii) $\mathbf{E}(Z_k^f)$ is calculated without computing the state $C|\psi_0\rangle$ and only a linear number of bits is needed. (Storing a stabilizer matrix requires $\Omega(n^2)$ bits.)
- (iii) Instead of propagating the state description forward, the final measurement is propagated backwards – one conjugates Z_k by the gates in \mathbf{C} in reverse. We call this *reverse simulation*.
- (iv) The overall runtime including single-qubit measurements is linear in n . (In the stabilizer formalism, it takes $O(n^2)$ -time to decide deterministic measurements even if one does not compute the post-measurement state.)
- (v) Since post-measurement states are not computed, one cannot simulate unitary gates that are applied adaptively based on measurement outputs.

Table 2.8 lists key differences between the two techniques. In general, given that intermediate measurements can always be moved to the end of a circuit (Theorem I.4),

SIMULATION PROPERTY	STATELESS	STABILIZERS
Input state	Product	Computational-basis
Gate conjugation order	Reverse	Forward
Gates conditioned on meas. output	Not allowed	Allowed
Multi-qubit output	Not allowed	Allowed
Memory	$O(n + m)$	$\Omega(n^2)$
Runtime	$O(g + m)$	$O(g \cdot n)$

Table 2.8: Comparison between direct (stabilizer formalism) and stateless simulation for n -qubit Clifford circuit with g gates and m ancilla qubits (used for multi-qubit measurements in stateless simulation).

property (3) does not impact performance. In Chapter VI, we describe an extension of Theorem II.40 that admits non-stabilizer circuits. In this case, we show that property (3) can have a considerable impact on the runtime performance of the simulation. We now summarize the simulation technique established by the results of Theorem II.40 in the following steps:

1. Initialize the Z_k -measurement operator.
2. Conjugate Z_k by each gate in C in reverse.
3. Compute $\mathbf{E}(Z_k^f)$ as per Equation 2.20. If the initial state is a computational basis state, $\mathbf{E}(Z_k^f) \in \{0, \pm 1\}$.
 - If $\mathbf{E}(Z_k^f) = 0$, then $p_0 = p_1 = \frac{1}{2}$.
 - If $\mathbf{E}(Z_k^f) = 1$, then $p_0 = 1$ and $p_1 = 0$.
 - If $\mathbf{E}(Z_k^f) = -1$, then $p_0 = 0$ and $p_1 = 1$.

Example II.41. Suppose $|\psi_0\rangle = |00\rangle$ and we want to simulate the EPR-pair circuit $\mathbf{C} = CNOT_{1,2} \cdot H_1$ from Example II.8 using a stateless approach. We compute the outcome probabilities for the first qubit as follows,

$$\begin{aligned} \mathbf{E}(Z_1^f) &= \langle 00 | (\mathbf{C}^\dagger) Z I(\mathbf{C}) | 00 \rangle \langle 00 | (H_1 \cdot CNOT_{1,2}) Z I(CNOT_{1,2} \cdot H_1) | 00 \rangle \\ &= \langle 00 | (H_1) Z Z (H_1) | 00 \rangle = \langle 00 | X Z | 00 \rangle = \langle 00 | 10 \rangle = 0 \end{aligned}$$

By Step 3 above, since the expected value is zero, the probability of measuring zero (one) for the first qubit is $\frac{1}{2}$.

An assumption of Theorem II.40 is that the output of the quantum computation is the result of measuring a single qubit. Recall from our discussion in Section 2.1.2 that decision problems (output is only a “yes/no” answer) are amenable to such a computational approach. Suppose that we are computing the solution to a decision problem, but we have to perform multiple measurements in order to arrive at our solution. In this case, one can adjoin a $|0\rangle$ -initialized ancilla qubit j for each measurement step and replace each measurement gate on qubit i by a CNOT with control i and target j . Qubit j is not used in any other way and its purpose is to decohere qubit i into the post-measurement mixture so that the final output of the “answer” qubit is unchanged. For an n -qubit stabilizer circuit with g unitary gates and m distinct measurement gates, this approach requires $O(n + m)$ bits of memory and $O(g + m)$ runtime.

In the proof of Theorem II.40 we used two properties: (i) since $P \in \mathcal{P}_n$, $\langle \psi_0 | P | \psi_0 \rangle$ can be computed in linear time for any product state $|\psi_0\rangle$, and (ii) since $\mathbf{C} \in \mathcal{C}_n$, $\mathbf{C}^\dagger P \mathbf{C}$ can also be computed in linear time. It turns out that both these properties can be generalized since they do not depend on any special group or algebraic structure on \mathcal{P}_n or \mathcal{C}_n [39]. Therefore, let \mathcal{S}_n be a set of n -qubit operators and \mathcal{U}_n be a class of unitary operators such that:

(i) for $S \in \mathcal{S}_n$, $\langle \psi_0 | S | \psi_0 \rangle$ can be computed in poly-time for any allowable input state $|\psi_0\rangle$,

(ii) $U^\dagger S U \in \mathcal{S}_n$ can be computed in poly-time for all $S \in \mathcal{S}_n$ and $U \in \mathcal{U}_n$.

The two properties above are sufficient for efficient stateless simulation (as outlined in the proof of Theorem II.40) over the sets of operators \mathcal{S}_n and \mathcal{U}_n .

We have focused on comparing details of the Clifford-circuit simulation techniques that are most relevant to our work. It is important to note that a more comprehensive analysis can be found in [40], which compares the classical simulation complexity of all combinations of the simulation properties outlined here: (i) strong vs. weak simulation, (ii) inputs being computational basis states vs. general product states, (iii) adaptive vs. non-adaptive gates for circuits with intermediate measurements and (iv) single vs. multi-qubit output. Figure 2.6 is borrowed from [40] as it provides a good summary of the results. In particular, observe that stabilizer-based simulation of Clifford circuits falls into the category: [NONADAPT, IN(BITS), OUT(MANY), STRONG]. If gates conditioned on measurement outcomes are allowed, then it falls in the [ADAPT, IN(BITS), OUT(MANY), WEAK] category. Both these categories admit efficient simulation on conventional computers (CI-P). Stateless simulation falls in the [NONADAPT, IN(PROD), OUT(1), STRONG] category and also admits efficient classical simulation. Figure 2.6 demonstrates a remarkable sensitivity of the classical simulation complexity of Clifford circuits under various small modifications and thus a surprising proximity of classical to quantum computing power.

2.5 Summary

In this chapter, we introduce our notion of quantum-circuit simulation and emphasize key properties of known simulation techniques that are relevant to our work. We review the stabilizer formalism, restating major results from the published literature, with proofs, emphasizing efficient computation. In particular, we design an algorithm to maintain the global phase of a stabilizer state during simulation. As we show in Chapter V, global-phase maintenance facilitates our technique for simulation of generic quantum circuits using superpositions of stabilizer states. Our technique avoids the shortcomings in prior work [2] and exhibits significant speedups over state-of-the-art simulators for specific circuits.

		NONADAPT		ADAPT	
		WEAK	STRONG	WEAK	STRONG
OUT(1)	IN(BITS)	Cl-P	Cl-P	Cl-P	#P-hard <i>(Theorem 2)</i>
	IN(PROD)	Cl-P	Cl-P <i>(Theorem 1)</i>	QC-hard <i>(Theorem 3)</i>	#P-hard
OUT(MANY)	IN(BITS)	Cl-P	Cl-P <i>(Theorem 4)</i>	Cl-P <i>(Theorem 5)</i>	#P-hard
	IN(PROD)	If Cl-P then PH collapses <i>(Theorem 7)</i>	#P-hard <i>(Theorem 6)</i>	QC-hard	#P-hard

Figure 2.6: Classical simulation complexities for sets of Clifford computational tasks [40]. IN(BITS) and IN(PROD) refer to allowing computational basis states and general product states as inputs. OUT(1) and OUT(MANY) refer to having single bit and multi-bit outputs. NONADAPT and ADAPT refer to circuits with intermediate measurements and gates conditioned on measurement outcomes (ADAPT). WEAK and STRONG refer to two notions of classical simulation as defined in Section 2.1.1. Cl-P denotes that classical efficient simulation is possible, QC-hard denotes that universal quantum computation is possible, and #P-hard asserts that classical simulation could be used to solve arbitrary problems in the classical class #P (and hence NP too).

CHAPTER III

Metric Geometry of Stabilizer States

Despite their compact representation, stabilizer states can exhibit multi-qubit entanglement and are encountered in many quantum information applications such as Bell states, GHZ states, error-correcting codes and one-way quantum computation. To better understand the role stabilizer states play in such applications, researchers have designed techniques to quantify the amount of entanglement [26, 37, 69] in such states and studied related topics such as purification schemes [19], Bell inequalities [35] and equivalence classes [15]. In particular, the authors of [16, 42, 48] show that the uniform distribution over the n -qubit stabilizer states is an exact 2-design¹. Therefore, similar to general random quantum states, stabilizer states exhibit a distribution that is close to uniform. Furthermore, the results from [10, 63] show that the entanglement of stabilizer states is nearly maximal and similar to that of general random states. This suggests the possibility of using stabilizer states as proxies for generic quantum states, e.g., represent arbitrary states by superpositions of stabilizer states. To this end, the work in [1] proposes a hierarchy for quantum states based on their complexity — the number of classical bits required to describe the state. Such a hierarchy can be used to describe which classes of states admit polynomial-size classical descriptions of various kinds. To explore the boundaries between these classes,

¹A state k -design is an ensemble of states such that, when one state is chosen from the ensemble and copied k times, it is indistinguishable from a uniformly random state [16, 36].

one can design new stabilizer-based representations (e.g., superpositions of stabilizer states and tensor products of such superpositions) that (i) admit polynomial-size descriptions of practical non-stabilizer states, and (ii) facilitate efficient simulation of generic quantum circuits.

In this chapter, we advance the understanding of the geometry of stabilizer states. This line of research helps us identify efficient techniques for representing and manipulating new classes of quantum states, rule out some techniques as inefficient, and quantify entanglement of relevant states. Our work contributes the following:

- (1) We quantify the distribution of angles between pairs of stabilizer states and characterize *nearest-neighbor stabilizer states*.
- (2) We study linearly-dependent sets of stabilizer states and show that every linearly-dependent triplet of such states that are non-parallel to each other includes two pairs of nearest neighbors and one pair of orthogonal states. Such triplets are illustrated in Figure 3.1. We also describe an orthogonalization procedure for stabilizer states that exploits this rather uniform nearest-neighbor structure.
- (3) We show that: (i) for any n -qubit stabilizer state $|\psi\rangle$, there are at least $5(2^n - 1)$ states $|\varphi\rangle$ such that $|\psi \wedge \varphi\rangle$ is a *stabilizer bivector*, and (ii) the norm of the

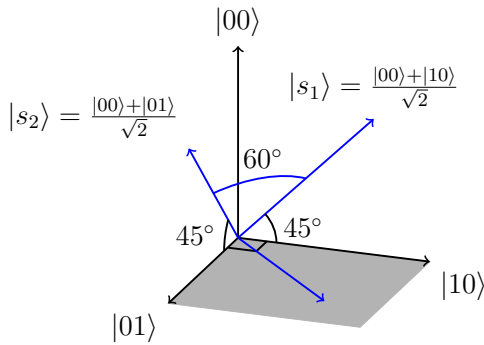


Figure 3.1: The angle between any stabilizer state and its nearest neighbors is 45° and the distance is $\sqrt{2} - \sqrt{2}$. Here, $|s_1\rangle$ is a nearest neighbor of both $|00\rangle$ and $|10\rangle$. Similarly, $|s_2\rangle$ is a nearest neighbor of $|00\rangle$ and $|01\rangle$. The angle between these two nearest neighbors of $|00\rangle$ is 60° . Consider the linearly-dependent triplets $\{|00\rangle, |10\rangle, |s_1\rangle\}$ and $\{|00\rangle, |01\rangle, |s_2\rangle\}$. Each set contains two pairs of nearest neighbors and one pair of orthogonal states.

wedge product between any two stabilizer states is $\sqrt{1 - 2^{-k}}$, where $0 \leq k \leq n$.

- (4) We explore the approximation of non-stabilizer states by single stabilizer states and short superpositions of stabilizer states.

3.1 Geometric Properties of Stabilizer States

Given $\langle \psi | \varphi \rangle = re^{i\alpha}$, we normalize the global phase of $|\psi\rangle$ to ensure, without loss of generality, that $\langle \psi | \varphi \rangle \in \mathbb{R}_+$.

Theorem III.1. *Let $S(|\psi\rangle)$ and $S(|\varphi\rangle)$ be the stabilizer groups for $|\psi\rangle$ and $|\varphi\rangle$, respectively. If there exist $P \in S(|\psi\rangle)$ and $Q \in S(|\varphi\rangle)$ such that $P = -Q$, then $|\psi\rangle$ and $|\varphi\rangle$ are orthogonal.*

Proof. Since $|\psi\rangle$ is a 1-eigenvector of P and $|\varphi\rangle$ is a (-1) -eigenvector of P , they must be orthogonal. \square

Theorem III.2. [2] *Let $|\psi\rangle, |\varphi\rangle$ be oblique stabilizer states. Let s be the minimum, over all sets of generators $\{P_1, \dots, P_n\}$ for $S(|\psi\rangle)$ and $\{Q_1, \dots, Q_n\}$ for $S(|\varphi\rangle)$, of the number of different i values for which $P_i \neq Q_i$. Then, $|\langle \psi | \varphi \rangle| = 2^{-s/2}$.*

Proof. Since $\langle \psi | \varphi \rangle$ is not affected by unitary transformations U , we choose a stabilizer circuit such that $U|\psi\rangle = |b\rangle$, where $|b\rangle$ is a basis state. For this state, select the stabilizer generators \mathcal{M} of the form $I \dots IZI \dots I$. Perform Gaussian elimination on \mathcal{M} to minimize the incidence of $P_i \neq Q_i$. Consider two cases. If $U|\varphi\rangle \neq |b\rangle$ and its generators contain only I/Z literals, then $U|\varphi\rangle \perp U|\psi\rangle$, which contradicts the assumption that $|\psi\rangle$ and $|\varphi\rangle$ are oblique. Otherwise, each generator of $U|\varphi\rangle$ containing X/Y literals contributes a factor of $1/\sqrt{2}$ to the inner product. \square

Corollary III.3. *Let $|\psi\rangle$ and $|\phi\rangle$ be oblique n -qubit stabilizer states such that $|\psi\rangle \neq e^{i\alpha} |\phi\rangle$. Then, $2^{-n/2} \leq |\langle \psi | \phi \rangle| \leq 2^{-1/2}$.*

Corollary III.4. *Given a stabilizer state $|\psi\rangle$, there exists an orthonormal basis including $|\psi\rangle$ and consisting entirely of stabilizer states.*

Proof. Observe that, by Theorem III.1, one can create a state $|\varphi\rangle$ that is orthogonal to $|\psi\rangle$ by changing the signs of an arbitrary non-empty subset of generators of $S(|\psi\rangle)$. Moreover, selecting two different subsets will produce two mutually orthogonal states. Thus, one can produce $2^n - 1$ additional orthogonal stabilizer states. Such states, together with $|\psi\rangle$, form an orthonormal basis. \square

Corollary III.4 is illustrated by Table 2.4 where each row constitutes an orthonormal basis. More generally, the orthogonality of two stabilizer states can sometimes be established by finding a pair of generators P and Q that satisfy the condition that $P = -Q$. For n -qubit states, this search can be performed in $\Theta(n^2 \log(n))$ worst-case time by sorting the generators (e.g., using the lexicographical ordering from Definition II.14), or even in $\Theta(n^2)$ time with radix sort. Alternatively, this search can be performed using hashing in expected $\Theta(n^2)$ time. Pre-sorting the generators does not improve asymptotic complexity, but if the generators are pre-hashed (or if the hashes are incrementally updated during simulation), then the search can be performed in $\Theta(n)$ time.

Unfortunately, the $P = -Q$ condition may not hold for any generators (from given generating sets) of two orthogonal stabilizer states.² In this case, orthogonality can be checked by computing the inner product, which takes $O(n^3)$ time as discussed in Section 4.2.

Lemma III.5. *Let $|\alpha\rangle$ and $|\beta\rangle$ be arbitrary states such that $\langle\alpha|\beta\rangle = 0$ and $\langle\alpha|\alpha\rangle = \langle\beta|\beta\rangle = 1$. Then, each of the four unbiased superpositions $|\psi_+\rangle = \frac{|\alpha\rangle+|\beta\rangle}{\sqrt{2}}$, $|\psi_-\rangle = \frac{|\alpha\rangle-|\beta\rangle}{\sqrt{2}}$, $|\psi_{+i}\rangle = \frac{|\alpha\rangle+i|\beta\rangle}{\sqrt{2}}$ and $|\psi_{-i}\rangle = \frac{|\alpha\rangle-i|\beta\rangle}{\sqrt{2}}$, satisfies $\langle\psi|\psi\rangle = 1$ and $|\langle\psi|\alpha\rangle| = |\langle\psi|\beta\rangle| = \frac{1}{\sqrt{2}}$.*

²Table 2.4 contains pairs of orthogonal states not in the same row for whose generators this condition does not hold.

Proof. Consider the case of $|\psi_{-i}\rangle$.

$$\langle\psi_{-i}|\psi_{-i}\rangle = \frac{\langle\psi_{-i}|\alpha - i\beta\rangle}{\sqrt{2}} = \frac{\langle\psi_{-i}|\alpha\rangle - i\langle\psi_{-i}|\beta\rangle}{\sqrt{2}} \quad (3.1)$$

Since the inner product is conjugate-linear on the first argument,

$$\frac{\langle\psi_{-i}|\alpha\rangle - i\langle\psi_{-i}|\beta\rangle}{\sqrt{2}} = \frac{\frac{1}{\sqrt{2}}\langle\alpha|\alpha\rangle - i\frac{i}{\sqrt{2}}\langle\beta|\beta\rangle}{\sqrt{2}} = 1 \quad (3.2)$$

Furthermore, $|\langle\psi_{-i}|\alpha\rangle| = \left|\frac{\langle\alpha|\alpha\rangle}{\sqrt{2}}\right| = \frac{1}{\sqrt{2}}$ and $|\langle\psi_{-i}|\beta\rangle| = \left|\frac{i\langle\beta|\beta\rangle}{\sqrt{2}}\right| = \frac{1}{\sqrt{2}}$. Other cases are analogous. \square

3.1.1 Inner products and k -neighbor stabilizer states

Definition III.6. Given an arbitrary state $|\psi\rangle$ with $\|\psi\| = 1$, a stabilizer state $|\varphi\rangle$ is a k -neighbor stabilizer state of $|\psi\rangle$ if $|\langle\psi|\varphi\rangle| = 2^{-k/2}$.

When two stabilizer states are 1-neighbors we will also refer to them as *nearest neighbors* since, by Corollary III.3, this is the maximal inner-product value $\neq 1$. Furthermore, the distance between a stabilizer state and any of its k -neighbors is $\sqrt{2 - 2^{1-k/2}}$. Therefore, the distance between closest stabilizer states (nearest neighbors) is $\sqrt{2 - \sqrt{2}} \approx 0.765$.

Proposition III.7. Consider two orthogonal stabilizer states $|\alpha\rangle$ and $|\beta\rangle$ whose unbiased superposition $|\psi\rangle$ is also a stabilizer state. Then $|\psi\rangle$ is a nearest neighbor of $|\alpha\rangle$ and $|\beta\rangle$.

Proof. Since stabilizer states are unbiased, $|\langle\psi|\alpha\rangle| = |\langle\psi|\beta\rangle| = \frac{1}{\sqrt{2}}$. Thus, $|\psi\rangle$ is a 1-neighbor or nearest-neighbor stabilizer state to $|\alpha\rangle$ and $|\beta\rangle$. Figure 3.1 illustrates this case. \square

Lemma III.8. Any two stabilizer states have equal numbers of k -neighbor stabilizer states.

Proof. Any stabilizer state can be mapped to another stabilizer state by a stabilizer circuit (Corollary II.12). Since such circuits effect unitary operators, inner products are preserved. \square

Lemma III.9. *Let $|\psi\rangle$ and $|\varphi\rangle$ be n -qubit stabilizer states such that $\langle\psi|\varphi\rangle \neq 1$. Then $\frac{|\psi\rangle+i^l|\varphi\rangle}{\sqrt{2}}$, $l \in \{0, 1, 2, 3\}$, is a stabilizer state iff $\langle\psi|\varphi\rangle = 0$ and $|\varphi\rangle = P|\psi\rangle$, where $P \in \mathcal{G}_n$.*

Proof. We first prove that, if $\langle\psi|\varphi\rangle = 0$ and $|\varphi\rangle = P|\psi\rangle$, an unbiased sum of such states is also a stabilizer state. Suppose $S(|\psi\rangle) = \langle g_k \rangle_{k=1,2,\dots,n}$ is generated by elements g_k of the n -qubit Pauli group. Let

$$f(k) = \begin{cases} 0 & \text{if } [P, g_k] = 0 \\ 1 & \text{otherwise} \end{cases}$$

and write $S(|\varphi\rangle) = \langle (-1)^{f(k)} g_k \rangle$. Conjugating each generator g_k by P we see that $|\varphi\rangle$ is stabilized by $\langle (-1)^{f(k)} g_k \rangle$. Let Z_k (respectively X_k) denote the Pauli operator Z (X) acting on the k^{th} qubit. By Corollary II.12, there exists an element \mathbf{C} of the n -qubit Clifford group such that $\mathbf{C}|\psi\rangle = |0\rangle^{\otimes n}$ and $\mathbf{C}|\varphi\rangle = (\mathbf{C}P\mathbf{C}^\dagger)\mathbf{C}|\psi\rangle = i^m |f(1)f(2)\dots f(n)\rangle$. The second equality follows from the fact that $\mathbf{C}P\mathbf{C}^\dagger$ is an element of the Pauli group and can therefore be written as $i^m X(v)Z(u)$ for some $m \in \{0, 1, 2, 3\}$ and $u, v \in \mathbb{Z}_2^k$. Therefore,

$$\frac{|\psi\rangle + i^l|\varphi\rangle}{\sqrt{2}} = \frac{\mathbf{C}^\dagger(|0\rangle^{\otimes n} + i^{t=(l+m)\bmod 4} |f(1)f(2)\dots f(n)\rangle)}{\sqrt{2}} \quad (3.3)$$

The state in parentheses on the right-hand side is the product of an all-zeros state and a GHZ state. Therefore, the sum is stabilized by $S' = \mathbf{C}^\dagger \langle S_{zero}, S_{ghz} \rangle \mathbf{C}$ where $S_{zero} = \langle Z_i, i \in \{k | f(k) = 0\} \rangle$ and S_{ghz} is supported on $\{k | f(k) = 1\}$ and equals $\langle (-1)^{t/2} X X \dots X, \forall i Z_i Z_{i+1} \rangle$ if $t = 0 \pmod 2$ or $\langle (-1)^{(t-1)/2} Y Y \dots Y, \forall i Z_i Z_{i+1} \rangle$ if $t = 1 \pmod 2$.

We now prove the opposite implication. Let $|u\rangle = \frac{|\psi\rangle + i^l |\varphi\rangle}{\sqrt{2}}$, where $|\psi\rangle$ and $|\varphi\rangle$ are n -qubit stabilizer states and $\langle\psi|\varphi\rangle \neq 1$. Let \mathbf{C}_1 and \mathbf{C}_2 be Clifford circuits such that $|\psi\rangle = \mathbf{C}_1 |\mathbf{0}\rangle$ and $|\varphi\rangle = \mathbf{C}_2 |\mathbf{0}\rangle$, where $|\mathbf{0}\rangle = |0\rangle^{\otimes n}$. Observe that $\mathbf{C}_1 \neq \mathbf{C}_2$ by our assumption that $\langle\psi|\varphi\rangle \neq 1$. Therefore, $|u\rangle = (\mathbf{C}_1 |\mathbf{0}\rangle + i^l \mathbf{C}_2 |\mathbf{0}\rangle)/\sqrt{2} = \mathbf{C}_1(|\mathbf{0}\rangle + i^l \mathbf{C}_1^\dagger \mathbf{C}_2 |\mathbf{0}\rangle)/\sqrt{2}$, and

$$\mathbf{C}_1^\dagger |u\rangle = \frac{|\mathbf{0}\rangle + i^l \mathbf{C}_1^\dagger \mathbf{C}_2 |\mathbf{0}\rangle}{\sqrt{2}}$$

Since $\mathbf{C}_1^\dagger \mathbf{C}_2 \neq I^{\otimes n}$ and $\mathbf{C}_1^\dagger |u\rangle$ is a stabilizer state, $\mathbf{C}_1^\dagger \mathbf{C}_2 |\mathbf{0}\rangle$ must simplify to a basis state $|b\rangle \neq |\mathbf{0}\rangle$ (otherwise, $\mathbf{C}_1^\dagger |u\rangle$ is either biased or has $2^n - 1$ basis states). It follows that $\langle\psi|\varphi\rangle = \langle\mathbf{0}|\mathbf{C}_1^\dagger \mathbf{C}_2 |\mathbf{0}\rangle = \langle\mathbf{0}|b\rangle = 0$. Let $i^l |b\rangle = P |\mathbf{0}\rangle$, where P is an element of the Pauli group,

$$\begin{aligned} \mathbf{C}_1^\dagger |u\rangle &= |\mathbf{0}\rangle + P |\mathbf{0}\rangle \\ |u\rangle &= \mathbf{C}_1 |\mathbf{0}\rangle + \mathbf{C}_1 P |\mathbf{0}\rangle = \mathbf{C}_1 |\mathbf{0}\rangle + (\mathbf{C}_1 P \mathbf{C}_1^\dagger) \mathbf{C}_1 |\mathbf{0}\rangle \\ &= \mathbf{C}_1 |\mathbf{0}\rangle + P' \mathbf{C}_1 |\mathbf{0}\rangle = |\psi\rangle + P' |\psi\rangle \end{aligned}$$

□

Corollary III.10. *Let $\frac{|\psi\rangle + i^l |\varphi\rangle}{\sqrt{2}}$ be a stabilizer state. Then the canonical stabilizer matrices for $|\psi\rangle$ and $|\varphi\rangle$ are similar.*

Proof. Let $\mathcal{M}^\psi = \{R_1, R_2, \dots, R_n\}$ be the canonical stabilizer matrix for $|\psi\rangle$. Since $|\varphi\rangle = P |\psi\rangle$ by Lemma III.9, $\mathcal{M}^\varphi = \{(-1)^c R_1, (-1)^c R_2, \dots, (-1)^c R_n\}$, where $c = 0$ if P and R_i commute, and $c = 1$ otherwise. □

Theorem III.11. *For any n -qubit stabilizer state $|\psi\rangle$, there are $4(2^n - 1)$ nearest-neighbor stabilizer states, and these states can be produced as described in Lemma III.9.*

Proof. The all-zeros basis amplitude of any stabilizer state $|\psi\rangle$ that is a nearest neighbor to $|0\rangle^{\otimes n}$ must be $\propto 1/\sqrt{2}$. Therefore, $|\psi\rangle$ is an unbiased superposition of $|0\rangle^{\otimes n}$

and one of the other $2^n - 1$ basis states, i.e., $|\psi\rangle = \frac{|0\rangle^{\otimes n} + P|0\rangle^{\otimes n}}{\sqrt{2}}$, where $P \in \mathcal{G}_n$ such that $P|0\rangle^{\otimes n} \neq \alpha|0\rangle^{\otimes n}$. As in the proof of Lemma III.9, we have $|\psi\rangle = \frac{|0\rangle^{\otimes n} + i^l|\varphi\rangle}{\sqrt{2}}$, where $|\varphi\rangle$ is a basis state and $l \in \{0, 1, 2, 3\}$. Thus, there are 4 possible unbiased superpositions, and a total of $4(2^n - 1)$ nearest-neighbor stabilizer states. Since $|0\rangle^{\otimes n}$ is a stabilizer state, all stabilizer states have the same number of nearest neighbors by Lemma III.8. \square

Corollary III.12. *For any n -qubit stabilizer state $|\psi\rangle$, there exists a set of states $V_\psi = \{|s_i\rangle\}_{i=1}^{2^n-1}$ such that each $|s_i\rangle$ is a nearest-neighbor to $|\psi\rangle$ and $\langle s_i|s_j\rangle = 1/2$ for $i \neq j$. This set V_ψ together with $|\psi\rangle$ forms a basis in $\mathcal{H}^{\otimes n}$.*

Proof. Without loss of generality, assume that $|\psi\rangle = |0\rangle^{\otimes n}$. Theorem III.11 shows that, for any given stabilizer states, its nearest neighbors come in groups of four. Taking any one representative from each group of four, we get a set of states $V_\psi = \left\{ |s_i\rangle = \frac{|0\rangle^{\otimes n} + i^l|b_i\rangle}{\sqrt{2}} \right\}_{i=1}^{2^n-1}$, where $l \in \{0, 1, 2, 3\}$ and $|b_i\rangle$ are computational-basis states other than $|\psi\rangle$. Thus, for all $|s_i\rangle$ and $|s_j\rangle$ such that $i \neq j$,

$$\langle s_i|s_j\rangle = \left(\frac{\langle 0^{\otimes n}| + \langle b_i|}{\sqrt{2}} \right) \left(\frac{|0^{\otimes n}\rangle + |b_j\rangle}{\sqrt{2}} \right) = \frac{1}{2} \quad (3.4)$$

V_ψ together with $|\psi\rangle$ form a linearly independent set (one can subtract $|\psi\rangle$ from each $|s_i\rangle$ to get an orthogonal set) and thus a basis. Figure 3.1 illustrates this nearest-neighbor structure for a small set of states. \square

In Table 2.4, one can find the twelve nearest-neighbor states of $|00\rangle$. We computed the angles between all-pairs of 3-qubit stabilizer states and confirmed that each was surrounded by exactly 28 nearest neighbors. The same procedure confirmed that the number of nearest neighbors for any 4-qubit stabilizer state is 60. In Section 4.3, we describe an orthogonalization procedure for linear combinations of stabilizer states that takes advantage of the nearest-neighbor structure described in Theorem III.11.

Alternatively, Theorem III.11 can also be proven using a counting argument. By Theorem III.2, we know that any nearest-neighbor stabilizer state to $|0^{\otimes n}\rangle$ can be represented by a stabilizer matrix that has exactly one generator (row) with at least one X/Y literal. Therefore, there are $2(4^n - 2^n)$ choices for the first generator P_1 . The factor of 2 accounts for the possible signs of P_1 . The remaining (independent) generators, P_2, \dots, P_n , are then selected such that they commute with P_1 and consist of Z/I literals only. Observe that such generators can be selected arbitrarily since they generate the same (Z/I) -element subgroup.

Example III.13. Suppose $P_1 = XII$, then P_2 and P_3 can be selected arbitrarily from the set $\{IZI, IZZ, IIZ\}$. To account for stabilizer matrices that describe the same state, consider that P_1 can be replaced by P_1Q , where Q is an arbitrary product of the Z/I generators.

Therefore, the number of nearest-neighbor stabilizer states $\mathcal{L}_n(1)$ is given by,

$$\mathcal{L}_n(1) = \frac{2(4^n - 2^n)}{2^{n-1}} = 4(2^n - 1) \quad (3.5)$$

The following theorem generalizes Equation 3.5 to the case of k -neighbor stabilizer states.

Theorem III.14. *For any n -qubit stabilizer state, the number of k -neighbor stabilizer states is,*

$$\mathcal{L}_n(k) = 2^{k(k+1-n)} \prod_{j=0}^{k-1} \frac{4^n/2^j - 2^n}{2^k - 2^j} \quad (3.6)$$

Proof. Without loss of generality, we count the k -neighbors of $|0^{\otimes n}\rangle$ (Lemma III.8). By Theorem III.2, we know that such states are represented by an n -qubit stabilizer matrix with k independent X/Y generators and $n - k$ independent Z/I generators. Assume, for $j \leq k$, that the X/Y generators P_1, \dots, P_{j-1} have been chosen, and let Q_j, \dots, Q_n be Z/I generators that commute with them. Given this generating set, we

count the possible choices for P_j to replace any one of the Z/I generators. Observe that P_j must commute with P_1, \dots, P_{j-1} and cannot be an element of the subgroup generated by $P_1, \dots, P_{j-1}, Q_j, \dots, Q_n$. Thus, there are $2(4^n/2^{j-1} - 2^n)$ choices for P_j . The factor of 2 accounts for the choice of sign. We need to account for choices of X/Y generators that describe the same state. Consider the subgroup generated by $P_{i \in \{1, \dots, k\}}$. There are $2^k - 1$ choices for P_1 , $2^k - 2$ for P_2 , $2^k - 4$ for P_3 , and so on. This gives a factor of $\prod_{j=1}^k (2^k - 2^{j-1})$. Furthermore, we can replace P_i by $P_i Q$, where Q is an arbitrary product of the Z/I generators. This gives k factors of 2^{n-k} .

$$\mathcal{L}_n(k) = \prod_{j=1}^k \frac{2}{2^{n-k}} \frac{4^n/2^{j-1} - 2^n}{2^k - 2^{j-1}}$$

□

Corollary III.15. *For an arbitrary n -qubit stabilizer state $|\psi\rangle$, the number of stabilizer states orthogonal to $|\psi\rangle$ is,*

$$\mathcal{L}_n(\perp) = \mathcal{N}(n) - \sum_{k=1}^n \mathcal{L}_n(k) - 1 = \frac{\mathcal{N}(n)(2^n - 1)}{3 \cdot 2^n} \quad (3.7)$$

where $\mathcal{N}(n)$ is the number of n -qubit stabilizer states (Proposition II.13). In other words, for any n -qubit stabilizer state $|\psi\rangle$ with a sufficiently large n , almost 1/3 of remaining stabilizer states are orthogonal to $|\psi\rangle$.

As an illustration of Theorem III.14 and Corollary III.15, Table 3.1 numerically describes the distribution of inner products between any one n -qubit stabilizer state and all other stabilizer states for $n \in \{1, \dots, 7\}$. This table shows that the number of nearest-neighbor stabilizer states as a fraction of all n -qubit stabilizer states approaches zero as n increases. We now formalize other trends gleaned from Table 3.1.

Lemma III.16. *For an arbitrary n -qubit stabilizer state, consider the quantity $a_{n,k} = \frac{\mathcal{L}_n(k)}{\mathcal{N}(n)}$, where $0 < k \leq n$. Then, for fixed m , the sequence $\{a_{n,n-m}\}^{n \rightarrow \infty}$ is monotonically convergent.*

Table 3.1: Distribution of inner products (angles) between any one n -qubit stabilizer state and all other stabilizer states for $n \in \{1, \dots, 7\}$. The last column indicates the ratio of orthogonal (\perp) states.

n	$\mathcal{N}(n)$	$\mathcal{L}_n(k)/(\mathcal{N}(n) - 1)$							
		$k = 1$ (45.00°)	$k = 2$ (60.00°)	$k = 3$ (69.30°)	$k = 4$ (75.52°)	$k = 5$ (79.82°)	$k = 6$ (82.82°)	$k = 7$ (84.93°)	\perp (90.00°)
1	6	80%							20%
2	60	20.34%	54.24%						25.42%
3	1080	2.59%	20.76%	47.45%					29.19%
4	36720	0.16%	3.05%	20.92%	44.62%				31.25%
5	2423520	0.01%	0.20%	3.27%	20.96%	43.27%			32.29%
6	315057600	$\approx 0\%$	0.01%	0.23%	3.39%	20.97%	42.60%		32.81%
7	81284860800	$\approx 0\%$	$\approx 0\%$	0.01%	0.24%	3.44%	20.97%	42.27%	33.07%

Proof. Using Theorem III.14 and the recurrence relation of Equation 2.2 we obtain,

$$\frac{a_{n,k}}{a_{n+1,k+1}} = \frac{\mathcal{L}_n(k)}{\mathcal{N}(n)} \cdot \frac{2(2^{n+1} + 1)\mathcal{N}(n)}{\mathcal{L}_{n+1}(k+1)} = 2(2^{n+1} + 1) \frac{\mathcal{L}_n(k)}{\mathcal{L}_{n+1}(k+1)} \quad (3.8)$$

where

$$\begin{aligned} \frac{\mathcal{L}_n(k)}{\mathcal{L}_{n+1}(k+1)} &= 2^{n-k-1} \prod_{j=0}^{k-1} \left(\frac{4^n/2^j - 2^n}{2^k - 2^j} \right) \prod_{j=0}^k \left(2 \frac{2^k - 2^{j-1}}{4^{n+1}/2^j - 2^{n+1}} \right) \\ &= 2^n \prod_{j=1}^k \left(\frac{4^n/2^{j-1} - 2^n}{2^k - 2^{j-1}} \right) \prod_{j=0}^k \left(\frac{2^k - 2^{j-1}}{4^{n+1}/2^j - 2^{n+1}} \right) \\ &= 2^n \left(\frac{2^k - 1/2}{4^{n+1} - 2^{n+1}} \right) \prod_{j=1}^k \left(\frac{1}{2} \frac{4^n/2^{j-1} - 2^n}{2(4^n/2^j) - 2^n} \right) = \frac{2^{k+1} - 1}{2^k(2^{n+3} - 4)} \end{aligned}$$

Inserting the above result in Equation 3.8 gives,

$$\frac{a_{n,k}}{a_{n+1,k+1}} = 2(2^{n+1} + 1) \frac{2^{k+1} - 1}{2^k(2^{n+3} - 4)} = \frac{2^{n+k+3} + 2^{k+2} - 2^{n+2} - 2}{2^{n+k+3} - 2^{k+2}} \quad (3.9)$$

Consider the case $0 < k < n$,

$$\frac{2^{n+k+3} + 2^{k+2} - 2^{n+2} - 2}{2^{n+k+3} - 2^{k+2}} < 1 \quad \text{and thus} \quad \frac{2^{k+2}}{2^{n+1} + 1} < 1$$

Therefore, for fixed $0 < m < n$, $\{a_{n,n-m}\}^{n \rightarrow \infty}$ is monotonically increasing. It converges because $0 < a_{n,k} < 1$. For the case $k = n$,

$$\frac{a_{n,n}}{a_{n+1,n+1}} = \frac{2^{2n+3} - 2}{2^{2n+3} - 2^{n+2}} = \frac{2^{n+1} - 2^{-n-1}}{2^{n+1} - 1} = 1 + \frac{1}{2^{n+1}} > 1 \quad (3.10)$$

Therefore, the sequence $\{a_{n,n}\}^{n \rightarrow \infty}$ is monotonically decreasing and convergent. \square

Theorem III.17. *For an arbitrary n -qubit stabilizer state, the limit of the sequence $\{a_{n,n-k}\}^{n \rightarrow \infty}$, $0 \leq k < n$, lies in the interval $[l(k), u(k)]$, where*

$$l(k) = \frac{M_5}{2^{k(k+5)/2}} \cdot \exp\left(\frac{1}{32} - \frac{2}{2^{k+5} - 1}\right) \quad u(k) = \frac{M_5}{2^{k(k+3)/2}} \cdot \exp\left(\frac{1}{15} - \frac{2}{2^{k+5} + 1}\right)$$

and

$$M_5 = \prod_{j=1}^5 \left(\frac{1}{1 - 2^{-j}}\right) \prod_{j=1}^5 \left(1 - \frac{2}{2^{j+k} + 1}\right)$$

Proof. From the proof of Lemma III.16 we know $0 < l(k) < u(k) < 1$. We now derive the sharper bounds, $l(k)$ and $u(k)$, for $\{a_{n,n-k}\}^{n \rightarrow \infty}$. Using Theorem III.14 and Proposition II.13 we obtain,

$$a_{n,n-k} = 2^{-k} \underbrace{\prod_{j=1}^{n-k} \left(\frac{1}{1 - 2^{-j}}\right)}_{A_{n-k}} \underbrace{\prod_{j=1}^{n-k} \left(1 - \frac{2}{2^{j+k} + 1}\right)}_{B_{n-k}} \underbrace{\prod_{j=1}^k \left(\frac{1}{1 + 2^j}\right)}_{C_k} \quad (3.11)$$

We now derive upper and lower bounds for the products A_{n-k} , B_{n-k} and C_k .

$$\begin{aligned} \prod_{j=1}^{n-k} \exp\left(\frac{1}{2^j}\right) &< A_{n-k} < \prod_{j=1}^{n-k} \exp\left(\frac{1}{2^j - 1}\right) \\ \prod_{j=1}^{n-k} \exp\left(\frac{-2}{2^{j+k} - 1}\right) &< B_{n-k} < \prod_{j=1}^{n-k} \exp\left(\frac{-2}{2^{j+k} + 1}\right) \\ \frac{1}{2^{k(k+5)/2}} = \prod_{j=1}^k \frac{1}{2^{j+1}} &< C_k < \prod_{j=1}^k \frac{1}{2^j} = \frac{1}{2^{k(k+1)/2}} \end{aligned}$$

Therefore, as $n \rightarrow \infty$,

$$\begin{aligned} A_5 \cdot \exp\left(\frac{1}{32}\right) &< A_\infty < A_5 \cdot \exp\left(\frac{1}{15}\right) \\ B_5 \cdot \exp\left(\frac{-2}{2^{j+k}-1}\right) &< B_\infty < B_5 \cdot \exp\left(\frac{-2}{2^{j+k}+1}\right) \end{aligned}$$

□

In particular, observe that $u(k) \approx 0$ for $k > 4$. Thus, the fraction of all stabilizer states that are oblique to some n -qubit stabilizer state $|\psi\rangle$ is dominated by its $(n-k)$ -neighbors, where $k \in \{0, 1, 2, 3, 4\}$. Limit values for $\mathcal{L}_n(n-k)/\mathcal{N}(n)$ are approximated in Figure 3.2a.

Theorem III.18. *For two n -qubit stabilizer states $|\psi\rangle$ and $|\phi\rangle$ drawn independently from a uniform distribution, the expected value $E[\langle\psi|\phi\rangle] \rightarrow 0$ as $n \rightarrow \infty$.*

Proof. Let $|\varphi_{n-k}\rangle$ be an $(n-k)$ -neighbor of $|\psi\rangle$. By Theorems III.2 and III.17,

$$E[\langle\psi|\phi\rangle] = \sum_{k=0}^{n-1} a_{n,n-k} |\langle\psi|\varphi_{n-k}\rangle| < \sum_{k=0}^{n-1} u(k) |\langle\psi|\varphi_{n-k}\rangle| \approx \sum_{k=0}^4 u(k) 2^{(k-n)/2} \quad (3.12)$$

Therefore, $E_\infty[\langle\psi|\phi\rangle] < \sum_{k=0}^4 u(k) 2^{(k-n)/2}$, which approaches zero as $n \rightarrow \infty$. □

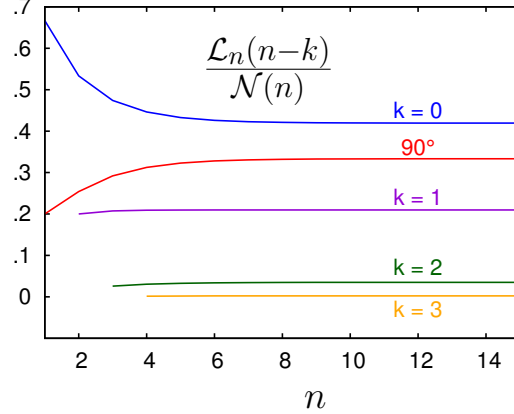
Theorem III.18 suggests that stabilizer states are distributed in a way similar to random quantum states [16, 42, 48].

3.1.2 Exterior products and stabilizer bivectors

In this section, we study pairs of stabilizer states and the parallelograms they form, which lie in a $2n$ -qubit Hilbert space. Such pairs of states are called *bivectors* and are obtained by computing the wedge product $|\psi \wedge \phi\rangle = |\psi\rangle \otimes |\phi\rangle - |\phi\rangle \otimes |\psi\rangle$ between stabilizer states. The wedge product is antisymmetric and yields zero for parallel vectors. The norm of a stabilizer bivector can be interpreted as the *signed area* of the parallelogram defined by non-parallel stabilizer states.

k	$\lim_{n \rightarrow \infty} \frac{\mathcal{L}_n(n-k)}{\mathcal{N}(n)}$
0	41.9422%
1	20.9712%
2	3.4952%
3	.2497%
4	.0083%
5	$\approx 0\%$
\vdots	\vdots
$n-1$	$\approx 0\%$

(a)



(b)

Figure 3.2: (a) For an n -qubit stabilizer state $|\psi\rangle$, the fraction ($\approx 2/3$) of all stabilizer states that are oblique to $|\psi\rangle$ is dominated by its $(n-k)$ -neighbors, where $k \in \{0, 1, 2, 3, 4\}$. (b) The stabilizer states orthogonal to $|\psi\rangle$ together with its $(n-k)$ -neighbors ($k < 3$) account for $\approx 99\%$ of all states.

Definition III.19. A *stabilizer bivector* $|\varphi\rangle$ is a $2n$ -qubit stabilizer state such that $|\varphi\rangle \propto |\psi \wedge \phi\rangle$, where $|\psi\rangle$ and $|\phi\rangle$ are n -qubit stabilizer states.

Example III.20. The wedge product of the stabilizer states $|\psi\rangle = |00\rangle + |11\rangle$ and $|\phi\rangle = |00\rangle - |11\rangle$ produces the stabilizer bivector $|\psi \wedge \phi\rangle = |1100\rangle + |0011\rangle$ (up to a phase). In contrast, given the states $|\psi\rangle = |00\rangle + |11\rangle$ and $|\phi\rangle = |10\rangle$, $|\psi \wedge \phi\rangle = |0010\rangle + |1100\rangle + |1110\rangle - |0011\rangle - |1000\rangle - |1011\rangle$ is not a stabilizer bivector because the number of constituent basis states is not a power of two (Corollary II.21-ii).

The work in [20, 38] derives measures of entanglement for general pure bipartite states based on the alternating tensor product space defined by the wedge product of two states. Furthermore, Hayashi et al. [23] conclude that the antisymmetric basis states (the wedge product of basis states) are more entangled than any symmetric basis states. Therefore, stabilizer bivectors are potential candidates for developing new entanglement monotones for quantifying quantum resources. In Section 4.4, we describe how to efficiently compute a generator set for stabilizer bivectors.

Theorem III.21. *For any n -qubit stabilizer state $|\psi\rangle$, there are at least $5(2^n - 1)$ distinct stabilizer states $|\phi\rangle$ such that the wedge product $|\psi \wedge \phi\rangle$ is a stabilizer bivector.*

Proof. By Proposition II.15, $|\psi\rangle|\phi\rangle$ and $|\phi\rangle|\psi\rangle$ are both stabilizer states. Consider two cases.

Case 1: $|\psi\rangle$ and $|\phi\rangle$ are orthogonal. From Lemma III.9 and Corollary III.10 we know that, if $|\psi\rangle|\phi\rangle - |\phi\rangle|\psi\rangle$ is a stabilizer state, the canonical matrices of $|\psi\rangle$ and $|\phi\rangle$ must be *similar* (Definition II.24). Since each *dissimilar* matrix (Definition II.24) can be used to represent 2^n states (the number of possible phase-vector permutations) and $|\psi \wedge \psi\rangle = 0$, there are $2^n - 1$ such wedge products that produce stabilizer bivectors.

Case 2: $|\psi\rangle$ and $|\phi\rangle$ are 1-neighbors. By Theorem III.11, $|\psi\rangle = \frac{|\phi\rangle + P|\phi\rangle}{\sqrt{2}}$ and $|\phi\rangle = \frac{|\psi\rangle + P'|\psi\rangle}{\sqrt{2}}$, where P and P' are Pauli operators. Therefore,

$$|\psi \wedge \phi\rangle = \left(\frac{|\phi\rangle + P|\phi\rangle}{\sqrt{2}} \right) \wedge |\phi\rangle = \frac{(P|\phi\rangle) \wedge |\phi\rangle}{\sqrt{2}} \quad (3.13)$$

Since $|\phi\rangle$ and $P|\phi\rangle$ are orthogonal with similar stabilizer matrices (Case 1), $|\psi \wedge \phi\rangle$ is a stabilizer bivector (up to a normalizing factor), and there are $4(2^n - 1)$ such wedge products.

We now show that for k -neighbors $|\psi\rangle$ and $|\phi\rangle$, $k > 1$, $|\psi \wedge \phi\rangle$ is not a stabilizer bivector. Without loss of generality, let $|\psi\rangle = |0\rangle$ and $|\phi\rangle = \sum_{i=0}^{2^k-1} \alpha_i |b_i\rangle$, where the $|b_i\rangle$ are computational-basis states.

$$|\psi \wedge \phi\rangle = |0\rangle \wedge \left(\sum_{i=0}^{2^k-1} \alpha_i |b_i\rangle \right) = \sum_{i=1}^{2^k-1} |0 \wedge b_i\rangle \quad (3.14)$$

Therefore, $|\psi \wedge \phi\rangle$ has $2^{k+1} - 2$ computational-basis states, which is not a power of 2 for $k > 1$. By Corollary II.21-*i*, $|\psi \wedge \phi\rangle$ is not a stabilizer bivector for $(k > 1)$ -neighbors. \square

Proposition III.22. *Given any two n -qubit stabilizer states $|\psi\rangle$ and $|\phi\rangle$, the area of*

the parallelogram formed by these states is $\|\psi \wedge \phi\| = \sqrt{1 - 2^{-k}}$, $0 \leq k \leq n$.

Proof. Consider the determinant of the *Gramian matrix* of the states,

$$\det \begin{vmatrix} 1 & \langle \psi | \phi \rangle \\ \langle \phi | \psi \rangle & 1 \end{vmatrix} = \|\psi \wedge \phi\|^2 = 1 - |\langle \psi | \phi \rangle|^2$$

Thus, by Theorem III.2, $\|\psi \wedge \phi\| = \sqrt{1 - 2^{-k}}$. □

3.1.3 Linear dependence of stabilizer states

We now characterize linearly-dependent triplets of stabilizer states and discuss properties of minimally-dependent sets. Such properties can help in exploring succinct representations of arbitrary pure states using linear combinations of stabilizer states.

Corollary III.23 (of Theorem III.11). *Every linearly-dependent triplet $\{|s_1\rangle, |s_2\rangle, |s_3\rangle\}$ of stabilizer states that are non-parallel to each other includes two pairs of nearest neighbors and one pair of orthogonal states. Furthermore, every nearest-neighbor pair gives rise to a triplet of linearly-dependent stabilizer states with two pairs of nearest neighbors and one pair of orthogonal states.*

Proof. Without loss of generality, assume that $|s_1\rangle = \alpha |s_2\rangle + \beta |s_3\rangle$ and $|s_1\rangle = |00\dots 0\rangle$. Recall from Corollary II.21 that the non-zero amplitudes of a stabilizer state $|s\rangle$ are $\pm 1/\sqrt{|s|}$ or $\pm i/\sqrt{|s|}$ and its support $|s|$ is a power of two. By our assumptions, the supports of $|s_2\rangle$ and $|s_3\rangle$ cannot differ by more than one.

Case 1: $|s_2| \neq |s_3|$. Then one of $|s_2\rangle$ and $|s_3\rangle$ has support one, while the other has support two. Suppose $|s_2| = 2$. By Lemma III.9, $|s_2\rangle = \frac{|00\dots 0\rangle + i^l |b\rangle}{\sqrt{2}}$, where $l \in \{0, 1, 2, 3\}$ and $|b\rangle$ is computational-basis state other than $|00\dots 0\rangle$. Thus, $|s_1\rangle$ and $|s_2\rangle$ are nearest neighbors. Since $|s_3| = 1$ and the triplet is linearly dependent, $|s_3\rangle = \sqrt{2}|s_2\rangle - |s_1\rangle = i^l |b\rangle$. Therefore, $|s_3\rangle$ is orthogonal to $|s_1\rangle$ and a nearest neighbor to $|s_2\rangle$.

Case 2: $|s_2| = |s_3|$. At least one of $|s_2\rangle$ and $|s_3\rangle$ must have a non-zero amplitude at $|00\dots 0\rangle$. If only one does, say $|s_2\rangle$, then $|s_3\rangle$ will have a non-zero amplitude at some other basis state where $|s_2\rangle$ has zero amplitude, preventing $|s_1\rangle = \alpha |s_2\rangle + \beta |s_3\rangle$ from being $|00\dots 0\rangle$. Thus, both $|s_2\rangle$ and $|s_3\rangle$ must have a non-zero amplitude at $|00\dots 0\rangle$ and, more generally, their non-zero amplitudes must all be at the same basis elements.

As non-zero amplitudes of stabilizer states must be either real or imaginary, we normalize the global phases of $|s_2\rangle$ and $|s_3\rangle$ so that the $|00\dots 0\rangle$ amplitudes are real, which implies that α and β are real. Using additional multiplication by ± 1 , we ensure that $\alpha, \beta > 0$. Considering the amplitudes at $|00\dots 0\rangle$, linear dependence implies $\pm \frac{\alpha}{\sqrt{|s_2|}} \pm \frac{\beta}{\sqrt{|s_3|}} = 1$. Furthermore, $\alpha, \beta > 0$ and $|s_2| = |s_3|$ leave three possibilities: $\alpha \pm \beta = \sqrt{|s_2|}$ and $\beta - \alpha = \sqrt{|s_2|}$. At any other basis element where $|s_2\rangle$ and $|s_3\rangle$ have non-zero amplitudes, there is the additional constraint that $\alpha = \beta$ since these amplitudes must cancel out while $\alpha, \beta > 0$. This eliminates two of the three possibilities above, implying $\alpha = \frac{1}{\sqrt{|s_2|}}$, $\beta = \frac{1}{\sqrt{|s_2|}}$ and $|s_2| = |s_3| = 2$. By Lemma III.9, $|s_2\rangle$ and $|s_3\rangle$ are nearest neighbors of $|s_1\rangle$ and $\langle s_2 | s_3 \rangle = 0$. \square

Minimally-dependent sets of stabilizer states. Given $k > 3$, we now consider sets of k stabilizer states that are *linearly dependent*, but such that all of their proper subsets are *independent*. One such possibility are sets that contain $k - 1$ mutually orthogonal stabilizer states and their sum. Consider the computational basis and a superposition of basis states that is also a stabilizer state. Theorem II.19 and Corollary II.21 suggest examples of such minimally-dependent sets with $k = 2^m + 1$ stabilizer states. Suppose $k = 2^m - d$ and $|s\rangle$ is a stabilizer state in a superposition of all 2^m computational-basis states. To construct a minimally-dependent set of size k , replace $2d + 2$ basis states with $d + 1$ stabilizer states formed as half-sums of $d + 1$ disjoint pairs of basis states. Note that all stabilizer states except for $|s\rangle$ remain orthogonal and contribute to $|s\rangle$.

Example III.24. Let $|s\rangle = \frac{1}{2\sqrt{2}} \sum_{i=0}^7 |b_i\rangle$ where each $|b_i\rangle$ is a computational-basis state. One can obtain a minimally-dependent set of size $k = 5$ as $\{(|b_0\rangle + |b_1\rangle), (|b_2\rangle + |b_3\rangle), (|b_4\rangle + |b_5\rangle), (|b_6\rangle + |b_7\rangle), |s\rangle\}$, where each state in parentheses is a half-sum of two basis states.

We close this section by outlining how to test the linear (in)dependence of a finite set of stabilizer states $\mathbf{s} = \{|s_1\rangle, \dots, |s_k\rangle\}$. Recall that the *Gramian* of \mathbf{s} is a matrix whose entries are given by $\langle s_j | s_i \rangle$, and that \mathbf{s} is linearly dependent if and only if the determinant of its Gramian matrix is zero. In Section 4.2, we describe an inner-product algorithm for stabilizer states. Thus, to test the linear (in)dependence of \mathbf{s} , generate the Gramian matrix by computing pairwise inner products using our algorithm.³ Then, compute the determinant of the Gramian matrix and compare it to zero.

3.2 The Embedding of Stabilizer Geometry in Hilbert Space

The geometric structure of stabilizer states described in Section 3.1.1 suggests an equally-spaced embedding in the finite-dimensional Hilbert space that can be exploited to study arbitrary quantum states. This is further evidenced by two types of results. First, the uniform distribution over stabilizer states is close to the uniform distribution of arbitrary quantum states in terms of their first two moments [16, 42, 48]. Second, the entanglement of stabilizer states is nearly maximal and similar to that of random states [10, 63].

Consider a linear combination of k stabilizer states $\sum_{i=1}^k \alpha_i |s_i\rangle$ and the task of finding a closest stabilizer state:

$$\arg \max_{|\phi\rangle \in \text{Stab}(\mathcal{H})} \left| \sum_{i=1}^k \alpha_i \langle \phi | s_i \rangle \right| \tag{3.15}$$

³The performance of pairwise computation of inner products between stabilizer states using techniques in Section 4.2 can be improved by pre-computing and storing a basis normalization circuit for each state, rather than computing it from scratch for each pair of states.

where $\text{Stab}(\mathcal{H})$ is the set of stabilizer states in the Hilbert space \mathcal{H} . Here, we can work with any representation of $|\phi\rangle$ that allows us to compute inner products with stabilizer states, e.g., a succinct superposition of stabilizer states. The large count of nearest-neighbor stabilizer states given by Theorem III.14 and the rather uniform structure of stabilizer geometry (Theorem III.18, Corollary III.12) motivate the use of local search to compute Formula 3.15. However, it turns out that greedy local search does not guarantee finding a closest stabilizer state.

3.2.1 Approximating an arbitrary quantum state with one stabilizer state

We analyze a simple *greedy local-search* algorithm that starts at the stabilizer state $|s_m\rangle$ in $|\psi\rangle = \sum_{i=1}^k \alpha_i |s_i\rangle$ with the largest $|\alpha_i|$. At each iteration, this algorithm evaluates the nearest neighbors $N(|s_m\rangle)$ of the current state $|s_m\rangle$ by computing $\max_{|\phi\rangle \in N(|s_m\rangle)} |\langle \phi | \psi \rangle|$, and then moves to a neighbor that satisfies this metric. The algorithm stops at a stabilizer state that is closer to $|\psi\rangle$ than any of its nearest neighbors. As an illustration, consider the state

$$|\psi\rangle = (1 + \varepsilon) |00\rangle + |01\rangle + |10\rangle + |11\rangle \quad (3.16)$$

Given a sufficiently small $\varepsilon > 0$, the unique closest stabilizer state to $|\psi\rangle$ is $|00\rangle + |01\rangle + |10\rangle + |11\rangle$. We start local search at $|s_m\rangle = |00\rangle$ which contributes most to $|\psi\rangle$. By Theorem III.11, all nearest neighbors of $|00\rangle$ have the form $\frac{|00\rangle + J|ab\rangle}{\sqrt{2}}$ where $ab \in \{01, 10, 11\}$ and $J \in \{\pm 1, \pm i\}$. Here we maintain global constants, as they make a difference. We also pick a representative value $ab = 01$ and note that the inner product with $|\psi\rangle$ is maximized by $J = 1$. Let $|r\rangle = (|00\rangle + |01\rangle)/\sqrt{2}$. When $\langle \psi | 00 \rangle = 1 + \varepsilon < \langle \psi | r \rangle = (2 + \varepsilon)/\sqrt{2}$ (up to the same constant), the algorithm sets $|s_m\rangle = |r\rangle$. Nearest neighbors of $|s_m\rangle = \frac{|00\rangle + |01\rangle}{\sqrt{2}}$ are its half-sums with its orthogonal

stabilizer states, hence we arrive at $\frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$. Consider the n -qubit state

$$|\psi_n\rangle = (1 + \varepsilon) |0 \dots 00\rangle + |0 \dots 01\rangle + \dots + |1 \dots 1\rangle \quad (3.17)$$

Its inner products with $|0 \dots 00\rangle$ and the full-superposition state (up to the same constant) are $(1 + \varepsilon)$ and $(2^n + \varepsilon)/2^{n/2}$, respectively. The full-superposition state will be closer as long as $\varepsilon < \frac{2^n - 2^{n/2}}{2^{n/2} - 1} = 2^{n/2}$. If we start local search at $|0 \dots 0\rangle$, it will terminate there when $\varepsilon > \frac{2 - \sqrt{2}}{\sqrt{2} - 1} = \sqrt{2}$, regardless of n . Thus, local search stops at a suboptimal stabilizer state when $\sqrt{2} < \varepsilon < 2^{n/2}$.

Even though constructive approximation by single stabilizer states appears difficult, it is important to know if good approximations exist. As we show next, the answer is negative for the more general case of approximating by stabilizer superpositions.

3.2.2 Approximating arbitrary states with superpositions of stabilizer states

The geometric structure of stabilizer states also motivates the approximation of arbitrary n -qubit unbiased states using superpositions of $\text{poly}(n)$ stabilizer states. In particular, some optimism for obtaining quality approximations using *small* superpositions is motivated by the count of n -qubit stabilizer states from Equation 2.2 — the $\Omega(2^{n^2/2})$ growth rate can be contrasted with the 2^n growth rate of the number of basis states.

To evaluate the quality of approximation by stabilizer states, we employ the quantities

$$\Upsilon = \liminf_{n \rightarrow \infty} \max_{|\psi\rangle} \max_{|s\rangle \in \mathcal{S}_n} \langle s | \psi \rangle \quad \text{and} \quad \Upsilon^{\text{poly}} = \sup_{\text{poly } p(n)} \liminf_{n \rightarrow \infty} \sup_{|\psi\rangle} \max_{|s\rangle \in \mathcal{S}_n^{p(n)}} \langle s | \psi \rangle \quad (3.18)$$

where $\mathcal{S}_n^{p(n)}$ contains superpositions of up to $p(n)$ states from the set of all \mathcal{S}_n for a

given polynomial $p(n)$. For illustration, replace \mathcal{S}_n with a set \mathcal{B}_n of 2^n orthogonal basis states. The state $|\psi\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |k\rangle$ minimizes $\max_{s \in \mathcal{B}_n} \langle s|\psi\rangle$ with the value $\frac{1}{2^{n/2}}$. Because this value approaches zero at an exponential rate, taking polynomial-sized superpositions of basis states (rather than single basis states) will still produce the zero limit. Perhaps, this result is not surprising given that $\max_{|s_1\rangle \neq |s_2\rangle \in \mathcal{B}_n} \langle s_1|s_2\rangle = 0$. However, since $\max_{|s_1\rangle \neq |s_2\rangle \in \mathcal{S}_n} \langle s_1|s_2\rangle = 1/\sqrt{2}$, and each stabilizer state $|s_1\rangle \in \mathcal{S}_n$ has $2^{n+2} - 4$ nearest neighbors $|s_2\rangle \in \mathcal{S}_n$ such that $\langle s_1|s_2\rangle = 1/\sqrt{2}$, one might hope that $\Upsilon = 1/\sqrt{2}$.

Lemma III.25. *Given an n -qubit stabilizer state $|s\rangle$ and two qubits q and r , supports and norms of all non-zero double cofactors $|s_{qr}\rangle$ are equal. In particular, since all supports are powers of two, the number of double cofactors with non-zero support cannot be three.*

Proof. The claim is trivial when only one double cofactor is non-zero. When only two are non-zero, they may originate from one single-qubit cofactor or from two different single-qubit cofactors. By Corollary II.21-iv, the supports and norms of cofactors must be equal. When all four double cofactors are non-zero, Corollary II.21-iv implies that their support must be one fourth of that of the initial state. Additionally, the orthogonality of cofactors implies that their norm must be one half of the original norm. Now we show that the case of exactly three non-zero double cofactors is impossible. Without the loss of generality, assume that $|\psi_{qr=11}\rangle = 0$, but other cofactors are $\neq 0$. Then single cofactors must have equal support. Therefore $|\psi_{r=1}| = |\psi_{qr=01}| = |\psi_{qr=00}| + |\psi_{qr=10}| = |\psi_{r=0}| = 2|\psi_{qr=10}|$, but also $|\psi_{q=1}| = |\psi_{qr=10}| = |\psi_{qr=00}| + |\psi_{qr=01}| = |\psi_{q=0}| = 2|\psi_{qr=01}|$. Thus $|\psi_{qr=01}| = 4|\psi_{qr=01}|$, which contradicts $|\psi_{qr=01}| \neq 0$. \square

Lemma III.25 is illustrated in Table 2.4 and Appendix A.

Theorem III.26. $\Upsilon = \Upsilon^{poly} = 0$.

Proof. Consider the family of $2n$ -qubit unbiased states $\mu^{2n} = \left(\frac{|00\rangle+|01\rangle+|10\rangle}{\sqrt{3}}\right)^{\otimes n}$, which are not stabilizer states per Lemma III.25. For an arbitrary stabilizer state $|s\rangle$, double cofactoring over $q = 2n - 1$ and $r = 2n - 2$ yields

$$\langle s|\mu^{2n}\rangle = \langle s_{qr=00}|\mu_{qr=00}^{2n}\rangle + \langle s_{qr=01}|\mu_{qr=01}^{2n}\rangle + \langle s_{qr=10}|\mu_{qr=10}^{2n}\rangle \quad (3.19)$$

because $|\mu_{qr=11}^{2n}\rangle = 0$. We can upper-bound this expression by over-estimating each non-zero term with $\langle v|w\rangle \leq \|v\| \cdot \|w\|$. To this end, $\|\mu_{qr}^{2n}\| = \frac{1}{\sqrt{3}}$, while the norm of (orthogonal) double cofactors of $|s\rangle$ depends on how many of them are non-zeros — one, two or four. In these cases, the norms are 1 , $\frac{\sqrt{2}}{2}$ and $\frac{1}{2}$, respectively, yielding upper bounds $\langle s|\mu^{2n}\rangle \leq \alpha = \frac{\sqrt{3}}{3}$, $\frac{\sqrt{6}}{3}$ and $\frac{\sqrt{3}}{2}$. Therefore, cumulatively, $\langle s|\mu^{2n}\rangle \leq \frac{\sqrt{3}}{2}$.

More accurate bounds can be obtained by rewriting the same three terms using

$$\langle v|w\rangle = \|v\| \cdot \|w\| \frac{\langle v|w\rangle}{\|v\| \cdot \|w\|},$$

then observing that $\langle v00|w00\rangle = \langle v|w\rangle$, $\langle v01|w01\rangle = \langle v|w\rangle$, and $\langle v10|w10\rangle = \langle v|w\rangle$. In particular, $\sqrt{3}|\mu_{qr=00}^{2n}\rangle$, $\sqrt{3}|\mu_{qr=01}^{2n}\rangle$ and $\sqrt{3}|\mu_{qr=10}^{2n}\rangle$ can be replaced by $|\mu^{2n-2}\rangle$, noting that $|\mu^{2n}\rangle$ is separable. Then the cofactors of $|s\rangle$ can be relaxed to a best-case $(2n - 2)$ -qubit stabilizer state s' to obtain

$$\langle s|\mu^{2n}\rangle \leq \frac{\sqrt{3}}{2} \langle s'|\mu^{2n-2}\rangle \leq \left(\frac{\sqrt{3}}{2}\right)^n \quad (3.20)$$

Therefore

$$\Upsilon = \lim_{n \rightarrow \infty} \inf_{|\psi\rangle} \max_{|s\rangle \in \mathcal{S}_{2n}} \langle s|\psi\rangle = \lim_{n \rightarrow \infty} \left(\frac{\sqrt{3}}{2}\right)^n = 0 \quad (3.21)$$

For any polynomial $p(n)$,

$$\lim_{n \rightarrow \infty} \inf_{|\psi\rangle} \sup_{|s\rangle \in \mathcal{S}_{2n}^{p(n)}} \langle s|\psi\rangle = \lim_{n \rightarrow \infty} p(n) \left(\frac{\sqrt{3}}{2}\right)^n = 0 \quad (3.22)$$

Therefore $\Upsilon^{poly} = 0$ as well. □

Theorem III.26 is somewhat surprising because no n -qubit state can be orthogonal to the set of all stabilizer states, which contains many basis sets. However, they establish *asymptotic orthogonality*, which can be viewed as an infinite-dimensional phenomenon. Such families of states that are asymptotically orthogonal to all stabilizer states can be neither represented nor approximated by polynomial-sized superpositions of stabilizer states.

Our construction of *stabilizer-evading* states can be modified to yield other *separable* states with similar properties, e.g., $\bar{\mu}^{3n} = \left(\frac{|001\rangle + |010\rangle + |100\rangle}{\sqrt{3}} \right)^{\otimes n}$. Superpositions such as $(\mu^{6n} + \bar{\mu}^{6n})/\sqrt{2}$ offer *entangled* states that cannot be approximated by polynomial-sized superpositions of stabilizer states.

The above results along with the results from Section 3.2.1 suggest that there are wide gaps between stabilizer states. The following proposition quantifies the size of such gaps.

Proposition III.27. *Consider 2^n -dimensional balls centered at a point on the unit sphere that maximizes the number of n -qubit non-stabilizer states in their interior but do not contain any stabilizer states. The radius of such balls cannot exceed $\sqrt{2}$, but approaches $\sqrt{2}$ as $n \rightarrow \infty$.*

Proof. Consider an arbitrary ball \mathbf{B} with radius $\sqrt{2}$ and centered on the unit sphere as shown in Figure 3.3. \mathbf{B} covers half of the unit sphere. Let $|s\rangle$ be a stabilizer state that does not lie on the boundary of \mathbf{B} . Then, either $|s\rangle$ or $-|s\rangle$ is inside \mathbf{B} . Furthermore, observe that the intersection of the unit sphere and the boundary of \mathbf{B} is contained in a hyperplane of dimension $n - 1$. If all stabilizer states were contained there, they would not have included a single basis set. However, since stabilizer states contain the computational basis, they cannot all be contained in that intersection. An asymptotic lower bound for the radius of \mathbf{B} is obtained using the family of *stabilizer-*

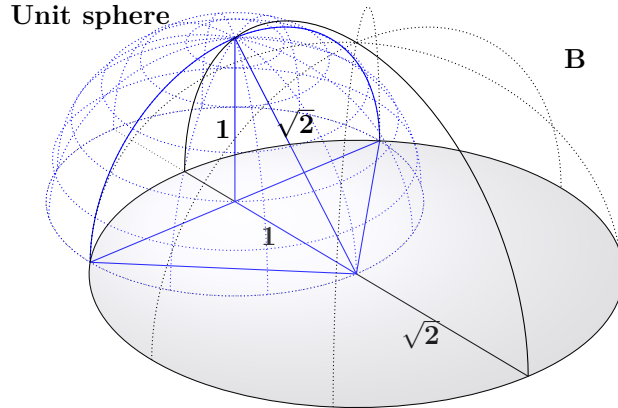


Figure 3.3: A ball \mathbf{B} with radius $\sqrt{2}$ centered on the unit sphere covers half of the unit sphere. Every such ball contains at least one stabilizer state (in most cases, half of all stabilizer states).

evading states defined in Theorem III.26. These states are asymptotically orthogonal to all n -qubit stabilizer states ($n \rightarrow \infty$). Therefore, the distance to their closest stabilizer state approaches $\sqrt{2}$. \square

3.3 Summary

The stabilizer formalism facilitates compact representation of stabilizer states and efficient simulation of stabilizer circuits [2, 33, 34]. Stabilizer states arise in applications of quantum information processing, and their efficient manipulation via geometric and linear-algebraic operations may lead to additional insights in quantum entanglement, quantum error correction and quantum-circuit simulation. Furthermore, stabilizer states are closely related to valence-bond states [65], cluster/graph states [5, 19, 37], and measurement-based/one-way quantum computation [57]. The emphasis of our work on the full set of (pure) n -qubit stabilizer states is justified by the desire to succinctly represent as many quantum states as possible [1]. This is in contrast to identifying structured representations of more specialized quantum states.

In this chapter, we characterize the nearest-neighbor structure of stabilizer states and quantify the distribution of angles between pairs of stabilizer states. Specifically,

we derive and prove combinatorial formulas for computing the number of k -neighbor stabilizer states. We use our formulas to characterize the distribution of real-valued inner products between any stabilizer state and its neighboring stabilizer states. We prove that each n -qubit stabilizer state $|\psi\rangle$ has exactly $4(2^n - 1)$ nearest-neighbor stabilizer states and that, as a fraction of all stabilizer states, the number of k -neighbors of $|\psi\rangle$ for all $k = 0, 1, \dots, n - 4$ cumulatively approaches zero as $n \rightarrow \infty$. Therefore, almost all stabilizer states are either orthogonal or nearly orthogonal to $|\psi\rangle$ as $n \rightarrow \infty$.

We introduce the notion of a stabilizer bivector $|\varphi\rangle$, which are $2n$ -qubit stabilizer states such that $|\varphi\rangle \propto |\psi \wedge \phi\rangle$, where $|\psi\rangle$ and $|\phi\rangle$ are n -qubit stabilizer states. We show that, for any stabilizer state $|\psi\rangle$, there are $5(2^n - 1)$ distinct stabilizer states $|\phi\rangle$ such that their wedge product is a stabilizer bivector. The stabilizer bivector obtained from the simple wedge product of k -neighbor states forms a parallelogram with area equal to $\sqrt{1 - 2^{-k}}$.

Although the geometric structure of stabilizer states is fairly uniform [16, 42, 48], we show that local search is not guaranteed to find the closest stabilizer state to an arbitrary quantum state. Furthermore, we define a family of unbiased states that cannot be approximated by polynomial-sized superpositions of stabilizer states, and prove that the maximal radius of any 2^n -dimensional ball centered at a point on the unit sphere that does not contain any n -qubit stabilizer states cannot exceed $\sqrt{2}$, but approaches $\sqrt{2}$ as $n \rightarrow \infty$.

CHAPTER IV

Computational Geometry of Stabilizer States

Section 3.2 illustrates how straightforward approaches to several natural geometric tasks fail despite the fairly uniform geometric structure of stabilizer states. In this chapter, we focus on more specialized, but no less useful tasks and develop efficient computation of distances, angles and volumes between stabilizer states. In Section 4.1, we discuss our algorithm for synthesizing new *canonical stabilizer circuits*. In Section 4.2, we turn our attention to the computation of inner products of stabilizer states and, by extension, their linear combinations. As Gram-Schmidt orthogonalization cannot be used directly with stabilizer states, we develop an alternative approach in Section 4.3. The efficient computation of generator sets for stabilizer bivectors is discussed in Section 4.4. We present an empirical evaluation of our circuit-synthesis and inner-product algorithms in Section 4.6

4.1 Synthesis of Canonical Stabilizer Circuits

A crucial step in the inner-product computation for stabilizer states (Theorem III.2) is the synthesis of a stabilizer circuit that brings an n -qubit stabilizer state $|\psi\rangle$ to a computational-basis state $|b\rangle$, which we represent by the following stabilizer matrix structure.

Definition IV.1. A stabilizer matrix is in *basis form* if it has the following structure.

$$\begin{array}{c} \pm \\ \pm \\ \vdots \\ \pm \end{array} \begin{bmatrix} Z & I & \cdots & I \\ I & Z & \cdots & I \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & Z \end{bmatrix}$$

Consider a stabilizer matrix \mathcal{M} that uniquely identifies $|\psi\rangle$. \mathcal{M} is reduced to basis form (Definition IV.1) by applying a series of elementary row and column operations. Recall that row operations (transposition and multiplication) do not modify the state, but column (Clifford) operations do. Thus, the column operations involved in the reduction process constitute a unitary stabilizer circuit \mathbf{C} such that $\mathbf{C}|\psi\rangle = |b\rangle$, where $|b\rangle$ is a basis state. Algorithm 4.1.1 reduces an input matrix \mathcal{M} to basis form and returns such a circuit \mathbf{C} .

Canonical forms are useful for synthesizing stabilizer circuits that minimize the number of gates and qubits required to produce a particular computation. This is particularly important in the context of quantum fault-tolerant architectures that are based on stabilizer codes. The work in [2] establishes a 7-block¹ canonical-circuit template with the sequence $H-C-P-C-P-C-H$. Furthermore, the work in [14] proves the existence of canonical circuits with the shorter sequence $H-C-X-P-CZ$, where the X and CZ blocks consist of NOT and Controlled- Z (CPHASE in Figure 2.2a) gates, respectively. However, the synthesis algorithm sketched in [14] employs the \mathbb{Z}_2 -representation for states (Theorem II.19) rather than the stabilizer formalism. Thus, no detailed algorithms are known for obtaining such canonical circuits *given an arbitrary generator set*. Algorithm 4.1.1 takes as input a stabilizer matrix \mathcal{M} and synthesizes a 5-block canonical circuit with template $H-C-CZ-P-H$ (Figure 4.1). We now describe the main steps in the algorithm. The updates to the phase vector under row/column operations are left out of the discussion as such updates do not affect the overall execution of the algorithm.

¹Theorem 8 in [2] actually describes an 11-step procedure to obtain such circuits. However, the last four steps are used to reduce destabilizer rows, which we do not consider here.

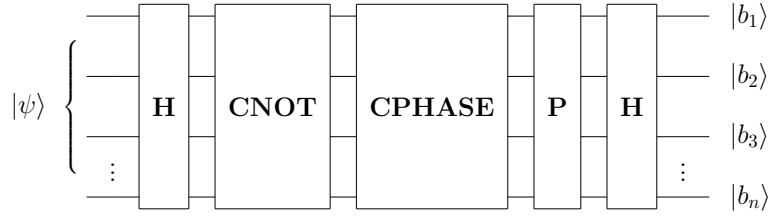


Figure 4.1: Template structure for the basis-normalization circuit synthesized by Algorithm 4.1.1. The input is an arbitrary stabilizer state $|\psi\rangle$ while the output is a basis state $|b_1 b_2 \dots b_n \in \{0, 1\}^n\rangle$.

1. Reduce \mathcal{M} to canonical form.
2. Use row transposition to diagonalize \mathcal{M} . For $j \in \{1, \dots, n\}$, if the diagonal literal $\mathcal{M}_{j,j} = Z$ and there are other Pauli (non- I) literals in the row (qubit is entangled), conjugate \mathcal{M} by H_j . Elements below the diagonal are Z/I literals.
3. For each above-diagonal element $\mathcal{M}_{j,k} = X/Y$, conjugate by $\text{CNOT}_{j,k}$. Elements above the diagonal are now I/Z literals.
4. For each above-diagonal element $\mathcal{M}_{j,k} = Z$, conjugate by $\text{CPHASE}_{j,k}$. Elements above the diagonal are now I literals.
5. For each diagonal literal $\mathcal{M}_{j,j} = Y$, conjugate by P_j .
6. For each diagonal literal $\mathcal{M}_{j,j} = X$, conjugate by H_j .

Proposition IV.2. For an $n \times n$ stabilizer matrix \mathcal{M} , the number of gates in the circuit \mathbf{C} returned by Algorithm 4.1.1 is $O(n^2)$.

Proof. The number of gates in \mathbf{C} is dominated by the CPHASE block, which consists of $O(n^2)$ gates. This agrees with previous results regarding the number of gates needed for an n -qubit stabilizer circuit in the worst case [9, 13]. \square

Algorithm 4.1.1 Synthesis of a basis-normalization circuit

Require: Stabilizer matrix \mathcal{M} for $S(|\psi\rangle)$ with rows R_1, \dots, R_n

Ensure: (i) Unitary stabilizer circuit \mathbf{C} such that $\mathbf{C}|\psi\rangle$ equals basis state $|b\rangle$, and (ii) reduce \mathcal{M} to basis form

- \Rightarrow GAUSS(\mathcal{M}) reduces \mathcal{M} to canonical form (Figure 2.4)
- \Rightarrow ROWSWAP(\mathcal{M}, i, j) swaps rows R_i and R_j of \mathcal{M}
- \Rightarrow ROWMULT(\mathcal{M}, i, j) left-multiplies rows R_i and R_j , returns updated R_i
- \Rightarrow CONJ(\mathcal{M}, α_j) conjugates j^{th} column of \mathcal{M} by Clifford sequence α

```
1: GAUSS( $\mathcal{M}$ ) ▷ Set  $\mathcal{M}$  to canonical form
2:  $\mathbf{C} \leftarrow \emptyset$ 
3: for  $j \in \{1, \dots, n\}$  do ▷ Apply block of Hadamard gates
4:    $k \leftarrow$  index of row  $R_{k \in \{j, \dots, n\}}$  with  $j^{\text{th}}$  literal set to  $X$  or  $Y$ 
5:   if  $k$  exists then
6:     ROWSWAP( $\mathcal{M}, j, k$ )
7:   else
8:      $k_2 \leftarrow$  index of last row  $R_{k_2 \in \{j, \dots, n\}}$  with  $j^{\text{th}}$  literal set to  $Z$ 
9:     if  $k_2$  exists then
10:      ROWSWAP( $\mathcal{M}, j, k_2$ )
11:      if  $R_j$  has  $X, Y$  or  $Z$  literals in columns  $\{j + 1, \dots, n\}$  then
12:        CONJ( $\mathcal{M}, H_j$ )
13:         $\mathbf{C} \leftarrow \mathbf{C} \cup H_j$ 
14:      end if
15:    end if
16:  end if
17: end for
18: for  $j \in \{1, \dots, n\}$  do ▷ Apply block of CNOT gates
19:   for  $k \in \{j + 1, \dots, n\}$  do
20:     if  $k^{\text{th}}$  literal of row  $R_j$  is set to  $X$  or  $Y$  then
21:       CONJ( $\mathcal{M}, \text{CNOT}_{j,k}$ )
22:        $\mathbf{C} \leftarrow \mathbf{C} \cup \text{CNOT}_{j,k}$ 
23:     end if
24:   end for
25: end for
26: for  $j \in \{1, \dots, n\}$  do ▷ Apply a block of Controlled- $Z$  gates
27:   for  $k \in \{j + 1, \dots, n\}$  do
28:     if  $k^{\text{th}}$  literal of row  $R_j$  is set to  $Z$  then
29:       CONJ( $\mathcal{M}, \text{CPHASE}_{j,k}$ )
30:        $\mathbf{C} \leftarrow \mathbf{C} \cup \text{CPHASE}_{j,k}$ 
31:     end if
32:   end for
33: end for
34: for  $j \in \{1, \dots, n\}$  do ▷ Apply block of Phase gates
35:   if  $j^{\text{th}}$  literal of row  $R_j$  is set to  $Y$  then
36:     CONJ( $\mathcal{M}, P_j$ )
37:      $\mathbf{C} \leftarrow \mathbf{C} \cup P_j$ 
38:   end if
39: end for
40: for  $j \in \{1, \dots, n\}$  do ▷ Apply block of Hadamard gates
41:   if  $j^{\text{th}}$  literal of row  $R_j$  is set to  $X$  then
42:     CONJ( $\mathcal{M}, H_j$ )
43:      $\mathbf{C} \leftarrow \mathbf{C} \cup H_j$ 
44:   end if
45: end for
46: return  $\mathbf{C}$ 
```

Observe that, for each gate added to \mathbf{C} , the corresponding column operation is applied to \mathcal{M} . Since column operations run in $\Theta(n)$ time, it follows from Proposition IV.2 that the runtime of Algorithm 4.1.1 is $O(n^3)$. Audenaert and Plenio [6] described an algorithm to compute the fidelity between two mixed stabilizer states. Similar to our use of basis-normalizing circuits, the approach from [6] relies on a stabilizer circuit to map a stabilizer state to a normal form where all basis states have non-zero amplitudes. However, the circuits generated by the Audenaert-Plenio algorithm exhibit two disadvantages: (i) they are not canonical and (ii) they have, on average, twice as many gates as our basis-normalization circuits (Section 4.6).

We note that our canonical stabilizer circuits can be optimized using the techniques from [49], which describes how to restructure circuits to facilitate parallel quantum computation. The authors show that such parallelization is possible for circuits consisting of H and CNOT gates, and for diagonal operators. Since CPHASE gates are diagonal, one can apply the techniques from [49] to parallelize our canonical circuits and produce equivalent circuits with $O(n^2)$ gates and parallel depth $O(\log n)$. Canonical stabilizer circuits that follow the 7-block template structure from [2] can be optimized to obtain a tighter bound on the number of gates. As in our approach, such circuits are dominated by the size of the CNOT blocks, which contain $O(n^2)$ gates. The work in [54] shows that any CNOT circuit has an equivalent CNOT circuit with $O(n^2/\log n)$ gates. Thus, one simply applies such techniques to each of the CNOT blocks in the canonical circuit. It is an open problem whether one can apply some variation of the same techniques to CPHASE blocks, which would facilitate similar optimizations for our 5-block canonical form.

4.2 An Inner-product Algorithm

Let $|\psi\rangle$ and $|\phi\rangle$ be two stabilizer states represented by stabilizer matrices \mathcal{M}^ψ and \mathcal{M}^ϕ , respectively. Our approach for computing the inner product between these

two states is shown in Algorithm 4.2.1. Following the proof of Theorem III.2, Algorithm 4.1.1 is applied to \mathcal{M}^ψ in order to reduce it to basis form. The stabilizer circuit generated by Algorithm 4.1.1 is then applied to \mathcal{M}^ϕ in order to preserve the inner product. Then, we minimize the number of X and Y literals in \mathcal{M}^ϕ by applying Algorithm 2.2.1. Lastly, each generator in \mathcal{M}^ϕ that anticommutes with \mathcal{M}^ψ (since \mathcal{M}^ψ is in basis form, we only need to check which generators in \mathcal{M}^ϕ have X or Y literals) contributes a factor of $1/\sqrt{2}$ to the inner product. If a generator in \mathcal{M}^ϕ , say Q_i , commutes with \mathcal{M}^ψ , then we check orthogonality by determining whether Q_i is in the stabilizer group generated by \mathcal{M}^ψ . This is accomplished by multiplying the appropriate generators in \mathcal{M}^ψ such that we create Pauli operator R , which has the same literals as Q_i , and check whether R has an opposite sign to Q_i . If this is the case, then, by Theorem III.1, the states are orthogonal. The bottleneck of Algorithm 4.2.1 is the call to Algorithm 4.1.1, and the overall runtime is $O(n^3)$. As Section 4.6 shows, the performance of our algorithm is sensitive to the input stabilizer matrix and takes $O(n^2)$ time in important cases.

Complex-valued inner product. Recall that Algorithm 4.2.1 computes the *real-valued* (phase-normalized) inner product $r = e^{-i\theta} \langle \psi | \phi \rangle$. To compute the *complex-valued* inner product, one needs to additionally calculate $e^{-i\theta}$. This is accomplished by modifying Algorithm 4.2.1 such that it maintains the global phases generated when computing $\mathbf{C} |\psi\rangle$ and $\mathbf{C} |\phi\rangle$, where \mathbf{C} is the basis-normalization circuit from Algorithm 4.1.1. Let α and β be the global phases of $\mathbf{C} |\psi\rangle$ and $\mathbf{C} |\phi\rangle$, respectively. Both phases can be computed using the process outlined in Section 2.3.3. The complex-valued inner product is then calculated as $\alpha^* \cdot \beta \cdot 2^{-k/2}$.

Algorithm 4.2.1 Inner product for stabilizer states

Require: Stabilizer matrices (i) \mathcal{M}^ψ for $|\psi\rangle$ with rows P_1, \dots, P_n , and (ii) \mathcal{M}^ϕ for $|\phi\rangle$ with rows Q_1, \dots, Q_n

Ensure: Inner product between $|\psi\rangle$ and $|\phi\rangle$

\Rightarrow **BASISNORMCIRC**(\mathcal{M}^ψ) reduces \mathcal{M} to basis form, i.e. $\mathbf{C}|\psi\rangle = |b\rangle$, where $|b\rangle$ is a basis state, and returns \mathbf{C}

\Rightarrow **CONJ**(\mathcal{M}, \mathbf{C}) conjugates \mathcal{M} by Clifford circuit \mathbf{C}

\Rightarrow **GAUSS**(\mathcal{M}) reduces \mathcal{M} to canonical form (Figure 2.4)

\Rightarrow **LEFTMULT**(P, Q) left-multiplies Pauli operators P and Q , and returns the updated Q

```
1:  $\mathbf{C} \leftarrow \text{BASISNORMCIRC}(\mathcal{M}^\psi)$  ▷ Apply Algorithm 4.1.1 to  $\mathcal{M}^\psi$ 
2: CONJ( $\mathcal{M}^\phi, \mathbf{C}$ ) ▷ Compute  $\mathbf{C}|\phi\rangle$ 
3: GAUSS( $\mathcal{M}^\phi$ ) ▷ Set  $\mathcal{M}^\phi$  to canonical form
4:  $k \leftarrow 0$ 
5: for each row  $Q_i \in \mathcal{M}^\phi$  do
6:   if  $Q_i$  contains  $X$  or  $Y$  literals then
7:      $k \leftarrow k + 1$ 
8:   else ▷ Check orthogonality, i.e.,  $Q_i \notin S(|b\rangle)$ 
9:      $R \leftarrow I^{\otimes n}$ 
10:    for each  $Z$  literal in  $Q_i$  found at position  $j$  do
11:       $R \leftarrow \text{LEFTMULT}(P_j, R)$ 
12:    end for
13:    if  $R = -Q_i$  then
14:      return 0 ▷ By Theorem III.1
15:    end if
16:  end if
17: end for
18: return  $2^{-k/2}$  ▷ By Theorem III.2
```

4.3 Orthogonalization of Stabilizer States

We now shift our focus to the task of orthogonalizing a linear combination of stabilizer states $|\Psi\rangle = \sum_{j=1}^N c_j |\psi_j\rangle$, where each $|\psi_j\rangle$ is represented by its own stabilizer matrix. To simulate measurements of $|\Psi\rangle$, it is helpful to transform the set of states that define $|\Psi\rangle$ into an orthogonal set. Since a linear combination of stabilizer states is usually not a stabilizer state, Gram-Schmidt orthogonalization cannot be used directly. Therefore, we develop an orthogonalization procedure that exploits the nearest-neighbor structure of stabilizer states (Section 3.1.1) and their efficient manipulation via stabilizers.

Proposition IV.3. *Let $|\psi\rangle$ be a state represented by \mathcal{M}^ψ , which contains at least one row with at least one X or Y literal. Then $|\psi\rangle$ can be decomposed into a superposition*

$\frac{|\phi\rangle+i^l|\varphi\rangle}{\sqrt{2}}$, where $|\phi\rangle$ and $|\varphi\rangle$ are nearest neighbors of $|\psi\rangle$ whose matrices are similar to each other.

Proof. Let R_j be a row in \mathcal{M}^ψ with an X/Y literal in its j^{th} position, and let Z_j be a Pauli operator with a Z literal in its j^{th} position and I everywhere else. Observe that R_j and Z_j anticommute. If any other rows in \mathcal{M}^ψ anticommute with Z_j , multiply them by R_j to make them commute with Z_j . Let \mathcal{M}^ϕ and \mathcal{M}^φ be the matrices obtained by replacing row R_j in \mathcal{M}^ψ with Z_j and $-Z_j$, respectively. This operation is equivalent to applying $(\pm Z_j)$ -measurement projectors to $|\psi\rangle$ (Section 2.3). Thus, $|\phi\rangle \equiv \frac{(I+Z_j)|\psi\rangle}{\sqrt{2}}$ and $|\varphi\rangle \equiv \frac{(I-Z_j)|\psi\rangle}{\sqrt{2}}$, such that $|\langle\psi|\phi\rangle| = |\langle\psi|\varphi\rangle| = 1/\sqrt{2}$. The matrices \mathcal{M}^ϕ and \mathcal{M}^φ are similar since $\mathcal{M}^\varphi = X_j(\mathcal{M}^\phi)X_j^\dagger$ (and $\mathcal{M}^\phi = X_j(\mathcal{M}^\varphi)X_j^\dagger$), where X_j is a Pauli operator with an X literal in its j^{th} position and I everywhere else. This implies that the j^{th} qubit in $|\phi\rangle$ and $|\varphi\rangle$ is in the deterministic state $|0\rangle$ and $|1\rangle$, respectively. ($|\phi\rangle$ and $|\varphi\rangle$ are cofactors of $|\psi\rangle$ with respect to qubit j .) To produce $\frac{|\phi\rangle+i^l|\varphi\rangle}{\sqrt{2}}$, we need the i^l factor, which is not maintained by the stabilizer \mathcal{M}^φ . This factor can be obtained by the same global-phase computation procedure described in Section 4.2 in the context of complex-valued inner products. \square

Algorithm 4.3.1 takes as input a linear combination of n -qubit states $|\Psi\rangle$ represented by a pair consisting of: (i) a list of *canonical stabilizer matrices* $\mathbf{M} = \{\mathcal{M}^1, \dots, \mathcal{M}^N\}$ and (ii) a list of coefficients $\mathbf{c} = \{c_1, \dots, c_N\}$. The algorithm iteratively applies the decomposition procedure described in the proof of Proposition IV.3 until all the matrices in \mathbf{M} are *similar* (Definition II.24) to each other. At each iteration, the algorithm selects a pivot qubit based on the composition of Pauli literals in the corresponding column. The states in the linear combination are decomposed with respect to the pivot only if there exists a pair of matrices in \mathbf{M} that contain different types of Pauli literals in the pivot column. Since similar matrices contain distinct phase vectors, the states represented by the modified list of matrices are mutually orthogonal (Theorem III.1). The algorithm maintains the invariant that the

sum of stabilizer states (represented by the pair \mathbf{M} and \mathbf{c}) equals the (non-stabilizer) vector $|\Psi\rangle$. Observe that, for each pivot qubit that satisfies $b = 1$, the size of \mathbf{M} doubles. Let m be the maximum value in $\{m_1, \dots, m_N\}$, where m_j is the number of rows with X/Y literals in matrix \mathcal{M}^j . An orthogonalization approach that obtains $|\Psi\rangle$ in the computational basis (i.e., calculating computational-basis amplitudes of each $|\psi_j\rangle$ and summing them with weights c_j) requires exactly 2^m terms. In contrast, Algorithm 4.3.1 expands \mathbf{M} to 2^k terms, where $k \leq m$. In particular, if the matrices in \mathbf{M} are “close” to similar (Definition II.24), then $k < m$ and thus Algorithm 4.3.1 provides an advantage over direct computation of basis amplitudes.

Example IV.4. Let $\mathbf{M} = \{\mathcal{M}^1, \mathcal{M}^2\}$, where \mathcal{M}^1 and \mathcal{M}^2 are n -qubit matrices. Suppose \mathcal{M}^1 has X literals along its diagonal and I literals everywhere else. Similarly, assume \mathcal{M}^2 follows the same diagonal structure, but replaces an X at diagonal position p with Y . Algorithm 4.3.1 decomposes both matrices over the selected pivot qubit p using Proposition IV.3, which yields a list of two similar matrices. In contrast, calculating the basis amplitudes of the states represented by \mathcal{M}^1 and \mathcal{M}^2 would require summing over 2^n terms.

4.4 Computation of Stabilizer Bivectors

In Section 3.1.2, we defined the notion of a *stabilizer bivector*, which can be used to represent antisymmetric basis states [23] compactly on conventional computers. We now describe an algorithm to efficiently compute a generator set for stabilizer bivectors.

Let $|\psi\rangle$ and $|\phi\rangle$ be two stabilizer states represented by matrices \mathcal{M}^ψ and \mathcal{M}^ϕ , respectively. Algorithm 4.4.1 shows our approach to computing the stabilizer bivector $|\psi \wedge \phi\rangle$. First, we compute $\langle \psi | \phi \rangle$ and obtain the canonical matrices for \mathcal{M}^ψ and \mathcal{M}^ϕ . This allows us to determine whether the conditions established by the proof

Algorithm 4.3.1 Orthogonalization procedure for linear combinations of stabilizer states

Require: Linear combination of n -qubit states $|\Psi\rangle = \sum_{j=1}^N c_j |\psi_j\rangle$ represented by (i) a list of canonical stabilizer matrices $\mathbf{M} = \{\mathcal{M}^1, \dots, \mathcal{M}^N\}$ and (ii) a list of coefficients $\mathbf{c} = \{c_1, \dots, c_N\}$

Ensure: Modified lists \mathbf{M}' and \mathbf{c}' representing a linear combination of *mutually orthogonal* states
 \Rightarrow PAULI(\mathcal{M}, j) returns 0 if j^{th} column in \mathcal{M} has Z literals only (ignores I literals), 1 if it has X literals only, 2 if it has Y literals only, 3 if it has X/Z literals only, 4 if it has Y/Z literals only
 \Rightarrow REMOVE($\mathbf{M}, \mathbf{c}, j$) removes the j^{th} element in \mathbf{M} and \mathbf{c}
 \Rightarrow INSERT($\mathbf{M}, \mathbf{c}, \mathcal{M}, c$) appends \mathcal{M} to \mathbf{M} and c to \mathbf{c} if an *equivalent matrix* does not exist in \mathbf{M} ; otherwise, sets $c_j = c_j + c$, where c_j is the coefficient of the matrix in \mathbf{M} equivalent to \mathcal{M}
 \Rightarrow DECOMPOSE($\mathcal{M}, j, a \in \{0, 1\}$) implements the proof of Proposition IV.3 and returns the pair $[\mathcal{M}_{j=a}, \alpha]$, where $\mathcal{M}_{j=a}$ is the nearest-neighbor canonical matrix with the j^{th} qubit in state $|a\rangle$, and α is the phase factor

```

1: for  $j \in \{1, \dots, n\}$  do
2:    $b \leftarrow 0$ 
3:    $l \leftarrow \text{PAULI}(\mathcal{M}^1, j)$ 
4:   for each  $\mathcal{M}^i, i \in \{2, \dots, N\}$  do
5:      $k \leftarrow \text{PAULI}(\mathcal{M}^i, j)$ 
6:     if  $k \neq l$  or the rows with Pauli literals in the  $j^{\text{th}}$  column are distinct then
7:        $b \leftarrow 1$ 
8:       break
9:     end if
10:  end for
11:  if  $b = 1$  then
12:    for each  $\mathcal{M}^i, i \in \{1, \dots, N\}$  do
13:      if PAULI( $\mathcal{M}^i, j$ )  $\neq 0$  then
14:         $[\mathcal{M}_{j=0}^i, \alpha] \leftarrow \text{DECOMPOSE}(\mathcal{M}^i, j, 0)$ 
15:         $[\mathcal{M}_{j=1}^i, \beta] \leftarrow \text{DECOMPOSE}(\mathcal{M}^i, j, 1)$ 
16:        REMOVE( $\mathbf{M}, \mathbf{c}, i$ )
17:        INSERT( $\mathbf{M}, \mathbf{c}, \mathcal{M}_{j=0}^i, \alpha/\sqrt{2}$ )
18:        INSERT( $\mathbf{M}, \mathbf{c}, \mathcal{M}_{j=1}^i, \beta/\sqrt{2}$ )
19:      end if
20:    end for
21:  end if
22: end for

```

of Theorem III.21 are satisfied. In the case that $|\psi\rangle$ and $|\phi\rangle$ are 1-neighbors, we modify the *dissimilar* matrices as per the proof of Theorem III.21. This ensures that the input matrices are *similar* before computing the matrices for the tensor products $|\psi \otimes \phi\rangle$ and $|\phi \otimes \psi\rangle$.

Example IV.5. Suppose $\mathcal{M}^\psi = \{ZI, IZ\}$ and $\mathcal{M}^\phi = \{XI, IZ\}$, which represent a nearest-neighbor pair. Following Equation 3.13, we replace \mathcal{M}^ϕ with $X_1 \mathcal{M}^\psi X_1^\dagger = \{-ZI, IZ\}$ (or \mathcal{M}^ψ with $X_1 \mathcal{M}^\phi X_1^\dagger = \{-XI, IZ\}$) to make the input matrices similar.

The last step is to compute the difference $|\psi \otimes \phi\rangle - |\phi \otimes \psi\rangle$ as follows:

1. Obtain the basis-normalization circuit \mathbf{C} for both $\mathcal{M}^{\psi \otimes \phi}$ and $\mathcal{M}^{\phi \otimes \psi}$. Observe that its the same circuit since the matrices are similar.
2. Conjugate both matrices by \mathbf{C} to obtain the matrices for computational-basis states $|b_1\rangle$ and $|b_2\rangle$.
3. Following the proof of Lemma III.9, obtain the matrix $\mathcal{M}^{b_1-b_2}$ for the sum $|b_1\rangle - |b_2\rangle$.
4. Conjugate $\mathcal{M}^{b_1-b_2}$ by \mathbf{C}^\dagger to obtain the matrix $\mathcal{M}^{\psi \wedge \phi}$.

Proposition IV.6. *Let $|\psi\rangle$ and $|\phi\rangle$ be two n -qubit stabilizer states. The $2n$ -qubit stabilizer bivector $|\psi \wedge \phi\rangle$ can be computed in $O(n^3)$ time.*

Algorithm 4.4.1 Computation of stabilizer bivectors

Require: Stabilizer matrices \mathcal{M}^ψ and \mathcal{M}^ϕ representing $|\psi\rangle$ and $|\phi\rangle$, respectively

Ensure: Stabilizer matrix $\mathcal{M}^{\psi \wedge \phi}$ for bivector $|\psi \wedge \phi\rangle$, if it exists

- ⇒ BASISNORMCIRC(\mathcal{M}^ψ) returns circuit \mathbf{C} such that $\mathbf{C}|\psi\rangle = |b\rangle$, where $|b\rangle$ is a basis state
- ⇒ CONJ(\mathcal{M}, \mathbf{C}) conjugates \mathcal{M} by Clifford or Pauli operator \mathbf{C} and returns the modified matrix
- ⇒ GAUSS(\mathcal{M}) reduces \mathcal{M} to canonical form
- ⇒ TENSOR($\mathcal{M}^\psi, \mathcal{M}^\phi$) computes $|\psi\rangle \otimes |\phi\rangle$ and returns the resulting matrix $\mathcal{M}^{\psi \otimes \phi}$ (Proposition II.15)
- ⇒ INPROD($\mathcal{M}^\psi, \mathcal{M}^\phi$) computes and returns $\langle \psi | \phi \rangle$
- ⇒ SUM($\mathcal{M}^{b_1}, \mathcal{M}^{b_2}$) computes $|b_1\rangle - |b_2\rangle$ and returns the resulting matrix $\mathcal{M}^{b_1-b_2}$

- 1: $\alpha \leftarrow \text{INPROD}(\mathcal{M}^\psi, \mathcal{M}^\phi)$ ▷ Apply Algorithm 4.2.1
 - 2: GAUSS(\mathcal{M}^ψ) ▷ Set \mathcal{M}^ψ to canonical form
 - 3: GAUSS(\mathcal{M}^ϕ) ▷ Set \mathcal{M}^ϕ to canonical form
 - 4: **if** ($\alpha = 0$ **and** \mathcal{M}^ψ dissimilar from \mathcal{M}^ϕ) **or** ($\alpha = 1$) **or** ($0 < \alpha < 1/\sqrt{2}$) **then**
 - 5: **exit** ▷ By Theorem III.21
 - 6: **end if**
 - 7: **if** \mathcal{M}^ψ dissimilar from \mathcal{M}^ϕ **then** ▷ 1-neighbor case from Theorem III.21
 - 8: $\mathcal{M}^\phi \leftarrow \text{CONJ}(\mathcal{M}^\psi, P)$ ▷ \mathcal{M}^ϕ now represents $P|\psi\rangle$, where P is a Pauli operator
 - 9: **end if**
 - 10: $\mathcal{M}^{\psi \otimes \phi} \leftarrow \text{TENSOR}(\mathcal{M}^\psi, \mathcal{M}^\phi)$
 - 11: $\mathcal{M}^{\phi \otimes \psi} \leftarrow \text{TENSOR}(\mathcal{M}^\phi, \mathcal{M}^\psi)$
 - 12: $\mathbf{C} \leftarrow \text{BASISNORMCIRC}(\mathcal{M}^{\psi \otimes \phi})$ ▷ Obtains basis-normalization circuit
 - 13: $\mathcal{M}^{b_1} \leftarrow \text{CONJ}(\mathcal{M}^{\psi \otimes \phi}, \mathbf{C})$ ▷ Maps $\mathcal{M}^{\psi \otimes \phi}$ to basis state $|b_1\rangle$
 - 14: $\mathcal{M}^{b_2} \leftarrow \text{CONJ}(\mathcal{M}^{\phi \otimes \psi}, \mathbf{C})$ ▷ Maps $\mathcal{M}^{\phi \otimes \psi}$ to basis state $|b_2\rangle$
 - 15: $\mathcal{M}^{b_1-b_2} \leftarrow \text{SUM}(\mathcal{M}^{b_1}, \mathcal{M}^{b_2})$ ▷ Obtains matrix for $|b_1\rangle - |b_2\rangle$ (proof of Lemma III.9)
 - 16: $\mathcal{M}^{\psi \wedge \phi} \leftarrow \text{CONJ}(\mathcal{M}^{b_1-b_2}, \mathbf{C}^\dagger)$
 - 17: **return** $\mathcal{M}^{\psi \wedge \phi}$
-

Proof. The bottleneck in Algorithm 4.4.1 is the computation of the basis-normalization circuit for the $2n \times 2n$ stabilizer matrix $\mathcal{M}^{\psi \otimes \phi}$, which takes $O(n^3)$ time. \square

4.5 Applications of Geometric Algorithms

In this section, we describe additional applications for several of our algorithms. First, we describe how Gaussian elimination facilitates representation and manipulation of *mixed states* using the stabilizer formalism. Second, we explain how our circuit-synthesis algorithm can be used to simulate and analyze strongly correlated quantum many-body systems.

4.5.1 Mixed stabilizer states

The work in [2] describes *mixed stabilizer states*, which form *uniform distributions* over all k -qubit states in the subspace generated by an n -qubit stabilizer matrix with $k < n$. Recall from Lemma I.10-iii that the projection onto the $+1$ eigenspace of Pauli operator Q is given by $(I + Q)/2$. Therefore, the density matrix of an n -qubit state with stabilizer matrix $\mathcal{M} = \{Q_1, Q_2, \dots, Q_n\}$ can be written as,

$$\rho = \frac{1}{2^n} (I + Q_1)(I + Q_2) \cdots (I + Q_n) \quad (4.1)$$

Expanding Equation 4.1 yields a uniform sum of density-matrix terms. Since k -qubit mixed stabilizer states can be written as the partial trace of a pure n -qubit state, they can be represented compactly using stabilizer matrices. Recall from Theorem II.3 that the set of Pauli operators (rows) in a stabilizer matrix that represents a pure state are linearly independent. In contrast, to represent mixed stabilizer states, one needs to maintain stabilizer matrices with linearly-dependent rows [2, 6]. This implies that a subset of the rows in the matrix can be reduced to the identity operator. Since Algorithm 2.2.1 reduces a stabilizer matrix to its Gauss-Jordan form, it can be used

to find and eliminate linearly-dependent rows. Such rows will show up as I -literal only rows at the bottom of the matrix after Algorithm 2.2.1 is applied. Similarly, Algorithms 4.1.1 and 4.2.1, generalize to the case of mixed stabilizer states since one can identify linearly-dependent rows via Algorithm 2.2.1 and then ignore them throughout the rest of the computation. Therefore, if the input matrices represent mixed states, Algorithm 4.2.1 computes the fidelity between them.

As outlined in [2] and proven in [6], one can also compute the partial trace over qubit j as follows: (i) apply a modified version of Algorithm 2.2.1 such that the j^{th} column of the stabilizer matrix has at most one X/Y literal (if X/Y literals exist in the column) and one Z literal (if Z literals exist in the column), and (ii) remove the rows and columns containing such literals. The resulting stabilizer matrix represents the reduced mixed state.

4.5.2 Simulation of quantum systems

Verstraete et al. [66] describe how to construct finite-depth quantum circuits that allow one to simulate strongly correlated quantum many-body systems. Such circuits *diagonalize* or *disentangle* the Hamiltonian that describes the dynamics of the physical system under simulation. Disentanglement circuits convert the whole Hamiltonian into one corresponding to non-interacting particles (i.e., it yields the steady-state solution of the corresponding Schrödinger equation), and thus allow us to gain a better understanding of such systems. In particular, the Hamiltonian of a stabilizer state can be written as a sum of Pauli-generator terms. Thus, it can be derived directly from the stabilizer matrix that represents the state. We note that Algorithm 4.1.1 can be used to construct diagonalization circuits for such Hamiltonians because it disentangles a stabilizer state by reducing its stabilizer matrix to basis form.

4.6 Empirical Studies

Our circuit-synthesis (Section 4.1) and inner-product (Section 4.2) algorithms hold potential to be used in several practical applications including quantum error correction, quantum-circuit simulation, and the computation of geometric measures of entanglement. Therefore, we implemented these algorithms in C++ and empirically validated their performance. Recall that the runtime of Algorithm 4.1.1 is dominated by the two nested for-loops (lines 20-35). The number of times these loops execute depends on the amount of entanglement in the input stabilizer state. In turn, the number of entangled qubits depends on the the number of CNOT gates in the circuit \mathbf{C} used to generate the stabilizer state $\mathbf{C} |0^{\otimes n}\rangle$ (Theorem II.11). By a simple heuristic argument [2], one generates highly entangled stabilizer states as long as the number of CNOT gates in \mathbf{C} is proportional to $n \log_2 n$. Therefore, we generated random n -qubit stabilizer circuits for $n \in \{20, 40, \dots, 500\}$ as follows: fix a parameter $\beta > 0$; then choose $\beta \lceil n \log_2 n \rceil$ unitary gates (CNOT, Phase or Hadamard) each with probability $1/3$. Then, each random \mathbf{C} is applied to the $|00 \dots 0\rangle$ basis state to generate random stabilizer matrices (states). The use of randomly generated benchmarks is justified for our experiments because (i) our algorithms are not explicitly sensitive to circuit topology and (ii) random stabilizer circuits are considered representative [21]. Both our inner-product and exterior-product algorithms exhibited similar asymptotic runtime behavior since the bottleneck in both algorithms is the execution of Algorithm 4.1.1. Therefore, for conciseness, we present experimental results only for our inner-product algorithm. For each n , we applied Algorithm 4.2.1 to pairs of random stabilizer matrices and measured the number of seconds needed to compute the inner product. The entire procedure was repeated for increasing degrees of entanglement by ranging β from 0.6 to 1.2 in increments of 0.1. Our results are shown in Figure 4.2a.

The runtime of Algorithm 4.2.1 appears to grow quadratically in n when $\beta = 0.6$. However, when we double the number of unitary gates ($\beta = 1.2$), the runtime exhibits

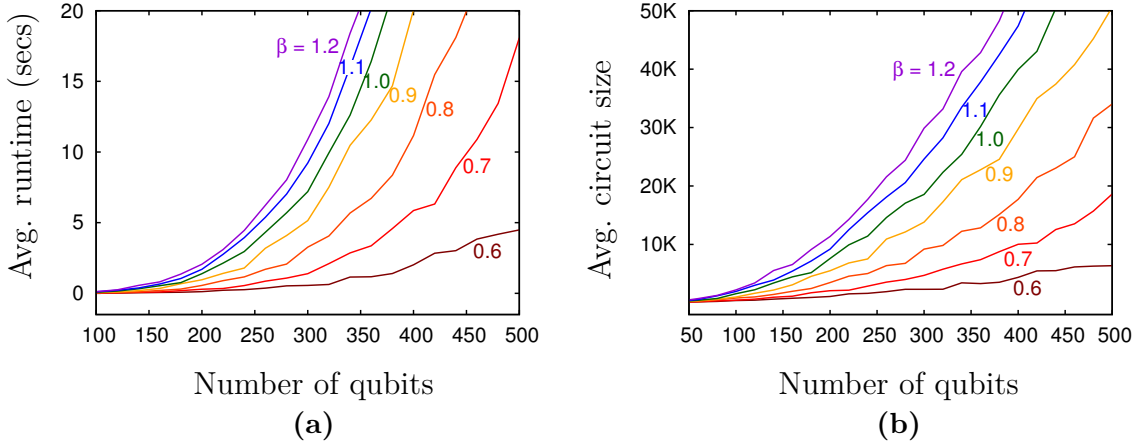


Figure 4.2: (a) Average runtime for Algorithm 4.2.1 to compute the inner product between two random n -qubit stabilizer states. The stabilizer matrices that represent the input states are generated by applying $\beta n \log_2 n$ unitary stabilizer gates to $|0^{\otimes n}\rangle$. (b) Average number of gates in the circuits produced by Algorithm 4.1.1.

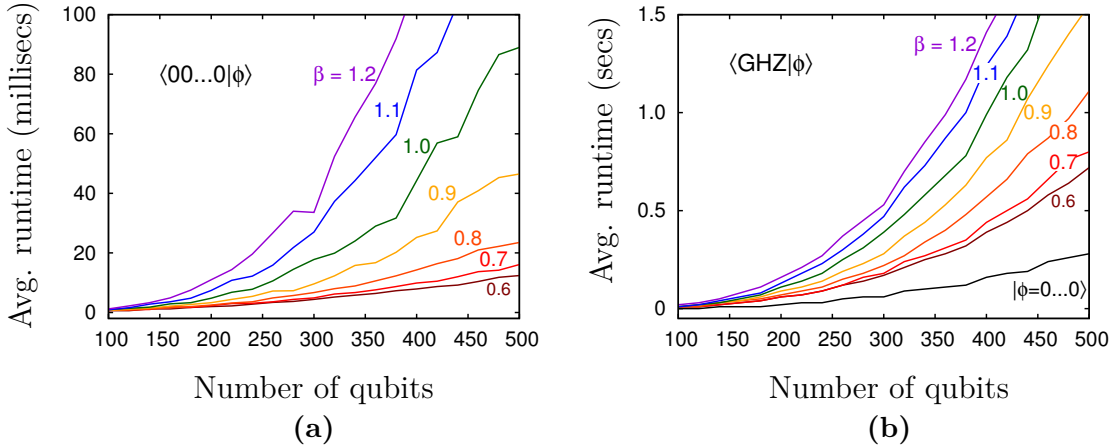


Figure 4.3: Average runtime for Algorithm 4.2.1 to compute the inner product between (a) $|0^{\otimes n}\rangle$ and random stabilizer state $|\phi\rangle$ and (b) the n -qubit GHZ state and random stabilizer state $|\phi\rangle$.

cubic growth. Therefore, Figure 4.2a shows that the performance of Algorithm 4.2.1 is highly dependent on the degree of entanglement in the input stabilizer states. Figure 4.2b shows the average size of the basis-normalization circuit returned by the calls to Algorithm 4.1.1. As expected (Proposition IV.2), the size of the circuit grows quadratically in n . Figure 4.3 shows the average runtime for Algorithm 4.2.1 to compute the inner product between: (i) the all-zeros basis state and random n -

qubit stabilizer states, and (ii) the n -qubit GHZ state and random stabilizer states. GHZ states are maximally entangled states of the form $|GHZ\rangle = \frac{|0^{\otimes n}\rangle + |1^{\otimes n}\rangle}{\sqrt{2}}$ that have been realized experimentally using several quantum technologies and are often encountered in practical applications such as error-correcting codes and fault-tolerant architectures. Figure 4.3 shows that, for such practical instances, Algorithm 4.2.1 computes the inner product in roughly $O(n^2)$ time (e.g., $\langle GHZ|0\rangle$). However, without a priori information about the input stabilizer matrices (e.g., a measure of the amount of the entanglement), one can only say that the performance of Algorithm 4.2.1 will be somewhere between quadratic and cubic in n . We compared Algorithm 4.1.1 to the circuit synthesis approach developed by Audenaert and Plenio (AP) [6] as part of their inner-product algorithm. The benchmark consists of randomly generated n -qubit stabilizer circuits with $n \log n$ unitary gates. Figure 4.4 shows that the AP algorithm produces (non-canonical) circuits with more than twice as many gates as our canonical circuits and takes roughly twice as long to produce them.

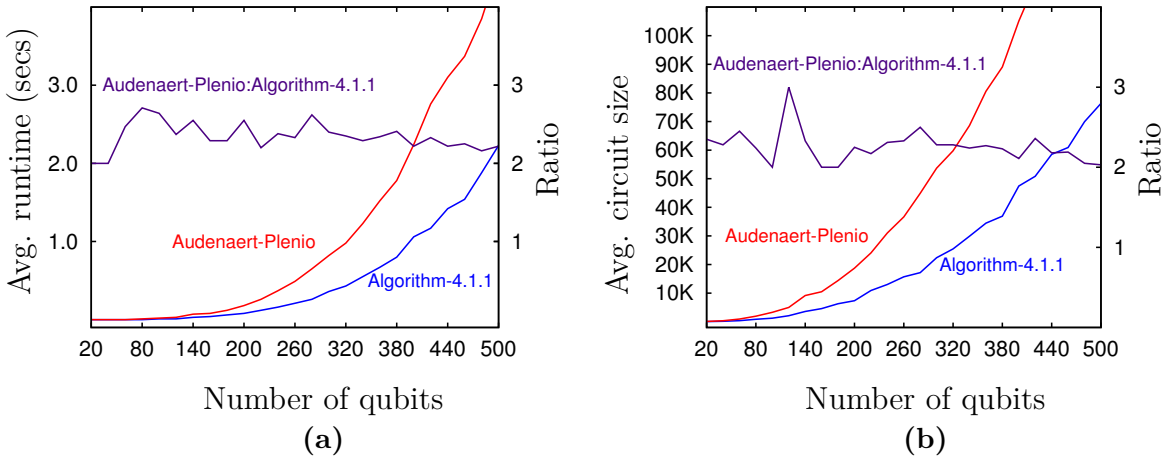


Figure 4.4: (a) Runtime and (b) circuit-size comparisons between Algorithm 4.1.1 and the circuit synthesis portion of the Audenaert-Plenio inner-product algorithm. On average, Algorithm 4.1.1 runs roughly twice as fast and produces canonical circuits that contain less than half as many gates. Furthermore, the Audenaert-Plenio circuits are not canonical.

4.7 Summary

In this chapter, we study algorithms for: (i) computing the inner product between stabilizer states, (ii) orthogonalizing a set of stabilizer states, and (iii) computing stabilizer bivectors. A crucial step of our inner-product algorithm is the synthesis of a circuit that transforms a stabilizer state into a computational-basis state. Our algorithm synthesizes such circuits using a 5-block canonical template structure using $O(n^2)$ stabilizer gates. Such canonical circuits play a key role in quantum fault-tolerant architectures since they minimize the number of gates required to initialize error-correcting codes. Our technique produces circuits with half as many gates as the approach from [6] using a block sequence that is shorter than the approach from [2]. Furthermore, we describe how our circuit-synthesis algorithm can be used to obtain diagonalization circuits for the Hamiltonian of a stabilizer state. We analyze the performance of our inner-product algorithm and show that, although its runtime is $O(n^3)$, it can take quadratic time in practice.

CHAPTER V

Engineering Stabilizer-based Simulation of Generic Quantum Circuits

As noted in Chapter II, the stabilizer gates by themselves do not form a universal set for quantum computation [2, 51]. However, the Hadamard and Toffoli gates do [4]. To simulate Toffoli and other *non-stabilizer gates*, we extend the formalism to include the representation of arbitrary quantum states as *superpositions of stabilizer states*.

Example V.1. Recall from Section 2.2.2 that computational-basis states are stabilizer states. Thus, any one-qubit state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ is a superposition of the stabilizer states $|0\rangle$ and $|1\rangle$. In general, any state decomposition in a computational basis is a stabilizer superposition.

Suppose $|\psi\rangle$ in Example V.1 is *unbiased*, i.e., $|\alpha_0|^2 = |\alpha_1|^2$, then $|\psi\rangle$ can be represented using a single stabilizer state instead of two (up to a global phase). *The key idea behind our technique is to identify and compress large unbiased superpositions on the fly during simulation to reduce resource requirements.*

5.1 Stabilizer Frames

Recall from Theorem I.14 that the Pauli operators form an orthonormal basis for linear operators and thus any unitary operator U has a unique Pauli expansion

(Proposition I.16). Consider the non-stabilizer gate U with Pauli expansion $f_p(U) = \sum_i \alpha_i P_i$ (Definition I.17). The action of U on the n -qubit stabilizer state $|\psi\rangle$ can be described by a superposition of states,

$$U|\psi\rangle = \sum_i \alpha_i P_i |\psi\rangle = \sum_i \alpha_i |\phi_i\rangle$$

Since each state $|\phi_i\rangle$ in the superposition is a stabilizer state, we can represent $U|\psi\rangle$ using a list of stabilizer matrices. Let the stabilizer matrix for $|\psi\rangle$ be $\mathcal{M} = \{R_1, \dots, R_n\}$, where R_i are the rows of the matrix. To compute each element of the list for $U|\psi\rangle$, we conjugate the rows of \mathcal{M} by P_i as follows.

$$U\mathcal{M}U^\dagger = \sum_i \alpha_i \{P_i R_1 P_i^\dagger, \dots, P_i R_n P_i^\dagger\} \quad (5.1)$$

For any pair of Pauli operators P and Q , $PQP^\dagger = (-1)^c Q$, where $c = 0$ if P and Q commute, and $c = 1$ otherwise. Therefore, Equation 5.1 can be represented by a list of *similar stabilizer matrices* (Definition II.24).

Definition V.2. An n -qubit *stabilizer frame* \mathcal{F} is a set of $k \leq 2^n$ stabilizer states $\{|\psi_j\rangle\}_{j=1}^k$ that forms an orthogonal subspace basis in the Hilbert space. We represent \mathcal{F} by a pair consisting of (i) a stabilizer matrix \mathcal{M} and (ii) a set of distinct *phase vectors* $\{\sigma_j\}_{j=1}^k$, where $\sigma_j \in \{\pm 1\}^n$. We use \mathcal{M}^{σ_j} to denote the ordered assignment of the elements in σ_j as the (± 1) -phases of the rows in \mathcal{M} . Therefore, state $|\psi_j\rangle$ is represented by \mathcal{M}^{σ_j} . The size of the frame, which we denote by $|\mathcal{F}|$, is equal to k .

Each phase vector σ_j can be viewed as a binary (0-1) encoding of the integer index that denotes the respective basis vector. Thus, when dealing with 64 qubits or less, a phase vector can be compactly represented by a 64-bit integer (modern CPUs also support 128-bit integers). To represent an arbitrary state $|\Psi\rangle$ using \mathcal{F} , one additionally maintains a vector of complex amplitudes $\mathbf{a} = (a_1, \dots, a_k)$, which

corresponds to the decomposition of $|\Psi\rangle$ in the basis $\{|\psi_j\rangle\}_{j=1}^k$ defined by \mathcal{F} , i.e., $|\Psi\rangle = \sum_{j=1}^k a_j |\psi_j\rangle$ and $\sum_{j=1}^k |a_j|^2 = 1$. Observe that each a_j forms a pair with phase vector σ_j in \mathcal{F} since $|\psi_j\rangle \equiv \mathcal{M}^{\sigma_j}$. Any stabilizer state can be viewed as a one-element frame.

Example V.3. Let $|\Psi\rangle = a_1(|00\rangle + |01\rangle) + a_2(|10\rangle + |11\rangle)$. Then $|\Psi\rangle$ can be represented by the stabilizer frame \mathcal{F} depicted in Figure 5.1.

5.1.1 Frame Operations

We now describe several *frame operations* that are useful for manipulating stabilizer-state superpositions.

ROTATE(\mathcal{F}, U). Consider the stabilizer basis $\{|\psi_j\rangle\}_{j=1}^k$ defined by frame \mathcal{F} . A stabilizer or Pauli gate U acting on \mathcal{F} maps such a basis to $\{U|\psi_j\rangle = e^{i\theta_j} |\varphi_j\rangle\}_{j=1}^k$, where $e^{i\theta_j}$ is the global phase of stabilizer state $|\varphi_j\rangle$. Since we obtain a new stabilizer basis that spans the same subspace, this operation effectively *rotates the stabilizer frame*. Computationally, we perform a frame rotation as follows. First, update the stabilizer matrix associated with \mathcal{F} as per Section 2.2.2. Then, iterate over the phase vectors in \mathcal{F} and update each one accordingly (Table 2.1). Let $\mathbf{a} = (a_1, \dots, a_k) \in \mathbb{C}^k$ be the

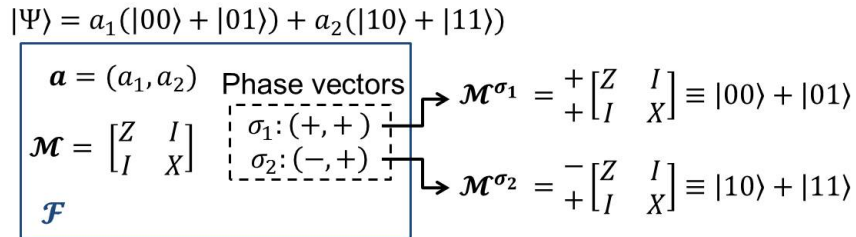


Figure 5.1: Example of a stabilizer frame that represents $|\psi\rangle$. Observe that while $|\psi\rangle$ is composed of four computational-basis amplitudes, its frame representation has only two phase vectors. For $|a_1|^2 = |a_2|^2$, one can manipulate \mathcal{F} to reduce its size. We discuss this technique in Section 5.1.2.

decomposition of $|\Psi\rangle$ onto \mathcal{F} . Frame rotation simulates the action of stabilizer gate U on $|\Psi\rangle$ since,

$$U|\Psi\rangle = \sum_{j=1}^k a_j U|\psi_j\rangle = \sum_{j=1}^k a_j e^{i\theta_j} |\varphi_j\rangle \quad (5.2)$$

Observe that the global phase $e^{i\theta_j}$ of each $|\varphi_j\rangle$ becomes relative with respect to $U|\Psi\rangle$. Therefore, our approach requires that we compute such phases explicitly using the approach outlined in Section 2.3.3 in order to maintain a consistent representation.

Recall from Section 2.3.3 that global-phase computation requires sampling of computational-basis amplitudes from the stabilizer matrix \mathcal{M} . By Observation II.17, \mathcal{M} needs to be in row-echelon form (Figure 2.4) to perform such sampling. Thus, simulating gates with global-phase maintenance would take $O(n^3|\mathcal{F}|)$ time for n -qubit stabilizer frames. To improve this, we introduce a simulation invariant.

Invariant V.4. The stabilizer matrix \mathcal{M} associated with \mathcal{F} remains in row-echelon form during simulation.

Since stabilizer gates affect at most two columns of \mathcal{M} , Invariant V.4 can be repaired with $O(n)$ row multiplications. Since each row multiplication takes $\Theta(n)$, the runtime required to update \mathcal{M} during global-phase maintenance simulation is $O(n^2)$. Therefore, for an n -qubit stabilizer frame, the overall runtime for simulating a single stabilizer gate is $O(n^2 + n|\mathcal{F}|)$ since one can memoize the updates to \mathcal{M} required to compute each a_j . Another advantage of maintaining this invariant is that the outcome of deterministic measurements (Section 2.2.2) can be decided in time linear in n since it eliminates the need to perform Gaussian elimination.

COFACTOR(\mathcal{F}, c). This operation facilitates measurement of stabilizer-state superpositions and simulation of non-stabilizer gates using frames. (Recall from Section 1.3 that post-measurement states are also called cofactors.) Here, $c \in \{1, 2, \dots, n\}$ is the cofactor index. Let $\{|\psi_j\rangle\}_{j=1}^k$ be the stabilizer basis defined by \mathcal{F} . Frame cofactoring

maps such a basis to $\{|\psi_j^{c=0}\rangle, |\psi_j^{c=1}\rangle\}_{j=1}^k$. Observe that, after a frame is cofactored, its size either remains the same (qubit c was in a deterministic state and thus one of its cofactors is empty) or doubles (qubit c was in a superposition state and thus both cofactors are non-empty). We now describe the steps required to cofactor \mathcal{F} . Suppose qubit c is in a deterministic case. Since \mathcal{M} is maintained in row-echelon form (Invariant V.4) no frame updates are necessary. In the randomized-outcome case, we apply the measurement algorithm described in Section 2.2.2 to \mathcal{M} while forcing the outcome to $x \in \{0, 1\}$ in order to generate the $|x\rangle$ -cofactor. This is done twice – once for each cofactor, and the row operations performed on \mathcal{M} are memoized each time. We then iterate over each phase vector in \mathcal{F} and permute its elements according to the memoized operations, and generate additional phase vectors corresponding to the cofactor states. As in the case of frame rotation, this operation is linear in the number of phase vectors and quadratic in the number of qubits. However, by the end of the operation, the number of phase vectors (states) in \mathcal{F} will have grown by a (worst case) factor of two. Furthermore, any state $|\Psi\rangle$ represented by \mathcal{F} is invariant under frame cofactoring.

5.1.2 Frame-based Simulation of Quantum Circuits

Let \mathcal{F} be the stabilizer frame used to represent the n -qubit state $|\Psi\rangle$. Following our discussion in Section 5.1, any stabilizer or Pauli gate can be simulated directly via frame rotation. Suppose we want to simulate the action of $TOF_{c_1 c_2 t}$, where c_1 and c_2 are the control qubits, and t is the target. First, we decompose $|\Psi\rangle$ into all four of its double cofactors (Section 1.3) over the control qubits to obtain the following equal superposition of orthogonal states:

$$|\Psi\rangle = \frac{|\Psi^{c_1 c_2=00}\rangle + |\Psi^{c_1 c_2=01}\rangle + |\Psi^{c_1 c_2=10}\rangle + |\Psi^{c_1 c_2=11}\rangle}{2}$$

Then, we compute the action of the Toffoli as,

$$TOF_{c_1 c_2 t} |\Psi\rangle = \frac{|\Psi^{c_1 c_2=00}\rangle + |\Psi^{c_1 c_2=01}\rangle + |\Psi^{c_1 c_2=10}\rangle + X_t |\Psi^{c_1 c_2=11}\rangle}{2} \quad (5.3)$$

where X_t is the Pauli gate (NOT) acting on target t . We simulate Equation 5.3 with the following frame operations. (An example of the process is depicted in Figure 5.2.)

Algorithm 5.1.1 Frame-based simulation of the Toffoli gate

1) COFACTOR(\mathcal{F}, c_1).

2) COFACTOR(\mathcal{F}, c_2).

3) Let Z_j be the Pauli operator with a Z literal in its j^{th} position and I everywhere else. Due to Steps 1 and 2, the matrix \mathcal{M} associated with \mathcal{F} must have two rows of the form Z_{c_1} and Z_{c_2} . Let u and v be the indices of such rows, respectively. For each phase vector $\sigma_{j \in \{1, \dots, |\mathcal{F}|\}}$, if the u and v elements of σ_j are both -1 (i.e., if the phase vector corresponds to the $|\Psi^{c_1 c_2=11}\rangle$ cofactor), flip the value of element t in σ_j (apply X_t to this cofactor).

Controlled-phase gates $R(\alpha)_{ct}$ can also be simulated using stabilizer frames. This gate applies a phase-shift factor of $e^{i\alpha}$ if both the control qubit c and target qubit t

$$\begin{array}{l}
 + \begin{bmatrix} X & I & I \\ I & X & I \\ I & I & Z \end{bmatrix} \xrightarrow{\text{CoF}_{1,2}} \begin{array}{l}
 \boxed{ \begin{array}{ll}
 |\Psi^{c_1 c_2=00}\rangle = |000\rangle & |\Psi^{c_1 c_2=10}\rangle = |100\rangle \\
 + \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} & - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 |\Psi^{c_1 c_2=01}\rangle = |010\rangle & |\Psi^{c_1 c_2=11}\rangle = |110\rangle \\
 + \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} & - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 |\Psi^{c_1 c_2=11}\rangle = |111\rangle & - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix}
 \end{array}
 } \\
 + \begin{bmatrix} X & I & I \\ I & X & I \\ I & I & Z \end{bmatrix} \xrightarrow{\text{CoF}_{1,2}} \begin{array}{l}
 |\Psi^{c_1 c_2=00}\rangle = |000\rangle \quad |\Psi^{c_1 c_2=10}\rangle = |100\rangle \\
 + \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \quad - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 |\Psi^{c_1 c_2=01}\rangle = |010\rangle \quad |\Psi^{c_1 c_2=11}\rangle = |110\rangle \\
 + \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \quad - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 |\Psi^{c_1 c_2=11}\rangle = |111\rangle \quad - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix}
 \end{array}
 \end{array}
 \xrightarrow{X_3} \begin{array}{l}
 - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix} \\
 - \begin{bmatrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{bmatrix}
 \end{array}
 \end{array}$$

Figure 5.2: Simulation of $TOF_{c_1 c_2 t} |\Psi\rangle$ using a stabilizer-state superposition (Equation 5.3). Here, $c_1 = 1$, $c_2 = 2$ and $t = 3$. Amplitudes are omitted for clarity and the (\pm) -phase vectors are shown as columns prefixed to their corresponding matrices. The X gate is applied to the third qubit of the $|\Psi^{c_1 c_2=11}\rangle$ cofactor.

are set. Thus, we compute the action of $R(\alpha)_{ct}$ as,

$$R(\alpha)_{ct} |\Psi\rangle = \frac{|\Psi^{ct=00}\rangle + |\Psi^{ct=01}\rangle + |\Psi^{ct=10}\rangle + e^{i\alpha} |\Psi^{ct=11}\rangle}{2} \quad (5.4)$$

Equation 5.4 can be simulated via frame-based simulation using a similar approach as discussed for *TOF* gates. Let $(a_1, \dots, a_{|\mathcal{F}|})$ be the decomposition of $|\Psi\rangle$ onto \mathcal{F} . First, cofactor \mathcal{F} over the c and t qubits. Then, for any phase vector $\sigma_{j \in \{1, \dots, |\mathcal{F}|\}}$ that corresponds to the $|\Psi^{ct=11}\rangle$ cofactor, set $a_j = a_j e^{i\alpha}$. Observe that, in contrast to *TOF* gates, controlled- $R(\alpha)$ gates produce biased superpositions. The Hadamard and controlled- $R(\alpha)$ gates are used to implement the quantum Fourier transform circuit, which plays a key role in Shor's factoring algorithm.

Prior work on simulation of non-stabilizer gates using the stabilizer formalism and related "beyond stabilizer" techniques can be found in [2]. The authors represent the resulting non-stabilizer state as a sum of density-matrix terms. Let ρ be an n -qubit stabilizer state whose stabilizer is generated by $\{Q_1, Q_2, \dots, Q_n\}$. Then,

$$\rho = \frac{1}{2^n} (I + Q_1)(I + Q_2) \cdots (I + Q_n) \quad (5.5)$$

If a stabilizer operation is performed, we keep track of ρ using the tableau technique.

Consider a non-stabilizer gate $U = \sum_i \alpha_i P_i$,

$$\begin{aligned} U\rho U^\dagger &= \frac{1}{2^n} \left(\sum_i \alpha_i P_i \right) \prod_j (I + Q_j) \left(\sum_k \alpha_k^* P_k \right) \\ &= \frac{1}{2^n} \sum_{i,k} \alpha_i \alpha_k^* P_i P_k \prod_j (I + (-1)^{Q_j \cdot P_k} Q_j) \end{aligned} \quad (5.6)$$

where $Q_j \cdot P_k$ is the symplectic inner product (Equation 2.16), which is 0 when Q_j and P_k commute and 1 otherwise. Simplifying Equation 5.6 yields a sum of at most 4^{2d} terms, where d is the number of qubits that U acts on. Each term is described by a Pauli operator $(P_i P_k)$ and a vector of eigenvalues for the stabilizer. Applying

a stabilizer gate to Equation 5.6 maps the Pauli operators in the sum of terms into other Pauli operators. If another non-stabilizer gate is applied, each term in Equation 5.6 is expanded in the same manner according to the Pauli expansion of the gate. Thus, after m non-stabilizer operations, the number of terms is at most 4^{2md} . In contrast, the number of states in our technique is $O(2^m)$ although we do not handle density matrices and perform more sophisticated bookkeeping.

5.1.3 Frame measurements

Since the states in \mathcal{F} are orthogonal, the outcome probability when measuring \mathcal{F} is calculated as the sum of the normalized outcome probabilities of each state. The normalization is with respect to the superposition amplitudes stored in \mathbf{a} (Example V.3). Thus, the overall measurement outcome may have a non-uniform distribution. Formally, let $|\Psi\rangle = \sum_i a_i |\psi_i\rangle$ be the superposition of states represented by \mathcal{F} , the probability of observing outcome $x \in \{0, 1\}$ upon measuring qubit m is,

$$p(x)_\Psi = \sum_{i=1}^k |a_i|^2 \langle \psi_i | P_x^m | \psi_i \rangle = \sum_{i=1}^k |a_i|^2 p(x)_{\psi_i}$$

where P_x^m denotes the measurement operator in the computational basis x as discussed in Section 1.3. The outcome probability for each stabilizer state $p(x)_{\psi_i}$ is computed as outlined in Section 2.2.2. Once we compute $p(x)_\Psi$, we flip a (possibly biased) coin to decide the outcome and cofactor the frame such that only the states that are consistent with the measurement remain in the frame.

5.2 Multiframe Simulation

Although a single frame is sufficient to represent a stabilizer-state superposition $|\Psi\rangle$, one can sometimes tame the exponential growth of states in $|\Psi\rangle$ by constructing a *multiframe representation*. Such a representation cuts down the total number of

states required to represent $|\Psi\rangle$ by at least a half, thus improving the scalability of our technique. Our experiments in Section 5.3 show that, when simulating certain instances of ripple-carry adders, the number of states in $|\Psi\rangle$ grows linearly when multiframes are used but exponentially when a single frame is used. To this end, we introduce an additional frame operation.

COALESCE(\mathcal{F}). One derives a multiframe representation directly from a single frame \mathcal{F} by examining the set of phase vectors and identifying *candidate pairs* that can be *coalesced* into a single phase vector associated with a different stabilizer matrix. Since we maintain the stabilizer matrix \mathcal{M} of a frame in row-echelon form (Invariant V.4), examining the phases corresponding to Z_k -rows (Z -literal in k^{th} column and I 's in all other columns) allows us to identify the columns in \mathcal{M} that need to be modified in order to coalesce candidate pairs. More generally, suppose $\langle\sigma_r, \sigma_j\rangle$ is a pair of phase vectors from the same n -qubit frame. Then $\langle\sigma_r, \sigma_j\rangle$ is considered a candidate iff it has the following properties: (i) σ_r and σ_j are equal up to $m \leq n$ entries corresponding to Z_k -rows (where k is the qubit the row stabilizes), and (ii) $a_r = i^d a_j$ for some $d \in \{0, 1, 2, 3\}$ (where a_r and a_j are the frame amplitudes paired with σ_r and σ_j). Let $\mathbf{e} = \{e_1, \dots, e_m\}$ be the indices of a set of differing phase-vector elements, and let $\mathbf{v} = \{v_1, \dots, v_m\}$ be the qubits stabilized by the Z_k -rows identified by \mathbf{e} . The steps in our coalescing procedure are:

Algorithm 5.2.1 Frame coalescing

- 1) Sort phase vectors such that *candidate pairs* with differing elements \mathbf{e} are next to each other.
 - 2) Coalesce candidate pairs into a new set of phase vectors $\boldsymbol{\sigma}'$.
 - 3) Create a new frame \mathcal{F}' consisting of $\boldsymbol{\sigma}'$ and matrix $\mathbf{C}\mathbf{M}\mathbf{C}^\dagger$, where $\mathbf{C} = \text{CNOT}_{v_1, v_2} \text{CNOT}_{v_1, v_3} \cdots \text{CNOT}_{v_1, v_m} \text{P}_{v_1}^d \text{H}_{v_1}$.
 - 4) Repeat Steps 2–3 until no candidate pairs remain.
-

The output of this coalescing operation is a list of n -qubit frames $\mathbf{F} = \{\mathcal{F}'_1, \mathcal{F}'_2, \dots, \mathcal{F}'_s\}$

(i.e., a multiframe) that together represent the same superposition as the original input frame \mathcal{F} . The runtime of this procedure is dominated by Step 1. Each phase-vector comparison takes $\Theta(n)$ time. Therefore, the runtime of Step 1 and our overall coalescing procedure is $O(nk \log k)$ for a single frame with k phase vectors.

Example V.5. Suppose we coalesce frame \mathcal{F} depicted in Figure 5.3. To obtain \mathcal{F}_1 , conjugate the second column of \mathcal{M} by an H gate and coalesce the first two phase vectors in \mathcal{F} . To obtain \mathcal{F}_2 , conjugate the second column by H, then conjugate the second and third columns by CNOT, and coalesce the last two phase vectors in \mathcal{F} . Observe that no P gates are applied since $d = 0$ for all pairs in \mathbf{a} .

Candidate pairs can be identified even in the absence of Z_k -rows in an n -qubit \mathcal{M} . By Corollary II.12, one can always find a stabilizer circuit \mathbf{C} that maps \mathcal{M} to the matrix structure depicted in Figure 2.4a, whose rows are all of Z_k form. We leverage the circuit synthesis algorithm described in Section 4.1 to extend our coalescing operation as follows:

Algorithm 5.2.2 Frame coalescing (extended)

- 1) Find \mathbf{C} that maps \mathcal{M} to computational-basis form (Algorithm 4.1.1).
 - 2) ROTATE(\mathcal{F} , \mathbf{C}).
 - 3) $\{\mathcal{F}'_1, \mathcal{F}'_2, \dots, \mathcal{F}'_s\} \leftarrow \text{COALESCE}(\mathcal{F})$.
 - 4) ROTATE(\mathcal{F}'_i , \mathbf{C}^\dagger) for $i \in \{1, \dots, s\}$.
-

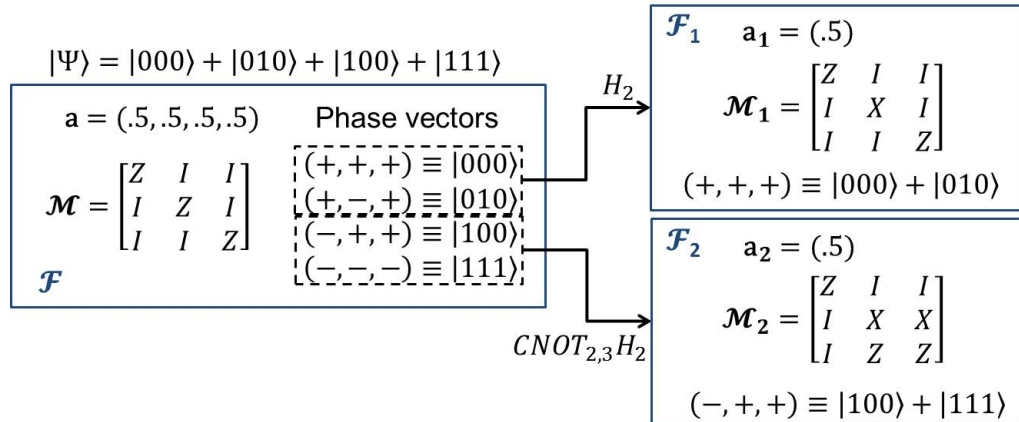


Figure 5.3: Example of how a multiframe representation is derived from a single-frame representation.

To simulate stabilizer, Toffoli and controlled- $R(\alpha)$ gates using multiframe \mathbf{F} , we apply single-frame operations to each frame in the list independently. For Toffoli and controlled- $R(\alpha)$ gates, additional steps are required:

Algorithm 5.2.3 Simulation-flow steps

- 1) Apply the coalescing procedure to each frame and insert the new “coalesced” frames in the list.
 - 2) Merge frames with equivalent stabilizer matrices.
 - 3) Repeat Steps 1–2 until no new frames are generated.
-

5.2.1 Orthogonality of Multiframes

We introduce the following invariant to facilitate simulation of quantum measurements on multiframes.

Invariant V.6. The stabilizer frames that represent a superposition of stabilizer states remain mutually orthogonal during simulation, i.e., every pair of (basis) vectors from any two frames are orthogonal.

Given multiframe $\mathbf{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$, one needs to consider two separate tasks in order to maintain Invariant V.6. The first task is to verify the pairwise orthogonality of the states in \mathbf{F} . The orthogonality of two n -qubit stabilizer states can be checked using the inner-product algorithm describe in Section 4.2, which takes $O(n^3)$ time. To improve this, we derive a heuristic based on Corollary III.4, which takes advantage of similarities across the (canonical) matrices in \mathbf{F} to avoid expensive inner-product computations in many cases. We note that, when simulating quantum circuits that exhibit significant structure, \mathbf{F} contains similar stabilizer matrices with equivalent rows (Pauli operators). Let $\mathbf{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$ be the set of n -qubit stabilizer matrices in \mathbf{F} . Our heuristic keeps track of a set of Pauli operators $\mathbf{P} = \{P_1, P_2, \dots, P_{k \leq n}\}$, that form an *intersection* across the matrices in \mathbf{M} .

Example V.7. Consider the multiframe from Figure 5.3. The intersection \mathbf{P} consists of the Pauli operator ZII (first row of \mathcal{M}_1 and \mathcal{M}_2).

By Corollary III.4, if two phase vectors (states) have different entries corresponding to the Pauli operators in \mathbf{P} , then the states are orthogonal and no inner-product computation is required. For certain practical instances, including the benchmarks described in Section 5.3, we obtain a non-empty \mathbf{P} and our heuristic proves effective. When \mathbf{P} is empty or the phase-vector pair is equivalent, we use the Algorithm 4.2.1 to verify orthogonality. Therefore, in the worst case, checking pairwise orthogonality of the states in \mathbf{F} takes $O(n^3k^2)$ time for a multiframe that represents a k -state superposition.

The second task to consider when maintaining Invariant V.6 is the orthogonalization of the states in $\mathbf{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$ when our check fails. To accomplish this, we iteratively apply the COFACTOR operation to each frame in \mathbf{F} in order to decompose \mathbf{F} into a single frame. This process is equivalent to the orthogonalization procedure described in Algorithm 4.3.1.

Observe that each iteration of Algorithm 4.3.1 can potentially double the number of states in the superposition. Since the algorithm terminates when a single frame remains, the resulting states are represented by distinct phase vectors and are therefore pairwise orthogonal.

The overall simulation flow of our frame-based technique is shown in Figure 5.4 and implemented in our software package `Quipu`.

5.2.2 Parallel Frame-based Simulation

Unlike other techniques based on compact representations of quantum states (e.g., using BDD data structures [67]), most frame-based operations are inherently parallel and lend themselves to multi-threaded implementation. The only step in Figure 5.4 that presents a bottleneck for a parallel implementation is the orthogonalization pro-

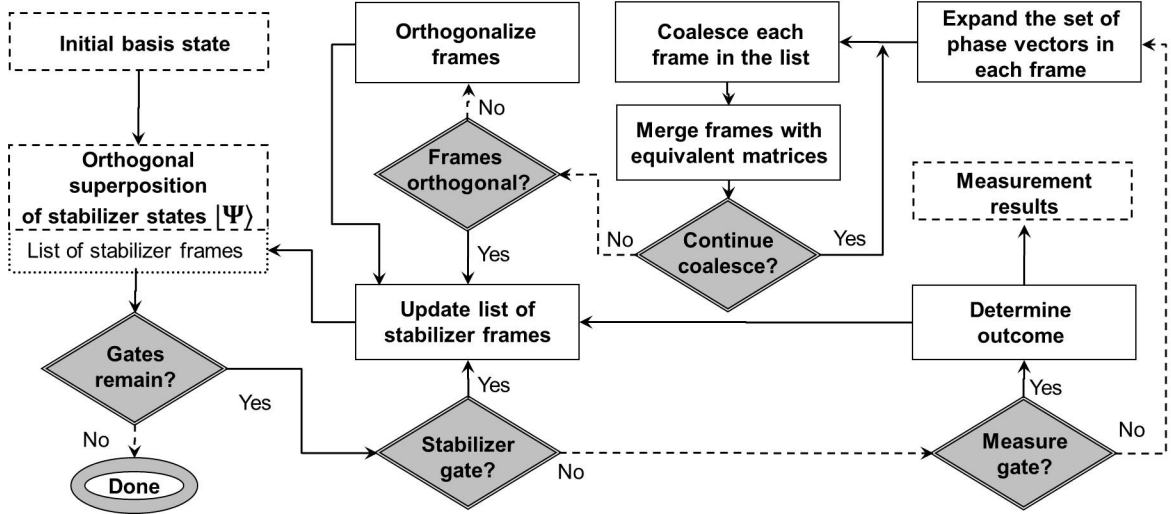


Figure 5.4: Overall simulation flow for Quipu.

cedure, which requires communication across frames. All other processes at both the single- and multi-frame levels can be executed on large subsets of phase vectors independently.

We implemented a multithreaded version of `Quipu` using the C++11 thread support library. Each worker thread is launched via the `std::async()` function. Figure 5.5 shows our wrapper function for executing calls to `std::async()`. The function `async_launch()` takes as input: (i) a frame operation (`Func f`), (ii) a range of phase-vector elements defined by `Iter begin` and `Iter end`, and (iii) any additional parameters (`Params... p`) required for the frame operation. Furthermore, the function returns a vector of `std::future` – the C++11 mechanism for accessing the result of an asynchronous operation scheduled by the C++ runtime support system. As Figure 5.5 shows, the workload (number of phase vectors) of each thread is distributed evenly across the number of cores in the system (`MTHREAD`). The results from each thread are joined only when orthogonalization procedures are performed since they require communication between multiple threads. This is accomplished by calling the `std::future::get()` function on each future. All stabilizer gates and measurements are simulated in parallel.


```

template<class Func, class Iter, class... Params>
auto async_launch(Func f, Iter begin, Iter end, Params... p)
    -> vector< decltype( async(f, begin, end, p...) ) >
{
    vector< decltype( async(f, begin, end, p...) ) > futures;
    int size = distance(begin, end);
    int n = size/MTHREAD;
    futures.reserve(MTHREAD);
    for(int i = 0; i < MTHREAD; i++)
    {
        Iter first = begin + i*n;
        Iter last = (i < MTHREAD - 1) ? begin + (i+1)*n : end;
        futures.push_back( async( f, first, last, p...) );
    }
    return futures;
}

```

Figure 5.5: Our C++11 template function for executing the frame operations (Func `f`) described in Section 5.1.2 in parallel. The function accepts a range of vector elements defined by iterators `Iter begin` and `Iter end`. `Params... p` is the variadic template argument that defines the parameters of Func `f`. The number of threads allowed is defined by `MTHREAD`. The function returns a vector of `std::futures`, which can be used to access the result of the asynchronous operations.

5.3 Empirical Validation

We tested a single-threaded and multi-threaded versions of `Quipu` on a conventional Linux server using several benchmark sets consisting of stabilizer circuits, quantum ripple-carry adders, quantum Fourier transform circuits and quantum fault-tolerant (FT) circuits.

5.3.1 Stabilizer circuits

We compared the runtime performance of single-threaded `Quipu` against that of `CHP` using a benchmark set similar to the one used in [2]. We generated random stabilizer circuits on n qubits, for $n \in \{100, 200, \dots, 1500\}$. The use of randomly generated benchmarks is justified for our experiments because (i) our algorithms are

not explicitly sensitive to circuit topology and (ii) random stabilizer circuits have been considered representative [21]. For each n , we generated the circuits as follows: fix a parameter $\beta > 0$; then choose $\beta \lceil n \log_2 n \rceil$ random unitary gates (CNOT, P or H) each with probability $1/3$. Then measure each qubit $a \in \{0, \dots, n - 1\}$ in sequence. We measured the number of seconds needed to simulate the entire circuit. The entire procedure was repeated for β ranging from 0.6 to 1.2 in increments of 0.1. Figure 5.6 shows the average time needed by `Quipu` and `CHP` to simulate this benchmark set. The purpose of this comparison is to evaluate the overhead of supporting generic circuit simulation in `Quipu`. Since `CHP` is specialized to stabilizer circuits, we do not expect `Quipu` to be faster. When $\beta = 0.6$, the simulation time appears to grow roughly linearly in n for both simulators. However, when the number of unitary gates is doubled ($\beta = 1.2$), the runtime of both simulators grows roughly quadratically. Thus, the performance of both `CHP` and `Quipu` depends strongly on the circuit being simulated. Although `Quipu` is $5\times$ slower than `CHP`, we note that `Quipu` maintains global phases whereas `CHP` does not. Nonetheless, Figure 5.6 shows that `Quipu` is asymptotically as fast as `CHP` when simulating stabilizer circuits that contain a linear number of measurements. Moreover, when simulating non-stabilizer circuits, the multi-threaded version of `Quipu` runs twice as fast as the single-threaded version (Figure 5.10). This speedup is not available when simulating stabilizer circuits.

5.3.2 Quantum ripple-carry adders

Our second benchmark set consists of n -bit ripple-carry (Cuccaro) adder [25] circuits, which often appear as components in many quantum arithmetic circuits [44]. The Cuccaro circuit for $n = 3$ is shown in Figure 5.7. Such circuits act on two n -qubit input registers, one ancilla qubit and one carry qubit for a total of $2(n + 1)$ qubits. We applied H gates to all $2n$ input qubits in order to simulate addition on a superposition of 2^{2n} computational-basis states. Figure 5.8 shows the average run-

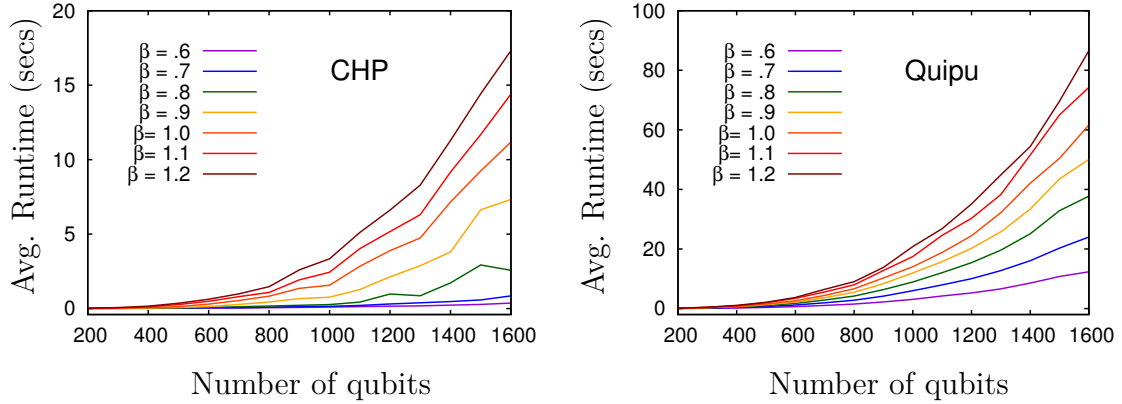


Figure 5.6: Average time needed by **Quipu** and **CHP** to simulate an n -qubit stabilizer circuit with $\beta n \log n$ gates and n measurements. **Quipu** is asymptotically as fast as **CHP** but is not limited to stabilizer circuits.

time needed to simulate this benchmark set using **Quipu**. For comparison, we ran the same benchmarks on an optimized version of **QuIDDPro**, called **QPLite**¹, specific to circuit simulation [67]. When $n < 15$, **QPLite** is faster than **Quipu** because the **QuIDD** representing the state vector remains compact during simulation. However, for $n > 15$, the compactness of the **QuIDD** is considerably reduced, and the majority of **QPLite**'s runtime is spent in non-local pointer-chasing and memory (de)allocation.

¹**QPLite** is up to $4\times$ faster since it removes overhead related to **QuIDDPro**'s interpreted front-end for quantum programming.

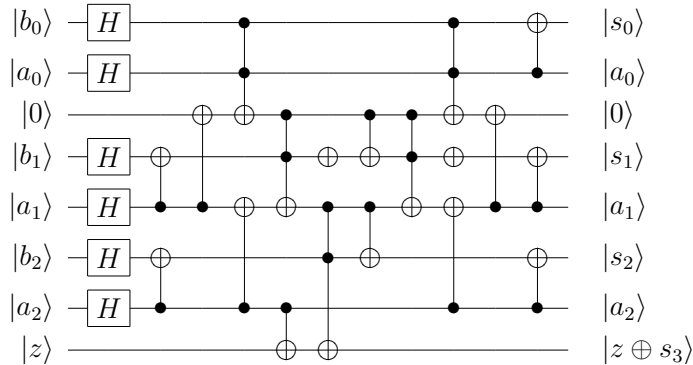


Figure 5.7: Ripple-carry (Cuccaro) adder for 3-bit numbers $a = a_0a_1a_2$ and $b = b_0b_1b_2$ [25, Figure 6]. The third qubit from the top is an ancilla and the z qubit is the carry. The b -register is overwritten with the result $s_0s_1s_2$.

Thus, `QPLite` fails to scale on such benchmarks and one observes an exponential increase in runtime. Memory usage for both `Quipu` and `QPLite` was nearly unchanged for these benchmarks. `Quipu` consumed 4.7MB on average while `QPLite` consumed almost twice as much (8.5MB).

We ran the same benchmarks using both the single-frame and multiframe approaches. In the case of a single frame, the number of states in a superposition grows exponentially in n . However, in the multiframe approach, the number of states grows linearly in n . This is because *TOF* gates produce large equal superpositions that are effectively compressed by our coalescing technique. Since our frame-based algorithms require $\text{poly}(k)$ time for k states in a superposition, `Quipu` simulates Cuccaro circuits in polynomial time and space for input states consisting of large superpositions of basis states. On such instances, known linear-algebraic simulation techniques (e.g., `QuIDDPro`) take exponential time while `Quipu`'s runtime grows quadratically (best quadratic fit $f(x) = 0.5248x^2 - 15.815x + 123.86$ with $R^2 = .9986$).

The work in [44] describes additional quantum arithmetic circuits that are based

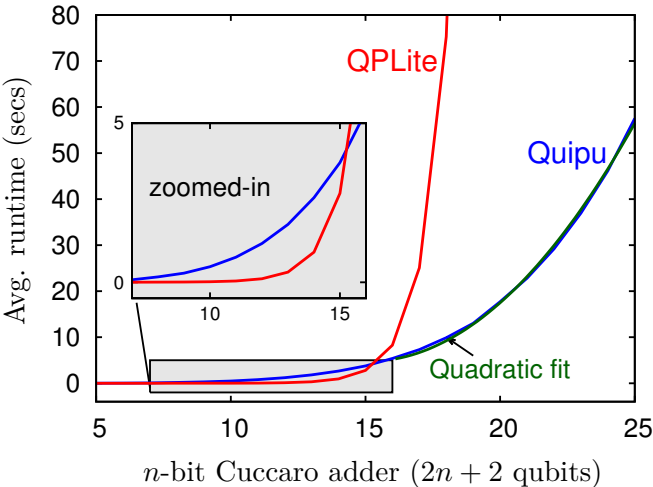


Figure 5.8: Average runtime and memory needed by `Quipu` and `QuIDDPro` to simulate n -bit Cuccaro adders after a superposition of all computational-basis states is obtained using a block of Hadamard gates (Figure 5.7). The quadratic function $f(x) = 0.5248x^2 - 15.815x + 123.86$ fits `Quipu`'s curve with $R^2 = .9986$.

on Cuccaro adders (e.g., subtractors, conditional adders, comparators). We used `Quipu` to simulate such circuits and observed similar runtime performance as that shown in Figure 5.8.

5.3.3 Quantum Fourier transform (QFT) circuits

Our third benchmark set consists of circuits that implement the n -qubit QFT, which computes the discrete Fourier transform of the amplitudes in the input quantum state. Let $|x_1x_2\dots x_n\rangle$, $x_i \in \{0,1\}$ be a computational-basis state and $\mathbf{x}_{1,2,\dots,m} = \sum_{k=1}^m x_k 2^{-k}$. The action of the QFT on this input state can be expressed as:

$$|x_1\dots x_n\rangle = \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2i\pi \cdot \mathbf{x}_n} |1\rangle \right) \otimes \left(|0\rangle + e^{2i\pi \cdot \mathbf{x}_{n-1,n}} |1\rangle \right) \otimes \dots \otimes \left(|0\rangle + e^{2i\pi \cdot \mathbf{x}_{1,2,\dots,n}} |1\rangle \right) \quad (5.7)$$

The QFT is used in many quantum algorithms, notably Shor’s factoring and discrete logarithm algorithms. Such circuits are composed of a network of Hadamard and controlled- $R(\alpha)$ gates, where $\alpha = \pi/2^k$ and k is the distance over which the gate acts. The three-qubit QFT circuit is shown in Figure 5.9. Figure 5.10 shows average runtime and memory usage for both `Quipu` and `QPLite` on QFT instances for $n = \{10, 12, \dots, 20\}$. `Quipu` runs approximately 10× faster than `QPLite` on average and consumes about 96% less memory. For these benchmarks, we observed that the number of states in our multiframe data structure was 2^{n-1} . This is because controlled- $R(\alpha)$ gates produce biased superpositions (Section 5.1.2) that cannot be effectively compressed using our coalescing procedure. Therefore, as Figure 5.10 shows, the runtime and memory requirements of both `Quipu` and `QPLite` grow exponentially in n for QFT instances. However, `Quipu` scales to 24-qubit instances whereas `QPLite` scales to only 18 qubits. The multithreaded implementation of `Quipu` exhibited a 2× speedup and used a comparable amount of memory on a four-core Xeon server.

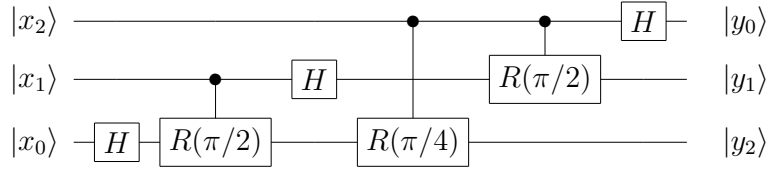


Figure 5.9: The three-qubit QFT circuit. In general, The first qubit requires one Hadamard gate, the next qubit requires a Hadamard and a controlled- $R(\alpha)$ gate, and each following qubit requires an additional controlled- $R(\alpha)$ gate. Summing up the number of gates gives $O(n^2)$ for an n -qubit QFT circuit.

We compared `Quipu` to a straightforward implementation of the state-vector model using an array of complex amplitudes. Such non-compact data structures can be streamlined to simulate most quantum gates (including Hadamard and controlled- $R(\alpha)$ gates) with limited runtime overhead, but scale to only around 30 qubits due to poor memory scaling. Our results showed that `Quipu` was approximately $3\times$ slower than an array-based implementation when simulating QFT instances. However, such implementations cannot take advantage of circuit structure and, unlike `Quipu` and `QPLite`, do not scale to instances of stabilizer and arithmetic circuits with > 30 qubits (see Figures 5.6 and 5.8).

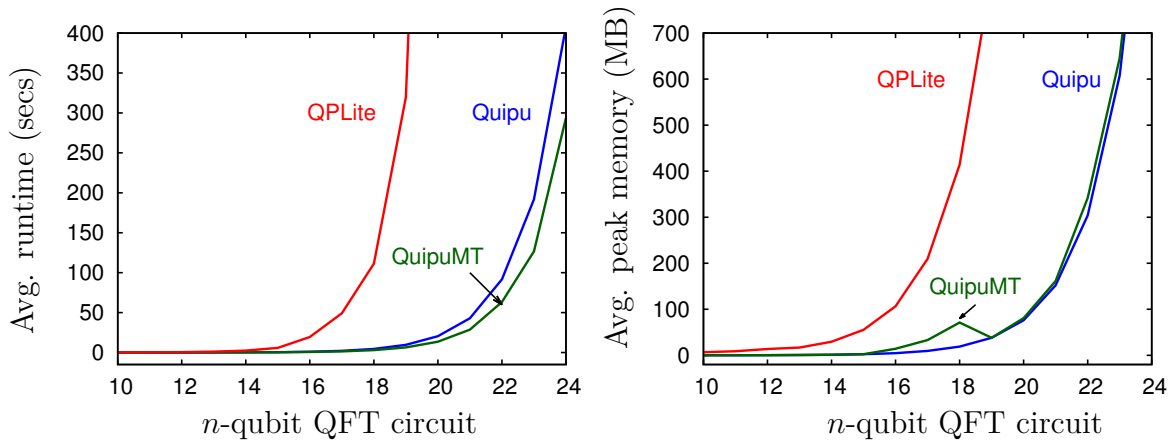


Figure 5.10: Average runtime and memory needed by `Quipu` (single-threaded and multi-threaded) and `QuIDDPro` to simulate n -qubit QFT circuits, which contain $n(n + 1)/2$ gates. We used the $|11\dots 1\rangle$ input state for all benchmarks.

5.3.4 Fault-tolerant (FT) circuits

Our last benchmark set consists of circuits that, in addition to preparing encoded quantum states, implement procedures for performing FT quantum operations (Section 1.4). One constructs FT stabilizer circuits by executing each stabilizer gate transversally across QECC-registers [34, 51, 56] as shown in Figure 1.4. Non-stabilizer gates need to be implemented using a FT architecture that often requires additional ancilla qubits, measurements and correction procedures conditioned on measurement outcomes. Figure 5.11 shows a circuit that implements a FT-Toffoli operation [56]. The presence of adaptable operations (gates conditioned on measurement outcomes) in such circuits implies that one needs to perform weak simulation (Section 2.1.1) of quantum fault-tolerant architectures. (Recall from Figure 2.6 that strong simulation of adaptable circuits is $\#P$ -hard as proven in [40, Theorem 2].) Quipu simulates adaptable gates by flipping a coin to decide a measurement outcome and then applying the corresponding circuit if the condition evaluates to true. Therefore, such simulation outputs a sample of the overall output probability distribution for the circuit. We note that weak simulation of circuits is sufficient for fault-tolerance threshold and reliability analysis.

We implemented a benchmark (listed as *toffoli* in Table 5.1) based on the circuit from Figure 5.11. Each line in Figure 5.11 represents a 5-qubit register implementing the DiVincenzo/Shor² code. We implemented FT benchmarks for the half-adder and full-adder circuits (Figure 5.12) as well as for computing $f(x) = b^x \bmod 15$. Each circuit from Figure 5.13 implements $f(x)$ with a particular co-prime base value b as a (2, 4) look-up table (LUT).³ The Toffoli gates in all our FT benchmarks are implemented using the FT architecture from Figure 5.11. Since FT-Toffoli operations

²The DiVincenzo/Shor code has been shown to function successfully in the presence of both bit-flip and phase-flip errors even if they occur during correction procedures [16].

³A (k, m) -LUT takes k read-only input bits and $m > \log_2 k$ ancilla bits. For each 2^k input combination, an LUT produces a pre-determined m -bit value, e.g., a (2, 4)-LUT is defined by values (1, 2, 4, 8) or (1, 4, 1, 4).

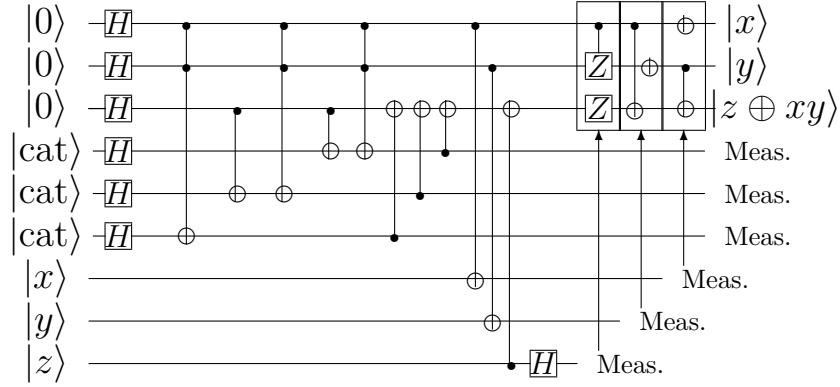


Figure 5.11: Fault-tolerant implementation of a Toffoli gate. Each line represents a 5-qubit register and each gate is applied transversally. The state $|cat\rangle = (|0^{\otimes 5}\rangle + |1^{\otimes 5}\rangle)/\sqrt{2}$ is obtained using a stabilizer subcircuit (not shown). The arrows point to the set of gates that is applied if the measurement outcome is 1; no action is taken otherwise. Controlled- z gates are implemented as $H_j CNOT_{i,j} H_j$ with control i and target j . z gates are implemented as P^2 .

require 6 ancilla registers, a circuit that implements t FT-Toffolis using a k -qubit QECC, requires $6tk$ ancilla qubits. Therefore, to compare with QPLite, we used the 3-qubit bit-flip code [51, Ch. 10] instead of the more robust 5-qubit code in our larger benchmarks. Our results in Table 5.1 show that Quipu is typically faster than QPLite by several orders of magnitude and consumes $8\times$ less memory for the *toffoli*, *half-adder* and *full-adder* benchmarks. Table 5.1 also shows that our coalescing technique is very effective as the maximum size of the stabilizer-state superposition is much smaller when multiple frames are used.

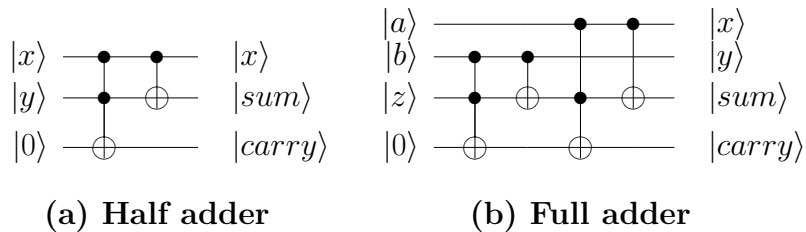


Figure 5.12: Adder circuits from our benchmarks. We used the 5-qubit DiVincenzo/Shor QECC and implemented Toffoli gates using the FT architecture from Figure 5.11.

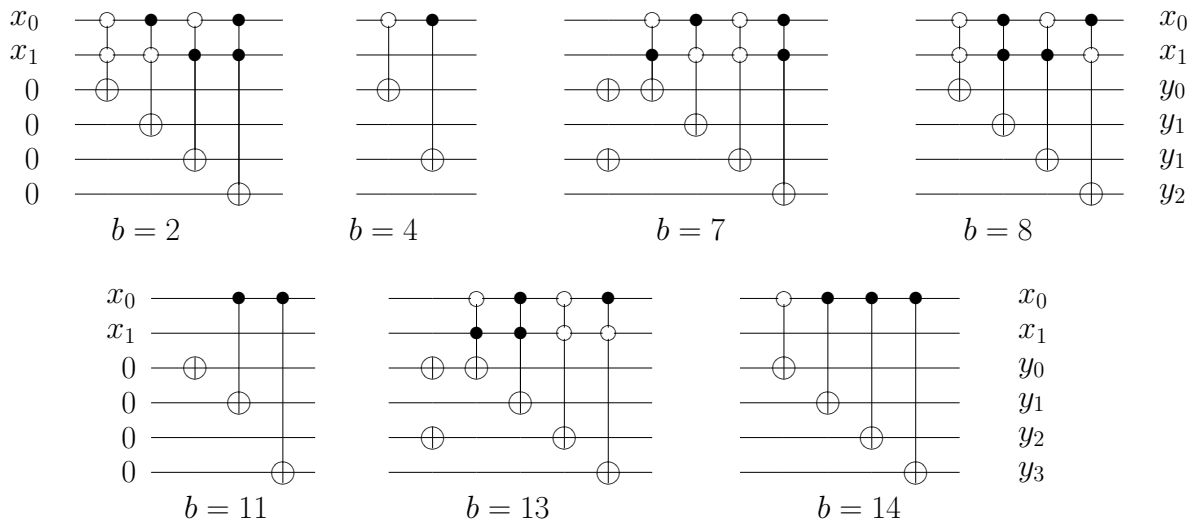


Figure 5.13: Mod-exp with $M = 15$ implemented as $(2, 4)$ -LUTs [44] for several co-prime base values. Negative controls are shown with hollow circles. We apply Hadamards to each x -qubit to generate a superposition of all the input values for x . Our benchmarks implement these computations using the 3-qubit bit-flip code [51, Ch. 10] and the FT-Toffoli architecture from Figure 5.11.

Table 5.1: Average time and memory needed by **Quipu** and **QPLite** to simulate our benchmark set of quantum FT circuits. The second column indicates the QECC used to encode k logical qubits into n physical qubits. We used the 3-qubit bit-flip code for larger benchmarks and the 5-qubit DiVincenzo/Shor code [16] for smaller ones (*). The third column shows the total number of qubits including ancillas required to implement FT-Toffoli gates. We used the $|00\dots 0\rangle$ input state for all benchmarks. The multithreaded version of **Quipu** exhibited similar runtime and memory requirements for these benchmarks since the total number of states observed is relatively small.

FAULT-TOLERANT CIRCUIT	QECC [n, k]	TOTAL QUBITS (INC. ANCILLA)	NUM. OF GATES		RUNTIME (SECS)		MEMORY (MB)		MAX SIZE(Ψ)	
			STAB.	TOFF.	QPLite	Quipu	QPLite	Quipu	SINGLE \mathcal{F}	MULTI \mathcal{F}
toffoli*	[15, 3]	45	155	15	43.68	0.20	98.45	12.76	2816	32
halfadd*	[15, 3]	45	160	15	43.80	0.20	94.82	12.76	2816	32
fulladd*	[20, 4]	80	320	30	84.96	0.88	91.86	12.94	2816	32
$2^x \bmod 15$	[18, 6]	81	396	36	4.81hrs	1.48	11.85	12.96	22528	64
$4^x \bmod 15^*$	[30, 6]	30	30	0	0.01	< 0.01	6.14	12.01	1	1
$7^x \bmod 15$	[18, 6]	81	402	36	11.25hrs	1.52	12.41	13.29	22528	64
$8^x \bmod 15$	[18, 6]	81	399	36	11.37hrs	1.52	12.48	13.29	22528	64
$11^x \bmod 15^*$	[30, 6]	30	25	0	0.02	< 0.01	6.14	12.01	1	1
$13^x \bmod 15$	[18, 6]	81	399	36	11.28hrs	1.56	11.85	12.25	22528	64
$14^x \bmod 15^*$	[30, 6]	30	40	0	0.02	< 0.01	6.14	12.01	1	1

5.4 p -blocked Multiframes

The quantum Fourier transform (QFT) experiments presented in Section 5.3 showed that the resources required to simulate such circuits using multiframes grew exponentially in the number of qubits. In this section, we present a modification to our multiframe data structure that facilitates simulation of QFT circuits in linear time and space. We call this new data structure a p -blocked multiframe since it is based on the approach proposed in [41], which separates a quantum state into p -blocks that do not grow exponentially in size. In this technique, states are termed p -blocked if no subset of $p + 1$ qubits is entangled. More generally, the set of all qubits in state $|\Psi\rangle$ is partitioned into k blocks B_1, B_2, \dots, B_k such that

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle \quad (5.8)$$

where each $|\psi_i\rangle$ is the state vector for block B_i . Since each block requires at least 2^p coefficients, the space complexity grows with the number of entangled qubits. Therefore, when p is small (i.e., k is large), the complexity of simulation may be reduced by decomposing the state into a tensor product of p -blocks. In contrast, when $k = 1$, no advantage is obtained using this technique. Operators that affect qubits within a single block B_i can be applied directly to $|\psi_i\rangle$. However, when simulating operators that extend across several blocks, one needs to combine all affected blocks via the tensor product into one large block. Then, the block is decomposed or *factored* into separate blocks if possible.

In our p -blocking technique, we represent each $|\psi_i\rangle$ in Equation 5.8 using a multiframe \mathcal{V}_i . Therefore, an n -qubit quantum state is represented as,

$$|\Psi\rangle = \mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \cdots \otimes \mathcal{V}_k = \left(\sum_i \alpha_i |\psi_i\rangle \right) \otimes \left(\sum_j \beta_j |\psi_j\rangle \right) \otimes \cdots \otimes \left(\sum_l \gamma_l |\psi_l\rangle \right) \quad (5.9)$$

where each $|\psi_i\rangle$, $|\psi_j\rangle$ and $|\psi_l\rangle$ is a stabilizer state. Let $\text{SIZE}(\mathcal{V}_i)$ denote the number of qubits in \mathcal{V}_i . Observe that $\sum_{i=1}^k \text{SIZE}(\mathcal{V}_i) = n$. Algorithm 5.4.1 describes our process for simulating gate operators using p -blocked multiframes.

Example V.8. The stabilizer frame obtained by simulating the 4-qubit QFT circuit on input state $|1111\rangle$ (Figure 5.14a) can be decomposed or factored into the tensor product of 4 single-qubit operations (Equation 5.7). Therefore, such a state can be represented using the p -blocked multiframe from Figure 5.14b.

As in the general p -blocking scheme from [41], operators that span a single multiframe \mathcal{V}_i are applied directly to the multiframe using the algorithms described in Section 5.2. Suppose we seek to simulate an operator that spans several multiframes. First, relevant multiframes are tensored together into a single t -qubit frame \mathcal{F} , where $t \leq n$ equals the sum of the sizes of the tensored multiframes. Then, we analyze the stabilizer matrix associated with \mathcal{F} to determine potential entanglement partitions for the frame (i.e., we identify separable sub-matrices). This is accomplished in $O(t^2)$ -time by examining the non- I literals in the rows of the canonical matrix. For each

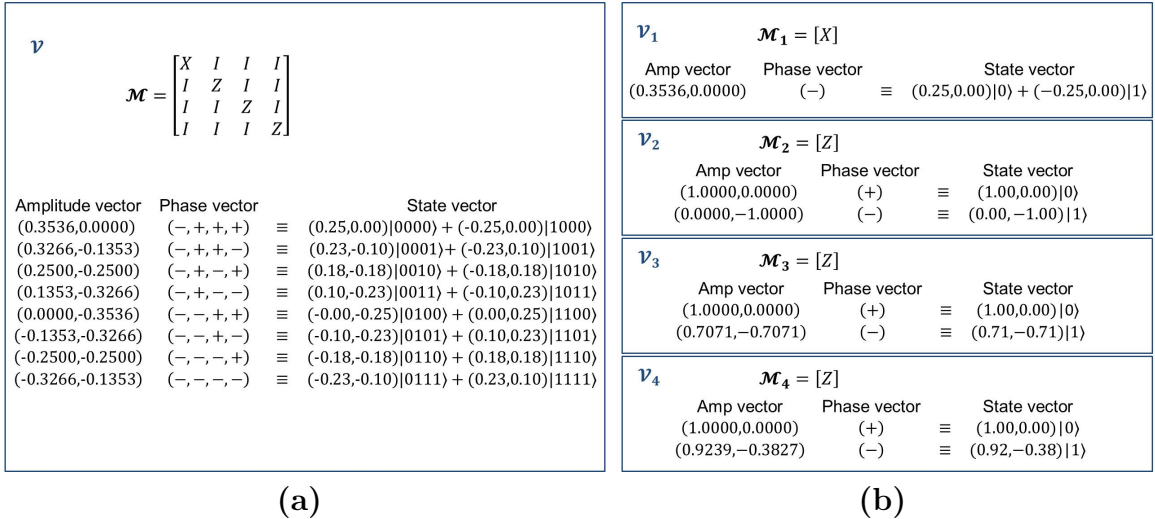


Figure 5.14: (a) Stabilizer frame (single-term multiframe) representation for $U|1111\rangle$, where U is the 4-qubit QFT circuit. (b) p -blocked multiframe representation for $U|1111\rangle$. Observe that $\mathcal{V} = \mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \mathcal{V}_3 \otimes \mathcal{V}_4$.

candidate partition, we apply a decomposition procedure that attempts to partition the phase vector-amplitude pairs of \mathcal{F} into a set of smaller multiframe. This set of multiframe is ordered⁴ such that their tensor product yields the original \mathcal{F} . If the decomposition is unsuccessful, \mathcal{F} is returned as a single-term multiframe. The worst-case runtime for decomposing the resulting frame is $t|\mathcal{F}| = O(t2^t)$. Therefore, the decomposition portion of Algorithm 5.4.1 should be executed only when we expect $|\mathcal{F}|$ to be relatively small.

Improved simulation of QFT circuits

We implemented p -blocked multiframe as an optional data structure in `Quipu` and tested our technique by simulating n -qubit QFT instances. Each controlled- $R(\alpha)$ gate in a QFT circuit is simulated using the approach outlined in Algorithm 5.4.1. First, the blocks corresponding to the input qubits of a controlled- $R(\alpha)$ gate are tensored to form a two-qubit multiframe with a single term. Then, the resulting frame \mathcal{F} is decomposed successfully into a tensor product of two one-qubit frames. Since $|\mathcal{F}| \leq 4$ always, the overall runtime of our technique is approximately linear in n (Figure 5.15).

Figure 5.15 shows that the decomposition technique used Algorithm 5.4.1 is effective when simulating QFT circuits. However, there is no guarantee that our decomposition approach yields the most compact p -blocking representation in general. One can design more sophisticated decomposition techniques that exploit state separability even further at the cost of increased runtime complexity. One approach is to generalize our technique by utilizing a modified *Schmidt decomposition* procedure specific to multiframe. In Section 8.3, we discuss future research along these lines.

⁴We keep track of such ordering separate data structure consisting of multiframe pointers. The pointers are updated during the decomposition process to maintain the correct order.

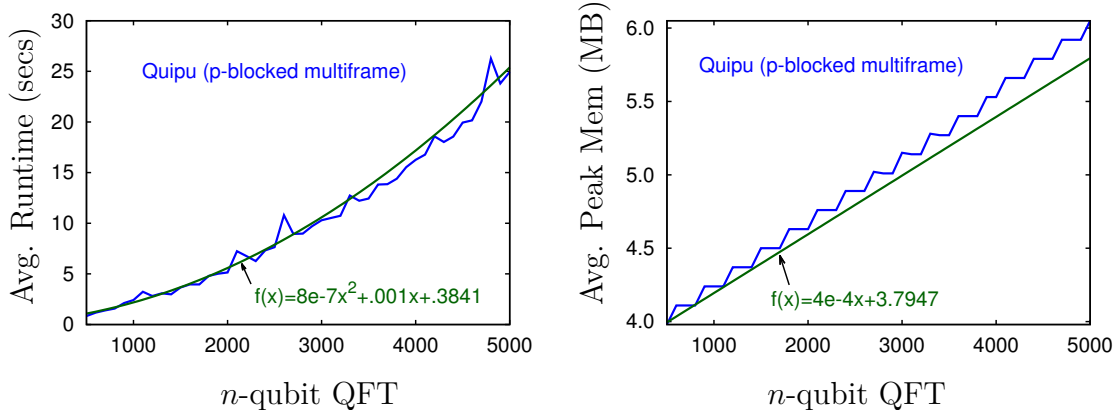


Figure 5.15: Average runtime and memory for Quipu to simulate n -qubit QFT circuits on input state $|11 \dots 1\rangle$ using p -blocked multiframes. The poly-fit functions for runtime and memory have $R^2 = .9886$ and $R^2 = .9964$, respectively.

5.5 Summary

In this chapter, we develop new techniques for quantum-circuit simulation based on superpositions of stabilizer states, avoiding shortcomings in prior work [2]. To represent such superpositions compactly, we design a new data structure called a *stabilizer frame*. We implemented stabilizer frames and relevant algorithms in our software package `Quipu`. Current simulators based on the stabilizer formalism, such as `CHP`, are limited to simulation of stabilizer circuits. Our results show that `Quipu` performs asymptotically as fast as `CHP` on stabilizer circuits with a linear number of measurement gates, but simulates certain quantum arithmetic circuits in polynomial time and space for input states consisting of equal superpositions of computational-basis states. In contrast, `QuIDDPro` takes exponential time on such instances. We simulate quantum Fourier transform (QFT) and quantum fault-tolerant circuits with `Quipu`, and the results demonstrate that our stabilizer-based technique leads to orders-of-magnitude improvement in runtime and memory as compared to `QuIDDPro`. While our technique uses more sophisticated mathematics and quantum-state modeling, it is significantly easier to implement and optimize. In particular, our multithreaded

implementation of `Quipu` exhibited a $2\times$ speed up on a four-core server.

To improve the performance of `Quipu` on QFT circuits, we design a tensor-product frame representation, called p -blocked multiframe. We develop an algorithm to decompose an arbitrary frame into a tensor-product of smaller multiframe (p -blocking). This decomposition technique facilitates simulation of QFT circuits in linear time and space using `Quipu` for computational-basis input states.

Algorithm 5.4.1 p -blocked multiframe simulation

Require: (i) p -blocked multiframe $\mathbf{V}_{in} = \mathcal{V}_1 \otimes \mathcal{V}_2 \otimes \cdots \otimes \mathcal{V}_k$ representing state $|\Psi\rangle$, (ii) m -qubit operator U , and (iii) indices i_1, i_2, \dots, i_m of acting qubits for U

Ensure: Updated p -blocked multiframe \mathbf{V}_{out} representing $U|\Psi\rangle$

- \Rightarrow BLOCKINDEX(i_1, i_2, \dots, i_m) returns indices b_1, b_2, \dots, b_m denoting corresponding block indices
- \Rightarrow FINDBLOCKS(\mathcal{F}) analyzes the stabilizer matrix \mathcal{M} associated with frame \mathcal{F} and returns ordered indices p_1, p_2, \dots, p_r denoting the first qubit of each entanglement partition
- \Rightarrow TENSOR(j_1, j_2, \dots, j_t) returns the frame representing the tensor product $\mathcal{V}_{j_1} \otimes \mathcal{V}_{j_2} \otimes \cdots \otimes \mathcal{V}_{j_t}$
- \Rightarrow INSERT(\mathbf{V}, \mathcal{V}) inserts \mathcal{V} into the ordered list of multiframes \mathbf{V}
- \Rightarrow INSERT($\mathcal{V}, \mathcal{M}, \sigma, c$) finds frame \mathcal{F} in multiframe \mathcal{V} with stabilizer matrix \mathcal{M} (if no such frame exists, \mathcal{F} is created and appended to \mathcal{V}) and inserts phase vector-amplitude pair (σ, c) into \mathcal{F}
- \Rightarrow GETAMP($\mathcal{V}, \mathcal{M}, \sigma$) finds frame \mathcal{F} in multiframe \mathcal{V} with stabilizer matrix \mathcal{M} and returns the amplitude c that forms a pair with phase vector σ ; returns 0 if no such phase vector exists
- \Rightarrow FACTOR(\mathcal{M}, p) factors the stabilizer matrix \mathcal{M} along qubit p creating a pair of smaller matrices \mathcal{M}_{left} and \mathcal{M}_{right} such that $\mathcal{M} = \mathcal{M}_{left} \otimes \mathcal{M}_{right}$

```

1:  $b_1, b_2, \dots, b_m \leftarrow$  BLOCKINDEX( $i_1, i_2, \dots, i_m$ )
2: if  $b_1, b_2, \dots, b_m$  are all equal then
3:   APPLY( $\mathcal{V}_{b_1}, U$ ) ▷ Simulate  $U$  via algorithms from Section 5.2
4:   return  $\mathbf{V}_{in}$ 
5: end if
6:  $\mathcal{F} \leftarrow$  TENSOR( $b_1, b_2, \dots, b_m$ )
7: APPLY( $\mathcal{F}, U$ )
8:  $p_1, p_2, \dots, p_r \leftarrow$  FINDBLOCKS( $\mathcal{F}$ )
9:  $\mathbf{V}_{out} \leftarrow \emptyset$ 
10: for  $p_l, l \in \{1, 2, \dots, r\}$  do
11:    $d \leftarrow 1$  ▷ Whether the decomposition is successful
12:    $\mathcal{V}_{left} \leftarrow \emptyset; \mathcal{V}_{right} \leftarrow \emptyset$ 
13:   for each phase vector-amplitude pair  $(\sigma_i, a_i), i \in \{1, \dots, |\mathcal{F}|\}$  do
14:     for each  $(\sigma_j, a_j), j \in \{i, \dots, |\mathcal{F}|\}$  do
15:        $\mathcal{M}^{\sigma_j} \leftarrow$  assign  $\sigma_j$  as the phase vector of the stabilizer matrix for  $\mathcal{F}$ 
16:        $[\mathcal{M}_{left}, \mathcal{M}_{right}] \leftarrow$  FACTOR( $\mathcal{M}^{\sigma_j}, p_l$ )
17:       if  $i = 1$  then ▷ Use first iteration to setup  $\mathcal{V}_{right}$ 
18:         INSERT( $\mathcal{V}_{right}, \mathcal{M}_{right}, \sigma_j, a_j/a_i$ )
19:       else ▷ Otherwise, check partitions are consistent
20:          $c \leftarrow$  GETAMP( $\mathcal{V}_{right}, \mathcal{M}_{right}, \sigma_j$ )
21:         if  $a_j \neq c \cdot a_i$  then ▷ Inconsistent partitions
22:            $d \leftarrow 0$ 
23:           break
24:         end if
25:       end if
26:     end for
27:     if  $d = 1$  then
28:       INSERT( $\mathcal{V}_{left}, \mathcal{M}_{left}, \sigma_i, a_i$ )
29:     else
30:       break
31:     end if
32:   end for
33:   if  $d = 1$  then
34:     INSERT( $\mathbf{V}_{out}, \mathcal{V}_{left}$ ); INSERT( $\mathbf{V}_{out}, \mathcal{V}_{right}$ ) ▷ Decomposed successfully
35:   else
36:     INSERT( $\mathbf{V}_{out}, \mathcal{F}$ ) ▷ Frames are single-term multiframes
37:   end if
38: end for
39: return  $\mathbf{V}_{out}$ 

```

CHAPTER VI

Stateless Simulation of Generic Quantum Circuits

In Section 2.4 we reviewed the stateless simulation approach for stabilizer circuits developed in [8, 11, 39], and compared it to direct simulation using stabilizer generators. In this chapter, we describe a generalization of the approach described in Section 2.4 that facilitates stateless simulation of generic quantum circuits and compare its performance to stabilizer frames (Chapter V). Furthermore, we explore compact data structures for generic stateless simulation based on multivalued decision diagrams.

6.1 Pauli Expansions of Linear Operators

Consider the Pauli expansion $f_p(U) = \sum_j \alpha_j P_j$ of a non-stabilizer gate U obtained using the operator function from Definition I.17. Such decompositions facilitate stateless simulation of quantum circuits that contain non-stabilizer operators.

Multi-controlled Z and X operators

We denote multi-controlled operators as $C^k(U)$, where k is the number of control qubits and U is the l -qubit operator applied when the control condition is satisfied. Thus $C^k(U)$ acts on $k + l$ qubits. For the case U is the one-qubit Z Pauli matrix (multi-controlled Z), $C^k(Z)$ is diagonal. Therefore, the terms in the Pauli expansion

of $C^k(Z)$ are composed of Z and I Pauli literals only. As an example, consider the case $k = 2$. Using the operator function from Definition I.17 we have,

$$f_p(C^2(Z)) = \frac{3 \cdot III + IIZ + IZI - IZZ + ZII - ZIZ - ZZI + ZZZ}{4} \quad (6.1)$$

To obtain the Pauli expansion of any operator $C^k(Z)$ without computing f_p we formulate a general form as follows. Let b_i denote the binary representation of integer i . We use b_i to construct a Pauli string Z_i as follows: if bit j in b_i is set, set the j^{th} literal of Z_i to Z ; otherwise the literal is set to I . For example, $Z_0 = II$, $Z_1 = IZ$, $Z_2 = ZI$, $Z_3 = ZZ$, etc. Let $h(Z_i)$ be the hamming weight with respect to Z_0 . The Pauli expansion of $C^k(Z)$ is defined as,

$$C^k(Z) = \frac{1}{2^{(k+1)}} \left(I^{\otimes(k+1)} + \sum_{i=1}^{2^{k+1}-1} (-1)^{h(Z_i)} Z_i \right) \quad (6.2)$$

Observe that Equation 6.2 can also be used to derive the Pauli expansion of $C^k(X)$ using the relation $HZH^\dagger = X$. Simply conjugate each term in Equation 6.1 by $I^{\otimes k}H$. For example in the case the *Toffoli gate* $C^2(X)$ we get,

$$f_p(C^2(X)) = \frac{3 \cdot III + IIX + IZI - IZX + ZII - ZIX - ZZI + ZZX}{4} \quad (6.3)$$

where the only difference from Equation 6.2 is the Pauli strings have an X literal instead of a Z in the last position.

Addition operators

It turns out that a recursive general form exists for deriving the Pauli expansion of a linear operator that adds two n -bit numbers $\mathbf{a} = a_1a_2 \dots a_n$ and $\mathbf{b} = b_1b_2 \dots b_n$. To simplify notation, let \mathbf{ab} denote the binary string $a_1b_1a_2b_2 \dots a_nb_2$. The n -qubit addition operator ADD_n maps a computational basis state $|\mathbf{ab}\rangle$ to another computa-

tional basis state $|\mathbf{ac}\rangle$, where \mathbf{c} is the n -bit result of the addition (ignoring the carry bit). Formally, we define the addition operator as,

$$ADD_n = \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} |\mathbf{a}_i \mathbf{b}_j\rangle \langle \mathbf{a}_i \mathbf{c}_{i+j}| \quad (6.4)$$

We now derive a recursive general form for the Pauli expansion of ADD_n . Our base case is the 1-bit adder, which is just the Pauli expansion of a CNOT. Let $A_1 = II + XI$ and $B_1 = IZ - XZ$, then $ADD_1 = A_1 + B_1$. For an i -bit quantum adder as defined by Equation 6.4 we have,

$$\begin{aligned} A_i &= II \otimes A_{i-1} + IZ \otimes B_{i-1} + XI \otimes A_{i-1} + iYI \otimes B_{i-1} \\ B_i &= II \otimes B_{i-1} + IZ \otimes A_{i-1} - XZ \otimes A_{i-1} - iYZ \otimes B_{i-1} \\ ADD_i &= II \otimes (A_i + B_i) + IZ \otimes (A_i + B_i) + XI \otimes A_i \\ &\quad - XZ \otimes A_i + iYI \otimes B_i - iYZ \otimes B_i \end{aligned} \quad (6.5)$$

6.2 Stateless Simulation of Generic Quantum Circuits

Let $\mathbf{E}(Z_k^f)$ denote the expectation value of performing a Z -measurement on qubit k in the final state $\mathbf{C}|\psi_0\rangle$, where \mathbf{C} is a generic n -qubit quantum circuit and $|\psi_0\rangle = |s_1\rangle \otimes |s_2\rangle \cdots |s_n\rangle$ is the initial product state.

1. Initialize the Z_k -measurement operator.
2. Conjugate Z_k by each gate U in circuit \mathbf{C} in reverse order. Without loss of generality, assume U is the first gate in \mathbf{C} .
 - If U is a Pauli or stabilizer gate, then $U^\dagger Z_k U = Q \in \mathcal{P}_n$.

- If U is a non-stabilizer gate, then

$$U^\dagger Z_k U = [f_p(U)]^\dagger Z_k [f_p(U)] = \left[\sum_i \alpha_i P_i \right]^\dagger Z_k \left[\sum_j \alpha_j P_j \right] = \sum_{i,j} \alpha_i^* \alpha_j Q_{i,j}$$

where $Q_{i,j} \in \mathcal{P}_n$.

3. Let $\mathbf{C}^\dagger Z_k \mathbf{C} = \sum_{i,j} \alpha_i^* \alpha_j Q_{i,j}$,

$$\mathbf{E}(Z_k^f) = \langle \psi_0 | \left(\sum_{i,j} \alpha_i^* \alpha_j Q_{i,j} \right) | \psi_0 \rangle = \sum_{i,j} \alpha_i^* \alpha_j \prod_{j=1}^n \langle s_j | Q_{i,j} | s_j \rangle \quad (6.6)$$

Observe that each term in the sum above is a variant of Equation 2.20 and thus can be computed in $O(n)$ time as outlined in Section 2.4.

Example VI.1. Consider the Pauli expansion of the non-stabilizer gate $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$,

$$f_p(T) = \left(\frac{1}{2} + \frac{1+i}{2\sqrt{2}} \right) I + \left(\frac{1}{2} - \frac{1+i}{2\sqrt{2}} \right) Z = \alpha_1 I + \alpha_2 Z \quad (6.7)$$

Suppose we want to simulate circuit $\mathbf{C} = HTH$ on initial state $|0\rangle$. We compute the outcome probabilities as follows,

$$\begin{aligned} \mathbf{E}(Z_1^f) &= \langle 0 | (\mathbf{C}^\dagger) Z (\mathbf{C}) | 0 \rangle \langle 0 | (HTH) Z (HTH) | 0 \rangle = \langle 0 | (HTH) X (TH) | 0 \rangle \\ &= \langle 0 | H(\alpha_1^* I + \alpha_2^* Z) X (\alpha_1 I + \alpha_2 Z) H | 0 \rangle \\ &= \langle 0 | H(\alpha_1 \alpha_1^* X - \alpha_2 \alpha_2^* X - i \alpha_1^* \alpha_2 Y + i \alpha_1 \alpha_2^* Y) H | 0 \rangle \\ &= \langle 0 | (\alpha_1 \alpha_1^* Z - \alpha_2 \alpha_2^* Z + i \alpha_1^* \alpha_2 Y - i \alpha_1 \alpha_2^* Y) | 0 \rangle = \alpha_1 \alpha_1^* - \alpha_2 \alpha_2^* \end{aligned}$$

Recall from Section 2.4 that $\mathbf{E}(Z_k^f) = p_0 - p_1$. Therefore, $p_0 = \alpha_1 \alpha_1^* = .8536$ and $p_1 = \alpha_2 \alpha_2^* = .1464$.

The above procedure suggests a simple data structure for stateless simulation of generic circuits: maintain a list of Pauli strings and their corresponding coefficients.

One then updates the Pauli strings when stabilizer gates are simulated, and (potentially) expands the list when simulating non-stabilizer gates. Although stateless simulation has the advantage of simplicity, we will show in Section 6.2.2 that it does not scale well. This is because the number of terms in the Pauli expansions of non-stabilizer gates is usually exponential in the number of acting qubits. The scalability of stateless simulation is similar to that of the density-matrix approach outlined in [2], where the authors represent a quantum state as a sum of $O(4^{2dk})$ density-matrix terms while simulating k non-stabilizer operations acting on d distinct qubits. Furthermore, since no state representation is maintained, one cannot design compression techniques similar to stabilizer-frame coalescing (Section 5.2). However, when the Pauli expansion of a non-stabilizer operator can be expressed using a recurrence relation (e.g., Equation 6.5), one can arrive at a compact representation using well-known graph techniques such as decision diagrams. We outline this technique and show examples in the next section.

6.2.1 Compact Representation using Decision Diagrams

Recall from Section 6.1 that n -bit addition operators can be expressed using the recurrence relation ADD_n from Equation 6.5. This suggests that such operators can be represented compactly. Specifically, if one regards Pauli literals as input variables (taking on one of four possible values) to a discrete function, ADD_n can be represented using *multivalued decision diagrams (MDDs)*. An MDD is a directed acyclic graph data structure consisting of internal nodes, terminal nodes and a set of edges connecting them. In contrast to the better-known *binary decision diagrams (BDDs)*, the number of outgoing edges in an internal node is not limited to two. MDDs may be ordered and reduced in a fashion analogous to the binary case and the resulting representation is termed a reduced ordered MDD.

Definition VI.2. Let $\sigma_1 \dots \sigma_n, \sigma_i \in \{I, X, Y, Z\}$ denote a term in the Pauli expansion

of quantum operator U . An n -qubit Pauli *QMDD* is a directed acyclic graph with the following properties:

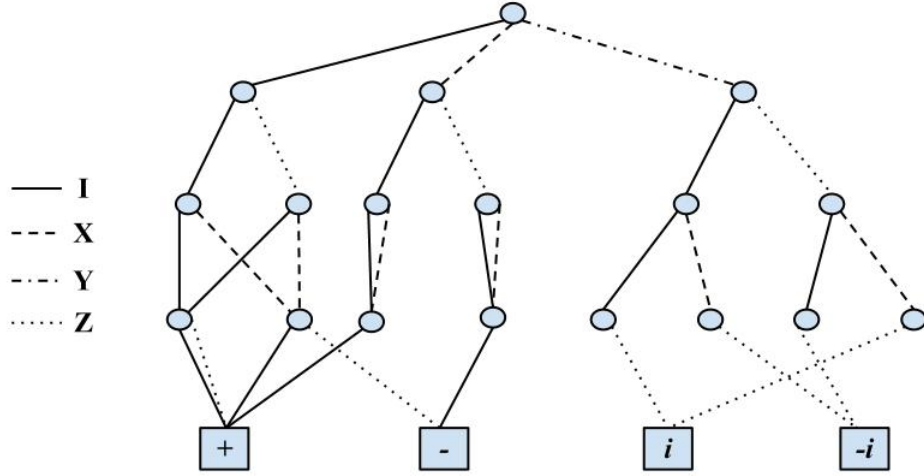
- (i) Each internal node x_i represents the Pauli literal σ_i .
- (ii) The outgoing edges of x_i represent the literal value assigned to σ_i for a particular term in the Pauli expansion.
- (iii) Each edge points downward, implying a top-down assignment of the values of the σ_i depicted by the internal nodes.
- (iv) Terminal nodes represent the possible phases $(\pm 1, \pm i)$ of the expansion terms.
- (v) Each path through a Pauli MDD from top to bottom represents a specific term in the Pauli expansion of U , and ends with the corresponding phase coefficient.

Example VI.3. Figure 6.1 shows the Pauli MDD for ADD_2 (Equation 6.5).

The advantage of using MDDs is that one can derive reduction rules to delete or share redundant nodes depending on the equal assignment of Pauli literals assigned to the internal nodes. An attractive direction for future work is to derive such rules for Pauli MDDs in order to: (i) efficiently represent other quantum arithmetic operators and (ii) design efficient analysis and simulation algorithms that take advantage of this compact representation. Furthermore, it is possible to extend MDDs by allowing arbitrary complex values in edge weights and terminals. Such improvements may lead to additional compression in a Pauli MDD but may also introduce complexities in terms of canonicity and algorithmic manipulation of the data structure.

6.2.2 Empirical Validation

We implemented the stateless technique described in Section 2.4 as a simulation option in *Quipu*. Our benchmark consists of circuits that follow the structure shown in Figure 2.1, where we assume U is a random n -qubit stabilizer circuit. Therefore, each



$$\begin{aligned}
 ADD_2 = & IIII + IIIZ + IIXI - IIXZ + IZII + IZIZ + IZXI - IZXX \\
 & + XIII + XIXI - XZII - XZXI + iYIIZ - iYIXZ - iYZIZ + iYZXZ
 \end{aligned}$$

Figure 6.1: MDD for addition operator ADD_2 . The circular (internal) nodes represent Pauli literals and the edges represent possible values for such literals. The square (terminal) nodes represent the possible phases ($\pm 1, \pm i$) of each term in ADD_2 .

benchmark circuit contains $2n + 1$ stabilizer gates. Figure 6.2 shows average runtime results for simulating such circuits with $n \in \{2000, 2100, \dots, 3500\}$ using **CHP** and the stateless version of **Quipu**. Stateless simulation runs in linear time for such instances and thus outperforms **CHP**, which takes quadratic time (Section 2.3.1). The stateless approach has the advantage of simulating such circuits on any product state while **CHP** is limited to computational-basis input states. However, stateless simulation does not allow multi-qubit measurement or gates conditioned on measurement outcomes. Additional comparisons of these two techniques can be found in Section 2.4.

We also compared the stateless approach to the frame-based approach described in Chapter V. For this comparison, we used the ripple-carry adder benchmarks used in Section 5.3 with a single measurement on the high-bit (qubit $|z\rangle$ in Figure 5.7). To simulate Toffoli gates, we used the Pauli expansion from Equation 6.3. As Figure 6.3a shows, the stateless approach has similar performance to **QuIDDPro** on such instances

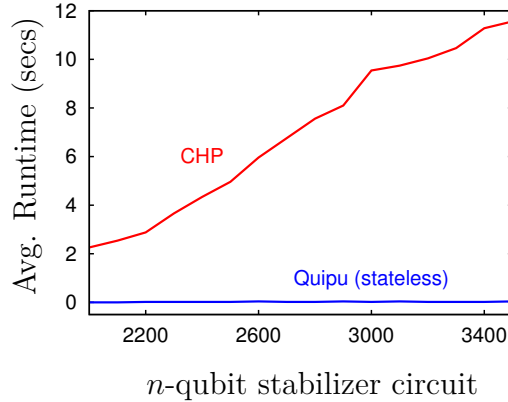
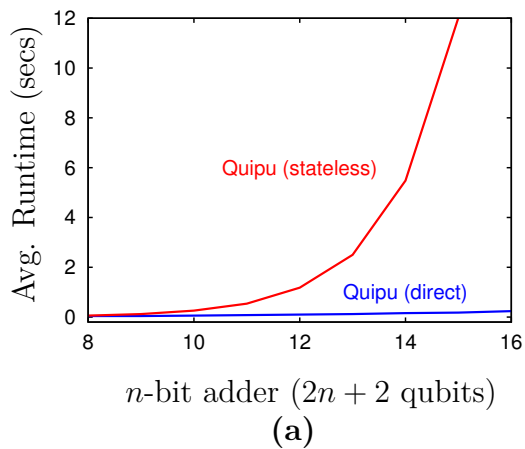


Figure 6.2: Average time needed by `Quipu` (stateless) and `CHP` to simulate an n -qubit stabilizer circuit with $2n + 1$ gates and a single measurement (circuit structure from Figure 2.1). `Quipu` runs in linear time for such instances while `CHP` takes quadratic time.

(Figure 5.7): it scales to 18-bit instances only and its runtime grows exponentially with circuit size. This limited scalability is due to the exponential growth in the size of the list-based data structure as shown in Figure 6.3b. In contrast, frame-based simulation (labeled `Quipu direct` in Figure 6.3) takes poly-time and space.

6.3 Summary

The notion of stateless simulation was introduced in [8, 11, 39] in the context of stabilizer (Clifford) circuits. Using our Pauli expansion techniques for linear operators described in Section 6.1, we generalize the stateless approach to admit simulation of generic quantum circuits. For stabilizer circuits that follow the structure from Figure 2.1, stateless simulation outperforms techniques based on the stabilizer formalism (e.g., `CHP`). Nonetheless, stateless simulation has its drawbacks as outlined in Section 2.4. Our experiments show that, when simulating ripple-carry adder circuits, stateless simulation requires exponential resources. We describe a compact representation for Pauli expansions called Pauli multivalued decision diagrams, which may mitigate the exponential increase in resources observed during stateless simulation.



SIZE OF REPRESENTATION		
n	DIRECT	STATELESS
8	16	766
9	18	1534
10	20	3070
11	22	6142
12	24	12286
13	26	24574
14	28	49150
15	30	98302
16	32	196606

Figure 6.3: (a) Average time needed by Quipu (stateless) and Quipu (direct) to simulate n -bit adders (Section 5.3) after a superposition of all computational-basis states is obtained. A single measurement is performed on qubit $2n + 2$, which is the highest carry bit. The stateless approach takes exponential time and scales to around 18-bit instances only. (b) Comparison of the number of terms in the direct frame-based approach (size of multiframe) and the stateless approach (size of the Pauli-string list).

CHAPTER VII

Graph-based Techniques and Matrix Product States

Chapters V and VI introduced new quantum-circuit simulation techniques that generalize the Heisenberg picture for quantum computers. Specifically, these techniques exploit symmetries in a quantum state to produce compact representations in certain cases. Another well-known technique that achieves a similar objective is the *matrix product state* (MPS) representation [27, 50, 55, 68]. The idea behind MPS is to write the 2^n complex amplitudes $c_{i_1 \dots i_n \in \{0,1\}^n}$ in a quantum state as entries in a large tensor with indices i_1, \dots, i_n . Such a tensor can be stored as a contraction of several smaller tensors – one for each qubit in the state. We describe the MPS representation in Section 7.1 along with empirical results obtained using a recently developed software package called ZKCM. In Section 7.2, we describe *quantum multi-valued decision diagrams* (QMDDs), which are symbolic data structures that can be used to represent certain matrix operators compactly [47]. QMDD algorithms have been developed that exploit the symbolic nature of the representation to improve the performance of quantum simulation [32]. An open research question is whether one can improve the performance of MPS simulation by designing graph-based techniques similar to QMDDs that directly model MPS. To this end, we generalize certain properties of QMDDs and derive MPS-based decision diagrams in Section 7.3.

7.1 Matrix Product State (MPS) Simulation

A matrix product state (MPS) is a weakly-entangled n -qubit quantum state,

$$|\psi\rangle = \sum_{i_1 \dots i_n \in \{0,1\}^n} c_{i_1 \dots i_n} |i_1\rangle \otimes \dots \otimes |i_n\rangle \quad (7.1)$$

that can be represented by a sum of product states. This approach is similar to our p -blocking technique for multiframe described in Section 5.4. Following the formulation from [68], let A denote a subset of the n qubits and B the rest of the qubits. The Schmidt decomposition [51, Section 2.5] of $|\psi\rangle$ with respect to partition $A:B$ is,

$$|\psi\rangle = \sum_{\alpha=1}^{\chi_A} \lambda_{\alpha} |\phi_{\alpha}^{[A]}\rangle \otimes |\phi_{\alpha}^{[B]}\rangle \quad (7.2)$$

where the vectors $|\phi_{\alpha}^{[A]}\rangle$ and $|\phi_{\alpha}^{[B]}\rangle$ are eigenvectors with eigenvalues $|\lambda_{\alpha}|^2 > 0$ of the reduced density matrices $\rho^{[A]}$ and $\rho^{[B]}$, respectively. The *local Schmidt rank* χ_A is a measure of entanglement between partitions A and B . The entanglement $|\psi\rangle$ can be quantified by the maximum χ_A over all possible bipartite partitions $A:B$, $\chi \equiv \max_A \chi_A$ [68, Equation 2]. Depending on the amount of entanglement, χ can range from 1 for fully separable states, to 2^n for fully entangled states.

The key idea driving MPS simulation is the local decomposition of the complex coefficients $c_{i_1 \dots i_n}$ from Equation 7.1 in terms of n tensors¹ $A^{[l]}$ and $n - 1$ vectors $v^{[l]}$,

$$c_{i_1 i_2 \dots i_n} = \sum_{\alpha_1, \dots, \alpha_{n-1}} A_{\alpha_1}^{[1]i_1} v_{\alpha_1}^{[1]} A_{\alpha_1 \alpha_2}^{[2]i_2} v_{\alpha_2}^{[2]} A_{\alpha_2 \alpha_3}^{[3]i_3} \dots A_{\alpha_{n-1}}^{[n]i_n} \quad (7.3)$$

where each α runs from 1 to χ_A . Therefore, the 2^n coefficients $c_{i_1 \dots i_n}$ are expressed in terms of roughly $(2\chi^2 + \chi)n$ parameters, which grows linearly in n for a fixed value χ . Observe that Equation 7.3 can be viewed as a concatenation of $n - 1$ local Schmidt decompositions, and defines a *canonical form* for MPS [55]. The $A^{[l]}$ tensors

¹A k -tensor can be viewed as a k -dimensional array.

correspond to a changes of basis between the different Schmidt bases (one for each qubit l) and the computational basis, and the vectors $v^{[l]}$ correspond to the Schmidt coefficients. The overall decomposition depends on the particular way qubits are ordered from 1 to n . We refer to [68] for a detailed explanation on how to obtain this decomposition.

Example VII.1. Consider the entangled state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$. Here $c_{00} = c_{01} = c_{10} = 1/2$ and $c_{11} = -1/2$. The canonical matrix product representation for $|\psi\rangle$ consists of two matrices $(A^{[1]}, A^{[2]})$ and one Schmidt vector $(v^{[1]})$ such that:

$$\begin{aligned} c_{00} &= \sum_{\alpha_1=1}^2 A_{\alpha_1}^{[1]0} v_{\alpha_1}^{[1]} A_{\alpha_1}^{[2]0} \approx \frac{1}{2} \\ c_{01} &= \sum_{\alpha_1=1}^2 A_{\alpha_1}^{[1]0} v_{\alpha_1}^{[1]} A_{\alpha_1}^{[2]1} \approx \frac{1}{2} \\ c_{10} &= \sum_{\alpha_1=1}^2 A_{\alpha_1}^{[1]1} v_{\alpha_1}^{[1]} A_{\alpha_1}^{[2]0} \approx \frac{1}{2} \\ c_{11} &= \sum_{\alpha_1=1}^2 A_{\alpha_1}^{[1]1} v_{\alpha_1}^{[1]} A_{\alpha_1}^{[2]1} \approx -\frac{1}{2} \end{aligned}$$

Here, $v_1^{[1]} = v_2^{[1]} = 1/\sqrt{2}$, $A_1^{[1]0} = .6030 - .0148i$, $A_2^{[1]0} = -.4297 - .6720i$, $A_1^{[1]1} = .7430 - .2900i$, $A_2^{[1]1} = .4786 + .3670i$, $A_1^{[2]0} = .9518 + .2155i$, $A_2^{[2]0} = .03461 + .2156i$, $A_1^{[2]1} = -.09900 - .1946i$ and $A_2^{[2]1} = -.6423 + .7347i$. Since the maximum value of α_1 is 2, the Schmidt rank $\chi = 2$ for $|\psi\rangle$.

To simulate one- and two-qubit gates, one uses algorithms that update the MPS representation of $|\psi\rangle$. The work in [68] gives algorithms that take $O(\chi^2)$ time for one-qubit gates [68, Lemma 1] and $O(\chi^3 + n\chi^2)$ time for two-qubit gates acting on consecutive qubits [68, Lemma 2]. In particular, applying two-qubit gates requires solving a potentially large eigenvalue problem to update the tensors and vectors associated with the input qubits only. Therefore, the computation cost of updating the MPS representation is independent of the number n of qubits, and grows polynomially in the overall Schmidt rank χ .

Empirical validation

To evaluate MPS-based simulation of quantum circuits, we used the ZKCM software package [58] written in C++. We compared the performance of ZKCM to that of p -blocked multiframes for the QFT benchmarks used in Sections 5.3 and 5.4.

Figure 7.1 shows that ZKCM simulates QFT circuits using approximate linear time and space for computational-basis input states. The maximum Schmidt rank χ observed for each QFT benchmark is 1 since the MPS decomposition always yields a separable state. Therefore, both the p -blocked multiframes (Section 5.4) and the MPS representations achieve exponential improvement over naive state-vector or decision-diagram implementations. However, the Quipu is orders of magnitude faster than ZKCM since it simulates all the benchmarks from Figure 7.1 in less than one second (Figure 5.15). This is not surprising since the decomposition procedure defined in Algorithm 5.4.1 is simpler than solving an eigenvalue problem – one of the steps required to update the MPS representation.

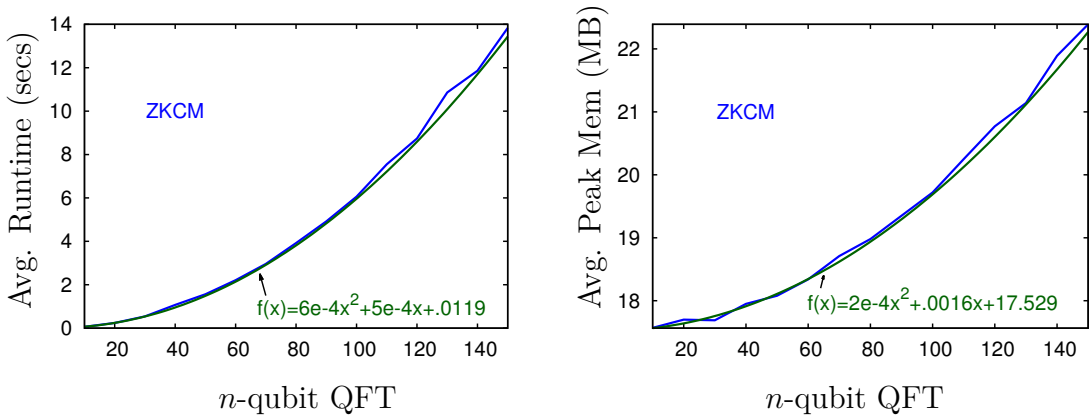


Figure 7.1: Average runtime and peak memory for ZKCM to simulate n -qubit QFT circuits on input state $|11 \dots 1\rangle$. The maximum Schmidt rank χ observed for each benchmark is 1. $R^2 = .999$ for polynomial fit $f(x)$ in both plots.

7.2 Quantum Multivalued Decision Diagrams (QMDDs)

As mentioned in Chapter II, the *quantum information decision diagram* (QuIDD) data structure can be used to represent certain quantum operators and states compactly. A detailed description of QuIDDs is given in [67, Chapter 7] along with other graph-based representations for modeling and synthesizing quantum operators. In Section 6.2.1, we defined new decision diagrams for representing Pauli expansions of quantum operators. Section 6.2.1 considered only basis-preserving quantum operators, and the Pauli decision diagrams (PDDs) we developed were limited to such operators. One particular advantage of PDDs over previous work on decision diagrams and MPS is that they can compactly represent highly entangled stabilizer states or operators. A more general form of PDDs can be developed using complex values on edges, as illustrated by Abdollahi and Pedram [3] in their data structure called *quantum decision diagrams* (QDDs). Miller and Thornton [47] developed *quantum multivalued decision diagrams* (QMDDs), which exploit the regular structure of the matrix operators that represent quantum circuits. Such matrices always have dimensions that are powers of two and can therefore be partitioned into quadrants, e. g., $\mathbf{M} = \begin{bmatrix} M_0 & M_1 \\ M_2 & M_3 \end{bmatrix}$. Each of the partitions M_i has dimensions $2^{n-1} \times 2^{n-1}$. The submatrices M_i can also be further partitioned into four smaller matrices. This process can recursively continue until the four submatrices become single values.

Definition VII.2. [47, Definition 5] An n -qubit *QMDD* is a directed acyclic graph with the following properties:

- (i) There is a single start vertex with only one incoming edge that itself has no source vertex.
- (ii) There is a single terminal node with no outgoing edges and fixed value 1.0.
- (iii) Each internal node (variable) $x_{i \in \{0,1,\dots,n-1\}}$ for d -valued logic has d^2 outgoing

edges. For quantum circuits, $d = 2$ and the outgoing edges are labeled 0, 1, 2, 3 corresponding to the subscripts of the submatrices M_i .

- (iv) The edges of each internal node are normalized such that the non-zero weight on the lowest index edge (one must exist or the vertex is redundant) divides all edge weights by the weight identified. The identified weight is then attached to the edge leading to the vertex.
- (v) For a fixed variable ordering $x_0 \prec x_1 \prec \dots \prec x_{n-1}$, each path through the QMDD from the top to a terminal value visits internal nodes in the reverse of the variable ordering, and each multiplicative edge simply multiplies the terminal at the end of the path by its value.
- (vi) Like most other DD-based data structures, internal nodes are irredundant and unique in that no two internal nodes contain the same outgoing edge values and destination nodes.

Example VII.3. The QMDD for the controlled- V gate is shown in Figure 7.2c.

A QMDD represents a complex-valued matrix uniquely up to variable reordering and certain quantum circuits can be represented compactly using QMDDs. One example is the cycle function benchmark *cycle17_3* from [46], which is a 20-qubit circuit with 48 Toffoli gates. Straightforward simulation using complex-valued matrices would require 48 multiplications of $2^{20} \times 2^{20}$ matrices while a QMDD for such a circuit contains only 236 nodes. Although most of the previous research on QMDDs has been devoted to their use in representing quantum circuits, the work in [32] describes how to represent quantum states using QMDDs. In this case, each node has exactly two outgoing edges corresponding to a recursive bipartitioning of the column vector that represents a quantum state. Therefore, states that are separable with respect to the computational basis can be represented compactly using QMDDs. The work

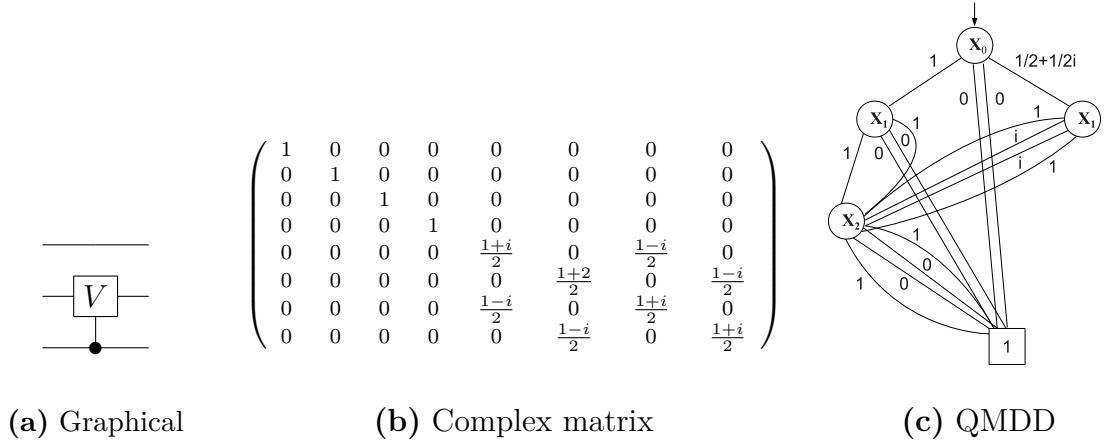


Figure 7.2: Several representations for the controlled- V gate on three qubits, where $V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$. The second and third qubits from the top are the control and target, respectively. The first qubit is not modified. The QMDD exploits the structure of the matrix and thus provides a more compact representation. Note that the edge weights are normalized as described in Definition VII.2.

in [47] outlines specialized QMDD algorithms for matrix multiplication, the tensor product, and initial construction. The use of QMDDs to simulate quantum logic is introduced in [32]. In the next section, we modify some of the properties introduced in Definition VII.2 to facilitate graph-based representation of matrix product states.

7.3 MPS-based Decision Diagrams

Section 7.2 described several quantum circuits whose operators can be compactly represented using QMDDs. Consider the matrix operator that defines the QFT operation on n qubits is,

$$F_n = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & & w^{N-1} \\ 1 & w^2 & \ddots & & w^{2(N-1)} \\ \vdots & & & & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)(N-1)} \end{bmatrix} \quad (7.4)$$

where $N = 2^n$ and $w = \exp(\frac{2\pi i}{N})$. Observe that recursive partitioning of F_n into quadrants does not yield many repeated submatrices. For example, the first level of partitioning for F_2 is $F_2^{[0]} = \begin{bmatrix} 1 & 1 \\ 1 & w \end{bmatrix}$, $F_2^{[1]} = \begin{bmatrix} 1 & 1 \\ w^2 & w^3 \end{bmatrix}$, $F_2^{[2]} = \begin{bmatrix} 1 & w^2 \\ 1 & w^3 \end{bmatrix}$ and $F_2^{[3]} = \begin{bmatrix} w^4 & w^6 \\ w^6 & w^9 \end{bmatrix}$, which are all distinct. This is also true in general for F_n . The absence of repeated submatrices limits compression and leads to an exponential increase in QMDD nodes. Therefore, QMDDs are not ideal for representing QFT operators. Nonetheless, as shown in Sections 5.4 and 7.1, it is possible to simulate the QFT operation efficiently. This suggests a potential generalization of QMDDs that incorporates sophisticated mathematical techniques such as those used in p -blocked multiframes and MPS. To this end, we modify some of the properties of QMDDs to define multivalued decision diagrams for modeling MPS.

Definition VII.4. Consider a quantum state $|\psi\rangle$ modeled by the MPS representation defined in Equation 7.3. An n -qubit *matrix product multivalued decision diagram* (MPMDD) for $|\psi\rangle$ is a directed acyclic graph with the following properties:

- (i) There is a single start vertex with only one incoming edge that itself has no source vertex.
- (ii) There is a single terminal node with no outgoing edges and fixed value 1.0.
- (iii) For each internal node (variable) $x_{l \in \{1, 2, \dots, n\}}$, the number of outgoing edges is twice the *local Schmidt rank* (χ_A in Equation 7.2).
- (iv) The variable ordering $x_1 \prec \dots \prec x_l \prec \dots \prec x_n$ is given by the MPS decomposition of $|\psi\rangle$. Node x_l is evaluated by assigning the pair of values (i_l, α_l) , which correspond to the indices of tensor $A_{\alpha_{l-1}\alpha_l}^{[l]i_l}$ in Equation 7.3. Observe that the value of index α_{l-1} is implied by an assignment of x_{l-1} .
 - (a) Each outgoing edge has a weight equal to $A_{\alpha_{l-1}\alpha_l}^{[l]i_l}$.
 - (b) Each internal node has weight equal to $v^{[l]\alpha_l}$.

- (v) A path through the MPMDD from the top to a terminal value corresponds to one term of the sum in Equation 7.3. Each multiplicative edge and node multiplies the terminal at the end of the path by its value.

Observe that MPMDDs are canonical since they directly model Equation 7.3 for an arbitrary quantum state. One advantage of using MPMDDs is that a particular assignment of variables can be evaluated using a (recursive) depth-first traversal of the graph. For an n -qubit MPS, the number of internal nodes in an MPMDDs is always n and its width (largest number of outgoing edges for any node) is 2χ . The general form of an MPMDD is shown in Figure 7.3. To compute a complex amplitude $c_{i_1 i_2 \dots i_n}$ of the MPS, one performs $O(\chi^2)$ depth-first traversals on the MPMDD – one for each term in the sum from Equation 7.3.

Proposition VII.5. *Any n -qubit quantum state $|\psi\rangle$ that admits an efficient QMDD representation using $\text{poly}(n)$ space also admits an efficient MPMDD representation.*

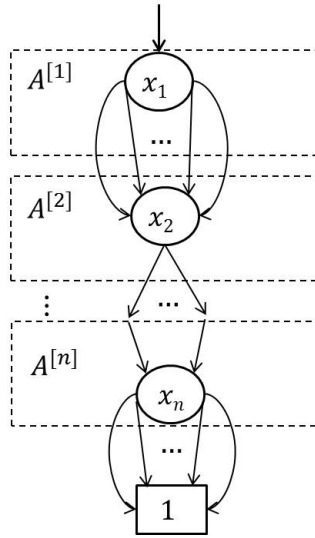


Figure 7.3: General form for an MPMDD. Level l of the diagram (dotted rectangles) corresponds to a tensor $A^{[l]}$ in Equation 7.3. The degree (number of outgoing edges) of each internal node in an MPMDD is equal to twice the local Schmidt rank (χ_A in Equation 7.2). Therefore, the width of the graph is twice the Schmidt rank χ for the state. The weights of nodes and edges depends on the particular variable assignment (Definition VII.4-iv).

Proof. A QMDD representation is efficient if the column vector for $|\psi\rangle$ can be recursively partitioned into $\text{poly}(n)$ blocks (Section 7.2). Such partitioning corresponds to a decomposition of $|\psi\rangle$ into separable blocks such that $|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_k\rangle$, where each $|\phi_i\rangle$ is a vector expressed in the computational basis. Since the concatenated Schmidt decomposition from Equation 7.3 has been shown to be canonical [55], any efficient computational-basis decomposition implied by a QMDD yields an MPS decomposition with Schmidt rank $\chi = \text{poly}(n)$. \square

Example VII.6. Figure 7.4 shows the QMDD and MPMDD for the separable state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle + |11\rangle)/2$.

Example VII.7. Figure 7.5 shows the QMDD and MPMDD for the entangled state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$.

Proposition VII.5 shows that MPMDDs generalize QMDDs and facilitate efficient graph-based representation of larger classes of quantum states. Examples VII.6 and VII.7 illustrate some of the differences between QMDDs and MPMDDs. For such simple instances, QMDDs are smaller than MPMDDs by a factor of n . However,

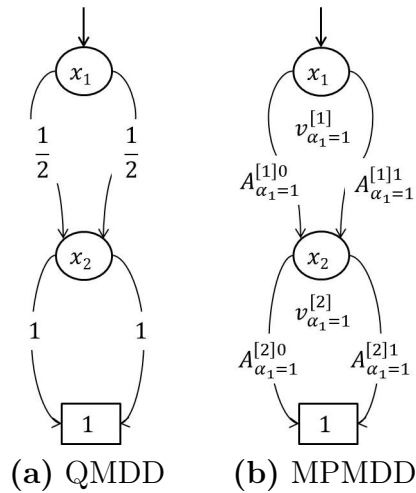


Figure 7.4: Two graph-based representations for the separable state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle + |11\rangle)/2$. Here, $v_1^{[1]} = v_1^{[2]} = 1.0$ and $A_1^{[1]0} = A_1^{[1]1} = A_1^{[2]0} = A_1^{[2]1} = 1/\sqrt{2}$.

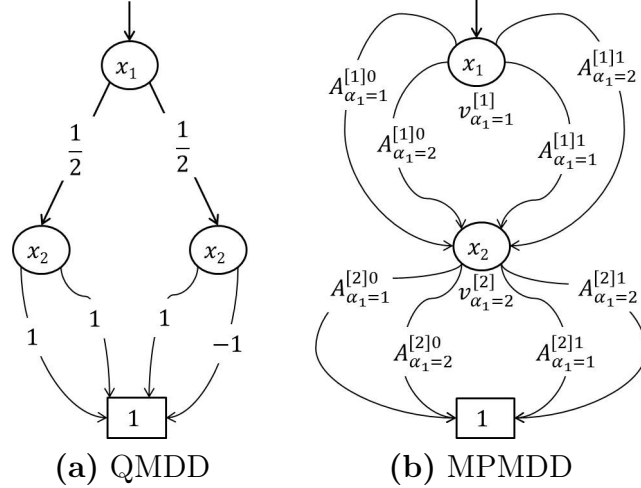


Figure 7.5: Two graph-based representations for the entangled state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$. The tensor (A) and vector (v) values for the MPMDD are listed in Example VII.1.

for more complex cases such as those involving QFT circuits, QMDDs observe exponential blowup while MPMDDs remain poly-sized (Section 7.1). Furthermore, since one can directly obtain any of the MPS tensors directly from an MPMDD using pointers, any MPS algorithms for quantum logic simulation can be extended to work on MPMDDs. An open challenge is to adapt the algorithms described in [68] for implementing one- and two-qubit operations to work directly on MPMDDs. Such algorithms may exploit graph structure to improve performance.

7.4 Summary

In this chapter, we described two well-known techniques for quantum simulation: matrix product states (MPS) and quantum multivalued decision diagrams (QMDD). The results from Section 7.1 suggest that it is possible to derive hybrid modeling techniques that combine MPS and QMDDs. Such an approach would expand the classes of states that can be represented via decision diagrams and potentially improve the performance of MPS simulation. To this end, we introduced matrix product multivalued decision diagrams (MPMDDs) in Section 7.3. We derived specific properties

for MPMDDs and showed that any quantum state that can be represented compactly using QMDDs also admits an efficient MPMDD representation.

CHAPTER VIII

Conclusions

Generic quantum-circuit simulation appears intractable for conventional computers since required resources grow exponentially with circuit width. As compared to digital logic circuits, quantum circuits have unique properties that make them significantly more complicated to analyze using traditional design-automation techniques. Another challenge is that quantum information is fragile and prone to several types of environmental errors, which experimental physicists find difficult to characterize. Nonetheless, as quantum information processing gains traction, quantum-circuit simulation becomes increasingly significant for engineering purposes – evaluation, testing and optimization – as well as for theoretical research. Therefore, quantum design and simulation tools must incorporate sophisticated models and efficient classical algorithms to overcome these challenges for classes of circuits with practical value. For example, Gottesman and Knill identified an important subclass, called *stabilizer circuits*, which can be simulated efficiently using group-theory techniques and insights from quantum physics. Realistic circuits enriched with quantum error-correcting codes and fault-tolerant procedures are dominated by stabilizer subcircuits and contain a relatively small number of non-stabilizer components. The work in this dissertation centers around developing new data structures and algorithms that facilitate parallel simulation of such circuits. In the remaining sections, a detailed discussion

of our contributions is offered, followed by a description of open challenges and a discussion of future applications.

8.1 Summary of Contributions

In this dissertation, we have evaluated several simulation techniques based on the Heisenberg model for quantum computers – also known as the stabilizer formalism. We designed a new simulation technique that facilitates simulation of generic quantum circuits via the stabilizer formalism and therefore exploits the speedups offered by such group-theory techniques. We developed and implemented a comprehensive software tool based on this technique. Our contributions directly facilitate further analysis of quantum speed-ups, exploitable structure in quantum information and error characterization. The contributions of this dissertation fall into several major categories: *(i)* analysis of geometric properties of stabilizer states and their embedding in Hilbert space (Chapters II and III), *(ii)* computational geometry of stabilizer states (Chapter IV), and *(iii)* design and implementation of several new stabilizer-based simulation techniques (Chapters V and VI).

Geometric properties of stabilizer states. We defined a nearest neighbor of an n -qubit stabilizer state $|\psi\rangle$ as a state $|\phi\rangle$ such that $|\langle\psi|\phi\rangle| = 1/\sqrt{2}$, which is the largest possible value $\neq 1$ (Corollary III.3). In Theorem III.11, we proved that every stabilizer state has exactly $4(2^n - 1)$ nearest neighbors. Theorem III.14 generalizes this result to the case of k -neighbors, where $0 < k \leq n$. We used these results to quantify the distribution of angles between any one stabilizer state and all other stabilizer states. We showed that, for sufficiently large n , $1/3$ of all stabilizer states are orthogonal to $|\psi\rangle$ (Corollary III.15) and the fraction of k -neighbors tends to zero for $0 < k < n - 4$ (Theorem III.17). Therefore, $2/3$ of all stabilizer states are oblique to $|\psi\rangle$ and this fraction is dominated by the k -neighbors, where $n - 4 \leq k \leq n$. These findings suggest

a rather uniform geometric structure for stabilizer states. This is further evidenced by two additional facts. First, for any n -qubit stabilizer state $|\psi\rangle$, there exists a set of $2^n - 1$ nearest neighbors to $|\psi\rangle$ that lie at 60° angles to each other (Corollary III.12). Second, every linearly-dependent triplet of stabilizer states that are non-parallel to each other includes two pairs of nearest neighbors and one pair of orthogonal states (Corollary III.23). Additionally, Theorem III.21 shows that there are $5(2^n - 1)$ states $|\varphi\rangle$ (including all nearest-neighbor states) such that the wedge product $|\phi\rangle = |\psi \wedge \varphi\rangle$ can also be represented compactly (up to a phase) using the stabilizer formalism. We call such a state a stabilizer bivector. In Proposition III.22, we proved that the norm of any stabilizer bivector and thus the area of the parallelogram formed by any two stabilizer states is $\sqrt{1 - 2^{-k}}$ for $0 \leq k \leq n$.

Embedding of stabilizer geometry in Hilbert space. In Section 3.2, we described how the discrete embedding of stabilizer geometry in Hilbert space complicates several natural geometric tasks. Our results on the geometric properties of stabilizer states imply that there are significantly more stabilizer states than the dimension of the Hilbert space in which they are embedded (Theorem III.18), and that they are arranged in a fairly uniform pattern (Corollaries III.12 and III.23). These factors suggest that, if one seeks a stabilizer state closest to a given arbitrary quantum state, local search appears a promising candidate. To the contrary, we showed that local search *does not* guarantee finding such stabilizer states (Section 3.2.1). The second natural task we consider is representing or approximating a given arbitrary quantum state by a short linear combination of stabilizer states. Again, having considerably more stabilizer states than the dimension of the Hilbert space suggested a positive result at first. However, we demonstrated a family of quantum states that exhibits *asymptotic orthogonality* to all stabilizer states and can thus be neither represented nor approximated by short superpositions of stabilizer states (Theorem III.26). Furthermore,

we showed in Proposition III.27 that the maximal radius of any 2^n -dimensional ball centered at a point on the unit sphere that does not contain any n -qubit stabilizer states cannot exceed $\sqrt{2}$, but approaches $\sqrt{2}$ as $n \rightarrow \infty$.

Computational geometry of stabilizer states. Angles between stabilizer states were discussed in [2], where the authors describe possible values for such angles and outline an inner-product computation that involves the synthesis of a basis-normalization stabilizer circuit that maps a stabilizer state to a computational basis state. We observe that this circuit-synthesis procedure is the computational bottleneck of the algorithm and thus any improvements to this synthesis process translate into increased performance of the overall algorithm. In Chapter IV, we described an algorithm for synthesizing canonical circuits with the block sequence $H-C-CZ-P-H$. Our circuits are close to the smallest known circuits proposed in [14]. Our canonical circuits improve the computation of inner products, helping us outperform the inner-product algorithm based on non-canonical circuits proposed in [6]. Algorithm 4.1.1 produces circuits with less than half as many gates on average and runs roughly $2\times$ faster. Furthermore, our synthesis approach produces canonical circuits given any input stabilizer state by first obtaining a *canonical generator set* for the state. Additionally, we exploited our analysis of the geometry of stabilizer states to design algorithms for the following fundamental tasks: (i) orthogonalization for linear combinations of stabilizer states, and (ii) efficient computation of stabilizer bivectors.

Stabilizer-based simulation of generic quantum circuits. We leveraged our results on the geometry of stabilizer states to design several data structures and algorithms for manipulating superpositions of stabilizer states. Our techniques ensure the scalability of quantum-circuit simulation when stabilizer gates dominate as demonstrated by our implementation which achieves significant speedups over state-

of-the-art simulators for such circuits. Our contributions along this line of research include the following:

- Development of a new algorithm for maintaining the global phase of a stabilizer state. This algorithm allowed us to overcome a key obstacle for representing quantum states as superpositions of pure stabilizer states since global phases become relative with respect to such superpositions.
- Design of the stabilizer frame data structure along with efficient algorithms for frame-based operations (`ROTATE`, `COFACTOR`) that facilitate simulation of quantum circuits with stabilizer, non-stabilizer and measurement gates.
- Development of the `COALESCE` operation for stabilizer frames. Such an operation exploits symmetries in a stabilizer frame and reduces the total number of states in a given superposition. Frame coalescing leads to a more compact multiframe data structure.
- Implementation of simulation invariants to speed up relevant algorithms. One invariant ensures that the stabilizer matrix for each frame is stored in row-echelon form to avoid expensive Gaussian elimination during simulation. Another invariant implements a heuristic approach to maintain the orthogonality of multiframe.
- Design of p -blocked multiframe, which generalize multiframe to incorporate the p -blocking technique from [41].
- Implementation of our software package, called `Quipu`, which incorporates all our simulation techniques and includes a software infrastructure for parallel frame-based simulation.
- Empirical validation of stabilizer frames, multiframe and p -blocked multiframe. `Quipu` simulates certain quantum arithmetic circuits (e.g., reversible ripple-carry

adders) in polynomial time and space for equal superpositions of n -qubits. On such instances, known linear-algebraic simulation techniques, such as the (state-of-the-art) BDD-based simulator `QuIDDP`, take exponential time. We simulate quantum Fourier transform (QFT) and quantum fault-tolerant circuits using `Quipu`, and the results demonstrate that our frame-based techniques empirically outperforms `QuIDDP` in all cases. In particular, the runtime of p -blocked multiframe simulation of QFT circuits is approximately linear in the number of qubits.

Generic stateless simulation and new graph-based techniques. Using our analysis of the Pauli expansions of quantum operators, we generalized the stateless approach from [8, 11, 39] to include simulation generic quantum circuits. Our contributions along this line of research include the following:

- Derivation of Pauli expansions of several useful quantum operators including multi-controlled X and Z operators.
- Design of a recurrence relation to express the Pauli expansion of addition operators.
- Design of Pauli multivalued decision diagrams, which can be used to represent Pauli expansions compactly.
- Development and implementation of an algorithm for stateless simulation of non-stabilizer gates using Pauli expansions.
- Empirical validation of stateless simulation. For stabilizer circuits that follow a specific structure, stateless simulation outperforms stabilizer-based techniques such as `CHP`. However, when simulating non-stabilizer circuits such as ripple-carry adders, stateless simulation takes exponential time and thus is outperformed by our frame techniques.

- Design of matrix product multivalued decision diagrams, which facilitate graph-based representation of larger classes of quantum states including matrix product states.

8.2 Open Challenges

As reviewed in Section 4.5.1, the work in [2, 6, 63] describes how to represent and manipulate mixed states using the stabilizer formalism. To this end, extensions of our results to mixed states could be of interest. Another attractive direction for future work is generalizing presented results and algorithms to the case of d -dimensional qudit states [12]. Further challenges related to the geometry of stabilizer states include:

- Characterization of possible volumes of high-dimensional parallelotopes formed by more than two stabilizer vectors (generalizing Proposition III.22).
- Characterization of minimally-dependent sets of stabilizer states and algorithms for efficient detection of such sets (extending our discussion in Section 3.1.3).
- Efficiency improvement for computing inner and exterior products of stabilizer states as well as the orthogonalization procedure from Algorithm 4.3.1.
- Characterization of the complexity analysis of finding a closest stabilizer state to a given non-stabilizer state (for appropriate representations of non-stabilizer states).
- Describing the Voronoi-diagram structure of stabilizer states (extending Theorem III.11).

Recently, new techniques for synthesizing single-qubit operators using Clifford and T gates has been developed [43, 59]. An open problem is whether such synthesis

techniques can be exploited to improve (stabilizer) frame-based simulation. In Section 5.2.2, we described a software framework for parallel simulation of multiframes and in Section 5.4 we introduced p -blocked multiframes, which admit poly-time simulation of quantum Fourier transform circuits. Two open challenges are:

- Designing software techniques for distributed simulation of p -blocked multiframes.
- Identifying additional circuits for which p -blocked multiframes outperform other frame-based techniques.

We defined Pauli decision diagrams (PDDs) in Section 6.2.1 and showed that PDDs can represent certain quantum operators compactly. Examples of such operators include highly entangled stabilizer operators and quantum addition operators, which previous decision diagram and matrix product techniques fail to represent efficiently. More general PDDs can be developed using complex-valued edge weights. Such edge-valued PDDs may be used identify new classes of quantum operators that can be represented compactly on conventional computers.

In Section 7.3, we introduced graph-based techniques for representing matrix product states. An attractive research direction is to design graph-based algorithms for quantum logic simulation using matrix product multivalued decision diagrams (MP-MDDs). Such algorithms can be adapted from the work in [68] and could improve performance of MPS simulation.

8.3 Future Directions

In this work, we have discussed different techniques to efficiently simulate various classes of quantum circuits depending on certain properties. Our successful design of hybrid modeling techniques that combine distinct mathematical approaches has led to theoretical insights and improved empirical performance for quantum-circuit

simulation. Nonetheless, it is still unclear how much overlap exists among stabilizer frames and other well-known techniques such as decision diagrams and matrix product states. To this end, we now discuss future directions for generalizing frame-based simulation to incorporate such techniques.

8.3.1 Frame-based simulation using Pauli expansions

As shown in Chapter V, frame-based simulation of n -qubit stabilizer circuits takes time polynomial in $|\mathcal{F}|$, the number of distinct phase vectors stored in the frame. Therefore, if an arbitrary state $|\phi\rangle$ can be decomposed into a stabilizer frame such that $|\mathcal{F}| = \text{poly}(n)$ and U is composed exclusively of stabilizer and Pauli gates, $U|\phi\rangle$ can be simulated *efficiently* on a classical computer. We showed examples of such states in Section 5.3. Furthermore, we described in Section 6.2.1 how certain quantum operators can be represented compactly using Pauli multivalued decision diagrams. Finding other quantum operators that admit Pauli MDD representations can help expand the practical value of both stateless and frame-based simulation techniques. Moreover, the overall performance of frame-based simulation may be improved if one designs efficient algorithms to apply Pauli MDD's directly to a stabilizer frame without expanding the Pauli decomposition represented by the MDD, which contains an exponential number of terms. Such algorithms would extend the class of operators that can be simulated using the stateless approach and facilitate analysis of intermediate states during the execution of a particular quantum algorithm such as Shor's number-factoring. This analysis can deepen our insight into why certain algorithms achieve exponential speed-ups over classical simulation.

8.3.2 Stabilizer-based matrix product states

Recall from Section 7.1 that Matrix product states (MPS) are an efficient approximation for weakly-entangled quantum states. There are important similarities

between p -blocked multiframe and MPS. Both approaches make use of a decomposition algorithm for updating the underlying representation. In fact, our decomposition algorithm (Algorithm 5.4.1) can be regarded as a modified version of Schmidt decomposition specific to stabilizer frames. Furthermore, the stabilizer matrix associate with a frame is analogous to the $A^{[l]}$ operators found MPS. Additionally, the phase vectors in a frame are analogous to the $v^{[l]}$ vectors in MPS. In similarity to the Schmidt rank, one can introduce the *stabilizer rank of a quantum state* – the smallest number of stabilizer states whose linear combination expresses a given quantum state. For example, while the GHZ state $(|000\rangle + |111\rangle)\sqrt{2}$ has Schmidt rank two, its stabilizer rank would be one since it can be represented using a single stabilizer state. One key difference is that MPS decomposition considers all possible bipartitions while Algorithm 5.4.1 does not. A generalized version of Algorithm 5.4.1 can lead to a *stabilizer-based matrix product state* representation that holds great promise for achieving new theoretical insights and improved empirical performance for quantum-circuit simulation.

APPENDIX

APPENDIX A

The 1080 **three-qubit stabilizer states**

Shorthand notation represents a stabilizer state as $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ where α_i are the normalized amplitudes of the basis states. Basis states are emphasized in bold. The \angle column indicates the angle between that state and $|000\rangle$, which has 28 1-neighbor states and 315 orthogonal states (\perp). The tables are intended for on-screen viewing under magnification.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] S. Aaronson. Multilinear formulas and skepticism of quantum computing. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 118–127, 2004.
- [2] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(052328), 2004.
- [3] A. Abdaollahi and M. Pedram. Analysis and synthesis of quantum circuits by using quantum decision diagrams. In *Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition*, pages 1–6, Munich, Germany, 2006.
- [4] D. Aharonov. A simple proof that toffoli and hadamard are quantum universal. arXiv:0301040, 2003.
- [5] S. Anders and H. J. Briegel. Fast simulation of stabilizer circuits using a graph-state representation. *Physical Review A*, 73(022334), 2006.
- [6] K. M. R. Audenaert and M. B. Plenio. Entanglement on mixed stabiliser states: normal forms and reduction procedures. *New Journal of Physics*, 7(170), 2005.
- [7] A. Calderbank, E. Rains, P. Shor, and N. Sloane. Quantum error correction via codes over $\text{gf}(4)$. *IEEE Transactions on Information Theory*, 44(4):1369–1387, 1998.
- [8] S. Clark, R. Jozsa, and N. Linden. Generalised clifford groups and simulation of associated quantum circuits. arXiv:quant-ph/0701103v1, 2007.
- [9] R. Cleve and D. Gottesman. Efficient computations of encodings for quantum error correction. *Physical Review A*, 56(1):1201–1204, 1997.
- [10] O. Dahlsten and M. B. Plenio. Exact entanglement probability distribution of bi-partite randomised stabilizer states. *Quantum Information and Computation*, 6(527), 2006.
- [11] C. M. Dawson, A. P. Hines, D. Mortimer, H. L. Haselgrove, M. A. Nielsen, and T. J. Osborne. Quantum computing and polynomial equations over the finite field \mathbb{Z}_2 . *Quantum Information and Computation*, 5(2):102–112, 2005.

- [12] N. de Beaudrap. A linear stabilizer formalism for systems of any finite dimension. *Quantum Information and Computation*, 13:73–115, 2013.
- [13] J. Dehaene and B. De Moor. Clifford group, stabilizer states, and linear and quadratic operations over $\text{gf}(2)$. *Physical Review A*, 68(042318), 2003.
- [14] M. Van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information and Computation*, 10:258–271, 2010.
- [15] M. Van den Nest, J. Dehaene, and B. De Moor. On local unitary versus local Clifford equivalence of stabilizer states. *Physical Review A*, 71(062323), 2005.
- [16] D. P. DiVincenzo, D. W. Leung, and B. M. Terhal. Quantum data hiding. *IEEE Transactions on Information Theory*, 48(3):580–599, 2002.
- [17] D. P. DiVincenzo and P. W. Shor. Fault-tolerant error correction with efficient quantum codes. *Physical Review Letters*, 77(3260), 1996.
- [18] I. Djordjevic. *Quantum Information Processing and Quantum Error Correction: an Engineering Approach*. Academic press, 2012.
- [19] W. Dur, H. Aschauer, and H. J. Briegel. Multiparticle entanglement purification for graph states. *Physical Review Letters*, 91(107903), 2003.
- [20] C. Emary. A bipartite class of entanglement monotones for n -qubit pure states. *Journal of Physics A*, 37(8293), 2004.
- [21] E. Knill et al. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1), 2007.
- [22] K. De Raedt et al. Massively parallel quantum computer simulator. *Computer Physics Communications*, 176(1–2):121–136, 2007.
- [23] M. Hayashi et al. Entanglement of multiparty stabilizer, symmetric, and anti-symmetric states. *Physical Review A*, 77(012104), 2008.
- [24] O. Boncalo et al. Using simulated fault injection for fault tolerance assessment of quantum circuits. In *Proceedings of the IEEE Simulation Symposium*.
- [25] S. A. Cuccaro et al. A new quantum ripple-carry addition circuit. arXiv:0410184v1.
- [26] D. Fattal, T. S. Cubitt, Y. Yamamoto, S. Bravyi, and I. L. Chuang. Entanglement in the stabilizer formalism. arxiv:0406168, 2004.
- [27] F. Fröwis, V. Nebendahl, and W. Dür. Tensor operators: constructions and applications for long-range interaction systems. *Physical Review A*, 81(062337), 2010.

- [28] H. J. García and I. L. Markov. Quipu: high-performance simulation of quantum circuits using stabilizer frames. In *Proceedings of the IEEE International Conference on Computer Design*, pages 404–410, Asheville, North Carolina, 2013.
- [29] H. J. García and I. L. Markov. Simulation of quantum circuits using stabilizer frames. *IEEE Transactions on Computers*, 2014. Under review.
- [30] H. J. García, I. L. Markov, and A. W. Cross. Efficient inner-product algorithm for stabilizer states. arXiv:1210.6646, 2012.
- [31] H. J. García, I. L. Markov, and A. W. Cross. On the geometry of stabilizer states. *Quantum Information and Computation*, 14(7–8):683–720, 2014.
- [32] D. Goodman and M. A. Thornton. Quantum logic circuit simulation based on the qmdd data structure. In *Proceedings of the Workshop on Applications of the ReedMuller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology*, pages 99–105, 2007.
- [33] D. Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997. argxiv:9705052.
- [34] D. Gottesman. The heisenberg representation of quantum computers. In *Proceedings of the twenty-second International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, 1998. arXiv:9807006v1.
- [35] O. Guehne, G. Toth, P. Hyllus, and H.J. Briegel. Bell inequalities for graph states. *Physical Review Letters*, 95(120405), 2005.
- [36] A. W. Harrow and R. A. Low. Random quantum circuits are approximate 2-designs. *Communications in Mathematical Physics*, 291(1):257–302, 2009.
- [37] M. Hein, J. Eisert, and H.J. Briegel. Multi-party entanglement in graph states. *Physical Review A*, 69(062311), 2004.
- [38] H. Heydari. Quantum entanglement measure based on wedge product. *Quantum Information and Computation*, 6:166–172, 2006.
- [39] R. Jozsa. Embedding classical into quantum computation. *Lecture Notes in Computer Science*, 5393:43–49, 2008.
- [40] R. Jozsa and M. Van den Nest. Classical simulation complexity of extended clifford circuits. arXiv:1305.6190, 2013.
- [41] R. Jozsa and N. Linden. On the role of entanglement in quantum computational speed-up. In *Proceedings of the Royal Society of London. Series A. Mathematical, Physical and Engineering Sciences*, volume 459, pages 2011–2032, 2003.
- [42] A. Klappenecker and M. Roetteler. Mutually unbiased bases are complex projective 2-designs. arXiv:0502031, 2005.

- [43] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by clifford and t gates. *Quantum Information And Computation*, 13(7–8):607–630, 2013.
- [44] I. L. Markov and M. Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Information and Computation*, 12(5), 2012.
- [45] I. L. Markov and Y. Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.
- [46] D. Maslov, G. Dueck, and N. Scott, 2011. <http://webhome.cs.uvic.ca/~dmaslov/>.
- [47] D. M. Miller and M. A. Thornton. Qmdd: A decision diagram structure for reversible and quantum circuits. In *Proceedings of the International Symposium on Multiple Valued Logic*, 2006.
- [48] A. Montanaro. On the distinguishability of random quantum states. *Communications in Mathematical Physics*, 273(3):619–636, 2007.
- [49] C. Moore and M. Nilsson. Parallel quantum computation and quantum codes. *SIAM Journal on Computing*, 31(3):799–815, 2001.
- [50] V. Murg, J.I. Cirac, B. Pirvu, and F. Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(025012), 2010.
- [51] A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [52] K. M. Obenland and A. M. Despain. A parallel quantum computer simulator. arXiv:9804039.
- [53] M. Oskin, F. T. Chong, and I. L. Chuang. A practical architecture for reliable quantum computers. *IEEE Computer*, 35(1):79–87, 2002.
- [54] K. N. Patel, I. L. Markov, and J. P. Hayes. Optimal synthesis of linear reversible circuits. *Quantum Information and Computation*, 8(3–4):282–294, 2008.
- [55] D. Perez-García, F. Verstraete, M.M. Wolf, and J.I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 7(401), 2007.
- [56] J. Preskill. Fault-tolerant quantum computation. In *Introduction to Quantum Computation*. World Scientific, 1998. arXiv:9712048.
- [57] R. Raussendorf, D. E. Browne, and H. J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(022312), 2003.

- [58] Akira SaiToh. Zkcm: a c++ library for multiprecision matrix computation with applications in quantum information. *Computer Physics Communications*, 184(8):2005–2020, 2013.
- [59] P. Selinger. Efficient clifford + t approximation of single-qubit operators. arXiv:1212.6253, 2012.
- [60] Y. Shi. Both toffoli and controlled-not need little help to do universal quantum computation. *Quantum Information And Computation*, 3(1):1–11, 2003.
- [61] P. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4), 1995.
- [62] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [63] G. Smith and D. Leung. Typical entanglement of stabilizer states. *Physical Review A*, 74(062314), 2006.
- [64] K. M. Svore, A. V. Aho, A. W. Cross, I. L. Chuang, and I. L. Markov. A layered software architecture for quantum computing design tools. *IEEE Computer*, 39(1):74–83, 2006.
- [65] F. Verstraete and J. I. Cirac. Valence-bond states for quantum computation. *Physical Review A*, 70(060302), 2004.
- [66] F. Verstraete, J. I. Cirac, and J. I. Latorre. Quantum circuits for strongly correlated quantum systems. *Physical Review A*, 79(032316), 2009.
- [67] G. F. Viamontes, I. L. Markov, and J. P. Hayes. *Quantum Circuit Simulation*. Springer, 2009.
- [68] G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(147902), 2003.
- [69] H. Wunderlich and M. B. Plenio. Quantitative verification of fidelities and entanglement from incomplete measurement data. *Journal of Modern Optics*, 56:2100–2105, 2009.