University College London

Department of Physics & Astronomy

Exploring Quantum Computation Through the Lens of Classical Simulation

Author:
Padraic Calpin

Supervisor:
Prof. Dan Browne

Submitted in partial fulfilment for the degree of **Doctor of Philosophy**

June 15, 2019

I, Padraic Calpin, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature

June 15, 2019

Date

Abstract

My research is about stuff.

It begins with a study of some stuff, and then some other stuff and things.

There is a 300-word limit on your abstract.

Impact Statement

UCL theses now have to include an impact statement. *(I think for REF reasons?)* The following text is the description from the guide linked from the formatting and submission website of what that involves. (Link to the guide: `http://www.grad.ucl.ac.uk/essinfo/docs/Impact-Statement-Guidance-Notes-for-Research-Students-and-Supervisors.pdf`)

> The statement should describe, in no more than 500 words, how the expertise, knowledge, analysis, discovery or insight presented in your thesis could be put to a beneficial use. Consider benefits both inside and outside academia and the ways in which these benefits could be brought about.
>
> The benefits inside academia could be to the discipline and future scholarship, research methods or methodology, the curriculum; they might be within your research area and potentially within other research areas.
>
> The benefits outside academia could occur to commercial activity, social enterprise, professional practice, clinical use, public health, public policy design, public service delivery, laws, public discourse, culture, the quality of the environment or quality of life.
>
> The impact could occur locally, regionally, nationally or internationally, to individuals, communities or organisations and could be immediate or occur incrementally, in the context of a broader field of research, over many years, decades or longer.
>
> Impact could be brought about through disseminating outputs (either in scholarly journals or elsewhere such as specialist or mainstream media), education, public engagement, translational research, commercial and social enterprise activity, engaging with public policy makers and public service delivery practitioners, influencing ministers, collaborating with academics and non-academics etc.

Further information including a searchable list of hundreds of examples of UCL impact outside of academia please see `https://www.ucl.ac.uk/impact/`. For thousands more examples, please see `http://results.ref.ac.uk/Results/SelectUoa`.

Acknowledgements

Acknowledge all the things!

Contents

Chapter 1

Introduction

1.1 A section

1.1.1 A subsection

Hello

Chapter 2

Methods for Simulating Stabilizer Circuits

2.1 Introduction

In the previous chapter (INSERT REFERENCE), we briefly introduced the notion of stabilizer circuits as a class of efficiently simulable quantum computations. In this chapter, we revisit stabilizer circuits in detail, with a focus on different classical data structures for encoding stabilizer states and the corresponding algorithms for simulations.

Several informal definitions of stabilizer circuits have been used in the quantum computing literature [1, 2, 3, 4]. However, what each definition has in common is that the operations $\mathcal{E}$ acting on an abelian subgroup $\mathcal{S} \subseteq \mathcal{P}_n$ generate a new subgroup $\mathcal{S}' \subseteq \mathcal{P}_n$. These groups $\mathcal{S}$ are also called a stabilizer groups.

In this thesis, we focus exclusively on stabilizer circuits acting on pure states $|\phi\rangle$ called stabilizer states. These can be entirely characterized by their associated stabilizer group as

$$s\,|\phi\rangle = |\phi\rangle \;\forall s \in \mathcal{S} \tag{2.1}$$

For an n-qubit state, the group $\mathcal{S}$ has 2^n elements [1]. As $\mathcal{S}$ is also abelian, this means it can be described by a generating set with n elements,

$$\mathcal{S} = \langle g_1, g_2, \dots, g_n \rangle \;: g_i \in \mathcal{S}, \tag{2.2}$$

which are commonly referred to as the 'stabilizers' of the state $|\phi\rangle$. We also note that this definition allows us to write

$$|\phi\rangle\langle\phi| = \frac{1}{2^n}\sum_{s\in\mathcal{S}} s = \frac{1}{2^n}\prod_{i=1}^{n}(\mathbb{I}+g_i) \tag{2.3}$$

Given that these circuits map stabilizer states to other stabilizer states, this means they must be built up of unitary operations U which map Pauli operators to other Pauli operators under conjugation. This set is commonly denoted as $\mathcal{C}_2$, or the 'second level of the Clifford hierarchy'

$$\mathcal{C}_2 \equiv \{U : UPU^\dagger \in \mathcal{P}_n \,\forall P \in \mathcal{P}_n\} \tag{2.4}$$

$$\mathcal{C}_j \equiv \{U : UPU^\dagger \in \mathcal{C}_{j-1} \,\forall P \in \mathcal{P}_n\} \tag{2.5}$$

where in Eq. 2.5 we have also introduced the (recursive) definition for level j of the Clifford hierarchy. From this definition

$$V\mathcal{S}V^\dagger = \langle V g_i V^\dagger\rangle = \langle g_i'\rangle = \mathcal{S}' \tag{2.6}$$

We also allow stabilizer circuits to contain measurements in the Pauli basis [1].

Simulating stabilizer circuits

From the above definitions, we can see that simulating a stabilizer circuit on n qubits corresponds to updating the n stabilizer generators for each unitary and measurement we apply. As the number of generators grows linearly in the number of qubits, if these group updates can be computed in time $O(\text{poly}(n))$ then it follows the circuits can be efficiently simulated clasically.

The first proof of this was given by Gottesman in [1], by showing through examples that stabilizer updates can be quickly computed for the CNOT, H and S gates, and for single qubit Pauli measurements. This is significant as the n qubit Clifford group can be entirely generated from these gates.

$$\mathcal{C}_2 = \langle CNOT_{i,j}, H_i, S_i : i,j \in \mathbb{Z}_n\rangle. \tag{2.7}$$

This result is typically referred to as the 'Gottesman-Knill' theorem.

A more formal proof follows from the work of Dehaene & de-Moor, who showed that the action of Clifford unitaries on Pauli operators corresponds to multiplication of $(2n+1)\times(2n+1)$ symplectic binary matrices with $(2n+1)$-bit binary vectors [5]. The dimension of these elements also grows just linearly in the number of qubits, and as matrix multiplication requires time $O(n^{2.37})$ it follows that we can update the stabilizers in $O(mn^{2.73})$ for m Clifford gates.

This work was then extended by Aaronson & Gottesman, who introduced an efficient data structure for stabilizer groups, and algorithms for their updates under Clifford gates and Pauli measurement [2]. This method avoids the need for matrix multiplications, instead providing direct update rules allowing stabilizer circuits to be simulated in $O(n^2)$.

Since 2004, there have been several papers looking at different data structures and algorithms for simulating stabilizer circuits of the type we consider here. For example, a method based on encoding stabilizer states as graphs [6], refinements of the Aaronson & Gottesman encoding [7], and an encoding using affine spaces and phase polynomials [3, 8].

In the rest of this section, we will discuss different aspects of simulating stabilizer circuits, focusing on updating stabilizer states under gates and measurements, computing stabilizer inner products, and the connections between stabilizer circuits and states.

2.1.1 Tableau Encodings of Stabilizer States

The method in [2] is based on a classical data structure they call the 'stabilizer tableau', a collection of Pauli matrices that define the stabilizer group, encoded using the binary symplectic representation of [5]

$$P = i^{\delta} - 1^{\epsilon} \bigotimes_{i=1}^{n} x_i z_i \tag{2.8}$$

where the Pauli matrix at qubit i is defined by two binary bits such that

$$x_i z_i = \begin{cases} I & x_i = z_i = 0 \\ X & x_i = 1, z_i = 0 \\ Z & x_i = 0, z_i = 1 \\ Y & x_i = z_i = 1 \end{cases} \tag{2.9}$$

Together with the δ and ϵ phases, a generic Pauli operator can be encoded in $2n+2$ bits; two bits to encode the phase, and two n-bit binary strings $\tilde{x}, \tilde{z} \in \mathbb{Z}_2^n$ to encode the Pauli acting on each qubit, commonly referred to as 'x-bits' and 'z-bits' respectively. In this picture, multiplication of Pauli operators corresponds to addition of x and z bits modulo 2, with some additional, efficiently computable function for correcting the phase [5]

$$PQ = i^{\delta_{pq}} - 1^{\epsilon_{pq}} \bigotimes_{i=1}^{n} x_i' z_i' \tag{2.10}$$

$$x_i' = x_{pi} \oplus x_{qi} \tag{2.11}$$

$$z_i' = z_{pi} \oplus x_{qi} \tag{2.12}$$

where $\delta_{pq} = \delta_p \oplus \delta_q$, $\epsilon_{qr} = f(\tilde{x}_p, \tilde{z}_p, \tilde{x}_q, \tilde{z}_q)$.

In stabilizer groups, we can restrict ourselves to considering Pauli operators with only real phase. This is because if $iP \in \mathcal{S}$, then $(iP)^2 = -I \in \mathcal{S}$. But, this implies that $-I\,|\phi\rangle = |\phi\rangle$, which is a contradiction.

While only n generators S_i are needed to characterize the stabilizer group $\mathcal{S}$, the tableau also includes an additional $2n$ operators called 'destabilizers' $D_i \in \mathcal{P}_n$. Together, these $2n$ operators generate all 4^n elements of $\mathcal{P}_n$.

There are many possible choices of destabilizer, but the tableau chooses operators such that [2]

$$[D_i, D_j] = 0\,\forall i,j \in \{1,\dots,n\}$$
$$[D_i, S_j] = 0 \iff i \neq j$$
$$\{D_i, S_i\} = 0$$

Altogether, the full tableau has spatial complexity $4n^2+2n$. These are sometimes referred to as 'Aaronson-Gottesman 'tableaux or 'CHP' tableaux, after the software implementation by Aaronson [9].

$$\begin{array}{c} \mathcal{D}_1 \\ \vdots \\ \mathcal{D}_n \\ \mathcal{S}_1 \\ \vdots \\ \mathcal{S}_n \end{array} \left[\begin{array}{ccc|ccc|c} x_{1,1} & \cdots & x_{1,n} & z_{1,n} & \cdots & z_{1,n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & \cdots & x_{n,n} & z_{n,1} & \cdots & z_{n,n} & r_n \\ \hline x_{n+1,n} & \cdots & x_{n+1,n} & z_{n+1,1} & \cdots & z_{n+1,n} & r_{n+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{2n,1} & \cdots & x_{2n,n} & z_{2n,1} & \cdots & z_{2n,n} & r_{2n} \end{array}\right] \tag{2.13}$$

Figure 2.1: Example of a 'CHP' tableau, where the first n rows are the Destabilizers and the next n rows are the stabilizers. The $2n+1$th column gives that phase -1^{r_i} for each operator.

Simulating Gates

Gate updates for each individual operator in the tableau can be computed constant time. For example, the Hadamard transforms single qubit Pauli matrices under conjugation as

$$HPH^\dagger = \begin{cases} I & P = I \\ Z & P = X \\ X & P = Z \\ -Y & P = Y \end{cases} \tag{2.14}$$

In the symplectic form, we then have to update the ith Pauli operator as

$$x_i' z_i' = (x_i \oplus p)(z_i \oplus p) \,:\, p = x_i \oplus z_i \tag{2.15}$$

and the phase as

$$\delta' = \delta \oplus (x_i \wedge z_i) \tag{2.16}$$

Similar update rules exist for the CNOT and S gates, which together generate the n qubit Clifford group. As there are $O(n)$ operators in the tableau, and each update is constant time, gate updates overall take $O(2n)$ [2]. This is in contrast to the $O(n^{2.37})$ complexity of [5]

Simulating Measurements

The addition of the destabilizer information is used to speed up the simulation of Pauli measurements on Stabilizer states. Measuring some operator P on a stabilizer state will always produce either a deterministic outcome, or an equiprobable random outcome [1].

If the outcome is deterministic, then $\pm P$ is in the stabilizer group, and the outcome is $+1$ or -1 respectively. Using the stabilizer genereators, this allows us to write

$$[P, S_i] = 0\,\forall S_i \in \mathcal{S} \implies \prod_i c_i S_i = \pm P. \tag{2.17}$$

for binary coefficients c_i.

Checking if the outcome is deterministic takes $O(n^2)$ time in general, using the symplectic inner product to check the commutation relations [5]. However, checking which measurement outcome occurs involves computing the coefficients c_i. In the symplectic form, thiscan be rewritten as

$$Ac = P$$

where c is a binary vector, A is a matrix with each stabilizer as a column vector, P is the operator to measure, and we have dropped the phase. Solving this would require inverting the matrix A, and take time $O(n^3)$.

Aaronson & Gottesman show that for single qubit mesurements, including destabilizer information instead allows us to compute the c_i and the resulting measurement outcome in $O(n^2)$. As this is a single qubit measurement, they also show that the commutivity relation requries checking only individual bits of the stabilizer vectors, also reducing that step to $O(n)$ time.

For random measurements, from Eq. 2.17, $\exists S_i : \{S_i, P\} = 0$, and it suffices to replace this stabilizer with P, and update the other elements of the group as $S'_j = PS_j$ iff $\{S_j, P\} = 0$ [1, 2].

'Canonical' Tableaux

There are multiple possible choices of generators for each stabilizer group/state. For example, for the Bell state $|\phi^+\rangle = \frac{1}{2}(|00\rangle + |11\rangle)$

$$\mathcal{S} = \{II, XX, -YY, ZZ\} = \langle XX, -YY\rangle = \langle XX, ZZ\rangle = \langle -YY, ZZ\rangle. \tag{2.18}$$

In simulation, tableau are fixed by choice of a convention. For example, it is possible to arrive at a 'canonical' set of stabilizer generators using an algorithm which strongly resembles Gaussian elimination [7]. This method rearranges the stabilizer rows of the tableau by multiplying and swapping generators, such that the overall stabilizer group is left unchanged. Computing this canonical form requires time $O(n^3)$ [7].

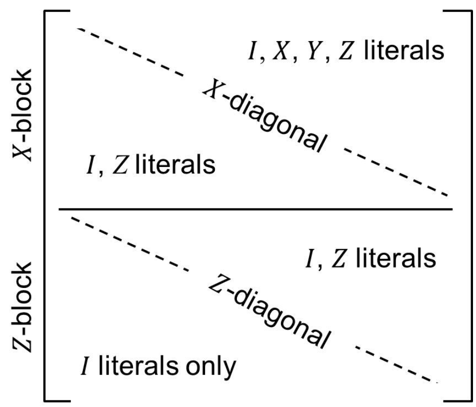

Figure 2.2: Representation of the canonical or 'row-reduced' set of stabilizer generators. Figure taken from [7].

These tableau can then be updated using the same methods as in [2], though this will in general not preserve the canonical form. Each Clifford gate will change one or two columns of the tableau, and thus an additional $O(n)$ row multiplications are required to restore it to canonical form, taking total time $O(n^2)$ per gate [10]. Importantly this canonical tableau can also be used to compute deterministic measurement outcomes in time $O(n)$, and so this method can simulate measurement outcomes more efficiently at the cost of more expensive gate updates [10].

In contrast, Aaronson & Gottesman fix the stabilizer tableau through an initial state, $|0\rangle^{\otimes n}$. The full tableau for this state looks like the identity matrix, with an additional zero-column for the phases. The tableau of a given state $|\phi\rangle$ is then built-up gate by gate using a stabilizer circuit $V : |\phi\rangle = V\,|0^{\otimes n}\rangle$.

2.1.2 Connecting Stabilizer States and Circuits

The convetion for 'CHP' stabilizer tableaux mentioned above, and the definition of stabilizer circuits given in Section 2.1, show that stabilizer states can also be defined by a stabilizer circuit and an initial state.

In [2], the authors derive examples of these 'canonical circuits', and show that its possible for any stabilizer state to be synthesised by a unique circuit acting on the $|0^{\otimes n}\rangle$ state

$$|\phi\rangle = V\,|0\rangle = H\,C\,S\,C\,S\,C\,H\,S\,C\,S\,|0^{\otimes n}\rangle \tag{2.19}$$

where each letter denotes a layer made up of only Hadamard (H), CNOT (C) or S gates. The proof is based on a sequence of operations reducing an arbitrary tableau to the identity matrix, each step of which corresponds to applying layers of a given Clifford gate [2]. As a corollary, the total number of gates in the canonical circuit for an n-qubit stabilizer state scales as $O(n\log(n))$ [2], based on previous work on synthesising $CNOT$ circuits with the $O(n\log(n))$ gates [11], and that each H and P layer can act on at most n-qubits.

A slightly simpler canonical form was derived in 2008, which allows a stabilizer circuit to be written as

$$|\phi\rangle = S\,CZ\,X\,C\,H\,|0^{\otimes n}\rangle \tag{2.20}$$

where the CZ and X layers are made up of Controlled-Z gates and Pauli X gates, respectively [3]. This circuit follows from the work of [5], who showed that any stabilizer state can be written as

$$|\phi\rangle = \frac{1}{\sqrt{2^k}}\sum_{x\in\mathcal{K}} i^{f(x)}\,|x\rangle\,. \tag{2.21}$$

In this equation, $\mathcal{K} \subseteq \mathbb{Z}_2^n$ is an affine subspace of dimension k, and $f(x)$ is a binary function evaluated $\bmod 4$. Thus, a stabilizer state is always a uniform superposition of computational basis strings, with individual phases ±i, ±1. The affine space $\mathcal{K}$ has the form

$$\mathcal{K} = \{Gu + h\}$$

for k-bit binary vectors u, an $n \times k$ binary matrix G, and an n-bit binary 'shift-vector' h.

Van den Nest notes that this representation can be directly translated into a stabilizer circuit; we begin by applying H to the first k qubits to initialize the state $\sum_u |u\rangle \otimes \left|0^{\otimes n-k}\right\rangle$. We then apply CNOTs to prepare $\sum_u |Gu\rangle$, and finally Pauli Xs to preapre $\sum_u |Gu \oplus h\rangle$ [3].

The phases can be further decomposed into two linear and quadratic binary functions $l, q : \mathbb{Z}_2^n \to \mathbb{Z}_2$, such that $i^{q(x)} = i^{l(x)}(-1)^{q(x)}$. The linear terms correpsond to single qubit phase gates, which can be generated by the S gate, and the quadratic terms to two-qubit phase gates, generated by the CZ [3]. Thus,

$$|\phi\rangle = \sum_{x \in \mathcal{K}} i^{l(x)}(-1)^{q(x)} |x\rangle = S\ CZ\ X\ C\ H\ |0\rangle \tag{2.22}$$

While [3] showed that these simpler canonical circuits exist, an algorithm to compute them first introduced in 2012 [7]. This method allowed such a circuit to be read off from the 'canonical' set of stabilizer generators introduced in Section 2.1.1.

2.1.3 Computing Inner Products

The final task we might consider in simulating stabilizer circuits is the problem of computing probability amplitudes $P(x) = |\langle x|\phi\rangle|^2$. As computational states are also stabilizer states, this corresponds more broadly to computing inner products between stabilizer states.

From the affine space form in Eq. 2.21, we can see that

$$\langle \varphi|\phi\rangle = \frac{1}{\sqrt{2^{k+k'}}} \sum_{x \in \mathcal{K} \cap \mathcal{K}'} i^{f(x) - f'(x)} \tag{2.23}$$

and the problem of computing the inner product corresponds to computing the magnitude of an 'exponential sum' of phase differences (±i, ±1) for each string x in the intersection of the two affine spaces [8]. From inspection, we can see that

$$|\sum_x i^{f(x)-f'(x)}| = \begin{cases} 0 \\ 2^{s/2} : s \in \{0, 1, \ldots, n\} \end{cases}$$

This sum can be solved in $O(n^3)$ time, using an algorithm developed by Sergey Bravyi [8, 12, 13]. An algorithm for computing this intersection was also described in [8], which we discuss further in Section 2.2.3.

Alternatively, the inner product can also be computed using the stabilizer generators directly. Consider two states $|\phi\rangle, |\varphi\rangle$ with respective generators G_i, H_i. If $\exists i, j : G_i = -H_j$, the states are orthogonal and the inner product is 0. Otherwise, the inner product is given by 2^{-s}, where s the number of generators $G_i \notin \{H_i\}$.

While there are multiple choices of stabilizer generators, we note that inner products are invariant under unitary operations U as

$$\langle\varphi|\phi\rangle = \langle\varphi|U^\dagger U|\phi\rangle.$$

Thus, given the canonical circuit $V : |\varphi\rangle = V|0^{\otimes n}\rangle$

$$\langle\varphi|\phi\rangle = \langle\varphi|V^\dagger V|\phi\rangle = \langle 0^{\otimes n}|V|\phi\rangle.$$

Each stabilizer G'_i of $|0^{\otimes n}\rangle$ has a single Pauli Z operator acting on qubit i. By simplifying the stabilizer H'_i of $V|\phi\rangle$ using Gaussian elimination, then we have

$$|\langle 0^{\otimes n}|V|\phi\rangle| = \begin{cases} 0 & \exists H'_i = \bigotimes_i Z_i \\ 2^{-s} & \exists H'_i : \{H'_i, G'_i\} = 0 \end{cases} \tag{2.24}$$

where s is the number of stabilizers that anticommute with the corresponding stabilizer G'_i [2]. The second case arises as if $\{H'_i, G'_i\} = 0$, then H'_i acts as either Pauli X or Y on qubit i. Thus, the qubit is in state $|\pm 1\rangle$ or $|\pm \mathrm{i}\rangle$, and $\langle 0|\pm i, 1\rangle = \frac{1}{\sqrt{2}}$. Because this method involves computing the canonical circuit and then applying gaussian elimination, it runs in time $O(n^3)$.

The first implementation of this algorithm was given in [7], where the authors first use their canonical form to construct a 'basis circuit' $B : |\varphi\rangle = B\,|b\rangle$ for some computational state $|b\rangle$, and then compute $\langle b|B|\phi\rangle$ using the same method outlined above [7].

2.2 Results

The main result of this chapter is to introduce two new classical representations of stabilizer states developed in collaboration with Sergey Bravyi [12]. We will discuss their algorithmic complexity, and implementation in software. We will also discuss the implementation of a classical datastructure based on affine spaces, introduced in [8].

Finally, we present data evaluating the performance of all three methods. For the affine space representation, we benchmark against existing implementations in MATLAB [8]. For the two novel representations, we present data comparing their performance to two pieces of existing stabilizer circuit simulation software [2, 6].

2.2.1 Novel Representations of Stabilizer States

Existing classical simulators have two important limitations. One is that they focus only on implementations of single qubit Pauli measurements made in the Z basis. Multi-qubit measurements, or measurements in different bases, need to be built up in sequence, or involve applying additional basis changes gates like H and S, respectively.

These simulators also do not track global phase information. For the case of simulating individual stabilizer circuits, this is sufficient as global phase does not affect measurement outcomes. However, if we wish to extend our methods to simulating superpositions of stabilizer states, then phase differences between terms in the decomposition must be recorded [10].

Here, we present two data structures, which we call the 'DCH' and 'CH' forms.

Definition 2.1. DCH Representation:

Any stabilizer state $|\phi\rangle$ can be written as

$$|\phi\rangle = \omega^e U_D U_{CNOT} U_H |s\rangle \tag{2.25}$$

where U_D is a diagonal Clifford unitary such that

$$U_D |x\rangle = i^{f(x)} |x\rangle,$$

U_{CNOT} is a layer of $CNOT$ gates, U_H is a layer of Hadamard gates, acting on a computational state $|s\rangle$, and with a global phase factor w^e where $\omega = \sqrt{\mathrm{i}}$ and $e \in \mathbb{Z}_8$.

Any diagonal Clifford matrix of the form U_D is described by its 'weighted polynomial' $f(x)$, evaluated mod 4, which can be expanded into linear and quadratic terms as

$$f(x) = \sum_i a_i x_i + 2\sum_{c,t} x_j x_k \mod 4 = L(x) + 2Q(x)$$

where the coefficients $a_i \in \mathbb{X}_4$ [3, 14]. This was also the expansion used in the definition of the affine space representation in Eq. 2.22.

We observe that the linear terms can be entirely generated by the S, Z and $S^\dagger$ gates acting on single qubits, and the quadratic terms by CZ gates acting on pairs of qubits [14]. Thus, any unitary U_D can be built up of these gates. As a corollary, we note that these 'DCH' circuits can be obtained from the 7-stage circuits given in Eq. 2.20, by commuting the X layer through to the beginning of the circuit and acting it on the $|0^{\otimes n}\rangle$ initial state. [3].

The computational string s can be encoded as an n-bit binary row-vector. This is also true of the Hadamard layer, which can be expanded in terms of a binary vector h as

$$U_H = \bigotimes_{i=1}^{n} H^{h_i}. \tag{2.26}$$

A $CNOT$ gate controlled on qubit c and targeting qubit t transforms the computational basis states as

$$CNOT_{c,t} |x\rangle = CNOT_{c,t} \bigotimes_{i=1}^{n} |x_i\rangle = \bigotimes_{i=1}^{n} |x_i \oplus \delta_{i,t} x_c\rangle$$

i.e. it adds the value of bit c to bit t, modulo 2. Thus, we can encode the action of U_{CNOT} as an $n \times n$ binary matrix E which is equal to the identity matrix, with an additional one at $E_{c,t}$, such that

$$CNOT_{c,t} \left|x\right\rangle = \left|xE\right\rangle \,:\, E_{i,j} = \begin{cases} 1 & i = j \\ 1 & i = c, j = t \\ 0 & otherwise \end{cases} \tag{2.27}$$

We can then build up U_{CNOT} from successicve CNOT gates as

$$U_{CNOT} \left|x\right\rangle = \left|xE_1E2E_3 \dots E_m\right\rangle \equiv \left|xW\right\rangle \tag{2.28}$$

where $W = E_1E_2 \cdots E_n$ is the matrix representing the full circuit, obtained by successive right multiplication of the matrices encoding a single CNOT.

Finally, we need to encode the action of U_D. The phase resulting from a single qubit diagonal Clifford is conditional on the qubits being in the $\left|1\right\rangle$ state. Thus, we can write the linear part of the weighted polynomial as Lx^T for some row-vector L of integers $\bmod 4$, which we call the linear phase vector. Each value in L can be stored using just 2 bits.

Each gate $CZ_{i,j}$ between qubits i and j also contributes a factor of 2 to the overall phase, conditioned on the ith and jth qubits being in the $\left|1\right\rangle$ state. For a given computational string x, the overall phase from the CZ gates is thus $2\sum_{i,j:CZ_{i,j}} x_i x_j$.

We can encode the action of the CZ gates using an $n \times n$ symmetic binary matrix Q where $Q_{i,j} = Q_{j,i} = 1$ if we apply $CZ_{i,j}$, and zero otherwise, which we call the quadratic phase matrix. We can then compute the phase from the CZ gates as

$$\begin{aligned}
xMx^t &= \sum_p x_p \left(Qx^T\right) \\
&= \sum_p x_p \left(\sum_q Q_{p,q} x_q\right) \\
&= \sum_{p,q} x_p x_q Q_{p,q} \\
&= 2\sum_p \sum_{q>p} x_p x_q Q_{p,q} \\
&= 2 \sum_{i,j:CZ_{i,j}\in U_D} x_i x_j
\end{aligned}$$

where the last line follows from the definition of the matrix Q. Altogether, this allows us to write [8]

$$U_D \left|x\right\rangle = i^{f(x)} \left|x\right\rangle = i^{Lx^T + xQx^T} \left|x\right\rangle = i^{xBx^T} \left|x\right\rangle \tag{2.29}$$

where B is a matrix such that $B_{ii} = L_i$, $B_{i,j} = Q_{i,j}$, as by definition Q has zero diagonal. We refer to B as simply the phase matrix, with diagonal elements stored mod 4 and off-diagonal elements stored mod 2.

Finally, we include the global phase factor, an integer modulo 8 and stored using just three bits, meaning overall the DCH representation is specified by the tuple (e, s, h, B, W). The spatial complexity is thus $\Theta(n^2)$. In order to optimize certain subroutines, which we discuss later in this section, we also store a copy of W^{-1}, the inverse of the CNOT matrix, and W^T, the transpose of the CNOT matrix. We further introduce two variables $p \in \{0, 1, \ldots, n\}$, $\epsilon = 0, 1$, which are used to ensure normalisation of the DCH state under certain operations. Together with the phase e, they define a coefficient we denote $c = 2^{-p/2}\epsilon\omega^e$. We store p as an unsigned integer, and ϵ as a single binary bit. Overall, then, the DCH form requires roughly $4n^2 + 4n + 36$ bits of memory.

Definition 2.2. CH Representation:

Any stabilizer state $|\phi\rangle$ can be written as

$$|\phi\rangle = \omega^e U_C U_H \left|s\right\rangle \tag{2.30}$$

where U_C is a Clifford operator such that

$$U_C \left|0^{\otimes n}\right\rangle = \left|0^{\otimes n}\right\rangle, \tag{2.31}$$

U_H is a layer of H gates, $|s\rangle$ is a computational basis state, and with global phase factor ω^e where $\omega = \sqrt{i}$ and $e \in \mathbb{Z}_8$.

The CH representation is based on a notion of a 'control-type' Clifford operator, which stabilizes the all zero computational basis state. Examples of control-type Clifford gates include the S, CZ and $CNOT$ gates. A control type operator U_C can be obtained from the DCH form, for example, by concatenating U_D and U_{CNOT} layers. Thus, we can see that any stabilizer state can be generated by a CH-type circuit.

Similarly to above, we encode the initial computational basis state s and the Hadamard layer U_H as n-bit binary row-vectors. The control-type layer we then encode using a stabilizer tableau, made up of $2n$ Pauli operators $U_C^\dagger X_i U_C$ and $U_C^\dagger Z_i U_C$. This tableau resembles a CHP-type tableau for the state $U_C \left|0^{\otimes n}\right\rangle$, where the Pauli X entries are the destabilizers and the Pauli Z entries are the stabilizers. Alternatively, we can see this as characterising the operator U_C by its action on the generators of the Pauli group.

Using a normal CHP-tableau, each Pauli would require $2n+1$ bits to encode. However, from the definition of the control-type operators, $U_C^\dagger Z_i U_C$ will never result in a Pauli X or Y operator, as otherwise $U_C \left|0^{\otimes n}\right\rangle \neq \left|0^{\otimes n}\right\rangle$. Thus, we can ignore the n 'x-bits' and phasebits of each of the Pauli Z rows. Specifically, we write

$$U_C^\dagger Z_j U_C = \bigotimes_{k=1}^{n} Z^{G_{j,k}} \tag{2.32}$$

$$U_C^\dagger X_j U_C = i^{\gamma_j} \bigotimes_{k=1}^{n} X^{F_{j,k}} Z^{M_{j,k}} \tag{2.33}$$

for binary matrices G, F, M, and a phase vector $\gamma : \gamma_i \in \mathbb{Z}_4$, as $Y = -\mathrm{i}XZ$. Note that this differs from the CHP method, where the string 11 encodes Pauli Y directly, without tracking a separate complex phase.

Finally, we again require three further bits to encode the global phase, and the CH representation is thus given by the tuple (e,s,h,G,M,F). Overall, the CH form also has spatial complexity $\theta(n^2)$. In order to optimize some subroutines, we additionally store copies of M^T and F^T, and again include the variables p and ϵ, requiring a total of $5n^2+4n+36$ bits of memory.

2.2.2 Simulating circuits with the DCH and CH Representations

In this section, we will outline how to update the DCH and CH representations under different stabilizer circuit operations, and how to compute the inner product. For both methods, gate updates can be split into two types: control-type operators, and Hadamard gates. The technique for treating the Hadamard also shares some aspects with applying Pauli projectors to the states, and deciding measurement outcomes. Some of the techniques employed will be common to both representations, differing only in their implementation on the underlying datastrucutre.

Gate updates: The DCH Representation

In the DCH picture, the complexity of a gate depends on whether it is a $CNOT$, or a diagonal Clifford operator S, Z, $S^\dagger$ or CZ. Diagonal gates can be simulated in constant time $O(1)$ by simply updating the linear or quadratic part of the diagonal layer. Single qubit gates applyed to qubit i update the ith element of the linear phase vector D, as they contribute only to the linear part of the weighted polynomial. Thus, we have

$$S_i\,|\phi\rangle \implies B_{i,i} \leftarrow B_{i,i}+1 \mod 4 \tag{2.34}$$

$$Z_i\,|\phi\rangle = S^2\,|\phi\rangle \implies B_{i,i} \leftarrow B_{i,i}+2 \mod 4 \tag{2.35}$$

$$S_i^\dagger = s^3\,|\phi\rangle \implies B_{i,i} \leftarrow B_{i,i}+3 \mod 4. \tag{2.36}$$

Similarly, a CZ gate applied to qubits i and j will change entries $B_{i,j}$ and $B_{j,i}$ of the quadratic phase matrix as

$$B'_{i,j} \leftarrow B_{i,j} \oplus 1, \tag{2.37}$$

and equivalently for $B_{j,i}$.

For $CNOT$ gates, we first need to commute them past the diagonal layer before updating U_{CNOT}. The overall effect on the DCH form is then

$$\begin{aligned}CNOT_{c,t}\left|\phi\right\rangle &= i^e\, CNOT_{c,t} U_D U_{CNOT} U_H \left|s\right\rangle \\ &= i^e\, CNOT_{c,t} U_D CNOT^{\dagger}_{c,t} U'_{CNOT} U_H \left|s\right\rangle \\ &= i^e\, U'_D U'_{CNOT} U_H \left|s\right\rangle \end{aligned} \tag{2.38}$$

updating U_{CNOT} using matrix multiplication as in Eq. 2.28, and where the last line relies on the following Lemma:

Lemma 1 *For any CNOT circuit U_{CNOT} and any diagonal Clifford circuit U_D, $U^{\dagger}_{CNOT} U_D U_{CNOT}$ is also a diagonal Clifford circuit U'_D with corresponding phase matrix $B' = WBW^T$.*

Proof of Lemma 1. Consider the case of a single CNOT gate on qubits c and t. We have

$$\begin{aligned}CNOT^{\dagger}_{c,t} U_D CNOT_{c,t}\left|x\right\rangle &= CNOT_{c,t} U_D CNOT_{c,t} \\ &= CNOT_{c,t} U_D \left|x + x_j e_k \mod 2\right\rangle \\ &= \mathrm{i}^{f(x+x_j e_k)} CNOT_{c,t} \left|x + x_j e_k \mod 2\right\rangle \\ &= \mathrm{i}^{f(x+x_j e_k)} \left|x + 2x_j e_k \mod 2\right\rangle \\ &= \mathrm{i}^{f(x+x_j e_k)} \left|x\right\rangle \end{aligned} \tag{2.39}$$

where we have used the fact that a single CNOT gate is self-inverse. Thus, $CNOT^{\dagger}_{c,t} U_D CNOT_{c,t}$ acts as a diagonal Clifford gate. As any CNOT circuit is a sequence of individual CNOT gates, $U^{\dagger}_C U_D U_C$ is also a diagonal Cliford circuit.

Using the matrix representation of the action of U_C, it is easy to show that

$$\begin{aligned}U^{\dagger}_C U_D U_C &= U_{C^{\dagger}} U_D \left|xW\right\rangle \\ &= \mathrm{i}^{(xW)B(xW)^T} U^{\dagger}_C \left|xW\right\rangle \\ &= \mathrm{i}^{(xW)B(xW)^T} \left|xWW^{-1}\right\rangle \\ &= \mathrm{i}^{xWBW^T x^t} \left|x\right\rangle, \end{aligned} \tag{2.40}$$

completing the proof. □

In general, computing the updated form of $U_{CNOT}^{\dagger} U_D U_{CNOT}$ would require time $O(n^2)$. However, for the case of a single gate $CNOT_{c,t}$, recall that the matrix E differs from the identity matrix at a single element, $E_{c,t} = 1$. This allows us to simplify the updates as

$$\left[E_{c,t} B E_{c,t}^T\right]_{i,j} = \sum_{k,l} E_{i,k} E_{j,l} B_{k,l} = \begin{cases} B_{i,j} & i,j \neq c \\ B_{c,j} + B_{t,j} & i = c,\, j \neq c \\ B_{i,c} + B_{i,t} & i \neq c,\, j = c \\ B_{c,c} + B_{t,t} + B_{c,t} + B_{t,c} & i = j = c \end{cases} \tag{2.41}$$

Additionally, we need to update W and W^{-1}. The inverse of U_C is the same sequence of CNOT gates, applied in reverse order. Thus, we have $W^{-1} = E_m E_{m-1} \cdots E_1$, and we update W^{-1} by left multiplication with the CNOT matrix. Using the definition of the CNOT matrix,

$$[WF]_{ij} = \quad \sum_k W_{i,k} F_{k,j} = \quad \begin{cases} W_{i,j} & j \neq t \\ W_{i,c} + W_{i,t} & j = t \end{cases}$$

$$[FW^{-1}]_{i,j} = \quad \sum_k F_{i,k} W_{k,j}^{-1} = \quad \begin{cases} W_{i,k}^{-1} & i \neq c \\ W_{c,j}^{-1} + W_{t,j}^{-1} & i = c \end{cases}$$

updating just the target column and the control row of W and W^{-1}, respectively.

Putting together these two pieces, we thus have

$$\begin{aligned} CNOT_{c,t}\,|\phi\rangle \implies & \mathrm{row}_c(B) \leftarrow \mathrm{row}_c(B) + \mathrm{row}_t(B) \\ & \mathrm{col}_c(B) \leftarrow \mathrm{col}_c(B) + \mathrm{col}_t(B) \\ & \mathrm{col}_t(W) \leftarrow \mathrm{col}_t(W) + \mathrm{col}_c(W) \\ & \mathrm{row}_c(W^{-1}) \leftarrow \mathrm{row}_c(W^{-1}) + \mathrm{row}_t(W^{-1}) \end{aligned} \tag{2.42}$$

These updates take $O(n)$ time, as we update a constant number of rows and columns.

Gate Updates: The CH Representaiton

For the CH representation, whenver we apply a new control-type operator C we need to update the stabilizer tableau by conjugating each element $U_C^\dagger X_i$, $Z_i U_C$ with the matrix C. This can be implemented using the usual rules for updating Pauli operators under Clifford operations, with the additional note that we have to adjust the updates to correctly track the phases of the Pauli X terms, and that we are conjugating as $U_C^{-1} P U_C$, rather than $U_C P U_C^{-1}$.

The control-type circuit is built out of individal operations $U_C = C_m C_{m-1} \dots C_1$. We we update U_C with some new operator C_{m+1}, change the tableau as

$$(C_{m+1} U_C)^\dagger P C_{m+1} U_C = U_C^\dagger \left(C_{m+1}^\dagger P C_{m+1} \right) U_C. \tag{2.43}$$

Because C_{m+1} is a Clifford operator, the term $C_{m+1}^\dagger P C_{m+1}$ is also a Pauli operator $P' = i^\alpha \prod_{i=1}^n X_i^{x_i} Z_i^{z_i}$ for some phase α and bit strings x and z. This allows us to write

$$\begin{aligned}
U_C^\dagger C_{m+1}^\dagger P C_{m+1} U_C &= i^\alpha U_C^\dagger \left(\prod_{i=1}^n X_i^{x_i} Z_i^{z_i} \right) U_C \\
&= i^\alpha \prod_{i=1}^n U_C^\dagger X_i^{x_i} Z_i^{z_i} U_C \\
&= i^\alpha \prod_{i=1}^n U_C^\dagger X_i^{x_i} U_C U_C^\dagger Z_i^{z_i} U_C \\
&= i^\alpha \prod_{i=1}^n \left(i^{\gamma_i} \prod_{j=1}^n X_i^{F_{i,j}} Z_i^{M_{i,j}} \right)^{x_i} \left(\prod_{i=1}^n Z_i^{G_{i,j}} \right)^{z_i} \qquad (2.44)
\end{aligned}$$

where the last line is a product of terms from the tableau of U_C.

As an example, consider the action of the S gate. For each term, we have

$$S^\dagger P S = \begin{cases} I & \to & I \\ X & \to & -\mathrm{i} XZ \\ Z & \to & Z \end{cases}$$

The Z stabilizers are unchanged, and the X/Y stabilizers flip from $\mathrm{i}^\alpha X^a Z^b$ to $\mathrm{i}^{\alpha+3} X^a Z^{b\oplus 1}$. On the tableau, acting an S gate on qubit q will only act non-trivially on the term $U_C^\dagger X_q U_C$, and thus

$$U_C^\dagger S^\dagger X_q S_q U_C = i^3 U_C^\dagger X_q U_C U_C^\dagger Z_q U_C \implies \begin{cases} \mathrm{row}_q(M) & \leftarrow & \mathrm{row}_q(M) + \mathrm{row}_q(G) \\ \gamma_q & \leftarrow & \gamma_q + 3 \mod 4 \end{cases}$$

We can compute the updates for CZ and CX in the same way, giving overall gate update rules

$$\begin{aligned} S & \begin{cases} \mathrm{row}_q(M) & \leftarrow & \mathrm{row}_q(M) + \mathrm{row}_q(G) \\ \gamma_q & \leftarrow & \gamma_q + 3 \mod 4 \end{cases} \\ CZ_{q,p} & \begin{cases} \mathrm{row}_q(M) & \leftarrow & \mathrm{row}_q(M) + \mathrm{row}_p(G) \\ \mathrm{row}_p(M) & \leftarrow & \mathrm{row}_p(M) + \mathrm{row}_q(G) \end{cases} \\ CNOT_{q,p} & \begin{cases} \mathrm{row}_p(G) & \leftarrow & \mathrm{row}_p(G) + \mathrm{row}_q(G) \\ \mathrm{row}_q(F) & \leftarrow & \mathrm{row}_q(F) + \mathrm{row}_p(G) \\ \mathrm{row}_q(M) & \leftarrow & \mathrm{row}_q(M) + \mathrm{row}_p(M) \\ \gamma_q & \leftarrow & \gamma_q + \gamma_p + 2\sum_i M_{q,i} F_{p,i} \mod 4 \end{cases} \end{aligned} \tag{2.45}$$

Where on the final line, we apply an extra phase correction that results from reordering the Pauli operators in the CNOT updates. This arises as, expanding out the action on the X stabilizers,

$$\begin{aligned} U_C^\dagger CNOT_{q,p} X_q CNOT_{q,p} U_C &= U_C^\dagger X_q X_p U_C \\ &= U_C^\dagger X_q U_C U_C^\dagger X_p U_C \\ &= i^{\gamma_q + \gamma_p} \prod_{i=1}^{n} X_i^{F_{q,i}} Z_i^{M_{q,i}} X_i^{F_{p,i}} Z_i^{M_{p,i}} \end{aligned}$$

and we pick up an extra phase of -1 each time $M_{q,i} = F_{p,i} = 1$ as $ZX = -XZ$. All of these updates take time $O(n)$, as we are updating the n-element rows of $n \times n$ matrices.

Hadamard gates and Pauli Measurements

Simulating Hadamard gates and arbitrary Pauli measurements is done using an algorithm with the same general structure in the DCH and CH representation. These routines employ an algorithm developed by Sergey Bravyi for application to the CH method, which can also be applied to the DCH case.

Hadamard gates and Pauli projectors can both be written as $\frac{1}{\sqrt{2}}(P_1+P_2)$ for some Pauli operators P_1, P_2. In the Hadamard case, we have $P_1 = X_i, P_2 = Z_i$, and in the projector case $P_1 = I, P_2 = P$. Given this structure, we then commute these operators through to the comutational basis state

$$\begin{aligned}\epsilon 2^{-p/2} i^e \frac{1}{\sqrt{2}}(P_1+P_2) U_C U_H |s\rangle &= \epsilon 2^{-(p+1)/2} i^e U_C U_H (P_1'+P_2') |s\rangle \\ &= \epsilon 2^{-(p+1)/2} i^{e'} U_C U_H \left(|t\rangle + i^\beta |u\rangle\right)\end{aligned}$$

where $P_{1,2}'$ can be efficiently computed as the circuit $U_C U_H$ is Clifford, $\beta \in Z_4$, and t and u are two new computational basis states obtained from the action of $P_{1,2}$ on s. Note that we are writing U_C here as a shorthand, as the circuit $U_D U_{CNOT}$ in the DCH represntation is also a control-type unitary.

Once in this form, we employ the following proposition, called Proposition 4 in [12]:

Proposition 1 *Given a stabilizer state $U_H\left(|t\rangle + i^\beta |u\rangle\right)$, we can construct a circuit W_C built out of $CNOT$, CZ and S gates, and a new Hadamard circuit U_H', such that we can write*

$$U_H\left(|t\rangle + i^\beta |u\rangle\right) = i^{\beta'} W_C U_H' |s'\rangle.$$

As a means of proving this proposition, we will go through and construct W_C and U_H'.

Proof of Proposition 1. Firstly, consider the case $t = u$. Then we have $s' = t$, and the result depends on the phase β. If $\beta = 0$, then the state is unchaged. If $\beta = 1,3$, then we have

$$\frac{1}{\sqrt{2}} U_H \left(1 + i^\beta\right) |s'\rangle = \frac{(1 \pm \mathrm{i})}{\sqrt{2}} U_H |s'\rangle$$

and it suffices to update the global phase term

$$\begin{aligned}\beta = 1 &\implies e \leftarrow e+1 \mod 8 \\ \beta = 3 &\implies e \leftarrow e+7 \mod 8\end{aligned}$$

Finally, if $\beta = 2$, we have $|s'\rangle - |s'\rangle$ and the state is cancelled out. We denote this by setting $\epsilon \leftarrow 0$. This only arises in the case of applying a Pauli projector that is orthogonal to the state.

If $t \neq u$, then we instead note that we can always define some sequence of $CNOT$ gates V_C such that

$$|t\rangle = V_C |y\rangle \quad |u\rangle = V_C |z\rangle$$

where y, z are two n-bit binary strings such that $y_i = z_i$ everywhere except bit q where $z_q = y_q + 1$. We can assume without loss of generality that $\exists q : t_q = 0, u_q = 1$, else we swap the two strings and update the phase accordingly. Then

$$V_C = \prod_{i: i \neq q,\, t_i \neq u_i} CNOT_{q,i}$$

and we can commute this circuit past U_H to obtain a new circuit V_C'. We can always freely pick $q : v_q = 0$, unless $v_i = 1 \forall i$, and thus V_C' is given by:

$$V_C' = \begin{cases} \prod_{i \neq q,\, v_i = 0} CNOT_{q,i} \prod_{i \neq q,\, v_i = 1} CZ_{q,i} & v_q = 0 \\ \prod_{i \neq q} CNOT_{i,q} & v_i = 1 \forall i \end{cases}$$

We complete the proof by considering the action of U_H on the new strings $|y\rangle + \mathrm{i}^{\beta} |z\rangle$. Again, fixing $y_q = 0, z_q = 1$, we can write

$$U_H \left(|y\rangle + i^{\beta} |z\rangle \right) = H^{v_q} S^{\beta} |+\rangle = \omega^a S_q^b H_q^c |d\rangle$$

for some bits $a, b, c, d \in \{0, 1\}$ that can be computed exactly from the values of β and v_q.

This completes the proof of Proposition 1, where $W_C = V_C' S_q^b$, $U_H' = U_H H_q^{v_q + c}$, and $s' = y \oplus d e_q$, where e_q is an indicator vector that is 1 at position q and zero elsewhere. $\square$

Computing the circuits W_C and U_H' given the two strings t, u takes time $O(n)$, as it involves inspecting the $n - bit$ strings t, u and v. Given this proposition, we now need to show how to commute a Pauli operator through the stabilizer circuit in both representations, and then how to update the layers $U_D U_{CNOT}$ and U_C by right multiplication with the circuit W_C. This can be rewritten in terms of binary vector-matrix multiplication, and we introduce the following notation:

$$\prod_{i=1}^{n} X_i^{x_i} \equiv X(x) \quad \prod_i Z_i^{z_i} \equiv Z(z)$$

for binary strings x and z.

Applying Proposition 1 to DCH States

When commuting a Pauli operator P through a Clifford circuit, it is important to fix the ordering of the X and Z terms, as Pauli operators can be expanded out as $P = i^a X(x)Z(z) = i^a(-1)^{x\cdot z} Z(z)X(x)$, as $XZ = -ZX$, and where we use $x \cdot z$ to denote the binary inner product

$$x \cdot z = \sum_i x_i z_i \mod 2.$$

In the DCH case, we fix $P = i^a Z(z)X(z)$, as this simplifies the phase terms when commuting past the U_D layer.

Pauli Z terms are unchanged by the DCH layer as they commute with diagonal Clifford oeprtors. To commute the X terms past the U_D layer, we use $X(x)U_D = U_D\left(U_D^\dagger X(x) U_D\right)$, and compute the new Pauli $U_D^\dagger P' U_D = i^{a'} Z(z')X(x)$.

The diagonal entries of the phase matrix B contribute as

$$(S^{B_{ii}})^\dagger X_i^{x_i} S^{B_{ii}} = \begin{cases} S^\dagger X^{x-i} S & \rightarrow & i(ZX)^{x_i} \\ ZXZ & \rightarrow & -X^{x_i} \\ SXS^\dagger & \rightarrow & -i(ZX)^{x_i} \end{cases} = i^{B_{ii}} X^{x_i} Z^{x_i B_{ii} \pmod 2}$$

We also have that $CZ(X \otimes I)CZ = XZ$, $CZ(I \otimes X)CZ = ZX$, i.e. a CZ conjugated with a Pauli X on the control (target) qubit adds a Pauli Z on the target (control) qubit. Qubit i picks up a Z operator each time there is a CZ between qubits i and j, and an X acting on qubit j. Using the off-diaognal entries of the phase matrix, we can write

$$Z_i^{z_i'} : z' = \sum_{j \neq i} x_j B_{j,i} \mod 2$$

Combining this with the fact we also pick up a Pauli Z from the diagonal if $B_{ii} = 1,3$, we can write $z_i = aB \mod 2$. Finally, we need to consider the extra -1 phase contributions for each $i : x_i z'_i = 1$, as a result of preserving the ordering of P'. Together with the diagonal phases, this can be simplified to

$$\sum_i x_i B_{ii} + 2\sum_i x_i \sum_{j\neq i} x_j B_{j,i} = xBx^T \mod 4$$

Overall then, we have

$$U_D^\dagger X(x) U_D = i^{xMx^T} Z(xM) X(x) \tag{2.46}$$

A similar result applies to commuting a Pauli operator through the U_{CNOT} layer. $CNOT$ has the property that it maps $I_c Z_t \to Z_c Z_t$ and $X_c I_t \to X_c X_t$ under conjugation. Thus, we can compute the new strings x', z' by applying an appropriate CNOT matrix.

For the X bits, we can simply apply $x' = xW^{-1}$, where we use the inverse matrix as we are computing $U_{CNOT}^\dagger X U_{CNOT}$ and thus the binary string is subject to the inverse sequence of CNOT gates.

For the string z, we need to apply a CNOT matrix with the controls and targets swapped. From the definition given in Eq. 2.27, we can see that if the binary matrix E encodes $CNOT_{c,t}$, then $CNOT_{t,c}$ is encoded by E^T. We then update the strign z under the sequence $E_m^t E_{m-1}^t \dots E_1^t = W^T$. Overall then, we have

$$U_{CNOT}^\dagger i^a Z(z) X(x) U_{CNOT} = i^a Z(zW^T) X(xW^{-1}). \tag{2.47}$$

As mentioned, we store copies of W^{-1} and W^T with the DCH representation. This helps to avoid the $O(n^3)$ computational cost associated with inverting W, and the $O(n^2)$ cost of transposing W. Overall then, we can compute this update in time $O(n^2)$.

Finally, to commute a Pauli operator past the U_H layer, we note that the Hadamard acts as

$$\begin{aligned} HXH &\to Z \\ HZH &\to X \\ HZXH &\to -ZX \end{aligned}$$

The x and z bits are only changed for those bits where $v_i = 1$, and so we can write

$$z_i' = z_i(1 - v_i) + x_i v_i$$

and vice-versa for the x bits. In terms of boolean operations, this can also be written as $z_i' = z_i \wedge \neg v_i \oplus x_i \wedge v_i$. Finally, we have the phase correction whenere $x_i = z_i = z_i = 1$. Thus, overall, we can write

$$U_H^\dagger i^a Z(z) X(z) U_H = i^{a + v\cdot(x\wedge z)} Z(z \wedge \neg v \oplus x \wedge v) X(x \wedge v) \tag{2.48}$$

and this update takes time $O(n)$ to compute.

To complete the application of Propositon 1, we also need to be able to update $U_D U_{CNOT}$ by right multiplication with W_C. We can split $W_C = W_{CNOT} W_D$, where W_D is made up of CZ gates and the single S gate.

The U_{CNOT} layer updates as $U_{CNOT}' = U_{CNOT} W_{CNOT}$. Because of the ordering of the circuits, we here update the matrix W by left multiplication, and update $W^\dagger$ by right multiplication. Thus, for each $CNOT$ gate in W_{CNOT}, we update the columns of W^{-1} and the rows of W using the rules given in Eq. 2.47.

We then need to commute the diagonal layer W_D past U_{CNOT}'. We can do this by adapting Eq. 2.40 to instead compute $U_{CNOT} W_D U_{CNOT}^\dagger$, giving a new phase matrix $C' = W^{-1} C W^{-1}$ where C encodes the action of W_D. This computation again benefits from storing W^{-1} in the DCH information, and can be further optimized by noting that many entries of C are zero. Finally, we can combine the two phase matrices by simplying adding all the elements, keeping the diagonal entries $\bmod 4$ and the off-diagonal entries $\bmod 2$. All together, including the Pauli updates, applying Proposition 1 takes time $O(n^2)$.

Applying Proposition 1 to CH States

Commuting a Pauli operator through the layers of the CH circuit can be done using methods already introduced in previous sections. Distinctly from the DCH case, here we fix $P = i^a X(x) Z(z)$.

To commute a Pauli past the U_C layer, we need to compute $U_C^\dagger P U_C$, and this can be expanded out in a similar manner to Eq. 2.44. This gives

$$\begin{aligned} U_C^\dagger X(x) U_C = &\quad \textstyle\prod_{i:x_i=1} U_C^\dagger X_i U_C \\ U_C^\dagger Z(z) U_C &\quad \textstyle\prod_{i:z_i=1} U_C^\dagger Z_i U_C \end{aligned}$$

We can thus build up P' term by term as

$$\begin{aligned} U_C^\dagger P U_C &= \prod_{j=1}^{n} x_j \left(\mathrm{i}^{\gamma_j} X(\mathrm{row}_j(F)) Z(\mathrm{row}_j(M))\right) \prod_{j=1}^{n} z_i \left(Z(\mathrm{row}_j(G))\right) \\ &= \mathrm{i}^{\sum_{j=1}^{n} x_j\gamma_j + 2\sum_{j=1}^{n}\sum_{k>j} x_j x_k (\mathrm{row}_j(F)\cdot \mathrm{row}_j(M))} X(xF)Z(xM+zG) \\ &= \mathrm{i}^{xJx^T} X(xF)Z(xM+zG). \end{aligned} \tag{2.49}$$

The extra factor of 2 in the phase arises from having to commute the Pauli Z terms in $U_C^\dagger X_j U_C$ past the following Pauli X terms. We can encode these commutation relations as a binary matrix

$$MF^T : \left[MF^T\right]_{i,j} = \mathrm{row}_i(M)\cdot \mathrm{row}_j(F),$$

which is additionally symmetric as

$$\left[U_C^\dagger X_j U_C, U_C^\dagger X_k U_C\right] = [X_j, X_k] = 0.$$

Similiar to the way we encode the phase polynomial in the DCH form, we can then simplify the overall phase calculation as

$$aJa^T : [J]_{i,j} = \begin{cases} \gamma_i & i=j \\ MF^T_{i,j} & i\neq j \end{cases}$$

where we pick up the correct factor of 2 from the symmetric nature of MF^T. Computing each of the matrix-vector multiplications to commute past $U_C$takes $O(n^2)$ time. We can then use the same update rule as for the DCH form to commute the Pauli operator past the U_H layer.

Finally, to finish applying Proposition 1, we need to update the tableau of U_C to

$U_C W_C$. We have

$$(U_C W_C)^\dagger X_i, Z_i (U_C W_C) = W_C^\dagger \left(U_C^\dagger X_i, Z_i U_C\right) W_C$$

an thus we need to update the Paulis in the tableau by conjugtion with $CNOT$, CZ and S gates. These rules for updating U_C by right-multiplication with a control type unitary are the same as for the CHP tableau, with some additional corrections for phase.

$$\begin{aligned}
S &\begin{cases} \mathrm{col}_q(M) & \leftarrow \quad \mathrm{col}_q(M) + \mathrm{col}_q(G) \\ \gamma & \leftarrow \quad \gamma - \mathrm{col}_q(F) \bmod 4 \end{cases} \\
CZ_{q,p} &\begin{cases} \mathrm{col}_q(M) & \leftarrow \quad \mathrm{col}_q(M) + \mathrm{col}_p(F) \\ \mathrm{col}_p(M) & \leftarrow \quad \mathrm{col}_p(M) + \mathrm{col}_q(F) \\ \gamma & \leftarrow \quad \gamma + \mathrm{col}_p(F) \cdot \mathrm{col}_q(F) \end{cases} \\
CNOT_{q,p} &\begin{cases} \mathrm{col}_q(G) & \leftarrow \quad \mathrm{col}_q(G) + \mathrm{col}_p(G) \\ \mathrm{col}_p(F) & \leftarrow \quad \mathrm{col}_p(F) + \mathrm{col}_q(F) \\ \mathrm{col}_q(M) & \leftarrow \quad \mathrm{col}_q(M) + \mathrm{col}_p(M) \end{cases}
\end{aligned} \tag{2.50}$$

There are $O(n)$ row and column updates to perform, and thus this final step runs in time $O(n^2)$. Overall, then, the complexity of applying Proposition 1 to the CH form is $O(n^2)$, arising from computing $U_C^\dagger P U_C$ and then updating the tableau under W_C.

Sampling Pauli Measurements with Proposition 1

Proposition 1 can also be extended to apply to sampling measurements of arbitrary Pauli operators. Measuring a Pauli operator P is closely related to applying a projector $\Pi_{\pm P} = \frac{1}{\sqrt{2}}(I \pm P)$. As mentioned previously, there are three possible outcomes for a Pauli measurement

$$\begin{aligned}
\Pi_{+P} |\phi\rangle = |\phi\rangle \quad & P|\phi\rangle = |\phi\rangle \quad && \text{Deterministic Outcome } +1 \\
\Pi_{+P} |\phi\rangle = 0 \quad & P|\phi\rangle = -|\phi\rangle \quad && \text{Determinitic Outcome } -1 \\
\Pi{+}P |\phi\rangle = |\phi\rangle + |\varphi\rangle \quad & P|\phi\rangle = |\varphi\rangle \quad && \text{Random Outome}
\end{aligned}$$

In terms of measuring an operator P, then we can begin by commuting the projector $I + P$ through the Clifford circuit as described in the previous sections. Dropping the normalisation, we have

$$\begin{aligned}(I+P)V\,|s\rangle &= V\left(I+V^{\dagger}PV\right)|s\rangle \\ &= V\left(|s\rangle + P'\,|s\rangle\right) = V\left(|s\rangle + i^{\beta}\,|s'\rangle\right)\end{aligned}$$

which is the equivalent to the statement of Proposition 1, with $t=s$ and $u=s'$.

If $s=s'$, then the measurement outcome is deterministic. As we have used the projector Π_{+P}, the measurement outcome is $+1$ unless $\beta=2$, in which case the outcome is -1. Otherwise, if $s\neq s'$, the measurement outcome is random and equiprobable. We can sample the ± 1 outcome using random number generation techniques, and then apply the corresponding projector $(I\pm P)$. As computing P' takes in general $O(n^2)$ time, deciding on the measurement outcome also takes $O(n^2)$ time. However, compare to other stabilizer simulators, we note that this algorithm works for arbitrary Pauli operators P as opposed to just single-qubit Pauli Z measurements.

Computational Amplitudes and Sampling Output Strings

Commuting Pauli operators through the layers of control type operators can also be used to compute the probability of a given computational basis state. Recall that a control-type Clifford circuit U_C is defined such that $U_C\,|0^{\otimes n}\rangle = |0^{\otimes n}\rangle$. Recall also that for the DCH representation, U_D and U_{CNOT} are also a control-type operators. Thus,

$$\begin{aligned}\langle 0^{\otimes n}|\phi\rangle &= w^{e}\,\langle 0^{\otimes n}|U_C U_H|s\rangle \\ &= w^{e}\left(\langle 0^{\otimes n}|\,U_C\right)U_H\,|s\rangle \\ &= w^{e}\,\langle 0^{\otimes n}|U_H|s\rangle\,.\end{aligned}$$

This trick, using the definition of a control-type operator to simplify the inner product, can be extended to any comptuational basis state. Writing $|t\rangle = X(t)\,|0^{\otimes n}\rangle$, we can then commute the X operators past the control-type layer (s) to obtain

$$\begin{aligned}\langle t|U_C U_H|s\rangle &= \langle 0^{\otimes n}|P' U_H|s\rangle \\ &= \langle 0^{\otimes n}|i^{\mu} Z(z')X(x')\,U_H|s\rangle = \langle x'|U_H|s\rangle \qquad (2.51)\end{aligned}$$

where we have used the 'ZX' convention in the definition of the Pauli operator. If instead we use the 'XZ' convention, then we pick up an additional phase factor of $-1^{x' \cdot z'}$.

The action of the Hadamard layer on a computational basis state can be expanded out as

$$U_H \left|s\right\rangle = 2^{-|v|/2} (-1)^{s \cdot v} \sum_{x \leq v} (-1)^{s \cdot x} \left|s \oplus x\right\rangle \tag{2.52}$$

where $x \leq v$ denotes the binary strings $x : x_i = v_i \iff v_i = 0$ and $|v|$ is the Hamming weight of the string v. Thus, we have overall that

$$\langle t|\phi| = \rangle \, 2^{-|v|/2} i^{\mu} \prod_{j:v_j=1} (-1)^{x'_j s_j} \prod_{j:v_j=0} \left\langle x'_j \middle| s \right\rangle, \tag{2.53}$$

which equals 0 if any $u_j \neq s_j$ for $v_j = 0$, and is propostional to $2^{-|v|/2}$ otherwise. As this requires commuting a Pauli operator through the C/DC layer (s), computing these amplitudes takes time $O(n^2)$.

This result can also be extened to sample strings from the probability distriution $P(x) = |\left\langle t|V|s\right\rangle|^2$, where V_C is a Clifford circuit such that $V_C = U_C U_H \equiv U_D U_{CNOT} U_H$. From the above, we know that any string with a non-zero amplitude occurs with equal probability. This, it is sufficient to start with a binary string

$$w : w_j = \begin{cases} s_j & v_j = 0 \\ 0 & \text{otherwise} \end{cases}$$

and then pick each of the remaining $|v|$ bits at random with equal probability.

Computing Inner Products

The computational basis are a special case of stabilizer state inner products. Here, we present a general method for computing inner products $\langle \varphi|\phi\rangle$ using the DCH and CH forms. Both methods proceed by combining the two control-type layers, and then breaking down the computation into a sum of different computational basis state amplitudes

$$\begin{aligned}\langle\varphi|\phi\rangle &= \langle t|V_H V_C^\dagger U_C U_H|s\rangle \\ &= \langle t|V_H|\Phi\rangle : |\Phi\rangle = V_C^\dagger |\phi\rangle\,.\end{aligned}$$

Proposition 2 *Given a stabilizer inner product of the form*

$$\langle t|V_H|\Phi\rangle$$

where $|\Phi\rangle$ is encoded in DCH or CH form, we can compute the inner product by computing the computational state ampliude $\langle t|\Phi'\rangle$ where $|\Phi'\rangle = V_H|\Phi\rangle$, in time $O(n^3)$.

Proof of Proposition 2a. In both the DCH and CH form, we can simulate the action of a single Hadamard gate in time $O(n^2)$. The Hadamard circuit V_H contains at most n Hadamard gates, and so we can compute $V_H|\Phi\rangle$ in time $O(n^3)$. The amplitude then reduces to computing the amplitude $\langle t|\Phi'\rangle$, which takes time $O(n^2)$. The overall worst-case complexity is thus $O(n^3)$. □

In the following sections, we will show how to compute $|\Phi\rangle$ from the DCH/CH data of $|\varphi\rangle$ and $|\phi\rangle$.

The DCH Case

In this representation, we need to compute $U'_D U'_{CNOT} = V^\dagger_{CNOT} V^\dagger_D U_D U_{CNOT}$. We begin by combining the two phase layers, noting that

$$U_D^\dagger |x\rangle = \mathrm{i}^{-xBx^t}|x\rangle$$

and thus given the two phase matrices A, B, the phase matrix encoding the combined circuit is

$$V_D^\dagger U_D|x\rangle = \mathrm{i}^{x(A-B)x^T}|x\rangle$$

where, as per the definition, the subtraction is $\bmod 2$ on the off-diagonal entries and $\bmod 4$ on the diagonal entries.

We then need to commute $V_{CNOT}^{\dagger}$ past the new U_D' layer, and combine it with U_{CNOT}. As this circuit is an inverse, it is characterised by the binary matrix Q^{-1}, and its inverse is Q. Thus

$$\begin{aligned} B' &\leftarrow Q^{-1}B'Q \\ W &\leftarrow WQ^{-1} \\ W^{-1} &\leftarrow QW^{-1} \end{aligned} \tag{2.54}$$

Altogether then, the updated DCH information of $|\Phi\rangle$ can be computed in time $O(n^2)$.

The CH Case

Given two tableau describing control-type unitaries V_C and U_C, we can combine them using Eq. 2.49, as

$$\begin{aligned} (V_C U_C)^{\dagger} X_j V_C U_C &= U_C^{\dagger}\left(V_C^{\dagger} X_j V_C\right) U_C \\ &= \mathrm{i}^{\gamma_j'} U_C^{\dagger} P U_C \\ &= \mathrm{i}^{\gamma_j' + \mathrm{row}_j(F') J \mathrm{row}_j(F')^T} X(\mathrm{row}_j(F')F) Z(\mathrm{row}_j(M')M), \end{aligned}$$

and similarly for the Z_j entries. Combining two tableau in this way will require time $O(n^3)$, as there are $2n$ entries and each update takes time $O(n^2)$. However, to compute the tableau of $|\Phi\rangle$, we will require the following Lemma:

Lemma 2 *Given the tableau of a control type operator U_C, specified by the binary matrices F, M and G, then the inverse tableau has matrices G', F' and M' such that*

$$\begin{aligned} G' &\equiv G^{-1} \\ F' &\equiv G^T \\ M' &\equiv M^T. \end{aligned} \tag{2.55}$$

Proof of Lemma 2. The entries of the tableau for $U_C^{\dagger}$ have the property

$$U_C\left(U_C^{\dagger} X_j, Z_j U_C\right) U_C^{\dagger} = U_C^{\dagger}\left(U_C X_j, Z_j U_C^{\dagger}\right) U_C = X_j, Z_j$$

Consider first the Pauli Z terms. Using Eq. 2.49, can see that

$$U_C\left(U_C^\dagger Z_j U_C\right)U_C^\dagger = Z(\mathrm{row}_j(G)G') = Z_j$$

for all $j \in \{1,2,\ldots,n\}$. Expanding out this requirement, we can see that $\mathrm{row}_j(G)\cdot\mathrm{col}_k(G') = \delta_{jk}\forall\, j,k$. If we change the order of the multiplications, we obtain the additional constraint $\mathrm{row}_j(G')\cdot\mathrm{col}_k(G) = \delta_{jk}$. We thus require that

$$GG' = G'G = I \tag{2.56}$$

and thus, $G' = G^{-1}$.

A feature of CHP tableaux is that the jth stabilizer and destabilizer anticommute. Here, similarly

$$U_C^\dagger X_j U_C\, U_C^\dagger Z_k U_C = (-1)^{\delta_{jk}}\, U_C^\dagger Z_k U_C\, U_C^\dagger X_j U_C$$

where the extra phase arises from the commutation relations of Pauli operators. In terms of the entries of the tableau, this tells us that

$$\mathrm{row}_j(F)\cdot\mathrm{row}_k(G) = \delta_{jk}\forall j,k \implies FG^T = I.$$

This also holds for the tableau of $U_C^\dagger$. From this, we can concldue that $F = \left(G^{-1}\right)^T$, and similarly $F' = G^T$.

Finally, consider the X_j entries. Again applying Eq. 2.49, we have

$$U_C\left(U_C^\dagger X_j U_C\right)U_C^\dagger = X(\mathrm{row}_j(F)F')Z(\mathrm{row}_j(F)M' + \mathrm{row}_j(M)G') = X_j.$$

As the Pauli Z terms cancel, we have

$$\begin{aligned}\mathrm{row}_j(F)\cdot\mathrm{col}_k(M') + \mathrm{row}_j(M)\cdot\mathrm{col}_k(G') = 0\;\forall j,k\\ \implies \mathrm{row}_j(F)\cdot\mathrm{col}_k(M') = \mathrm{row}_j(M)\cdot\mathrm{col}_k(G')\;\forall j,k.\end{aligned}$$

Using $F^T = (G^{-1})$, and Eq. 2.56, we thus have

$$\mathrm{row}_j(F) \cdot \mathrm{col}_k(M') = \mathrm{row}_j(M) \cdot \mathrm{row}_k(F)\ \forall j,k \implies M_{j,k} = M'_{k,j} \tag{2.57}$$

completing the proof. □

Specialization for 'Equatorial' Stabilizer States

A specialisation exists for computing the inner product when the state $|\varphi\rangle$ is of the form

$$|\varphi\rangle = \sum_{x \in \mathbb{Z}_2^n} i^{xAx^T} |x\rangle$$

a superposition of all 2^n computational basis states with relative phases. We call these 'equatorial' stabilizer states, as they are like n-qubit generalisations of single qubit states $|0\rangle + \mathrm{e}^{i\theta}|1\rangle$ which lie on the equator of the Bloch sphere.

Claim 1 *If $|\varphi\rangle$ is an equatorial state, we can write the inner product as*

$$\langle\phi|\varphi\rangle = 2^{-(n+|v|)/2} \mathrm{i}^{sKs^T + 2s \cdot v} \sum_{x \in \mathbb{Z}_2^{|v|}} \mathrm{i}^{xK(1,1)x^T + 2x[s+sK](1)^T} \tag{2.58}$$

where $s(1)$ denotes the elements of a vector $s_j : v_j = 1$, and $K(1,1)$ is the submatrix with rows i and columns j such that $v_i, v_j = 1$.

Proof of Claim 1. Let us assume that, given a control-type unitary $U_C \equiv U_D U_{CNOT}$, we can write $U_C^\dagger |\varphi\rangle = \sum_{x \in \mathbb{Z}_2^n} i^{xKx^T} |x\rangle$ for an appropriate phase matrix K. We will show in the following section how to construct this matrix K given the CH and DCH representation of a state $|\phi\rangle$. Given this form then, we have

$$\begin{aligned}\langle\varphi|\phi\rangle &= (\langle\phi|\varphi\rangle)^* \\ &= 2^{-n/2} \left(\sum_{x \in \mathbb{Z}_2^n} \mathrm{i}^{xKx^T} \langle s|U_H|x\rangle \right)^*\end{aligned}$$

Using Eq. 2.52 to expand out the left hand side of this expression, we obtain a sum over terms

$$\sum_{x\in\mathbb{Z}_2^n} \mathrm{i}^{xKx^T} \langle s|U_H|x\rangle = 2^{-|v|/2}(-1)^{s\cdot v}\sum_{y<v}(-1)^{s\cdot y}\sum_{x\in\mathbb{Z}_2^n} \mathrm{i}^{xKx^T} \langle s\oplus y|x\rangle$$

From the orthogonality of computational basis states, we can set $x = s\oplus y$ and drop all other terms in the sum. Doing so changes the phase calculation to

$$(s\oplus y)K(s\oplus y)^T = sKs^T + yKy^T + yKs^T + sKy^T = sKs^T + yKy^T + 2yKs^T$$

where the final equality follows from the symmetric nature of K. From the the definition of $y \leq v$, $y_j = 0 \iff v_j = 0$. Thus, we can take the global phase of sKs^T out and reduce the sum to the sum over strings $y \in \mathbb{Z}_2^{|v|}$, as in Claim 1.

To complete the proof, we need to show how to obtain K in both cases. In the DCH form, we have

$$\langle\phi|\varphi\rangle = \langle s|U_H U_{CNOT}^{-1} U_D^{-1}|\varphi\rangle .$$

Using the definition of an equatorial stabilizer state, we can write $|\varphi\rangle = V_D\left|+^{\otimes n}\right\rangle$, and simply compute $|\varphi'\rangle = U_D^{-1}V_D\left|+^{\otimes n}\right\rangle$ by combining the two phase layers to obtain a new phase matrix $(A-B)$.

Another feature of the state $\left|+^{\otimes n}\right\rangle$ is that it is invariant under $CNOT$ circuits, as it is a superposition of all computational basis states and subsequently invariant under their permutation. Applying Lemma 1, we can commute the circuit U_{CNOT}^{-1} past $U'_D = U_D^{-1}V_D$ and eliminate it. This gives a new phase matrix $K = G(A-B)G^T$.

In the CH case, using Eq. 2.49, we can write

$$U_C^{-1}|x\rangle = U_C^{-1}X(x)U_C\left|0^{\otimes n}\right\rangle = \mathrm{i}^{xJx^T}|xF\rangle$$

Applying this to $|\varphi\rangle$ thus gives

$$U_C^{-1} \sum_{x \in \mathbb{Z}_2^n} \mathrm{i}^{xAx^T} |x\rangle = \sum_{x \in \mathbb{Z}_2^n} \mathrm{i}^{x(A+J)x^T} |xF\rangle .$$

Using $FG^T = I$, as introduced in the previous section, and setting $x = yG^T$, we have

$$\sum_{y \in \mathbb{Z}_2^n} \mathrm{i}^{yG^T(A+J)Gy^T} |y\rangle = \sum_{y \in \mathbb{Z}_2^n} \mathrm{i}^{yKy^T} |y\rangle$$

as required where $K = G^T(A+J)G$. □

Once the calculaton is in this form, we can compute the inner product in time $O(|v|^3)$ using the algorithm for exponential sums developed by Sergey Brayyi [12]. Computing the phase matrix K takes time $O(n^2)$ in both cases, and thus as $|v| \leq n$ we have a general performance $O(n^3)$.

2.2.3 Implementations in Software

Efficient Binary Operations

Case Study: Affine Space Simulation

2.2.4 Performance Benchmarks

2.3 Discussion

Chapter 3

Stabilizer decompositions of Gates and Unitaries

Chapter 4

Simulating Quantum Circuits with the Stabilizer Rank Method

Chapter 5

General Conclusions

Bibliography

[1] D. Gottesman. The Heisenberg Representation of Quantum Computers (1998). arXiv:quant-ph/9807006.

[2] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, **70**, 052328 (2004). arXiv:quant-ph/0406196.

[3] M. Van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond (2008). arXiv:0811.0898.

[4] J. R. Seddon and E. Campbell. Quantifying magic for multi-qubit operations (2019). arXiv:1901.03322.

[5] J. Dehaene and B. de Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, **68**, 042318 (2003). arXiv:quant-ph/0304125.

[6] S. Anders and H. J. Briegel. Fast simulation of stabilizer circuits using a graph-state representation. *Phys. Rev. A*, **73**, 022334 (2006). arXiv:quant-ph/0504117.

[7] H. J. García, I. L. Markov, and A. W. Cross. Efficient inner-product algorithm for stabilizer states. page arXiv:1210.6646 (2012). arXiv:1210.6646.

[8] S. Bravyi and D. Gosset. Improved Classical Simulation of Quantum Circuits Dominated by Clifford Gates. *Phys. Rev. Lett.*, **116**, 250501 (2016). arXiv:1601.07601.

[9] CHP. https://www.scottaaronson.com/chp/. Last Accessed: 2019-05-13.

[10] H. J. García and I. L. Markov. Simulation of Quantum Circuits via Stabilizer Frames. *IEEE Transactions on Computers*, **64**, 2323 (2017).

`arXiv:1712.03554`.

[11] K. N. Patel, I. L. Markov, and J. P. Hayes. Efficient Synthesis of Linear Reversible Circuits. *arXiv e-prints*, pages quant–ph/0302,002 (2003).

[12] S. Bravyi, D. Browne, P. Calpin *et al.* Simulation of quantum circuits by low-rank stabilizer decompositions (2018). `arXiv:1808.00128`.

[13] S. Bravyi, D. Gosset, and R. König. Quantum advantage with shallow circuits. *Science*, **362**, 308 (2018).

[14] E. T. Campbell and M. Howard. Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost. *Phys. Rev. A*, **95**, 022316 (2017).