MA2 TPIVb report

Quantum information - Weak simulation

Quentin Pitteloud

June 6, 2021

Contents

1 Introduction

Reducing the complexity of simulating quantum circuits with a classical computer would be tremendously useful to check the results obtained with quantum computers in the NISQ era. In this ordeal, we will give a summary of the theory related to the question at hand and then apply it in the creation of a weak quantum simulator. This report will primarily serve as the logbook of the work done for the TPIVb lab work.

In this report, when referring to columns (respectively rows) j of a matrix M we will use the notation M_j (respectively M_j^T).

2 Theory

2.1 Stabilizer formalism

Circuits generated by the set of quantum gates $\{H, S, CNOT\}$ are called Clifford circuits. These circuit do not provide any speedup in the computations they implement over classical circuits. This fact can be demonstrated by the simulation techniques of such circuits, dubbed the "stabilizer formalism" [Gottesman, 1998]. The core idea to this technique is to represent a state by its group of stabilizers. These are Clifford operators which have said state as an eigenstate of eigenvalue 1, or in mathermatical terms: if $U|\psi\rangle = |\psi\rangle$, then U stabilises $|\psi\rangle$.

The original technique by D. Gottesman focuses on this group, and by iterating on the gates of the circuit it evolves the state by updating the group of stabilizers with the gate in the Heisenberg representation. For exemple, the stabilizer group of $|0\rangle$ is {Z}. Updating the state with a Hadamard changes the stabilizer to $\{\mathrm{H}^\dagger\mathrm{ZH}\} = \{\mathrm{X}\}$ which corresponds to the state $2^{-1/2}(|0\rangle + |1\rangle)$, as expected.

While useful in theory to show the limits of computational power of Clifford circuits, this method is not really used in practice. It has however been improved upon with S. Aaronson and D. Gottesman's Tableau algorithm [Aaronson and Gottesman, 2004]. The algorithm represents the state by a 2n by 2n+1 matrix with binary entries, with the first n rows represent the "destabilizers" (Pauli operators that change the state) and the last n rows the stabilizers of the state. The first n columns represent X stabilizers for every qubit, the next n columns Z stabilizers and the last column is a "phase" vector that determines if the phase of the associated stabilizers is positive or negative. A prototypical tableau is has the following form:

$$\left(\begin{array}{ccc|ccc|c}
x_{11} & \dots & x_{1n} & z_{11} & \dots & z_{1n} & r_1 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
x_{n1} & \dots & x_{nn} & z_{n1} & \dots & z_{nn} & r_n \\
\hline
x_{(n+1)1} & \dots & x_{(n+1)n} & z_{(n+1)1} & \dots & z_{(n+1)n} & r_{n+1} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
x_{2n1} & \dots & x_{2nn} & z_{2n1} & \dots & z_{2nn} & r_{2n}
\end{array}\right), \tag{1}$$

where $x_{ab} = 1$ if the stabilizer of qubit a has an X operator at the b^{th} place in its tensor product, similarly with z_{ab} for Z operators and $r_a = 1$ if the stabilizer has negative phase. Similarly to the original formalism, the tableau is updated iteratively for every gate in the circuit following the rules stated in the paper by Aaronson and Gottesman [Aaronson and Gottesman, 2004]. The measurement process is a bit trickier however, but a useful property of Clifford circuits is that the measurement of a qubit with always either be deterministic or probabilistic with equal probabilities of outcomes. When measuring qubit q, if there exists a $p \in \{n+1, ..., 2n\}$ such that $x_{qp} = 1$, then the measurement will be probabilistic and the result of the measurement can simply be obtained by choosing uniformly randomly between 0 and 1. Otherwise, the result will be deterministic. The result can be determined by setting the last row identically to zero, and then inputing all rows such that $x_{ia} = 1$ ($i \in (1, ..., n)$) iteratively into a subroutine called *rowsum*. The subroutine essentially takes 4 qubits (x_1, z_1, x_2, z_2) and determines the power to which the phase i is raised when multiplying the stabilizers reprented by (x_1, z_1) and (x_2, z_2). The result is given by the value of r_n at the end of the iteration.

Stabilizer states are states obtained from the ground state $|0^{\otimes n}\rangle$ with Clifford circuits. One useful property of these states is that they form an overcomplete basis of the Hilbert space of dimension 2^n [Bravyi and Gosset, 2016]. Hence, all states can be decomposed as a linear combination of stabilizer states. The number of elements in the decomposition is called the stabilizer rank, and this property will be relevant for the creation of a weak quantum simulator.

2.2 Strong versus weak simulation

In order to simulate a quantum circuit, there are two main approaches, the strong and weak simulations. Strong simulation is the most popular approach and denotes the computation of the probability of outcomes. Weak simulation denotes the sampling of the probability distribution. These two approaches are not equivalent as we'll see in the next section with so-called "Hadamard-Toffoli" circuits. This type of circuit is efficiently computable in the weak sense but simulating it in the strong sense equates to a NP-complete problem.

2.3 Normal form circuits

A Clifford circuit is said to be in Normal Form (NF) if it is composed of 3 blocks, composed in the following way. The first block is composed only of Hadamard gates, the second one is composed of NOT and CNOT gates while the last one is made of (controlled-)phase gates. This form of circuit is interesting for their simulatability is manifest [Nest, 2009]. This form can be visualised on figure 1. Indeed, NF circuits are a special case of Hadamard-Toffoli (HT) circuits, which are first made of a Hadamard block followed by "classical" gates (*i.e.* NOT, CNOT, Toffoli) are efficiently classically simulatable in the weak sense. This can been seen by the following reasoning (figure 2). The action of the block of Hadamards on the ground state ($|0^{\otimes n}\rangle$) generate the state $|x\rangle := |+^m\rangle\, |0^{n-m}\rangle$ by appropriately reorganizing the qubits. The second block implements an efficiently computable function $f : \{0,1\}^n \to \{0,1\}^n$, giving a final state $|f(x)\rangle$. Let $\alpha \in \{0,1\}$, and without loss of generality suppose we measure the first qubit. The probability of measuring this qubit in the computational basis and finding the outcome α is

$$\pi(\alpha) = \frac{||\{y \in \{0,1\} : f(x)_1 = \alpha\}||}{2^m}, \tag{2}$$

where $f(x)_1$ means the first qubit of the bitstring $f(x)$, and $|| \cdot ||$ means the cardinal of the set. This probability distribution can then simply sampled by constructing a uniformly random n-bit string, computing $f(x)$ and setting $\alpha := f(x)_1$. This is also valid for NF circuits as the phase gates do not impact the probability distribution π.

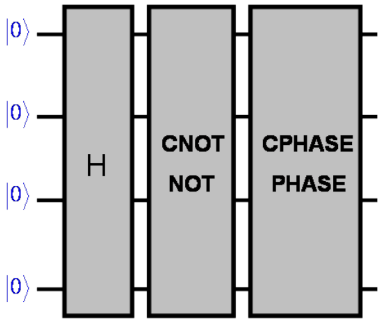

Figure 1: Normal Form circuit [Nest, 2009]

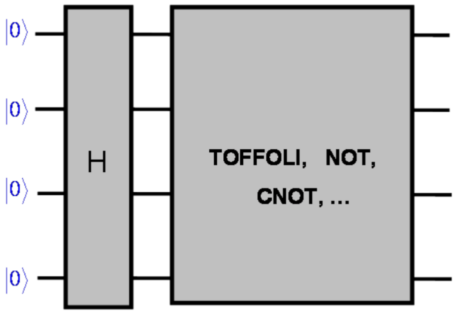

Figure 2: Hadamard-Toffoli circuit [Nest, 2009]

Normal form circuit are mainly useful since for any Clifford circuit there exists an equivalent Normal Form circuit. Here the equivalence is in regard to the action on the ground state $|0^{\otimes n}\rangle$, *i.e.* $\forall$ Clifford circuit C, $\exists$ NF circuit C' such that $C'|0^{\otimes n}\rangle = C|0^{\otimes n}\rangle$ This is proven by theorem 2 of M. Van Den Nest's paper on weak simulation [Nest, 2009]. The NF circuits highlight the limitations of the computational power of Clifford circuits. Indeed, the last block of NF circuits, composed of phase gates, do not contribute to the probability of a certain outcome therefore making the circuit equivalent (in a general sense) to a HT circuit. Additionally, NF circuits effectively only have 1 round of interfering gates placed at the very start of the circuit while the other two blocks do not produce any interference at all. This shows that Clifford circuits do not take advantage of the interference to obtain a speedup over classical circuits. The relation between the computation power of Clifford circuits and classical circuits is further explained by the fact that HT circuits can simulate any classical probabilistic algorithm.

Strong simulation of HT circuit is a NP-complete problem. This can be proven by the fact that counting problems are NP problems, and that computing the probability of a certain outcome is equivalent to computing the number of zeros n_f of the function f implemented by the HT circuit. Indeed, suppose we only measure the first qubit then the probability of 0 outcome is $\pi(0) = n_f/2^n = 1 - \pi(1)$, which shows the previous affirmation.

It has been shown in reference [Dehaene and De Moor, 2003] that the most general state generated by Clifford circuits is of the form

$$|\psi\rangle \propto \sum_{x\in A} i^{l(x)}(-1)^{q(x)}|x\rangle\,, \tag{3}$$

where l (respectively q) is a linear (respectively quadratic) function of n-bit string x and $A = \{Ru + t | u \in \mathbb{Z}_2^m\}, m \leq n$, is an affine subspace of $\mathbb{Z}_2^n$ with R a $n \times m$ matrix and $t \in \mathbb{Z}_2^n$. This result is also easy to prove for NF circuits as the first block of m Hadamard gates will create a state $|+^{\otimes m}\rangle|0^{\otimes n-m}\rangle \equiv \sum_u |u\rangle$, then the CNOT gates with change it to $\sum_u |Ru\rangle$, the NOT gates to $\sum_u |Ru + t\rangle \equiv \sum_x |x\rangle$ and finally the last block will generate the phase coefficients $i^{l(x)}(-1)^{q(x)}$. Any stabilizer state can be respresented using only the triplet (A, q, l), and this representation will be refered to as the A-form.

2.4 Gadget-based simulation method

In order to simulate Clifford + T circuits with better efficiency than with a brute force algorithm, one can replace T gates with a Clifford circuit called a "T gadget" [Bravyi and Gosset, 2016]. This gadget consumes one copy of a "magic state" $|T\rangle = 2^{-1/2}(|0\rangle + e^{i\pi/4}|1\rangle)$, initialized in a ancillary qubit which is entangled to the qubit on which the T gate was originally applied. The ancilla qubit is then measured in the computational basis and the result is used in a control for a classically controlled S gate on the original qubit.

The magic state $|T\rangle$ is not directly obtainable from a Clifford circuit. However, recall that stabilizer states form an overcomplete basis of the Hilbert space [Bravyi and Gosset, 2016]. As such, the magic state can be obtained as a linear combination of stabilizer states. In fact, it can be decomposed as

$$|T\rangle = 2^{-1/2}(\mathbb{1} + e^{i\pi/4}X)\,|0\rangle\,. \tag{4}$$

The T gadget is displayed in figure 3.

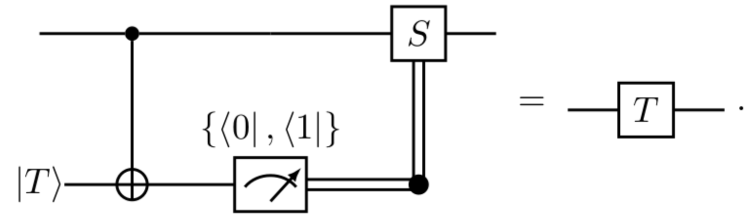

Figure 3: Circuit of T gadget [Pashayan et al., 2021]

The T gadget can also be written in a "reverse" way, avoiding the need to initialize the ancilla qubit in the magic state [Pashayan et al., 2021]. Instead, the ancilla will be initialized in the ground state $|0\rangle$. However, this implementation of the gadget requires the use of a non computational basis measurement. Indeed, the computational basis measurement of the acilla is replaced with a measurement in the basis $\{|T\rangle\,, |T^{\perp}\rangle\} = \{2^{-1/2}(|0\rangle + e^{i\pi/4}\,|1\rangle), 2^{-1/2}(|0\rangle - e^{i\pi/4}\,|1\rangle)\}$. The rest of the gadget is left mostly unchanged, only the result of the measurement will serve in a controlled Z gate instead of an S gate. This formulation of the gadget is also useful as it can be generalised to simulate an arbitrary phase gate by measuring the ancilla qubit in the basis $\{|T_\phi\rangle\,, |T_\phi^{\perp}\rangle\}$, where $|T_\phi\rangle = 2^{-1/2}(|0\rangle + e^{-i\phi}\,|1\rangle)$ and $|T_\phi^{\perp}\rangle = 2^{-1/2}(|0\rangle - e^{-i\phi}\,|1\rangle)$. The circuit for the arbitrary phase reverse T gadget is displayed in figure 4.

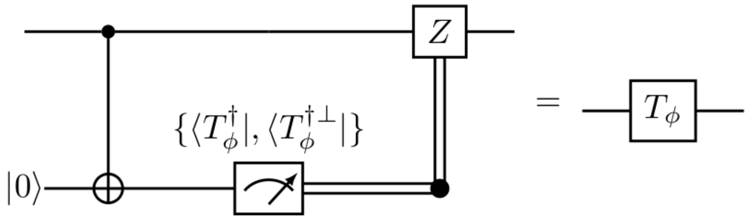
Z
$\{\langle T_\phi^\dagger|, \langle T_\phi^{\dagger\perp}|\}$
$|0\rangle$
$=$
T_ϕ

Figure 4: Circuit of reverse arbitrary phase (T_ϕ) gadget [Pashayan et al., 2021]

For both types of T gadgets, the measurements are equiprobable. As such, one can use postselection of the measurement outcome to get rid of the measurement altogether in the gadget at the cost of a normalization factor in the output probability. For instance, postselecting the reverse gadget on the $|T_\phi\rangle$ outcome, it simplifies (up to a factor) to the addition of an ancilla qubit, one Clifford gate (a CNOT) and a projector onto $|T_\phi\rangle$. This postselection creates the smallest possible T gadget, possibly improving the performance of algorithms using it. One drawback from using postselection is that the stabilizer decomposition of the postselected state must be exact; the stabilizer rank of the state must be known. The postselected reverse gagdet is presented in figure 5.

To avoid the issue of having to post-select on outcomes with exact stabilizer decomposition, one could use random postselection of outcomes [Bravyi et al., 2019]. This would require to randomly select outcomes for every gadget and that selection would change the gadgetized circuit accordingly, effectively creating a random final circuit. Since the state $|T\rangle$ has an exact stabilizer decomposition, this technique is will not be used in this report.

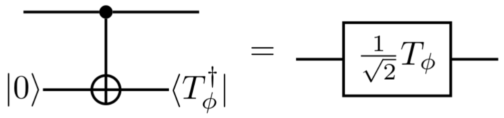

Figure 5: Circuit of postselected reverse arbitrary phase (T_ϕ) gadget [Pashayan et al., 2021]

2.5 Metropolis sampling method

To sample a probability distribution of a sum of Clifford circuits, one can use a Metropolis-type Markov Chain as in reference [Bravyi et al., 2019]. This type of Markov Chain is useful in general to sample a probability distribution $P(x)$ provided we have access to some function $f(x)$ which is proportional to P. Let $\Omega = \{x \in \{0,1\}^n | P(x) > 0\}$ be the state space of the chain. This algorithm starts by computing the probability of outcome of some random $x_{in} \in \Omega$. The chain then evolves step by step. Say we are at an arbitrary state $x \in \Omega$ in the chain, the Metropolis step $x \rightarrow x'$ is summarized in 3 parts:

1. Pick some $j \in \{1, ..., n\}$ uniformly at random, define $y := x \oplus e_j$, where $e_j = (0, ..., 1, ..., 0)^T$ with 1 at position j
2. If $\frac{P(y)}{P(x)} \geq 1$, set $x' = y$ and proceed to the next step
3. Otherwise, randomly generate some bit $b \in \{0,1\}$ with $P(b=1) = \frac{P(y)}{P(x)}$. If $b = 1$, set $x' = y$. If not, set $x' = x$. Proceed to the next step.

Notice now that

$$\frac{P(y)}{P(x)} = \left|\frac{\sum_a c_a \langle y|\phi_a\rangle}{\sum_b c_b \langle x|\phi_b\rangle}\right|^2 = \frac{f(y)}{f(x)}, \tag{5}$$

and thus one does only requires knowledge of the function f, which in this case is related to the probability amplitudes. The probability distribution P can be sampled by this algorithm since it is the only steady state of the Markov chain. In order to achieve this steady state, a large number of Metropolis steps must be done. This process then outputs some x with probability given by P, effectively sampling the distribution. This method was not needed to sample the probability distribution and another simpler one has been devised, hence the Metropolis algorithm has not been implemented.

3 Results

The main result presented in this report is a weak quantum simulator based on the T gagdet method using the A-form respresentation. As it will become apparent later, the choice of using this representation of states will not prove to be a practical one as the litterature on this representation is rather lacking. As such, a large amount of time has been spent rederiving and extrapolating the update rules of the state representation for every type of supported gate. This is especially true regarding the functions q and l of the A-form, as very little is said about those in reference [Nest, 2009].

The implementation is divided in three versions differing mainly from the way the probability P is obtained. These versions are presented to display the evolution of the simulator itself and showing the creation and learning process associated with it. The range of applicability of each version is limited, for some versions more than others.

3.1 Generalities about the implementation

Let us dive a little deeper into the implementation of the simulator. This algorithm can be broken down in four steps:

1. Gadgetize the circuit using postselected reverse T gadgets
2. Update the triplet (A, q, l) for all gates in the circuit but the projectors on $|T\rangle$
3. For every stabilizer projector in the bra equivalent of equation 4, update a copy of the triplet. This will give a set of A-form states corresponding to the final states $\{|\psi_i\rangle\}_{i=1,2}$.
4. Sample the probability distribution $p(x)$

There are numerous remarks and precisions to be made about this implementation. Here we will go through some of the more important ones.

The way circuits are represented is through a matrix with 2 rows: the first row contains a string denoting the type of gate applied and the second contains strings with the qubit(s) on which they are applied. The gadgetization process is thus simply done by increasing the number of qubits and replacing every T gate by a $CNOT$ liking the original qubit to the new one. The benefits of this representation of the circuit is that it is very simple conceptually and light but it can quickly become unwieldy even for moderately-sized circuits. As such, most results obtained from the algorithm will be on shallow circuits with few qubits.

While not essential, the use of postselection in the T gadgets in the simulator improves performance by keeping the complexity cost of gadgets small. The multiplication of the number of elements in the sum of projectors by $2 (= \chi$ the stabilizer rank of $|T\rangle)$ for every T gate in the initial circuit clearly indicates an exponential dependance $2^{\alpha t}$ in the complexity cost of the algorithm, with t the T count and α some real factor. However, updating the A-form does only require a cost $poly(n)$ since the size of matrix R and vector T scale linearly with n and the functions l and q are functions of the bitstring $x \in \mathbb{Z}_2^n$. The cost of the sampling process will depend on the version of the simulator.

The update rules for the triplet (A, q, l) under the action of a gate $G \in \{H, S, CNOT, T, X\}$ has been rederived from the few indications in section 5 and appendix A of reference [Nest, 2009]. In an effort to improve readability of this report, the explicit update rules and initial value of the triplet will be detailed in appendix A. It has to be said that these rules derived for this implementation may not suit all possible circuits despite an effort being made to have the most general implementation possible. The cause for this drawback may be that the action of a Hadamard on the state can change the rank of the matrix R (determining the subspace A) which has not been considered thoroughly in the implementation due to its empirical nature. Indeed, a Hadamard gate may add a column to R which can change its rank.

Moreover, the (A, q, l) formalism has been expanded a little to accommodate the needs of the implementation. Specifically, the rules in the case of a projection of a qubit onto the state $|0\rangle$ had to be devised from scratch. In general, these rules work as intended. However for some cases where multiple qubits are entangled to an ancilla, the rules can sometimes give erroneous results for R. It is in the realm of possibilites that no set of rules would fit all possible solutions, as the formalism was not devised to integrate projectors. The set of rules does seem adequate in most situations, even though this limitation further hurts the range of applicability of the implementation. As with the other update rules, it is specified in details in appendix A and the issues with entangled qubits is discussed in appendix C.

It must also be noted that although it has been designed with an arbitrary Clifford + T circuit in mind, the simulator does not handle well circuits with multiple H or T gates in a row. The former is probably due to the lack of tracking of the rank of the matrix R as the theory requires it to be full rank, and the latter is probably caused by errors in the update rules when projecting onto stabilizer states.

3.2 First version

The first version of the simulator did not take into account the individual phases of the states in the superposition. In this version, the functions q and l determining the phases are not considered. As one would expect, it does only yield appropriate results on NF-style circuits and full Clifford circuits. In other words, this limitation requires that phases to be irrelevant in the output probability distribution. Indeed, a non-Clifford, non NF-style circuit can be approximated as a sum over Clifford circuit and the probability distribution of a sum will depend on the individual phases of each element.

The sampling process is identical to the one originally concieved by Van Den Nest: Generate uniformly random m-bit strings u which gives a n-bit string from the state space Ω of the circuit using the relation $x = Ru + t$. The probability distribution can then be sampled by repeating this process a large number of times. The main advantage of this process is that it is very efficient, since R is a $n \times m$ matrix and $t \in \mathbb{Z}_2^m$, it is obvious that the cost of sampling the probability distribution is $poly(n, m)$. This sampling process was first implemented to create a Clifford simulator but also as a placeholder for a future version of the simulator.

3.2.1 Results of the first version

Some results using the first version of the simulator applied on simple circuits will be presented here. First, consider some Clifford circuit. Let us choose arbitrarily the very simple 2 qubit circuit $H_1 CNOT_{1,2} X_1$ where the index denotes the qubit on which it is applied (in the format $control, target$ for the CNOT gate). This simple circuit is relevant for our demonstration purposes because it contains one of all non-diagonal Clifford gates and contains entanglement which is an important property to be able to simulate. The raw (unnormalized) result for the circuit with 2048 samples is shown in figure 6. As expected, the simulator correctly gives a probability 1/2 for each outcome. Additional results for a larger Clifford circuit is diplayed in figure 20 in appendix B.

Similarly, the implementation gives good results for NF-style circuits. Since we have already shown Clifford circuits work as intended with this implementation, it is sufficient to consider the simplest NF-style circuit containing a T gate: the one qubit $H_1 - T_1$ circuit. The raw results are presented in figure 7 and display a uniform probability of outcomes as expected.

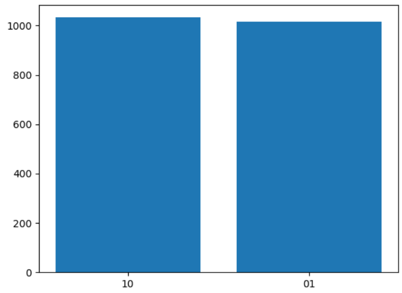

Figure 6: Raw result of the version 1 on circuit $H_1 CNOT_{1,2} X_1$ with 2048 samples

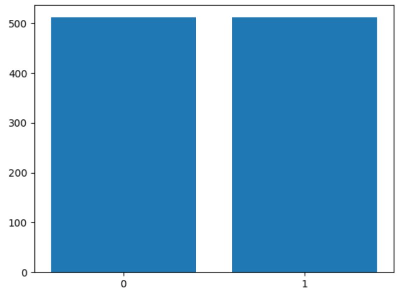

Figure 7: Raw result of the version 1 on circuit $H_1 T_1$ with 1024 samples

However, this version does not give correct results for other types of circuits. An example is displayed in figure 9.

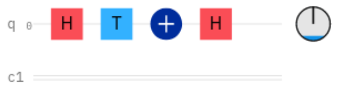

Figure 8: Example of a circuit [IBM, 2021]

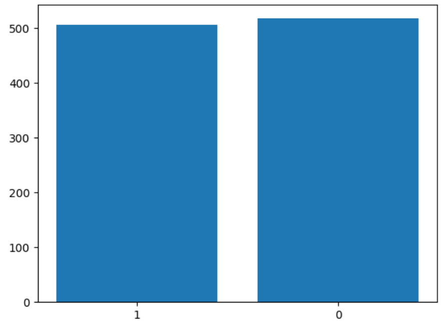

Figure 9: Incorrect results of the implementation for the circuit on the left (ver.1 - 1024 shots)

3.3 Second version

As explained previously, the main limiting factor of the previous version of the simulator was the lack of tracking of the individual phases. Hence, this version introduced the tracking of functions q and l. Additionally, this version would serve as a testbed for a different way of computing the probability distribution $P(x)$. The new method is much more computationally heavy but does offer an exact result. It was devised in prevision to the implementation of the Metropolis algorithm in section 2.5 to compute the probability $P(x_{in})$ and then computing $P(y)/P(x)$ during the Metropolis steps. It has also been expanded to a complete algorithm to compute $P(x)$.

This method is based on directly extrapolating the value of the probability amplitudes $\langle x|\psi\rangle$ (where $|\psi\rangle$ is the final state of the circuit, *i.e.* a superposition of stabilizers) from R, t and the phase functions. For instance, if the j^{th} row of R is non-zero, then qubit j will be in a uniform superposition of $|0\rangle$ and $|1\rangle$, hence it will participte to P with a factor $2^{-1/2}$. Also, we can deduce which qubits are entangled with each other from R. From these considerations, a simple algorithm to compute $P(x)$ can be constructed. Below we will detail the computing process for some outcome $x \in \mathbb{Z}_m^n$.

```
$P(x) \leftarrow 0$;
for circuit in the sum of Clifford do
    $Amp(x) \leftarrow 1$ // Probability amplitude $\langle x|\psi\rangle$
    for $k$ from 1 to $n$ do
        // qubit $k$ is in a superposition of $|0\rangle$ and $|1\rangle$
        if $R_k$ is non-zero then
            if qubit $k$ is not entangled with any qubit in 1 to $k-1$ then
                $Amp(x) \leftarrow 2^{-1/2} \cdot Amp(x)$;
        // qubit $k$ is in state $|t_k\rangle$, $t_k \in \{0,1\}$
        else if $x_k \neq t_k$ then
            $Amp(x) \leftarrow 0$;
    $P(x) \leftarrow P(x) + (-1)^{q(x)} i^{l(x)} \cdot Amp(x)$;
$P(x) \leftarrow |P(x)|^2$
```

This algorithm performs a strong simulation of the circuit, and is thus computationally heavy for a Clifford $+$ T circuit. Indeed, using this method to compute P makes the task a NP problem as it would need 2^n calls to the subroutine outputing $P(x)$ for a given x. Additionally, its range of applicability has been observed to be somewhat mediocre. In fact, non-Clifford circuits which output pure states do not work well with this version. The reason of this limited range has not been studied as this algorithm does not fulfill the purpose of this report. Moreover, the exact computation of $P(x)$ was not required as the Metropolis algorithm was not implemented due to another simpler way of sampling the distribution being concieved (shown in version 3). Nonetheless, some results have been obtained from this version of the simulator.

3.3.1 Results of the second version

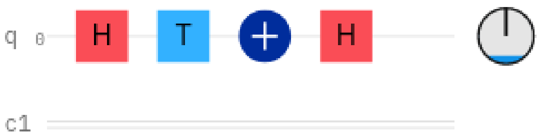

Figure 10: Example of a simple 1 qubit Clifford + T circuit [IBM, 2021]

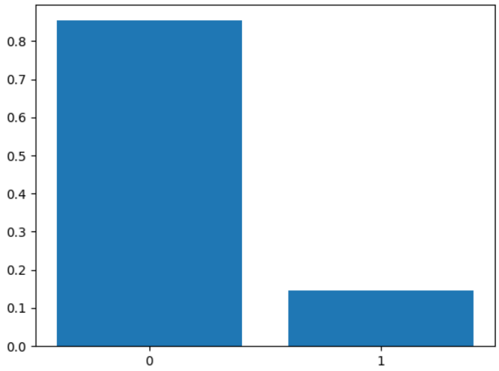

Figure 11: Result of the implementation (ver.2)

This version does not handle well independent qubits as is displayed in figure 13. This limitation is believed to arise from issues in the code. However as a more effecient version more in-line with the objective of this work was concieved, it was deemed unnecessary to improve this version further.

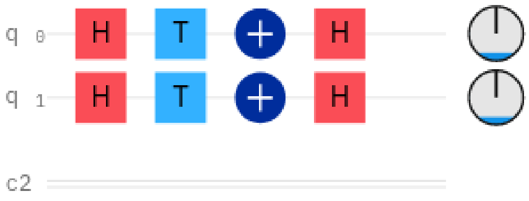

Figure 12: Single qubit Clifford + T circuit simultaneously applied on 2 independent qubits [IBM, 2021]

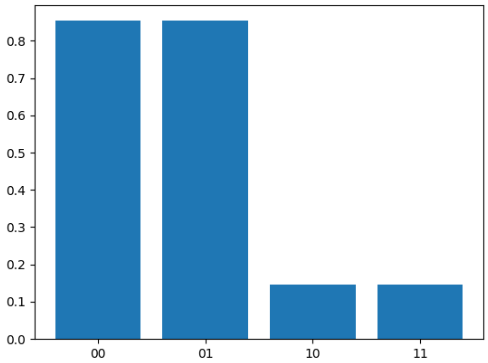

Figure 13: Incorrect results of the implementation (ver.2)

3.4 Third version

The third version re-uses the sampling process of the first version but incorporates the individual phase tracking of the second version. This results in a much more efficient simulator which can reliably give correct results. This version also has a broader range of applicability than the previous versions, but it still has issues simulating an arbitrary circuit due to the empirical nature of the update rules. The most commonly encountered issue arises when multiple T gates are applied to different entangled qubits.

Let us now give a brief review of the complete sampling process for clarity's sake. Recall that we are trying to sample a probability distribution from what is essentially a sum of Clifford circuits. Indeed, after all the projection on the stabilizer decomposition of $|T\rangle$ is said and done we are left with multiple triplets (A, q, l) each corresponding to the original circuit and a projector on one of the stabilizers. We will now review the sampling process for each triplet. First generate uniformly randomly a m-bit string u and use it to generate some n-bit string from the state space: $Ru + t = x \in \Omega$. Compute the associated phase $(-1)^{q(x)} i^{l(x)}$ and add that phase to the sampled probability amplitude corresponding to outcome x. After repeating the process a large number of times for each element in the sum of Clifford, take the square modulus of the resulting sampled probabilities for each outcomes and normalise the result. This gives a probability distribution which approximates the real distribution $P(x)$ of the circuit.

As said in section 3.2, the cost of computing x is $poly(n, m)$. Now for the cost of computing the phases. Let us use the notation $l(x) = d^T x$, $q(x) = x^T C x + b^T x$ where $d, b \in \mathbb{Z}_2^n$ and C is a square $n \times n$ matrix with coefficients in $\mathbb{Z}_2$. From these considerations, the polynomial cost in n of computing the phase becomes manifest. Note that the total complexity cost of the algorithm still depends exponentially in the T-count t as each T gate would double the number of elements in the sum of Clifford.

3.4.1 Results of the third version

In this section are displayed some results of the third version of the simulator. More results are presented in appendix B. The results displayed here have been observed to vary more between runs of the algorithm than the first version, hence a crude method of estimating the error was implemented. This method relies on running multiple times the sampling process and computing the mean and standard error (*i.e.* standard deviation of the mean) of the set of results. Note that the precision of the results can also be increased by using a larger number of shots in the sampling, thus lowering the error obtained in the estimation.

For most simple circuits, the implementation works reasonably well as shown in figure 15 for the circuit shown in figure 14.

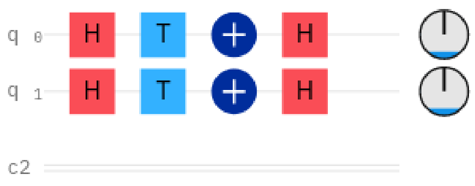

Figure 14: Example of a simple 2 qubit Clifford + T circuit with independent qubits [IBM, 2021]

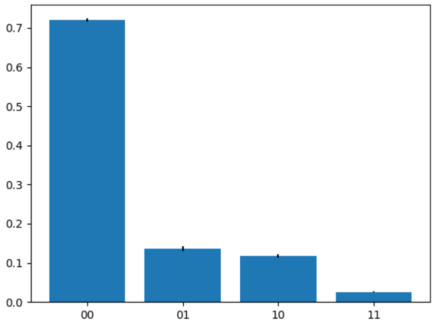

Figure 15: Normalized result of the implementation (ver.3 - 1024 shots)

For less trivial circuits, the simulator outputs less than ideal results. For instance, take the circuit shown in figure 16. This circuit has the advantage of only having at most one T gate per group of entangled qubits, and thus should be handled without issues by the simulator. However, when comparing with results with IBM's *quantum experience* simulator, the outputs do not match exactly as shown in figures 17 and 18. Though very unlikely, it is possible that the differences might partially originate from an error when converting the circuit to take into acount the little Endian convention of IBM's simulator. While the values of the probabilities displayed and the state space of the output match between the two results, the simulator results appear as if it were a "scrambled" version of the IBM's simulator.

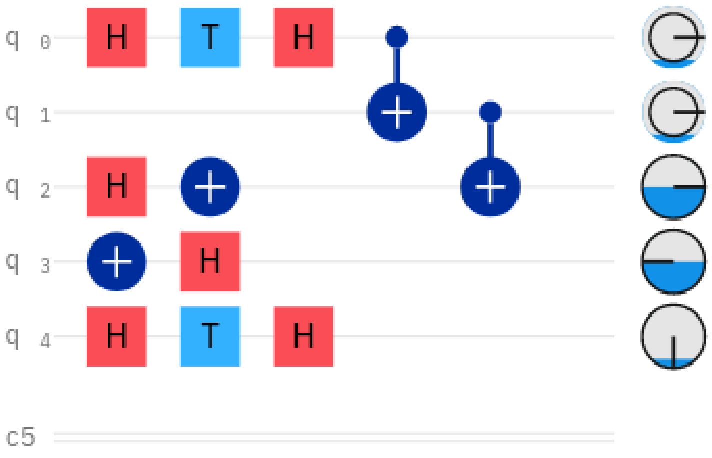

Figure 16: Example of a non trivial 5 qubit circuit [IBM, 2021]

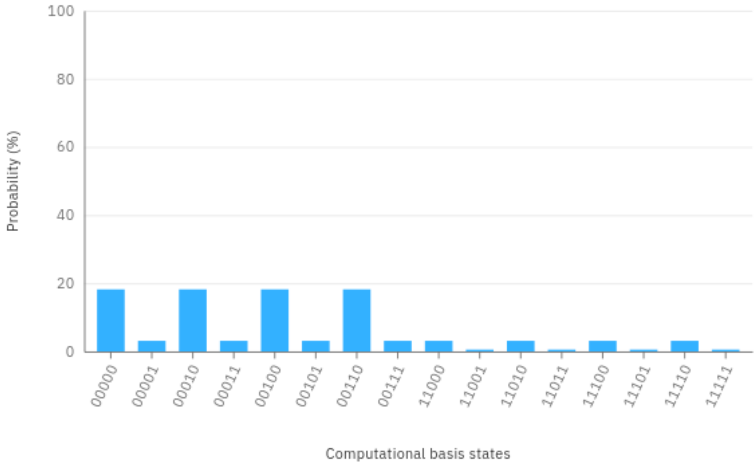

Figure 17: *IBM quantum experience* results for the circuit in figure 16 [IBM, 2021]

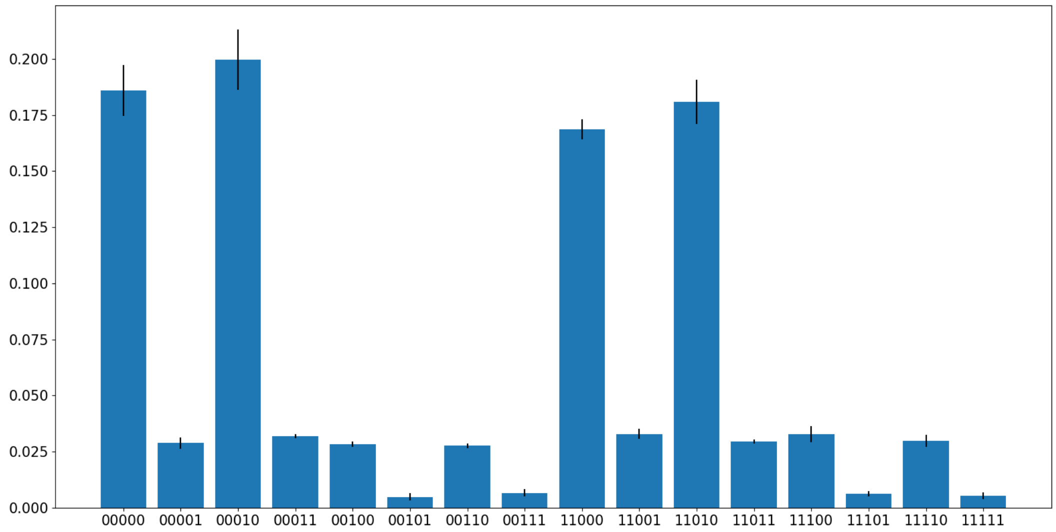

Figure 18: Normalized result of the A-form simulator for the circuit in figure 16 (ver.3 - 1024 shots)

4 Conclusion

This report logged the work done during the semester as part of the TPIVb lab work. Theory regarding weak simulation of quantum circuits was presented. Additionally, multiple versions of an original (A, q, l)-based weak simulator were presented, each with some results and limits discussed. The simulator was ultimately plagued by issues rooted in the lack of litterature regarding the A-form.

The final version of the simulator offers some good results for small circuits but they start to quickly degrade with the depth and T-count of the circuit. The reason is thought to reside mainly in the evolution of the phase functions q and l, especially during with a projector. However, the evolution rules for the A subspace have proven to be quite robust even for larger circuits. One could get rid of projectors by using the set of gates $\{H, S, CNOT, R(\theta)\}$, where $R(\theta)$ is an arbitrary rotation gate, instead of the set $\{H, S, CNOT, T\}$. This would eliminate the need of gadget methods and instead use the "sum over Cliffords" method [Bravyi et al., 2019], thanks to the relation

$$R(\theta) = (\cos(\theta/2) - \sin(\theta/2))\mathbb{1} + \sqrt{2}e^{-i\pi/4}\sin(\theta/2)S. \tag{6}$$

Using such methods could possibly improve the robustness of the simulator and improve performance by eliminating the need to add ancilla qubits.

A Explicit update rules for (A, q, l)

We will now explicitely give the update rules for each element of the triplet (A, q, l). First let us give the initial conditions used in this implementation. As there was no explicit condition given, they have been set to the following values. Let n be the number of qubits, and recall $l(x) = d^T x$, $q(x) = x^T C x + b^T x$.

$$R = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{M}_{n\times 1}(\mathbb{Z}_2), \quad t = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{Z}_2^n \tag{7}$$

$$c = \mathbb{1} \in \mathbb{M}_{n\times n}(\mathbb{Z}_2), \quad b = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{Z}_2^n, \quad d = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{Z}_2^n \tag{8}$$

For clarity's sake, the update rules in the next sections will be given in pseudocode.

A.1 Updating (A, q, l) for a Hadamard gate

The case of a Hadamard is, along with the projector, the most non trivial one. Let j be the index of the qubit on which H is applied and e_j a vector of all zeros but a one at position j. For the update of R the algorithm is the following.

```
if ∃ column $R_j \in R$ that is zero then
    $R_j \to 1$;
else
    if the $j^{th}$ row $R_j^T$ of $R$ is zero then
        Append $e_j$ to $R$ as a column;
    else
        Append to $R$ the sum of every column where the $j^{th}$ element is 1 ;
        Add 1 to the $j^{th}$ element of the new column ;
```

For the update of C and b, the algorithm is given below. v is a vector in $\mathbb{Z}_2^n$ such that its i^{th} element is 1 it the i^{th} qubit is entangled with the j^{th} qubit. Note that the matrix R contains all the information we need, since if two qubits are entangled together there will be at least one column in R with value 1 in both of their corresponding rows. The notation $\oplus$ means the addition mod 2.

```
// Construct the vector $v$ - find all qubits entangled with $j$
$v \leftarrow \vec{0}$;
for $k$ in the row $R_j^T$ do
    if $k$ = 1 then
        for $l$ in the column containing $k$ do
            if $l$ = 1 then
                $v_l \leftarrow 1$ ;
// Update $C$ and $b$
$C_j \leftarrow C_j \oplus v$ ;
$b_j \leftarrow b_j \oplus t_j$ ;
```

A.2 Updating (A, q, l) for a CNOT gate

Let i be the control qubit and j the target one. We will abuse the notation a little and simultaneously use $\oplus$ for the element-wise addition in $\mathbb{Z}_2^n$ and for the addition in $\mathbb{Z}_2$.

```
$R_j^T \leftarrow R_j^T \oplus R_i^T$ ;
$t_j \leftarrow t_j \oplus t_i$ ;
```

A.3 Updating (A, q, l) for a X gate

As before, let j denote the qubit on which X is applied.

```
$t_j \leftarrow t_j \oplus 1$ ;
$b_j \leftarrow b_j \oplus 1$ ;
```

A.4 Updating (A, q, l) for a S or Z gate

Note that the update rules for S and Z have not been implemented in the simulator mainly due to time constraints, but also because because other issues were more limiting in the applicability range of the simulator. Nevertheless, the update rules will be explicitely stated in this section.

For a gate S applied on qubit j, only the matrix C and vector d are affected. We will once more use the notation e_j for the vector with all zeros but a one at position j and abuse the notation $\oplus$ as in section A.2. Note that the order of the the update rules is important here.

```
$C_j^T \leftarrow C_j^T \oplus d$ ;
$d_j \leftarrow d_j \oplus 1$ ;
```

Now using the fact that $Z = S^2$, and that for any vector a in $\mathbb{Z}_2^n$, $2 \cdot a = \vec{0}$, we can deduce that a Z applied on qubit j will only affect the matrix C. The notation $\vec{1} \in \mathbb{Z}_2^n$ denotes a vector with 1 in all entries.

```
$C_j^T \leftarrow C_j^T \oplus \vec{1}$ ;
```

A.5 Updating (A, q, l) for a projector on a stabilizer state

In this simulator, only the projection onto the state $|0\rangle$ has been devised. For other stabilizer states, the Hermitianity of Pauli and H operators is used. Indeed, if one wishes to project onto say $|1\rangle$, the projector can be replaced by a projector on $|0\rangle$ preceded by an X gate. The same procedure holds for all projector onto Hermitian Clifford operators. Let n be the number of qubits and j be the index of the qubit on which the projector $|\phi_p\rangle = U_p |0\rangle$ is applied with U_p some Clifford circuit that we will assume Hermitian. This assumption is reasonable for our purposes as the stabilizer decomposition of $|T\rangle$ $(= (\mathbb{1} + e^{i\pi/4}X) |0\rangle)$ only contains Hermitian operators. The method can be generalised to include any stabilizer projector, only requiring the addition of the update rules for $S^\dagger$ in the method.

We will use the algorithm to construct the vector v from section A.1 and assume that we already obtained it.

```
// Update $(A, q, l)$ with the Clifford circuit $U_p$ on qubit $j$ according to
   the updates rules
for $Gate$ in $U_p$ do
    $(A, q, l) \leftarrow update((A, q, l), \text{Gate})$
// Add rows/elements together to simulate the projection
for $k$ from 1 to $n$ do
    if $v_k = 1$ then
        $R_k^T \leftarrow R_k^T \oplus R_j^T$;
        $T_k \leftarrow T_k \oplus T_j$;
        $C_k^T \leftarrow C_k^T \oplus C_j^T$;
        $b_k \leftarrow b_k \oplus b_j$;
        $d_k \leftarrow d_k \oplus d_j$;
// Now delete unnecessary rows and columns if possible
$q \leftarrow$ smallest $q$ such that $R_{q,j} = 1$;
if Number of columns of $R > 1$ then
    delete $R_q$;
delete $R_j^T$;
delete $T_j$;
delete $C_j^T$ // row
delete $C_j$ // column
delete $b_j$;
delete $d_j$;
```

B Additional results of the simulator

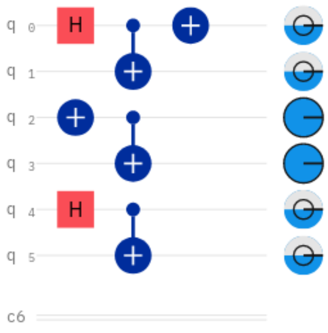

Figure 19: Example of larger Clifford circuit [IBM, 2021]

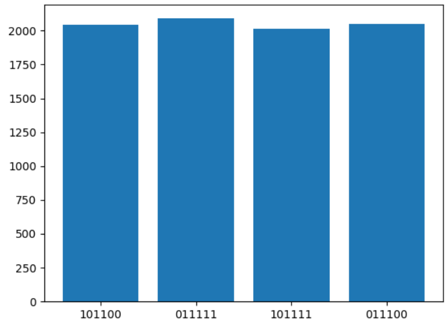

Figure 20: Result of the implementation (ver.1 - 8196 shots)

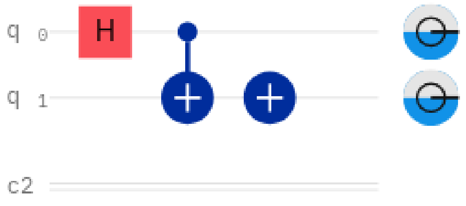

Figure 21: Example of a simple 1 qubit Clifford circuit [IBM, 2021]

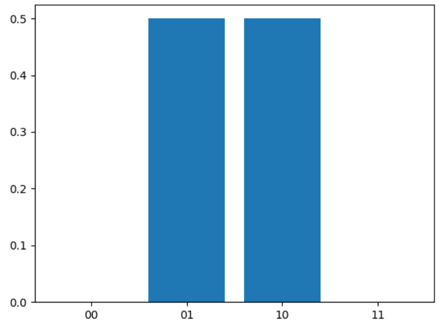

Figure 22: Result of the implementation (ver.2)

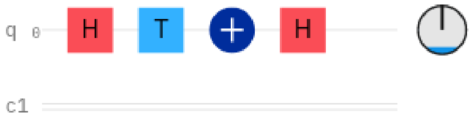

Figure 23: Example of a simple 1 qubit Clifford + T circuit [IBM, 2021]

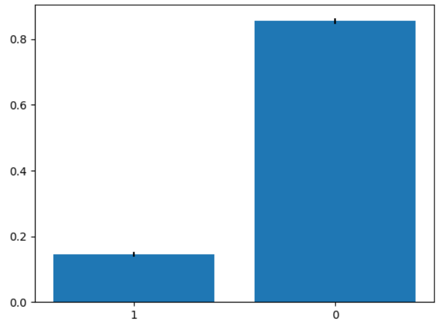

Figure 24: Normalized result of the implementation (ver.3 - 1024 shots)

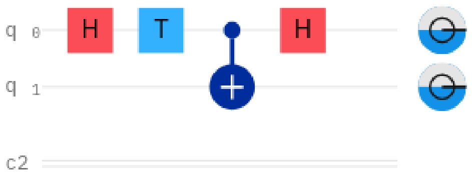

Figure 25: Example of a simple 2 qubit Clifford + T circuit [IBM, 2021]

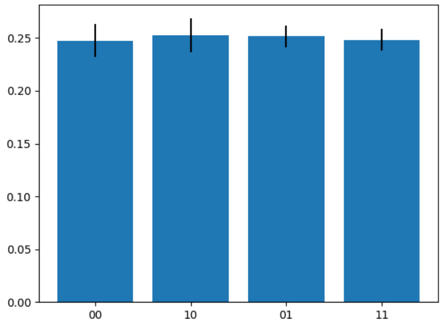

Figure 26: Normalized result of the implementation (ver.3 - 1024 shots)

C Multiple T gates on entangled qubits (Version 3 of simulator)

The issue caused by having multiple T gates on different entangled qubits can be observed for instance using the circuit in figure 27. The expected results are displayed in figure 28. The results obtained using the update rules of appendix B are displayed in figure 29. However, changing the update rules of an X gate to the ones below immediately gives the expected results (figure 30). These new update rules give incorrect results for most other circuits which previously worked as expected. Due to time constraints, better rules could not be found.

```
    $t_j \leftarrow t_j \oplus 1$ ;
    $d_j \leftarrow d_j \oplus b_j$ ;
    $b_j \leftarrow b_j \oplus 1$ ;
```

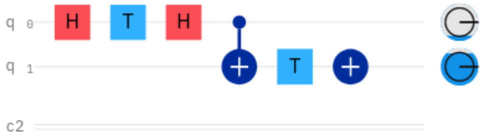

Figure 27: Simple circuit exhibiting two T gates applied of different entangled qubits [IBM, 2021]

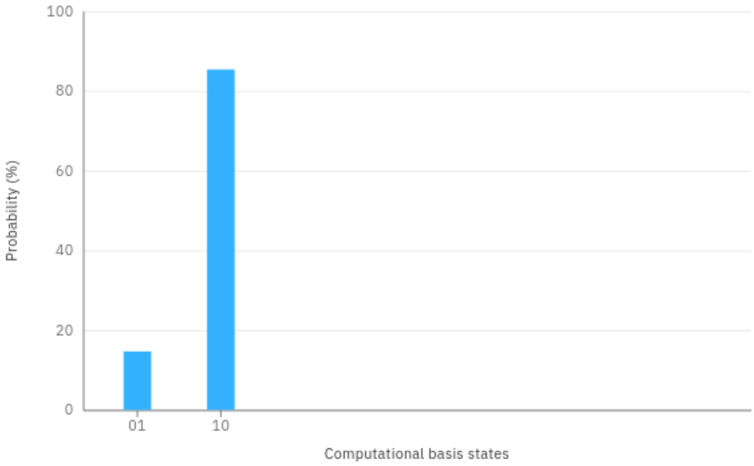

Figure 28: *IBM quantum experience* results for the circuit in figure 27 [IBM, 2021]

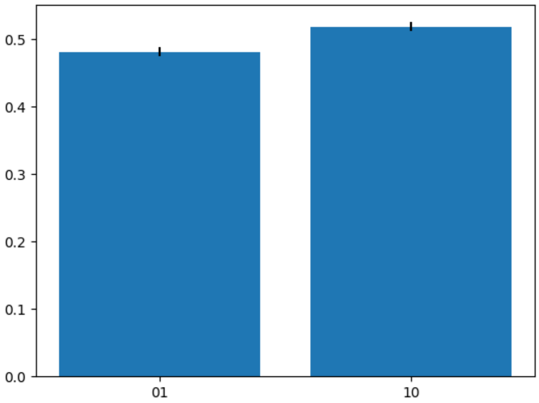

Figure 29: Incorrect normalized result of the A-form simulator for the circuit in figure 27 (ver.3 - 1024 shots)

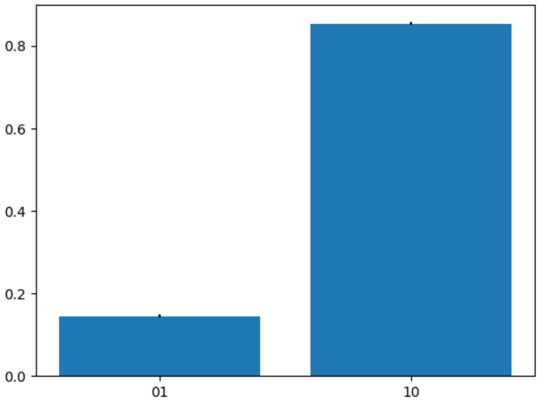

Figure 30: Correct normalized result of the A-form simulator for the circuit in figure 27 (ver.3 - 1024 shots)

References

[Gottesman, 1998] Gottesman, D. The Heisenberg Representation of Quantum Computers. *arXiv:quant-ph/9807006*, 1998. ArXiv: quant-ph/9807006.

[Dehaene and De Moor, 2003] Dehaene, J. and De Moor, B. The Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Phys. Rev. A*, 68(4):042318, 2003. ISSN 1050-2947, 1094-1622. doi:10.1103/PhysRevA.68.042318. ArXiv: quant-ph/0304125.

[Aaronson and Gottesman, 2004] Aaronson, S. and Gottesman, D. Improved Simulation of Stabilizer Circuits. *Phys. Rev. A*, 70(5):052328, 2004. ISSN 1050-2947, 1094-1622. doi: 10.1103/PhysRevA.70.052328. ArXiv: quant-ph/0406196.

[Nest, 2009] Nest, M. V. d. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *arXiv:0811.0898 [quant-ph]*, 2009. ArXiv: 0811.0898.

[Bravyi and Gosset, 2016] Bravyi, S. and Gosset, D. Improved classical simulation of quantum circuits dominated by Clifford gates. *Phys. Rev. Lett.*, 116(25):250501, 2016. ISSN 0031-9007, 1079-7114. doi:10.1103/PhysRevLett.116.250501. ArXiv: 1601.07601.

[Bravyi et al., 2019] Bravyi, S., Browne, D., Calpin, P., Campbell, E., Gosset, D., and Howard, M. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3:181, 2019. ISSN 2521-327X. doi:10.22331/q-2019-09-02-181. ArXiv: 1808.00128.

[IBM, 2021] IBM. Quantum Experience. 2021.

[Pashayan et al., 2021] Pashayan, H., Reardon-Smith, O., Korzekwa, K., and Bartlett, S. D. Fast estimation of outcome probabilities for quantum circuits. *arXiv:2101.12223 [quant-ph]*, 2021. ArXiv: 2101.12223.