MA1 TPIV report

Quantum information

Quentin Pitteloud

December 29, 2020

Contents

1 Introduction

Quantum Computing is a relatively recent field which uses quantum mechanics to process information, similarly as classical computers have done for the past decades. Quantum computers are still somewhat far from having practical applications but a lot of the theoretical framework has already been laid out. Indeed, there already exist many algorithms that promise increased performance over their best classical counterpart, yet we are still unable to benefit from many of those as quantum hardware is still very noisy. Nonetheless, it is still relevant, or at the very least interesting, to review some theoretical elements of this field. In this report, we will go over purely theoretical notions such as the Gottesman–Knill theorem. We will then also take interest in multiple algorithms with a particular emphasis on Grover's algorithm. This report serves as an extensive summary of the work that has been done during the semester.

2 Gottesman-Knill theorem

The Gottesman-Knill theorem is an important theorem in quantum information, and more specifically in the simulation of quantum gates. The statement of the theorem is as follows:

Theorem: Any quantum computer performing only: a) Clifford group gates, b) measurements of Pauli group operators, and c) Clifford group operations conditioned on classical bits, which may be the results of earlier measurements, can be perfectly simulated in polynomial time on a probabilistic classical computer.

The Clifford group gates is composed of $\{H, S, CNOT\}$. One might notice that the Clifford group is only a T gate away from being universal for quantum computation, as the group $\{H, S, T, CNOT\}$ is universal. Their action on some elements of the Pauli group are shown in fig. 1.

R	$X \to Z$ $Z \to X$
P	$X \to Y$ $Z \to Z$
CNOT	$X \otimes I \to X \otimes X$ $I \otimes X \to I \otimes X$ $Z \otimes I \to Z \otimes I$ $I \otimes Z \to Z \otimes Z$

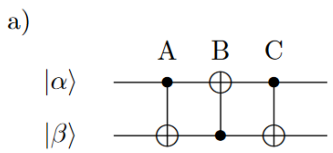
Figure 1: Effect of Clifford gates on elements of the Pauli group [1]. Note that in this figure, the Hadamard gate is denoted by R and P is the phase gate

The most important consequence of this theorem is that any circuit that verifies the conditions (a),(b) and (c) cannot have an exponential speedup over classical algorithms. Indeed, if it were to be the case for some quantum algorithm, the polynomial overhead due to the simulation would be largely compensated by the exponential speedup. Thus, the simulation of this quantum algorithm on a classical computer would simply become the fastest classical one. The quantum process would have the same performance as this newly found classical counterpart.

Inversely, any (part of a) circuit that offers only a polynomial speedup can be written using only clifford gates.

Additionally, the polynomial overhead of the simulation compensates any polynomial speedup gained by a quantum circuit. As such, it would not be faster to simulate the Quantum search algorithm (which has a quadratic speedup) on a classical computer than running a classical search algorithm.

The method used to simulate the circuit is not to compute the state of the computer after each gate, but rather to compute the operation the circuit will execute, without doing any matrix calculus. This is done by using the identities in figure 1 and computing what the circuit does to the elements of the Pauli group. An example of the method is displayed in fig. 2.



b)

A: CNOT($1 \to 2$)

$\overline{X}_1$	$X \otimes X$
$\overline{X}_2$	$I \otimes X$
$\overline{Z}_1$	$Z \otimes I$
$\overline{Z}_2$	$Z \otimes Z$

B: CNOT($2 \to 1$)

$\overline{X}_1$	$I \otimes X$
$\overline{X}_2$	$X \otimes X$
$\overline{Z}_1$	$Z \otimes Z$
$\overline{Z}_2$	$Z \otimes I$

C: CNOT($1 \to 2$)

$\overline{X}_1$	$I \otimes X$
$\overline{X}_2$	$X \otimes I$
$\overline{Z}_1$	$I \otimes Z$
$\overline{Z}_2$	$Z \otimes I$

Figure 2: Example of the method used to simulate a circuit [1]. Note that in this figure, the bar on top of a gate indicates that it is the initial gate.

The proof of the theorem has been given as a set of simple circuits on which the method is applied.

3 Deutsch-Jozsa, Bernstein-Vazirani and Simon's algorithms

3.1 Deutsch-Jozsa algorithm

This algorithm is a demonstration of the quantum parallelism but in practice it does not solve any important problem. Its circuit can be seen in figure 3. Using this algorithm, one can know whether a function $f : \{0,1\}^n \to \{0,1\}$ is balanced (half of the possible input values yield 0 and the other half 1) or constant, in a single run of the algorithm.

This algorithm is also deterministic, meaning that the probability to find the correct result is 1.

The algorithm works by using one register of n qubits and an ancilla qubit initialized at $|1\rangle$. It uses an oracle to implement the operation $|x\rangle_n |y\rangle \to |x\rangle_n |y \oplus f(x)\rangle$, where $|\cdot\rangle_n$ denotes the state of the n-qubit register. If the function is constant, measuring the first register will always yield $|0\rangle^n$ while if the function is balanced the inverse is true.

3.2 Bernstein-Vazirani algorithm

This algorithm aims to prove the separation of classical and quantum complexity. The algorithm provides the s bitstring for a function $f : \{0,1\}^n \to \{0,1\}$ defined as $f(x) = x \cdot s$, where $\cdot$ is defined as the dot product modulo 2. This time, the oracle performs the transformation

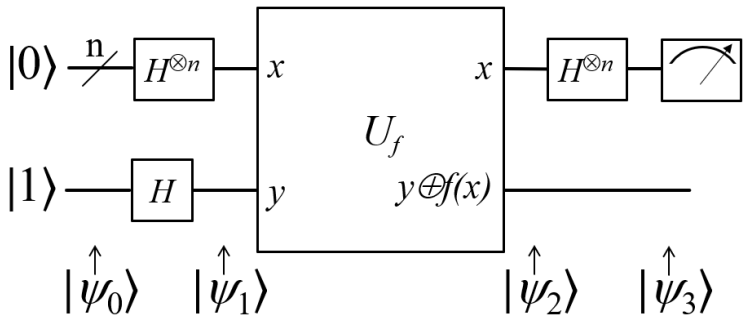


Figure 3: Circuit of the Deutsch-Jozsa algorithm [2]

$|x\rangle \rightarrow (-1)^{f(x)} |x\rangle$. The circuit is otherwise identical to the one of Deutsch-Jozsa algorithm. The s bitstring is obtained by measuring the n qubit register.

3.3 Simon's algorithm

This algorithm solves a similar problem as Bernstein-Vazirani algorithm. The problem is as follows: given $f : \{0,1\}^n \rightarrow \{0,1\}^n$ such that $f(x) = f(x \oplus s)$, find s. The output is taken randomly from the set of values y such that $y \cdot s = 0$, with $\cdot$ once more the scalar product over $\mathbb{Z}_2^n$. Hence, to obtain s, the algorithm must be run multiple times in order of obtain n independant values y in order to solve the system of equations composed of the outputs $y \cdot s = 0$. The circuit can be found in figure 4.

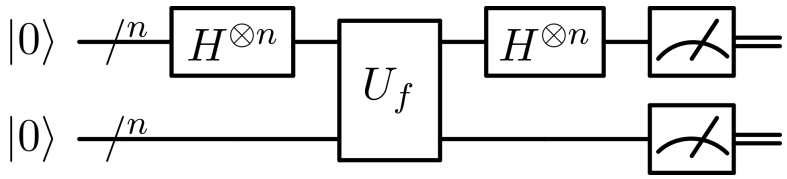


Figure 4: Circuit of Simon's algorithm [2]

3.4 Similarities and discussion

All 3 algorithms use quantum parallelism and an oracle to obtain a speedup over classical algorithms. Indeed, each algorithm requires only one use of the oracle per run to produce an output while in the classical case, the oracle can be called up to $O(2^{n-1}+1)$ times. Additionally, the use of quantum parallelism indicates a large amount of intrication between the qubits. However, with only a polynomial speedup in $N = 2^n$ this would mean that while intrication is used by the algorithms, it is not used to speed up the computation as much as some other algorithms do. Furthermore, a polynomial speedup is in accordance with the fact that the circuits of the algorithms contain only Clifford gates (aside from the oracle).

4 Grover's Algorithm

Grover's algorithm, also called the Quantum Search Algorithm, is a probabilistic process that is able to find an element in an unordered set in $O(\sqrt{N/M})$ operations, where $N = 2^n$ is the number of elements in the set and M the number of solutions. Note that its classical counterpart will require $O(N/M)$ operations to find a solution. The efficiency of this algorithm

is asymptotically optimal. Indeed, it has been shown by Bennett *et al.* [3] that a quantum computer could at best speed up search algorithms quadratically.

In the case $M = 1$, this search process can be seen as a way to invert a function $f(x) = y$, as it returns x for a given y.

The input of the algorithm is a uniform superposition of states $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$. The process can be decomposed in 4 steps:
1) apply the oracle O, 2) apply Hadamard gates over the n qubits, 3) Do a conditional phase shift defined as $|x\rangle \to -(-1)^{\delta_{x,0}} |x\rangle$ and finally 4) apply Hadamard over the n qubits. In operator notation, this becomes $G = H^{\otimes n}(2 |0\rangle \langle 0| - I)H^{\otimes n}O = (2 |\psi\rangle \langle \psi| - I)O$.

The circuit of the algorithm is presented in fig. 5 and the circuit of the iterator is presented in fig. 6.

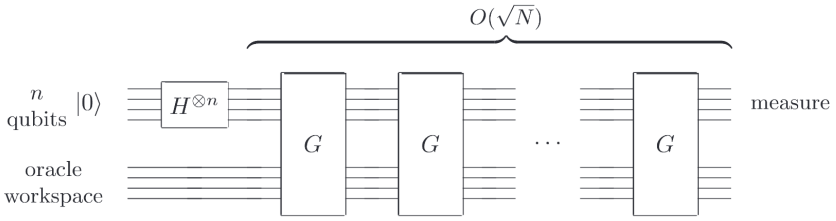


Figure 5: Circuit of Grover's algorithm [2]

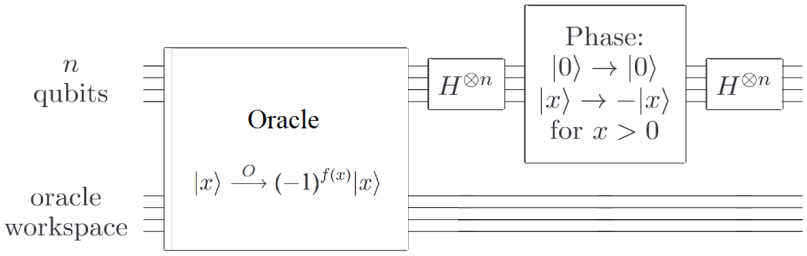


Figure 6: Circuit of the iterator G [2]

This algorithm can be visualized geometrically in the 2D space spanned by the uniform superposition $|\psi\rangle$ of the computational basis and the uniform superposition of the solutions, noted $|\phi\rangle$. The oracle O does a reflection about the orthogonal of $|\phi\rangle$ while the conditional phase shift and Hadamard (*i.e.* the operator $2 |\psi\rangle \langle \psi| - I$) does a reflection about $|\psi\rangle$. Hence, the iterator $G = (2 |\psi\rangle \langle \psi| - I)O$ is a rotation in this 2D space. The angle of this rotation is around $2\pi/\sqrt{N}$ (for $M = 1$) and rotates the states towards $|\phi\rangle$. Thus, after $O(\sqrt{N})$ iterations, the probability to measure a solution is much larger than measuring a wrong solution.

This algorithm is very useful, as many problems reduce to a search problem, such as decision problems or the prime factoring problem. The former represents problems that can be posed as simple yes/no question of the input. The latter is an important problem for internet security, but it has to be noted that Grover's algorithm is less efficient than Shor's algorithm.

In light of the Gottesman-Knill theorem, one might notice the quadratic speedup of this algorithm means that it can be efficiently simulated using a classical computer. However, this is only the case as long as the oracle can itself be constructed from Clifford gates. As such, in the following sections we will take a closer look on the oracle in this algorithm.

5 Construction of an oracle for Grover's algorithm

An oracle is a "black box" piece of code which provides a precise computation that is non trivial. In the case of Grover's algorithm, the oracle has to implement the computation

$$|x\rangle \to (-1)^{f(x)} |x\rangle \,, \tag{1}$$

where $f(x)$ is a function that verifies $f(x) = 1$ if x is a solution and $f(x) = 0$ otherwise. In the following sections, we study in detail this oracle in different cases.

5.1 Constructing a single target oracle for n qubits

While Grover's algorithm works for any number of solutions M, the particular case of $M = 1$ is interesting both for the fact that it is the most computationally heavy case and that it can be seen as inverting a function. Hence, we will try to build an oracle with only a single target. The central idea to build it is to use a multiple controlled Z gate (abbreviated MCZ) on all qubits, as it already implements an oracle with target $|11...1\rangle$. In order to change the target, X gates can be applied on individual qubits before and after the MCZ gate. For example, the oracle for $|00...0\rangle$ will have X gates applied on all qubits.

The implementation of the MCZ gate is done the same way as shown in figure 7. This implementation for n qubits requires $n-1$ ancilla qubits and $2(n-1)$ Toffoli gates. This makes this implementation not particularly efficient as the Toffoli gates require many elementary gates to be implemented. Indeed, the implementation of the Toffoli gate is shown in fig. 8.

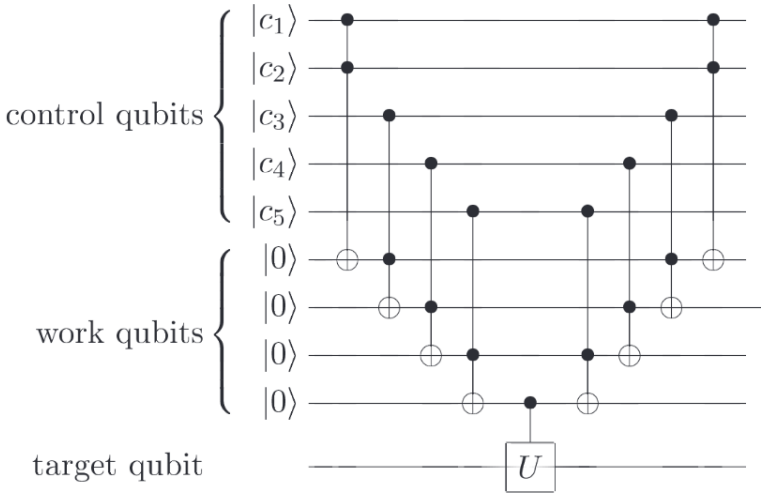


Figure 7: Implementation of the multiple controlled U gate for $n = 5$ qubits [2]

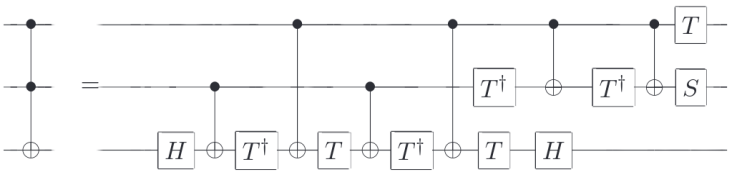


Figure 8: Implementation of the Toffoli gate [2]

5.2 Improving the efficiency of the oracle

In order to improve the performance of the oracle, one could seek alternative ways to implement the relevant gates. As it happens, there exists a way to efficiently construct diagonal gates using only CNOTs and single qubit gates, as presented in the article *Programmable networks for quantum algorithms* [4]. Following the method described in this paper, a MCZ gate does not require Toffoli gates nor does it needs ancilla qubits in its implementation. The implementation of a CCCZ gate (Triple controlled Z gate) is shown in figure 9 where the ± gates are defined as

$$[\pm] = R_z(\pm\pi/16) = \begin{pmatrix} 1 & 0 \\ 0 & exp(\pm i\pi/8) \end{pmatrix}. \tag{2}$$

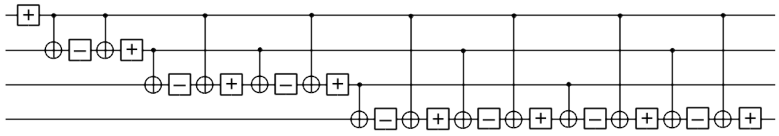


Figure 9: Alternative implementation of the CCCZ gate [4]

5.3 Oracles built with Clifford gates and their limitations

We've seen that Grover's algorithm provides a polynomial speedup and thus it can be built using only Clifford gates. However, this does not take into account the complexity of the oracle. One might then wonder: Is it possible to build an oracle for Grover's algorithm using only Clifford gates and if so, are there constraints to these oracles? This question is answered fully by the following proposition.

Proposition: For n>2 qubits, it is not possible to build an oracle for Grover's algorithm using only Clifford gates with fewer targets than $2^n/4$.

The proof is done by induction. Let's start by the base case n=3. A single target oracle can be written in this case using X gates and a CCZ gate, which in turn can be written as

$$CCZ = (I \otimes I \otimes H)T_{of}(I \otimes I \otimes H), \tag{3}$$

where I is the single qubit identity gate and T_{of} is the Toffoli gate. As we know, the Toffoli gate cannot be written using only Clifford gates and thus neither can the single target oracle [5]. If the oracle has $2 = 2^3/4$ targets, it can be written for instance with a CZ gate on any pair of qubits. As the CZ gate can be written using a CNOT and Hadamard gates using the same technique as in equation 3, this oracle can be constructed from Clifford gates. This proves the base case.

Suppose now that the proposition is true up to an arbitrary number of qubits n. Let us prove the proposition for $n+1$ qubits. We will consider oracles with m targets, $1 \leq m < N/4$ with the notation $N = 2^{n+1}$, in order to prove the proposition. The upper bound $N/4$ can be easily verified to be accessible by Clifford oracles, for instance by using a single CZ gate on any pair of qubits. The key idea here is to decompose the m-targets circuit into subcircuits of 1 or 2 targets. The subcircuits with 2 targets must have them distinct only by the first qubit, *i.e.* targets of the form $|0x_1x_2x_3...\rangle$ and $|1x_1x_2x_3...\rangle$ where x_i designates the value of the $(i+1)^{th}$ qubit. We will consider subcircuits individually.

First consider the case of a subcircuit with two targets. Using the way we have built our 2-targets subcircuits, one can write one of those with a n qubit single target oracle. However, following the induction assumption, it is not possible to write this oracle using Clifford gates. Now consider the case of a single target subcircuit. Without loss of generality, we can assume the target to be $|0\rangle$ (in the notation $|x\rangle$ where x is the value of the bitset in base-10). Then one can write the oracle in matrix form ($N \times N$) as

$$O = \begin{pmatrix} -1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}. \tag{4}$$

To see whether this oracle is in the Clifford group, we will see its action on some element of the Pauli group. Take for instance

$$(I^{\otimes n} \otimes X) = \begin{pmatrix} X & & & \\ & X & & \\ & & \ddots & \\ & & & X \end{pmatrix}, \tag{5}$$

where I is the 1 qubit identity gate, X is the 1 qubit X gate (NOT gate), and the matrix is written in block diagonal form for convenience. Note that the resulting matrix is $N \times N$ as is the previous oracle. Now, one study the effects of O on this particular element:

$$O(I^{\otimes n} \otimes X)O^\dagger = O(I^{\otimes n} \otimes X)O = \begin{pmatrix} -X & & & \\ & X & & \\ & & \ddots & \\ & & & X \end{pmatrix} \notin P_{n+1}, \tag{6}$$

where P_{n+1} is the Pauli group on $n+1$ qubits. Hence, O is not in the Clifford group, and thus cannot be written using only Clifford gates. This concludes the proof of the proposition.

As said in the proof, an oracle with $2^n/4$ targets can be written using a simple CZ gate. We can give a simpler (but unrigorous) argument to show that this is the best result achievable using only clifford gates. The only controlled gate in the Clifford group is the CZ gate (the CNOT gate being equivalent to the CZ gate). Additionally, it is a diagonal gate, so it commutes with any other diagonal gate. Thus, one can only apply a single CZ gate to any pair of qubits before it cancels itself out. Moreover, in the limit of many qubits, an oracle made only of Clifford gates (single qubits gates and CZ gates) will tend to have $2^n/2$ targets. Indeed, to prove this last statement, one must first notice that the single qubit gates do not alter the number of targets. Secondly, consider the ratio R of targets to total number of possible outcomes, *i.e.* $R = (\text{number of targets})/2^n$. For instance, a CZ gate has a ratio $R(CZ) = 1/4$, and $R(CZ \otimes CZ) = 3/8 = 1/4 + 1/8$. Hence,

$$\lim_{n\to\infty} R((CZ)^{\otimes n}) = \lim_{n\to\infty} \sum_{j=2}^{n} \left(\frac{1}{2}\right)^j = -1 - \frac{1}{2} + \sum_{j=0}^{\infty} \left(\frac{1}{2}\right)^j = -\frac{3}{2} + \frac{1}{1-1/2} = \frac{1}{2}, \tag{7}$$

where the solution of a geometric series has been applied to find the result. We then see that applying more than a single CZ gate will only increase the number of targets in our oracle, and since the CZ gate has a ratio $R = 1/4$ it cannot be improved using Clifford gates.

6 Python implementation of the Grover algorithm

The Quantum Search Algorithm can be easily implement in Python thanks to Qiskit. In an attempt to improve readability, the python files will not be included in this report. The first python implementation of the algorithm has been done using the method of constructing MCZ gates shown in fig. 7. Considering the *little endian* convention of Qiskit, the order of the qubits in the quantum register is reversed. For 3 qubits, this gives the circuit shown in fig. 10. From now on, we will consider that the target is $|00...0\rangle$.

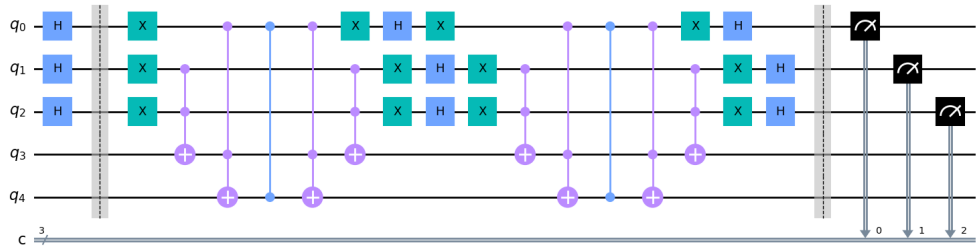


Figure 10: Circuit of Grover's algorithm for 3 qubits (target = $|000\rangle$)

This circuit can easily be improved by noting that a single Toffoli gate along with two Hadamard gates can produce a single target oracle for 3 qubits. Indeed, recall from equation 3

$$(I \otimes I \otimes H)T_{of}(I \otimes I \otimes H) = CCZ. \tag{8}$$

The imporved circuit is shown in figure 11.

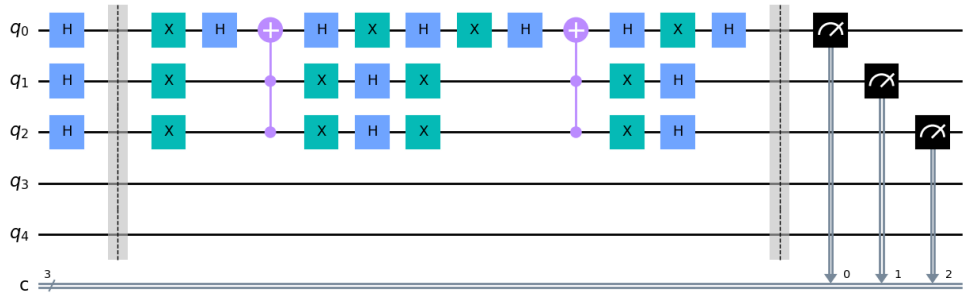


Figure 11: Improved circuit of Grover's algorithm for 3 qubits

For more qubits, the technique from paper [4] to implement a multiple controlled Z gate (MCZ gate) can be used. For 4 qubits for instance, recall that the circuit for the CCCZ gate is shown in figure 9. One can then use this circuit in Grover's algorithm, which gives the circuit in figure 12.

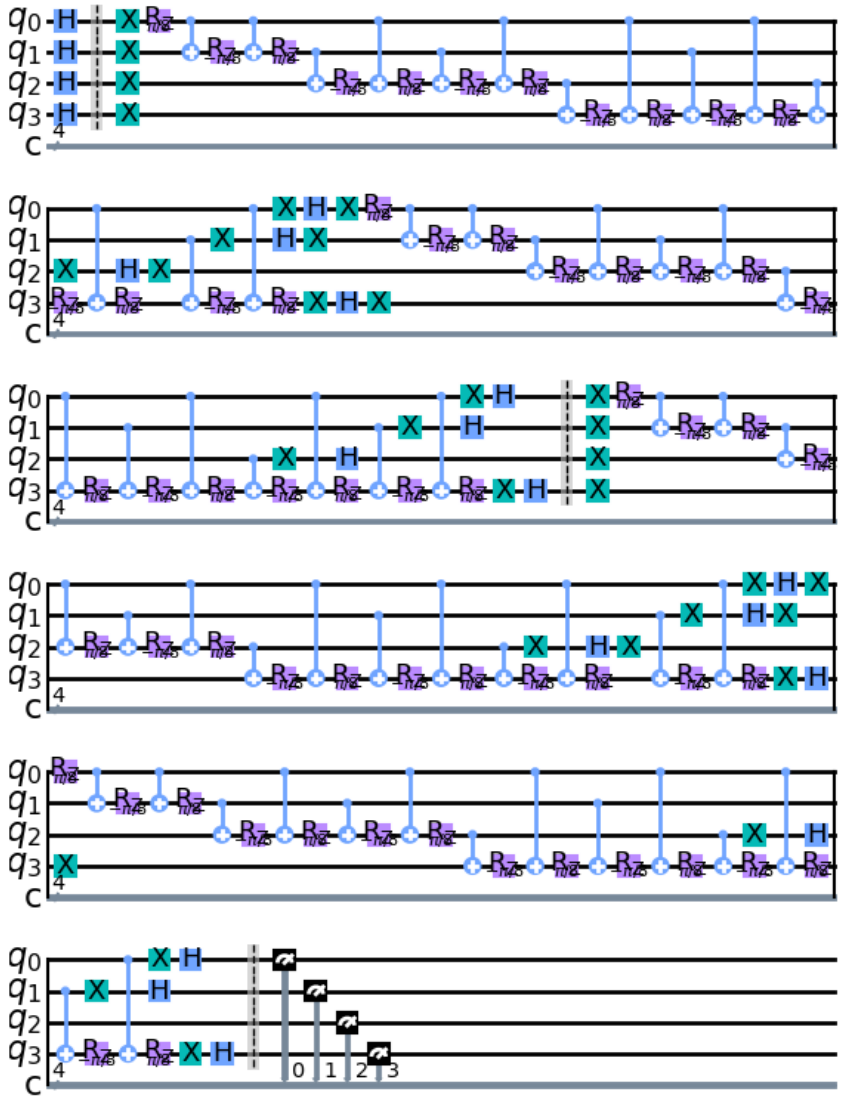


Figure 12: 4 qubits circuit for Grover's algorithm

6.1 Noise resistance

Up until now, we've only considered applying Grover's algorithm to a small number of qubits. For any practical application, this is not sufficient. However, before trying to run the algorithm with many qubits, it is wise to check that the algorithm is still applicable for more qubit in spite of the noise. Hence, a convergence study has been done to study the behaviour of the algorithm from 2 to 10 qubits. The figure 13 displays the convergence study. The upper boundary was chosen due to computational power constraints. This range is still satisfactory as it covers most of the pubicly accessible quantum computers. The study observes the evolution of the probability to find the correct target (arbitrarily chosen to be $|000...\rangle$) and the probability to

land on noise versus the number of qubits. The errors were very crudely estimated by doing the simulation multiple times for every number of qubits.

As one can see in the aformentioned figure, the target is still distinguishable from the noise for 10 qubits. Moreover, both curves display the same overall behaviour. Hence, they do not appear like the critical number of qubits (defined as the number of qubits where noise renders target and non-targets undistinguishable) is close to the upper limit of the considered range of qubits. However, these results are to be taken with a grain of salt since a simulator was used to obtain these results. Doing the same convergence study on a real quantum computer would likely display a lower critical number qubits until the target is indistinguishable from noise.

Furthermore, the circuit used to obtain these results uses the first unoptimized version of the oracle (figure 7). Hence the real maximum number of qubits considered in this study is 19 qubits (10 qubits + 9 ancilla).

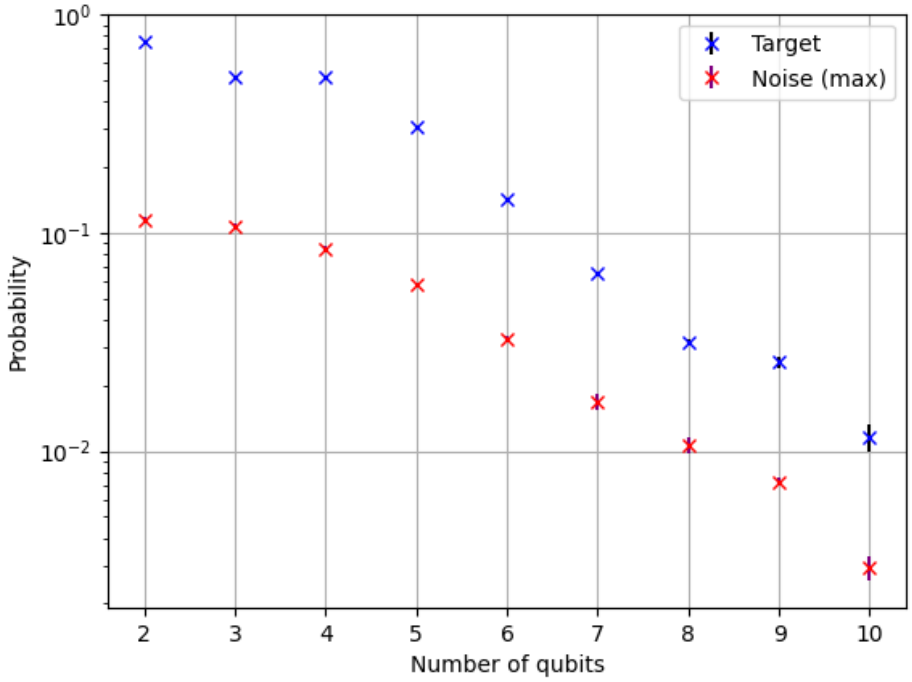


Figure 13: Convergence study of Grover's algorithm

The same graph as fig. 13 without errorbars, in linear scaling and with a wider range of qubits can be seen in appendix A, figure 25. Note that this figure shows the critical number of qubits to be around 14. Also, 14 qubits seemed to be a hard limit for the simulator as it required a total of 27 qubits (of which 13 are ancilla qubits).

6.2 Compiled circuit

In the previous sections, we used a simulator to run the algorithm for convenience. The simulator provided by Qiskit does not use any specific qubit architecture. This implies that the results obtained with a real device would differ substantially from those obtained with the simulator. In order to mitigate this error factor, we will now take into account the architecture of the computer and its effects on the circuits. We will thus consider compiled circuits.

The chosen architecture for our study is the one of IBM's quantum computer in Valencia. This choice was made as it was the least busy computer at the time. The specifics of the computer are shown in figure 14.

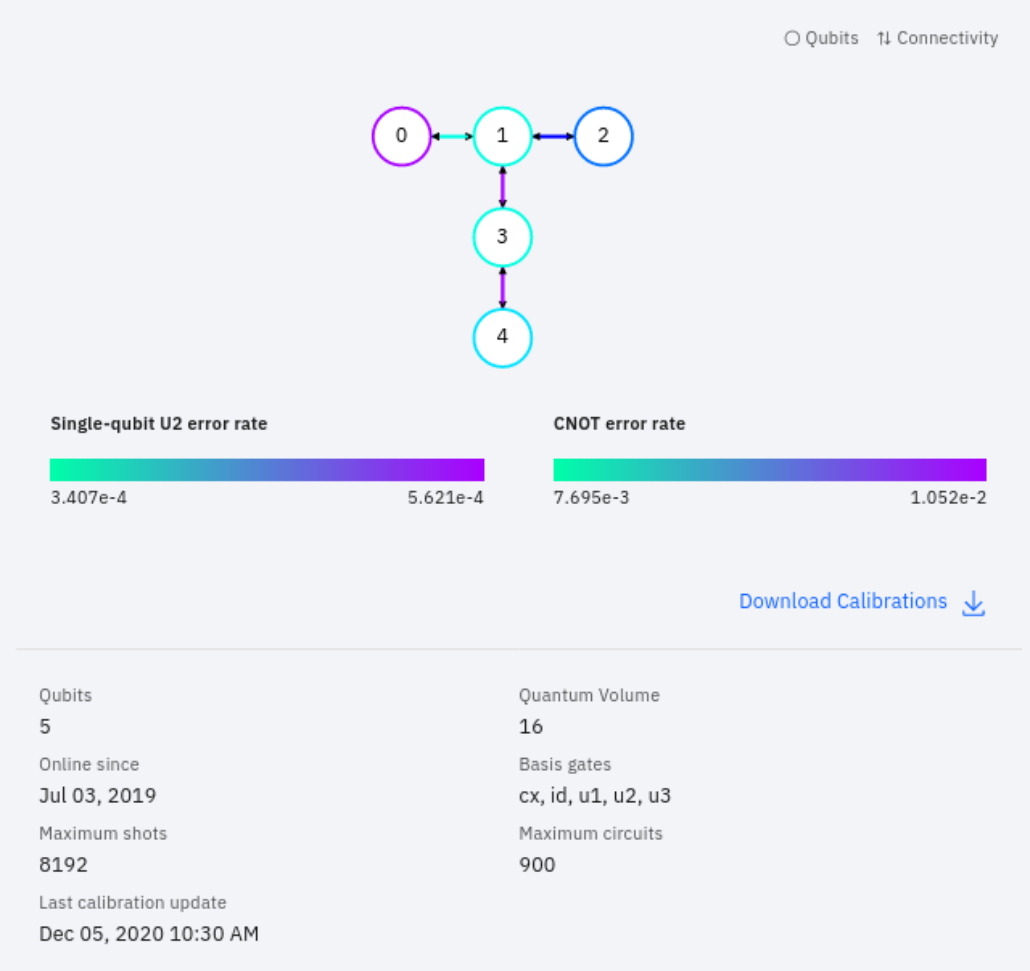


Figure 14: Qubit layout and specifics of IBM's computer in Valencia [6]

By inputing the circuit shown in figure 10 in the transpiler for the selected architecture, one obtains the circuit shown in figure 15. To improve readability, a zoom of the first quarter of the circuit is provided in figure 16. This circuit was then used both in a simulator with a noise model (from the device) and with the real device. The results with compiled circuit are presented in fig. 17 for the simulator and fig. 18 for the real device. The simulator does not follow the behaviour of the real device in the slightest. The target is still distinct from the rest of the possibilites with the simulator, while with the real device, the target is undistinguishable from the noise.

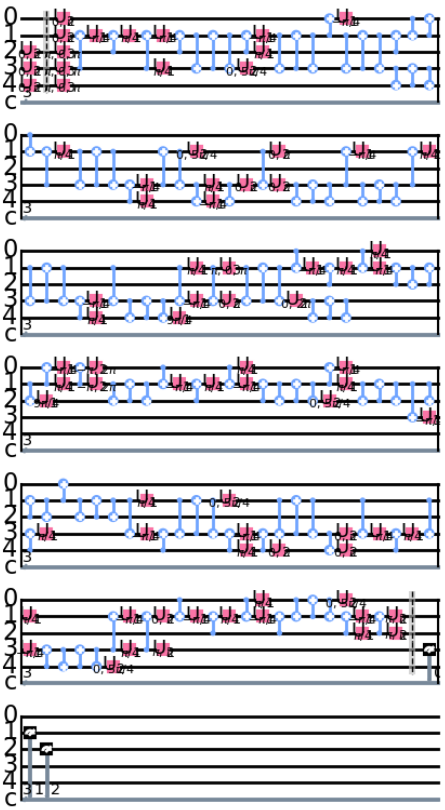


Figure 15: Compiled circuit on IBM Valencia

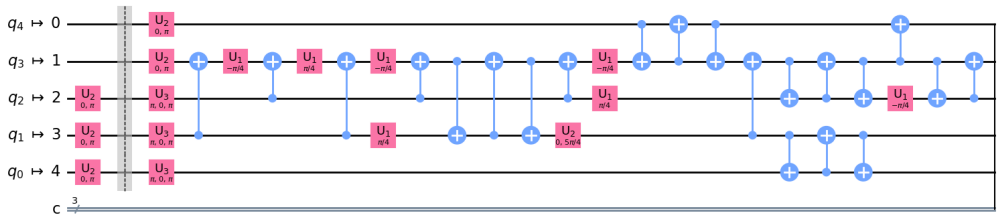


Figure 16: Zoom of the compiled circuit on IBM Valencia

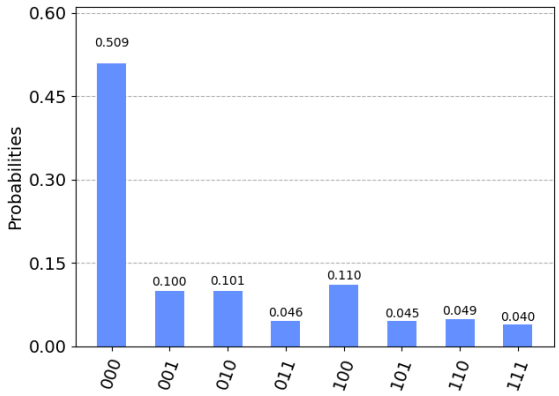


Figure 17: Results from simulator with noise on compiled circuit (fig. 15)

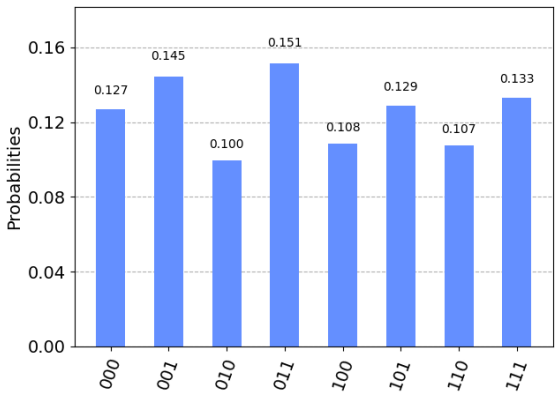


Figure 18: Results from real device on compiled circuit (fig. 15)

In order to improve the results of figures 17 and 18, the improved circuit of figure 11 will be used. The compiled circuit is presented on figure 19, while the results with the simulator (on the compiled circuit) is presented in figure 20 and with a real device in figure 21. The results of the simulator is much closer to the ones of the quantum computer, with both results still clearly distinguishing the target from the noise. However, some slight differences can be seen. With the quantum computer, the noise is slightly more homogeneous than with the simulator. This difference could however be attributed to the probabilistic nature of quantum mechanics, but it would still be of interest to try and improve the results of the simulator if possible.

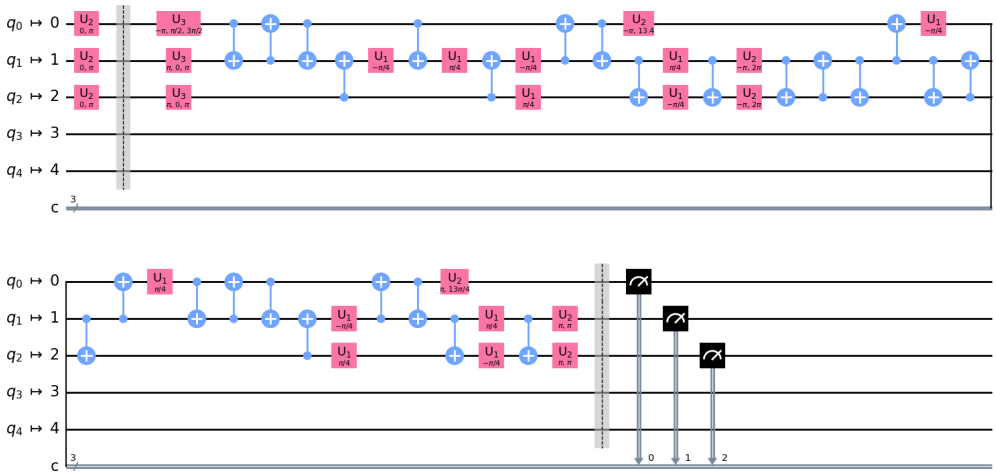


Figure 19: Improved circuit compiled on IBM Valencia

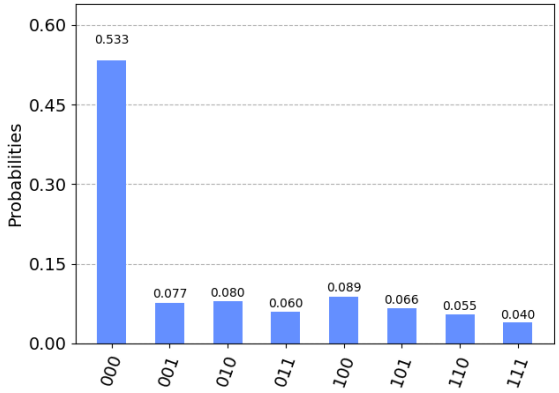


Figure 20: Results from simulator with noise on improved compiled circuit (fig. 19)

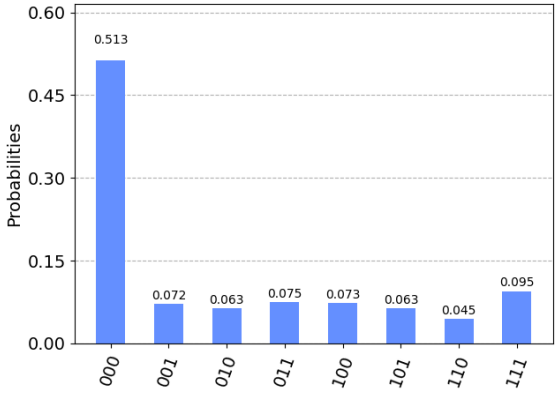


Figure 21: Results from real device on improved compiled circuit (fig. 19)

One main drawback of the simulator is that it does not seem to simulate errors on idle qubits [7]. In an attempt to remedy this, identity gates have been applied to idle qubits at every available space. The modified circuit can be visualized in figure 22. The results shown in figures 23 and 24 indicate that this method of simulating noise in idle qubits does not improve the results with the simulator. Indeed, the newly obtained results differ more from the previous results obtained with a quantum computer (figure 21) than the previous simulator results did (figure 20). Moreover, the real device and the simulator now show outputs ever more different than they did previously.

This method does not allow one to simulate noise on idle qubits, but it does show that identity gates are not simulated accurately. Also, it is to be noted that the model does include measurement errors by default. Hence, the built-in noise model from Qiskit does not always faithfully replicate the behaviour of a real device. This discrepancy is exacerbated for circuits containing identity gates or qubits that stay idle for a relatively long while.

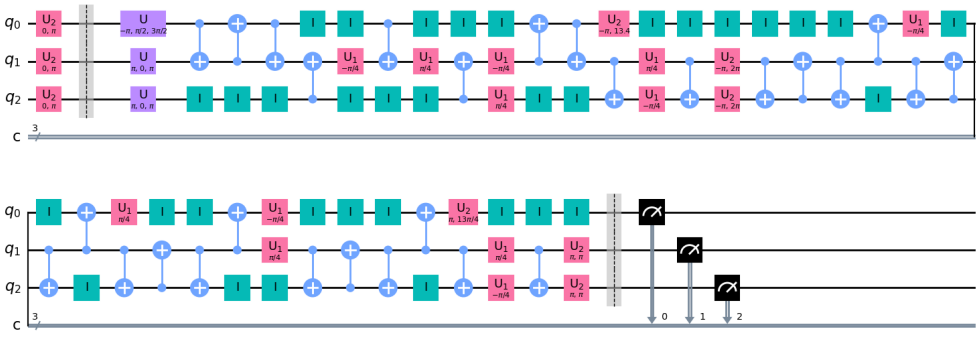


Figure 22: Modifed circuit

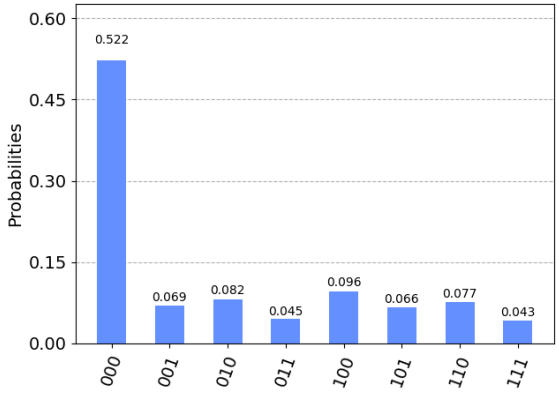


Figure 23: Results from simulator with noise on the modified circuit (fig. 22)

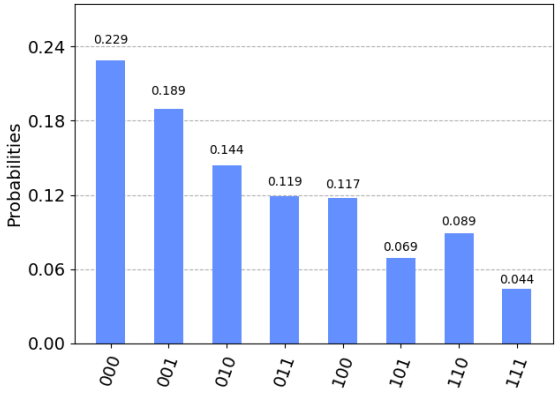


Figure 24: Results from real device on modified circuit (fig. 22)

7 Conclusion

In this report we've presented some of the fundamental oracle-based quantum algorithms, such as the Deutsch-Jozsa and the Quantum Search Algorithm. We have dug deeper in the latter mainly by trying to build an optimized oracle. This optimization came in two ways, first by trying to write it using only Clifford gates which we've shown that it comes with the limitation that there will be at least $2^n/4$ targets, and secondly by limiting at much as possible the use of gates and ancilla qubits. This second way of optimization was already solved in the paper *Programmable networks for quantum algorithms* [4]. The algorithm was then implemented it in python where its resistance to noise was studied. Finally, the compiled circuits were used to evaluate the fidelity of the simulator with respect to the real device.

A Python implementation

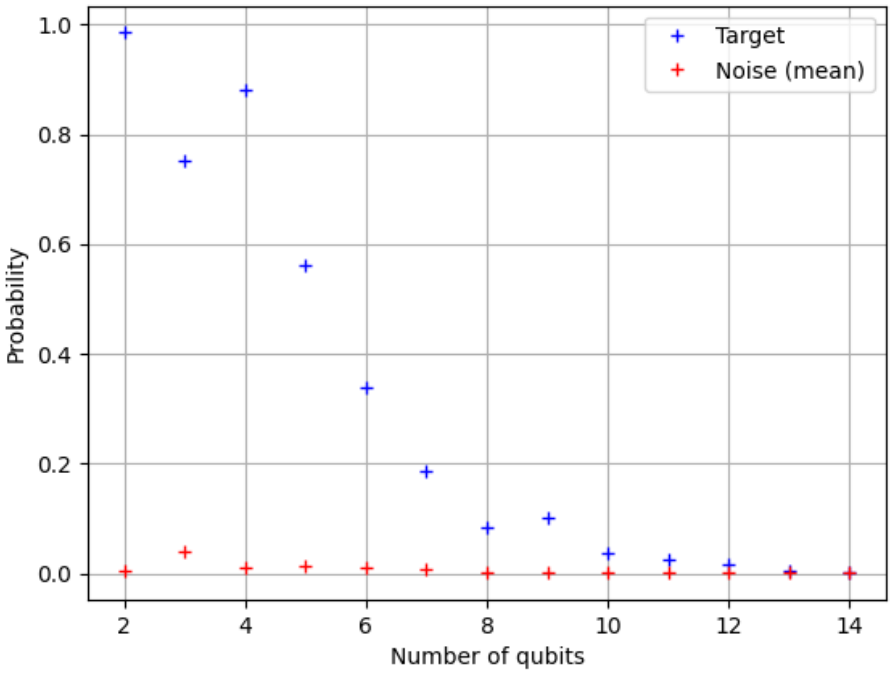


Figure 25: Convergence study, similar to figure 13

References

[1] D. Gottesman, *The Heisenberg Representation of QUantum Computers*. (2008) https://arxiv.org/pdf/quant-ph/9807006v1.pdf

[2] M. Nielsen, I. Chuang, *Quantum information and quantum computation*, Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[3] C. H. Bennett, E. Bernstein, G. Brassard, U. Vazirani, *Strengths and Weaknesses of Quantum Computing*. SIAM Journal on Computing. https://arxiv.org/pdf/quant-ph/9701001.pdf

[4] N. Schuch, J. Siewer, *Programmable networks for quantum algorithms*. 2003. https://arxiv.org/pdf/quant-ph/0303063.pdf

[5] D. Aharov, *Toffoli and Hadamard are Quantum Universal*, https://arxiv.org/pdf/quant-ph/0301040.pdf

[6] *IBM Quantum Experience website*, url: https://quantum-computing.ibm.com/,retrieved December 2020.

[7] *Qiskit documentation on the noise_model function*, url: https://github.com/Qiskit/qiskit-aer/blob/master/qiskit/providers/aer/noise/noise_model.py, retrieved December 2020.

[8] Cuffaro, M. E. *On the Significance of the Gottesman–Knill Theorem*. The British Journal for the Philosophy of Science. (2005) axv016. doi:10.1093/bjps/axv016

[9] J. Preskill's lecture notes, Caltech, url: http://theory.caltech.edu/~preskill/ph219/index.html#lecture retrieved December 2020.