Investigating the applicability of quantum algorithms in current LIGO/Virgo gravitational wave data-analysis methods

Jonas Tjepkema
Supervised by Dr. Gideon Koekoek

December 2021

Abstract

Future generations of gravitational wave detectors will output massive amounts of data that will eclipse current super computer algorithmic capabilities. In preparation for this, the scientific community needs to come up with solutions for handling this data. This paper investigates the use of quantum computing as a tool for improving the computational power of gravitational wave data analysis. This is done by exploring the replacement of parts of classical matched filtering with a quantum alternative using Grover's algorithm. A fully functional quantum circuit for SNR value comparison and extraction from a database is presented and a full theoretical deconstruction made. An analysis of it's capabilities is done, partly on real world data, as well as observations on its behavior when used in quantum counting or in multi-Grover iteration circuits. This is followed by a study of the evolution of circuit complexity in function of the size of the database and of the SNR values. The measure of complexity was defined as a combination of the gates used in the final circuit. Finally, the benefits of splicing the database to optimize complexity and modifying the circuit transpilation method are also discussed.

Contents

1 Introduction

The theory of General Relativity first developed by Albert Einstein is one of the most successful theories in the physical sciences. It describes with stunning accuracy currently visible gravitational effects and predicts phenomena that were and still are inexplicable with classical Newtonian physics, like special planetary motions (Naeye, 2009), the existence of black holes (Schwarzschild, 1916), the expansion of the universe (Friedmann, 1922) and most importantly for this project, gravitational waves (Einstein, 1916). These waves are ripples of spacetime created by any travelling time-varying mass quadrupole moment and are extremely weak and difficult to measure as they only affect matter very subtly. Consequently, scientists have only been able to observe them as products of the most violent collisions between very compact objects like neutron stars and black holes (Abbott et al., 2021). Unlike electromagnetic waves, gravitational waves do not scatter easily when travelling through space, giving us a unique way to observe events that happened in the very early universe as well as events that do not emit any light or electromagnetic effects. The current tools to observe these waves are very large laser-interferometers that beam light through two perpendicular arms and mirrors with the purpose of creating an interference pattern that can be analysed into a signal. This pattern will change very slightly when a gravitational wave (GW) passes through and stretches one arm of the interferometer in relation to the other (Barish, 1999). This change in signal can then be deconstructed into the shape of the wave as well as an estimation of the source of the wave. There are currently four of these detectors in the world, of which only Virgo (Italy) and LIGO (USA) have made gravitational wave detections thus far.

But analyzing these signals is no small task, illustrated by the fact that the first gravitational wave, GW150914, was only observed in 2015 and processed in 2016 (Abbott et al., 2016). The waves create absolutely minuscule changes in the distance between the interferometers mirrors, and their extreme sensitivity (LIGO for example can detect a difference of 1/10,000th the width of a proton) leads to enormous amounts of noise from their surroundings, as any vibration will be detected (Camp, 2004). When a wave is detected it is directly fed into a data processing pipeline that will denoise it and make it usable for data-analysis in the shape of a frequency-time signal. This contains the information leading to and including the collision, called the "chirp" of the signal. But this is not where the analysis ends, as it is difficult to extract the information from the chirp. The process relies on creating many simulations of compact body collisions with different attributes (like the mass and spin of the bodies, the distance at which the collision happened, etc...) using a combination of the Einstein field equations and post-Newtonian formulas.

These simulated events are called templates and take a lot of time to compute using super-computers because of the high dimensionality of the parameter space. Once the template bank is complete, each template is compared to the signal and the Signal-to-Noise (SNR) is computed, which can be used as a measure of how well the template fits the signal. This last process is called matched filtering and it is central to gravitational wave data-analysis. Unfortunately, it has a very high computational cost due to the number of comparisons that have to be made between the template and the signal. After the best templates have been found, it is possible to determine more precisely the characteristics of the system that emitted the wave.

Until now, LIGO and Virgo have been observing gravitational waves at a rate of approximately one per week (Castelvecchi, 2020) during their active observational periods and this will only increase with time, as many optimisations and changes are made on the systems to improve the range and resolution of the observations. The success of these experiments have motivated the planning of future gravitational wave detectors around the world, especially in Europe, where talks about building the Einstein Telescope (Punturo, 2010), a new type of interferometer, have picked up pace (Cho, 2021). This detector is predicted to have an increase of an order of magnitude in amplitude sensitivity and will be able to detect frequencies as low as 1 Hz (Hild et al.,

2008). These improvements will allow observations from the whole visible universe, all the way to its dark age. These improvements, of course, also add to the amount of data to be analysed (Barkett, 2019). So much, in fact, that it has been demonstrated that without a revolution in data-analysis techniques, the amount of data received from the detectors would largely surpass any type of modern super-computer capabilities (Bird, 2019).

In parallel to all these developments, the field of quantum computing (QC) has also exploded since its birth in the 1980's (Benioff, 1980). The current state of the art quantum computer has 127 quantum-bits (also called Qbits)(Gibney, 2019) and IBM has announced the creation of a 1000 Qbit computer by 2023 (Cho, 2020). The benefits of QC are multiple, but the main reason for the excitement surrounding it is that it promises incredible speed-ups in computing time. A quantum computer, contrarily to a classical computer, can superpose the individual states of its Qbits. This means that there are many more possible outcomes from the computation itself, and operations can be performed on every Qbit at the same time. This is called quantum parallelism and is what increases the computational speed of certain algorithms exponentially compared to classical versions. Of course it does come with some caveats: as soon as we want to read the outcome, the superposition state collapses into one and you lose a lot of information. This means that the computational acceleration can only be achieved by crafting very specific algorithms that manipulate the Qbits into a state that will give us the outcome we want. Some algorithms have already been developed, like the Quantum Fourier Transform, which is exponentially faster ($O(n^2)$) than the classical Fast Fourier Transform ($O(n2^n)$) or Shor's algorithm, which promises to solve prime number factorization (Nielsen et al., 2000).

Once quantum computers become more powerful, full scale algorithms like these will certainly be used on problems that current super-computers would have trouble solving in an acceptable time. As such, a lot of research is currently focused on the usability of quantum computing in different fields like machine learning, computational biology or generative chemistry among others. As described above, gravitational waves data-analysis is in dire need of improvements on the computational side and quantum computing could very well be a solution.

This thesis project took place in the context of Maastricht University (UM) collaborating with other Dutch universities on the study of using quantum computing as a tool to increase performance in scientific and more specifically GW research (see Acknowledgements section). The goal of this specific enterprise was to identify possible ways to improve the efficiency of data processing in the current GW data pipeline. these ways could be parameter estimation techniques, matched filtering methods, machine learning training or any other innovative ideas on incorporating quantum computing. In light of the short time allocated for a bachelor thesis at Maastricht University, this project mostly focused on matched filtering, it being a fairly simple data analysis process and as such a good first target for attempting to adapt gravitational wave research to quantum devices. When starting the research, it was decided, after some initial study, that Grover's algorithm would be tried as an implementation method. It was a little later discovered that Gao et al. (2021) recently published on the same topic and, as a result, this paper covers some of the same areas of focus, but will diverge distinctly in significant ways. An emphasis will be put onto practical implementations of quantum circuits and the ramifications certain practical choices will have onto the general circuit and time complexities. The following text has the goal of being both a report on the work accomplished from September to December 2021 as well as an information resource for the next individuals who will work on this UM quantum computing project.

This paper will start with a theoretical section covering the details of matched filtering and the necessary information about Grover's algorithm as well as the components of the final quantum circuit. It will be followed by an explanation of the methods needed to make these implementations as well as the technical information on quantum circuit simulations. We will then cover the results and conduct an in depth discussion on them, including difficulties and setbacks. This last section will also look at the possible next steps of the larger research this project is part of.

2 Theory

Each subsection of this theoretical part is a block necessary to the construction of a quantum circuit that can carry out a simplified gravitational wave search process. We will first go through a summary of how matched filtering works, as it is the process that we are trying to speed up. We will then examine the inner functions of Grover's algorithm and quantum counting to better understand the basic structure of the final quantum circuit created during this project. This circuit will be described in the last part of the section. An understanding of the basics of quantum computing is necessary for this part.

2.1 Matched Filtering

As mentioned in the introduction, matched filtering is a step in gravitational wave data analysis that tries to single out the approximate parameters of the system that created a specific gravitational wave. The output of a GW detector is a time-series output $s(t)$ hopefully composed of a GW signal, $h(t)$ as well as some noise $n(t)$ due to the measurement environment and tools used. This results in the signal equation being $s(t) = h(t)+n(t)$. Due to the nature of gravitational waves and the type of detector used, we have $h(t) \ll n(t)$, which is an issue, as it means that the signal will be difficult to recover from the noise levels. We can, however, take advantage of the fact that we know the type of noise (additive stochastic noise), as well as the shape of the supposed gravitational waves, to discover the original signal.

Conceptually, the idea is to convolve a certain filter K over the output of the detector to find the signal-to-noise ratio (SNR), which is defined as

$$\frac{S}{N} = \frac{\int_{-\infty}^{\infty} \langle s(t) \rangle K(t) df}{[\langle \hat{s}^2(t) \rangle - \langle \hat{s}(t) \rangle^2]^{1/2}_{h(t)=0}} \tag{1}$$

with S being the expected value of the inner product of the filter and the observed signal (also written as $\hat{s}$) and N the root-mean-square of $\hat{s}$ when the GW signal is absent. The calculations are generally made in the frequency domain, and assuming $\langle n(t) \rangle = 0$, the equation to calculate the SNR becomes

$$\frac{S}{N} = \frac{\int_{-\infty}^{\infty} \tilde{h}(f) \tilde{K}^*(f) df}{\sqrt{\int_{-\infty}^{\infty} (1/2) S_n(f) |\tilde{K}^*(f)|^2}} \tag{2}$$

which is the equation that we want to maximize to have a clear proof that the hoped signal is present. It is possible to demonstrate that to optimize this process, the filter in frequency domain, is of the shape

$$\tilde{K}(f) = c \cdot \frac{\tilde{h}(f)}{S_n(f)} \tag{3}$$

with c being an inconsequential constant and $S_n(f)$ being the single-sided noise power spectral density (Maggiore, 2000). This is the definition of the matched filter, also known as the Wiener filter. Combining equations (3) and (2) we can get our final formula for the optimal value of the SNR,

$$\left(\frac{S}{N}\right)^2 = 4 \int_0^{\infty} \frac{|\tilde{h}(f)|^2}{S_n(f)} df = \rho(f)^2 \tag{4}$$

which applies to continuous functions. To use this on a discrete data set of M time steps spaced by Δt, similar to what the detectors output, we simply have, back to the time-domain,

$$\rho(t_j)^2 = \frac{4}{M\Delta t} \sum_{k=1}^{(M-1)/2} \frac{|\tilde{h}(f_k)|^2}{S_n(f_k)} e^{2\pi i j k/M} \tag{5}$$

For the whole derivation of these results and an in depth description of the steps, please refer to Maggiore (2007), or other literary resources.

As shown by these equations, two items are required to do gravitational wave data-analysis: the output signal from the detector and a bank of simulated signals, which hopefully will match the GW signal contained in the detector output to a certain degree. These simulated signals are called templates, and, in practice, a template bank contains thousands of items that are created to cover a specific portion of the total possible parameter space. Each one of these has then to be compared to the detector signal to calculate the ρ at each time-step. If any of those is greater than some predefined ρ_{thresh} (chosen depending on the expected noise and signal strength produced by the detector), the template is considered to be a matched template and will require more precise parameter estimation.

2.2 Grover's Algorithm

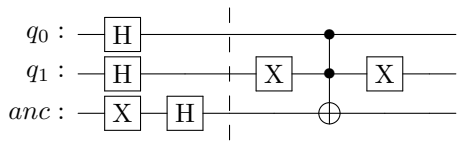
Searching an unstructured database of N items classically requires $O(N)$ operations to find a solution, as it has no order and you have to, at worse, look at each element to find what you are looking for. Grover's algorithm is a general purpose quantum search algorithm that solves the same problem in only $O(\sqrt{N})$ operations, which is a substantial improvement.

To function, this algorithm relies on two important components: the oracle and a diffusion operator. To begin the process, all solutions have to be put into equal superposition. To simplify the explanation in this section, we assume that we are simply doing a search on the computational bases of the quantum computer. Assuming our quantum computer has n qbits, we will start our algorithm with the state $|0\rangle^{\otimes n}$ and apply Hadamard gates to each one of them. This results in the state

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{6}$$

with $N = 2^n$ representing the number of computational bases, as well as items in the database. The next step is to apply the oracle.

The oracle, U_ω, is an operation that, depending on the specifics of the search, can recognize whether a specific basis is a solution or not and modify its global phase to -1. Constructing an oracle is generally an arduous process as it requires the logical encoding of the problem at hand. For example, in the search of a specific number, the oracle has to be able to recognize if a specific basis represents that number and then act on it. For this reason, this section will stay vague about the structure of an oracle, but we will revisit this topic in section 2.4. The switching of the global phase, however, is not too complicated. This step is commonly done using an extra qbit (called ancillary qbit) initiated in the state $|-\rangle$, NOT controlled by all superposed qbits. The following circuit is initiated as decribed before, but the control and phase kickback from the ancillary is only done when $q_0 = |1\rangle$ and $q_1 = |0\rangle$.



Let's look at the states. We start with the initialization and factor everything out,

$$
\begin{aligned}
|\psi\rangle &= [H^{\otimes 2}|00\rangle][XH|0\rangle] \\
&= |++-\rangle \\
&= \left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \\
&= \frac{1}{2\sqrt{2}}[|000\rangle-|001\rangle+|010\rangle-|011\rangle+|100\rangle-|101\rangle+|110\rangle-|111\rangle]
\end{aligned}
$$

if we now apply the multi-control NOT, we see that it only works when the first qbit is 1 and the second is 0 (the double X gate has this effect: an incoming 1 gets modified into a 0 and is modified back on the other side)

$$
\begin{aligned}
|\psi\rangle &= \frac{1}{2\sqrt{2}}[|000\rangle-|001\rangle+|010\rangle-|011\rangle+|101\rangle-|100\rangle+|110\rangle-|111\rangle] \\
&= \frac{1}{2\sqrt{2}}[|00\rangle(|0\rangle-|1\rangle)+|01\rangle(|0\rangle-|1\rangle)+|10\rangle(|1\rangle-|0\rangle)+|11\rangle(|0\rangle-|1\rangle)] \\
&= \frac{1}{2\sqrt{2}}[|00\rangle(|0\rangle-|1\rangle)+|01\rangle(|0\rangle-|1\rangle)-|10\rangle(|1\rangle-|0\rangle)+|11\rangle(|0\rangle-|1\rangle)] \\
&= [|00\rangle|-\rangle+|01\rangle|-\rangle-|10\rangle|-\rangle+|11\rangle|-\rangle] \\
&= [|00\rangle+|01\rangle-|10\rangle+|11\rangle]|-\rangle
\end{aligned}
$$

we see through this calculation that the negative sign from the ancillary only affected the global phase of the state $|01\rangle$ and left the ancillary qbit untouched. This will have no effect if a measurement was made after the phase kickback, as all the computational bases still have the same amplitude (no change in probability), but it does mark the state of interest. It is also important to note that depending on how the controls are done, it is also possible to mark multiple states. And it brings us to the next phase of the algorithm, the diffusion operator.

The state of the computer is now in equal superposition of all computational bases but has modified the global phase of the solutions. The goal of the diffusion operator is to modify the amplitudes such that the marked states are more likely to happen than the non-marked states. This can be done by an *inversion about the mean* operation, U_s, which is defined by the equation

$$
\begin{aligned}
U_s &= H^{\otimes n}(2|0\rangle^{\otimes n}\langle 0|^{\otimes n}-I)H^{\otimes n} \\
&= 2|s\rangle\langle s|-I
\end{aligned}
\tag{7}
$$

$|s\rangle$ representing the state of equal superposition over all working qbits (meaning the qbits used as the database, not the ancillary and other extra qbits that might be needed for the oracle). This operation gives a phase shift of -1 to all computational bases except $|0\rangle^{\otimes n}$, meaning that it operates an inversion around the mean, adding amplitude to the negative phased bases and removing from the ones with positive phase. The combined operations of the oracle and the diffusion operator is called a Grover iteration and is the basis of the algorithm.

These definitions are quite abstract and it is generally easier to understand the action of a Grover iteration through geometric interpretation. Instead of working on the full vector space spanned by the computational basis, let's adopt a new basis spanned by the linear combination of solutions to the search, $|w\rangle$ and the linear combination of non-solutions, $|w_\perp\rangle$. As such, this basis is orthonormal. If we assume r solutions to the search, we can recreate the initial state as

$$
|s\rangle = \sqrt{\frac{r}{N}}|w\rangle + \sqrt{\frac{N-r}{N}}|w_\perp\rangle \tag{8}
$$

The oracle then simply gives a negative phase to all the solutions, meaning

$$U_\omega |s\rangle = -\sqrt{\frac{r}{N}} |w\rangle + \sqrt{\frac{N-r}{N}} |w_\perp\rangle \tag{9}$$

followed by the diffusion operator, which is, in essence, a reflection about the initial state $|s\rangle$. By doing a little bit of geometry, and defining $\frac{\theta}{2}$ as being the angle between $|s\rangle$ and $|w_\perp\rangle$, we find that operating the full Grover iteration results in a rotation

$$G |s\rangle = U_s U_\omega |s\rangle = \cos\frac{3\theta}{2} |w\rangle + \sin\frac{3\theta}{2} |w_\perp\rangle \tag{10}$$

which is graphically represented in Figure 1. The closer the final vector is to $|w\rangle$, the higher the probability of measuring the correct solutions at the end of the algorithm

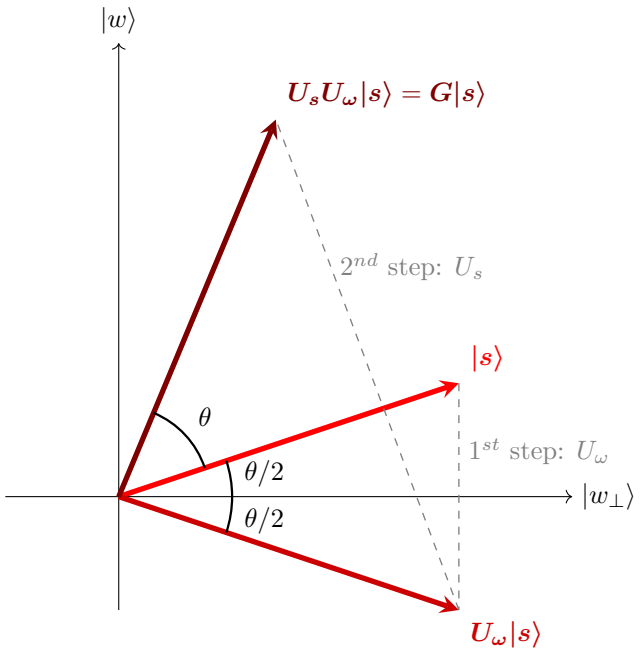


Figure 1: Graphical representation of a full Grover iteration

We can also see that this rotation does not have to end at one cycle. It is visibly possible to apply multiple Grover iterations to improve the chances of getting the right solutions as an output (Nielsen, 2000). In fact, k operations of G result in

$$G^k |s\rangle = \cos\left(\frac{2k+1}{2}\theta\right) |w\rangle + \sin\left(\frac{2k+1}{2}\theta\right) |w_\perp\rangle \tag{11}$$

To generalize even further, with the right choice of ϕ, such that $\sin\phi = \frac{2\sqrt{r(N-r)}}{N}$ (Nielsen, 2000), we can write the Grover iteration in this basis as rotation matrix of the form

$$G = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \tag{12}$$

which will be useful in the future.

This geometric visualization also allows us to find the optimal number of these operations. From Figure 1 and Equation 10,

$$\begin{aligned}\tan\theta &= \frac{\sqrt{r/N}}{\sqrt{(N-r)/N}} \\ &= \frac{\sqrt{r}}{\sqrt{N-r}}\end{aligned}$$

which gives us,

$$\theta \approx \sqrt{\frac{r}{N}} \tag{13}$$

if N is big enough. Seeing as $|w\rangle$ and $|w_\perp\rangle$ are orthonormal ($\pi/2$ radiants between them), we find the optimal number of Grover iterations, R, is

$$\begin{aligned}R &= \frac{\pi}{2}\frac{1}{2\theta} \\ &\approx \frac{\pi}{4}\sqrt{\frac{N}{r}}\end{aligned} \tag{14}$$

From this result, we see that if we want to optimally use Grover's algorithm, we need to know the size of the database, N, as well as the number of solutions in that database, r. This last variable is generally a problem, as we don't usually know how many elements match our search. This is also where we can see that the Grover algorithm requires only $O(\sqrt{N/r})$ Grover operations, compared to the classical counterpart with $O(N/r)$ operations needed (Nielsen, 2000).

2.2.1 Diffusion Operator Circuit

As said above, the oracle is tricky to create and does not have a general method of creation; it has to be done case by case. From its definition however, the diffusion operator seems to have a fixed structure that can be implemented into a circuit. Let's start from its definition and build forward (to simplify notation $^{\otimes n}$ is left out from a certain point),

$$\begin{aligned}U_s &= 2\left|s\right\rangle\left\langle s\right| - I \\ &= 2H^{\otimes n}\left|0\right\rangle^{\otimes n}\left\langle 0\right|^{\otimes n}H^{\otimes n\dagger} - HIH^R \\ &= H(2\left|0\right\rangle\left\langle 0\right| - I)H \\ &= H[2\begin{pmatrix}1\\0\\\cdots\\0\end{pmatrix}\begin{pmatrix}1 & 0 & \cdots & 0\end{pmatrix} - I]H \\ &= H[2\begin{pmatrix}1 & 0 & \cdots & 0\\0 & 0 & \cdots & 0\\\vdots & \vdots & \ddots & \\0 & 0 & & 0\end{pmatrix} - I]H \\ &= H\begin{pmatrix}1 & 0 & \cdots & 0\\0 & -1 & \cdots & 0\\\vdots & \vdots & \ddots & \\0 & 0 & & -1\end{pmatrix}H\end{aligned}$$

We can see in this equation that if we take out -1 we almost get to a CZ gate, except that the minus sign should be on the $(n,n)^{th}$ index and not the $(1,1)^{th}$,

$$U_s = -H \begin{pmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & & 1 \end{pmatrix} H$$

To resolve this we can simply add an X gate on all qbits, which gives

$$\begin{aligned} U_s &= -HX \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 \\ \vdots & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} XH \\ &= [HXC^Z XH]^{\otimes n} \end{aligned} \tag{15}$$

We see that the diffusion operator can be fully built by simply using H, X and C^Z gates, as shown in figure 2,

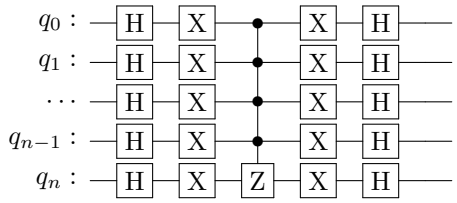


Figure 2: Circuit drawing of the diffusion operator

We can then condense the oracle and diffusion operator into a single G operator, that can be used multiple times before doing a measurement. Figure 3 is an example with 5 working qbits and 1 ancillary qbit.

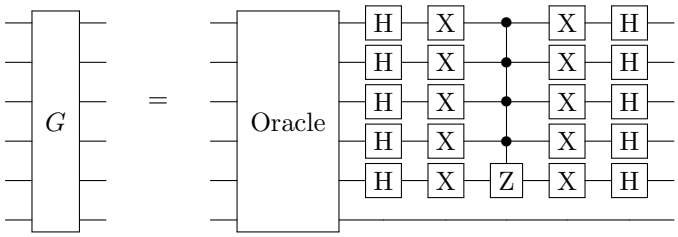


Figure 3: Example circuit for a full Grover iteration

2.3 Quantum Counting

In the previous section, we noted that estimating the number of Grover iterations needed for an optimal search was directly linked to the number of solutions in the search. This is a problem because in most cases of a search, this is exactly the kind of question that is trying to be answered.

Luckily there is a way around this issue: using a process called quantum phase estimation (QPE) and quantum counting.

2.3.1 Quantum Phase Estimation

Let's suppose a generic unitary operator U has an eigenvector $|\psi\rangle$ with an eigenvalue $e^{2\pi i\theta}$, giving us

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle \tag{16}$$

The goal of QPE is to estimate the value of θ inside the eigenvalue. It does this by having two registers of qbits, a counting register with t qbits and a working register on which U operates. The counting register is initialized with Hadamard gates, while the second register starts in the state $|\psi\rangle$. We then apply controlled-U operations in successive powers of two, as can be seen in the first part of Figure 4. The state of the register after these operations is easily calculated with the help of Eq. 16, as the eigenvalue only applies to the state $|1\rangle$,

$$\frac{1}{2^{t/2}}\Big(|0\rangle + e^{2\pi i 2^{t-1}\theta}|1\rangle\Big)\Big(|0\rangle + e^{2\pi i 2^{t-2}\theta}|1\rangle\Big)\cdots\Big(|0\rangle + e^{2\pi i 2^{0}\theta}|1\rangle\Big) = \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi i k\theta}|k\rangle \tag{17}$$

which looks very similar to the Quantum Fourier Transform (see Appendix A)

$$QFT : |j\rangle \longrightarrow \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n}|k\rangle \tag{18}$$

Looking carefully, the only missing part is a 2^t in the denominator of the exponent. As such, to get an estimate of the phase on the eigenvalue, we have to do an inverse QFT on the counting register and then read the state of the first register, as is shown on Figure 4. This process will give us a good estimate of θ, which can be improved by having more qbits in the counting register.

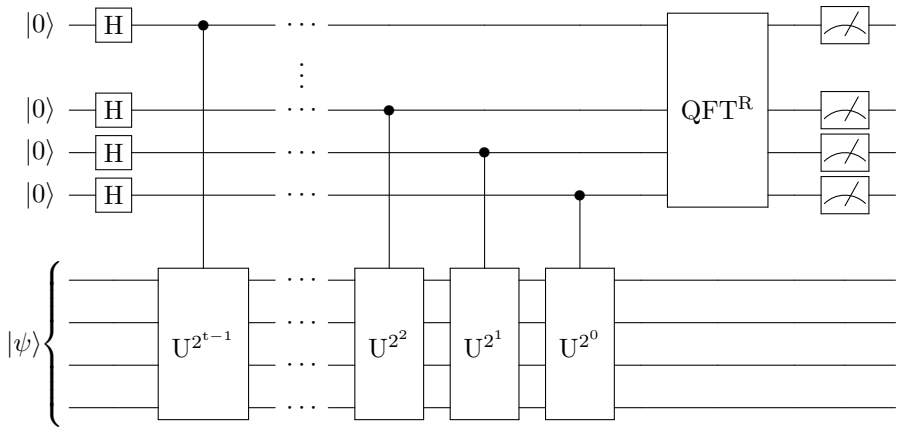


Figure 4: Circuit for Quantum Phase Estimation

As seen above, the output of the measurement on the computational bases will actually give you a measurement of $|2^t\theta\rangle$ on the counting qbits. So, to read your final answer, you simply need to convert the binary reading to decimal and then divide by 2^t. Doing so will give you θ, which you can then use with Eq. 13 and 14 to have the number of solutions as well as the iterations needed.

2.3.2 Quantum Counting

To see if QPE can be used on Grover's algorithm, we need to demonstrate that the assumptions that were made for the QPE unitary operator also work on the Grover operator. Let the Grover iteration be defined in the matrix shape found in Eq. 12. Then we can demonstrate that

$$|s_+\rangle = \begin{pmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \text{and} \, |s_-\rangle = \begin{pmatrix} \frac{-i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{19}$$

are eigenvectors of G with eigenvalues $e^{\pm i\theta}$ with the following equality,

$$\begin{aligned} G\,|s_+\rangle &\stackrel{?}{=} e^{i\theta}\,|s_+\rangle \\ \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} &\stackrel{?}{=} (\cos\theta + i\sin\theta) \begin{pmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \\ \frac{1}{\sqrt{2}} \begin{pmatrix} i\cos\theta - \sin\theta \\ i\sin\theta + \cos\theta \end{pmatrix} &= \frac{1}{\sqrt{2}} \begin{pmatrix} i\cos\theta - \sin\theta \\ \cos\theta + i\sin\theta \end{pmatrix} \end{aligned}$$

We also need to show that a linear combination of those eigenvectors can construct the basis vectors of $|s\rangle$. Assuming

$$\frac{1}{\sqrt{2}}\Big(\alpha\,|s_+\rangle + \beta\,|s_-\rangle\Big)$$

with $\alpha, \beta \in \mathbb{C}$, we can construct the two basis vectors in the following manner,

$$\begin{aligned} \frac{1}{\sqrt{2}}\Big(\,|s_+\rangle + |s_-\rangle\Big) &= \frac{1}{\sqrt{2}} \left(\begin{pmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} + \begin{pmatrix} \frac{-i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \right) \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ \frac{2}{\sqrt{2}} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= |0\rangle \end{aligned}$$

and

$$\begin{aligned} \frac{1}{\sqrt{2}}\Big(-i\,|s_+\rangle + i\,|s_-\rangle\Big) &= \frac{1}{\sqrt{2}} \left(-i \begin{pmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} + i \begin{pmatrix} \frac{-i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \right) \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{2}{\sqrt{2}} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= |1\rangle \end{aligned}$$

showing that any computational basis that is a combination of $|0\rangle$ and $|1\rangle$ can be built from $|s_+\rangle$ and $|s_-\rangle$. Knowing these facts, we can simply apply QPE to find the number of iterations needed for Grover's algorithm by simply replacing the unitary operator U by the Grover operator G. Applying QPE in Grover's algorithm is called *quantum counting*.

2.4 Implementing Matched Filtering to Grover's Algorithm

We have now covered all the essentials needed to create a circuit that implements matched filtering. Unfortunately, there was no time in the three-month project to calculate the actual SNR values from a time series using a quantum circuit, as this would have required more knowledge than was accessible. As such, the following circuit assumes the SNRs to be calculated and stored in an array of the shape $[\rho_1, \rho_2, \cdots, \rho_n]$ which we call our database. The circuit that we will construct in the following sections will take in this database, encode the elements into an index and value quantum register, then compare these values to a certain threshold value, ρ_{thresh}, and finally increase the probability of measuring the index of the values that are higher than this threshold. We are, in essence, making an algorithm, that given an array of data, will give us the location of all the values that are higher than a certain threshold. As such, the only part missing for a fully quantum matched filtering algorithm is the creation of a circuit that would be able to convert the time-series data into SNRs, which might be a future step in the continuation of this project.

2.4.1 Database Encoding

The same way as a classical array needs both a value and an index into which to place that value, encoding an array onto a quantum computer requires two quantum registers, the index register $|i\rangle$ and the value register $|v\rangle$. The idea behind this circuit is to entangle the two registers together in a way that keeps both the index and the value together. This is done by having multi-controlled-X gates (MCX), with the controls on the index register and the target in the value register. Each qbit in both registers represents a specific power of two and the combination of them creates the index or value in binary form. It is important that $|i\rangle$ is initiated in a state of superposition, as this allows each possible computational basis of this register to be linked to a certain value. The controls will be surrounded by X gates, or not, depending on the index that needs to be encoded. The number of qbits needed in each registry depends on the database as well. If we have n,m qbits in $|i\rangle$,$|v\rangle$ respectively, we can encode 2^n values with a maximum value limit of $2^m - 1$.

Let's consider an example for clarity. Figure 5 is an implementation of the description above for an array [3,4,9]. This means we have a value 3 at index 0, value 4 at index 1 and value 9 at index 2. The circuit has 3 qbits in $|i\rangle$, meaning that the array could, in theory, store 8 values (there are only 3 present) and 4 qbits in $|v\rangle$, meaning that the maximum value that can be stored is 15 (the maximum at the moment is 9).

Looking at implementing index 1, value 4, which can be found in the middle of Figure 5 in between the two limiting barriers, we first convert these two number to binary, keeping in mind the number of qbits of both registry,

$$\text{index(3qbits)}: \ 1 = (001)_2 \quad \text{value(4qbits)}: \ 4 = (0100)_2$$

From above, we understand that we want to apply the MCX gate only when the state is 001 on $|i\rangle$ to entangle the value to that index. This means that we want the controls on the second and third index qbits to be surrounded by X gates. Mathematically, we have

$$\begin{aligned} |i\rangle|v\rangle &= \underbrace{\frac{1}{\sqrt{2^3}}\Big[\,|000\rangle + |001\rangle + |010\rangle + \cdots + |111\rangle\,\Big]}_{|i\rangle} \otimes \underbrace{|0000\rangle}_{|v\rangle} \\ &= \frac{1}{\sqrt{2^3}}\Big[\,|000\rangle|0000\rangle + |001\rangle|0000\rangle + |010\rangle|0000\rangle + \cdots + |111\rangle|0000\rangle\,\Big] \end{aligned}$$

and if we apply the CNOT entangles the two registers,

$$\text{CNOT}\,|i\rangle|v\rangle \Rightarrow \frac{1}{\sqrt{2^3}}\Big[\,|0000000\rangle + |0010100\rangle + |0100000\rangle + \cdots + |1110000\rangle\,\Big]$$

the final state carries the information of which index has which value. It is also clear that the circuit encodes zeros to every index that has not been specifically made to carry a value. In the eventuality of the value to be encoded needs more than a single bit in the $|v\rangle$ registry, we simply need to repeat the MCX onto another qbit. This could also be encoded as a multi-control multi-target gate (MCMT), but for the sake of clarity, the more simple MCX was used.

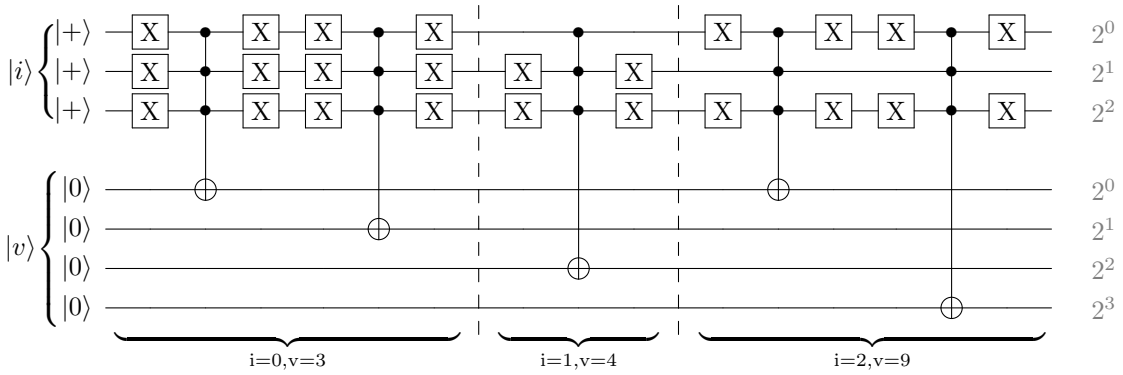


Figure 5: Example of encoding integers on a quantum circuit (it can be made simpler by using multi-control multi-target gates, but it is conceptually simpler as shown). Three values are being encoded, each one separated by a dotted vertical line. The index is encoded on the first register, $|i\rangle$, and the value itself is encoded onto the second register, $|v\rangle$. The very leftmost section is encoding the value 3 onto the index 0 by a series of controlled-not. In the case of the first value, the 3 will only be encoded onto the state consisting of $|000\rangle$ on $|i\rangle$, as seen by the X gates before and after the control. The gray numbers on the side represent the value of encoded the bit string.

This base structure for encoding values can also be used directly as part of an oracle to identify single values, but as this project was mostly interested in finding all values above a certain threshold, which is covered in the next section, we do not touch on this topic here.

2.4.2 Bit string Comparator

As mentioned above, the final algorithm is focused on in comparing individual values to each other, not to finding a single one. This means that we need a quantum bit string comparator. A bit string comparator, as its name indicates, takes two bit strings as inputs and informs us which one is smaller, equal or greater than the other. The design of such a circuit has already been done and, as such, this section will simply summarize what can be found in more detail in the paper by Oliveira and Ramos (2007). The circuit works on the principle of binary addition, and it takes in two n qbit states, $|a\rangle = |a_n \cdots a_1 a_0\rangle$ and $|b\rangle = |b_n \cdots b_1 b_0\rangle$ behaving as binary strings (these can be seen as two quantum registers).

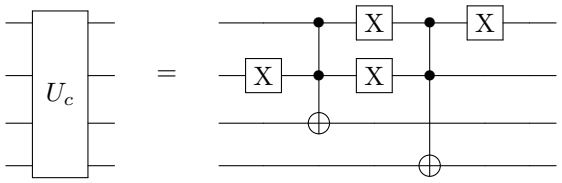


Figure 6: Sub-circuit used in the quantum bit string comparator (see Figure 7)

When two strings are used as input, the information about the two can be measured on the qbits $|o_2 o_1\rangle$ (see Figure 7). The relations are,

$$\begin{aligned} &\text{if } |o_2 o_1\rangle = |10\rangle \text{ then } a > b \\ &\text{if } |o_2 o_1\rangle = |01\rangle \text{ then } a < b \\ &\text{if } |o_2 o_1\rangle = |00\rangle \text{ then } a = b \end{aligned} \tag{20}$$

If this was all that this circuit could do it would not be very useful, as we would need to check every single value of an array one by one, just like in a classical computer. This is a quantum circuit, however, and as such, we can use it to make this comparison on multiple values at the same time by simply putting one of the inputs in superposition and running this circuit as an oracle in a Grover iteration.

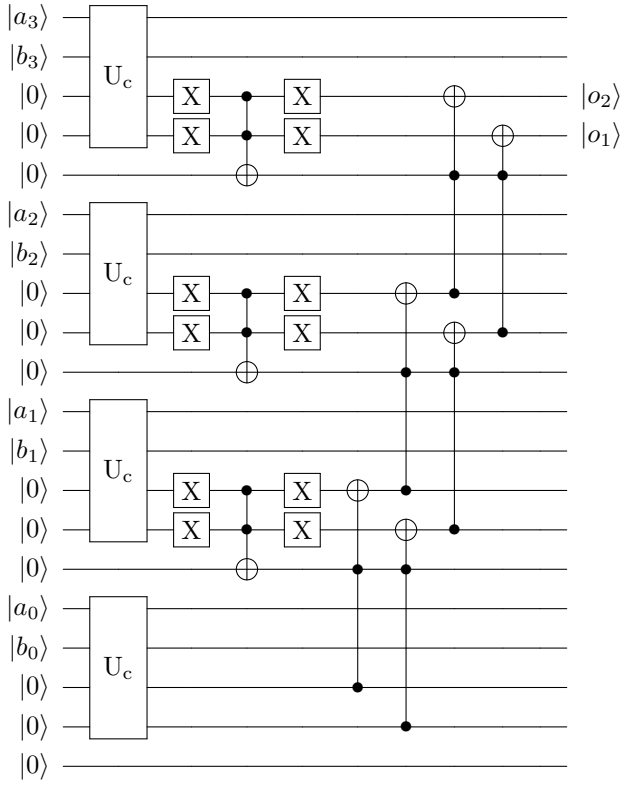


Figure 7: Example of a quantum bit string comparator circuit. In this specific case, the size of the strings to be compared is four bits. There is an extra unused qbit at the bottom which can be used as an ancillary qbit in the eventuality this circuit is used in Grover's Algorithm. The $|o_2 o_1\rangle$ qbits are used as outputs.

To illustrate this, let's give an example with a circuit that takes bit strings of size 3 as inputs (Figure 8 is a schematic of this example). In such a situation, the circuit would take a certain value a as input on the $|a_2 a_1 a_0\rangle$ qbits. If we decide that $a = 4$, then the qbits will be initialized to $|a_2 a_1 a_0\rangle = |100\rangle$. The second bit string is initialized into an equal superposition state, $|b_2 b_1 b_0\rangle = |+++\rangle$. Those states are passed through the circuit and $|o_2 o_1\rangle$ will have different values on each of the computational bases according to Eq. 20. By controlling the ancillary on $|o_1\rangle$ the circuit will do a phase kickback to all the states that are greater than a (see the relations

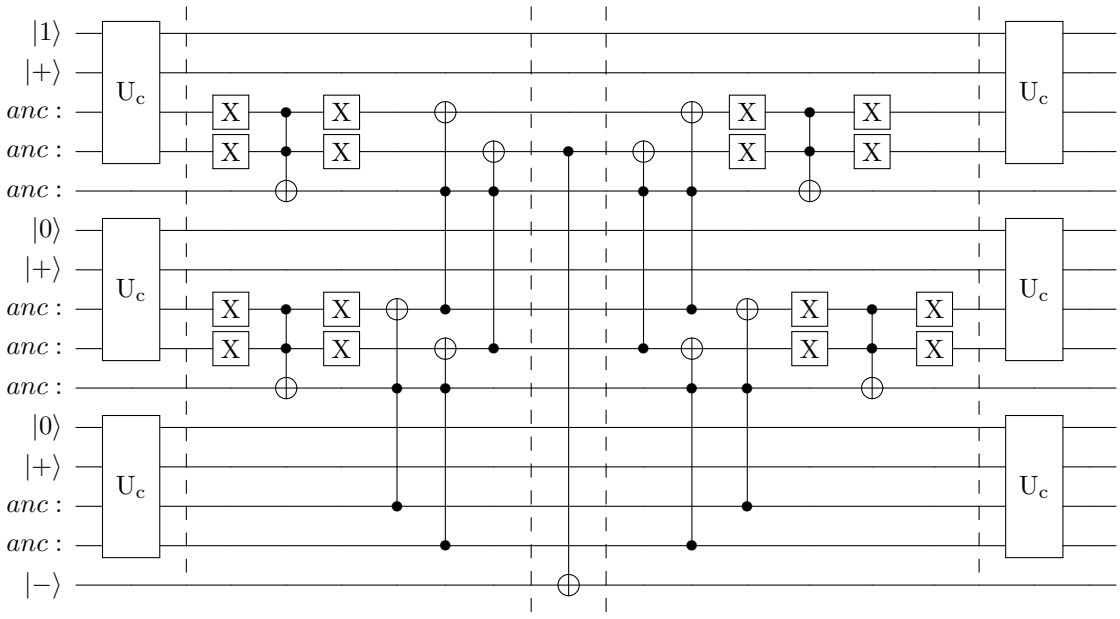


Figure 8: Example circuit of a quantum bit string comparator used as an oracle. All the *anc* qbit are initialized to $|0\rangle$, but would have visually obstructed the initialization of $|a\rangle = |100\rangle = |4\rangle$ and $|b\rangle = |+++\rangle$ and made the example less clear. We can see that the phase kickback ancillary is controlled by $|o_1\rangle$

.

in Eq. 20). At that point, the registers need to be disentangled by repeating the whole process in reverse. This ensures that the later operations necessary in Grover's algorithm impact only the qbits that were originally in superposition, without making any of the changes that would necessarily happen if the states were still entangled. This step is also called 'uncomputation'. The final step is to simply have a diffusion operator on the $|b_2b_1b_0\rangle$ qbits and then measure them. In the case of this example, we would have the highest probability of measuring the computational states $|101\rangle$, $|110\rangle$ and $|111\rangle$, as they represent the values 5, 6 and 7 in binary.

2.4.3 Quantum Matched Filtering

With these two independent circuits, it is now possible to create the final circuit that firstly encodes an array of 2^n different values (ranging from 0 to $2^m - 1$), then compares them to a specific threshold ρ_{thresh} and finally outputs the indices that store the values exceeding the threshold with a high probability. Logically then, the first part of the circuit will be the quantum array encoder with its index register, $|i\rangle = |i_n \cdots i_2 i_1 i_0\rangle$, in a superposition state $|+\rangle^{\otimes n}$ and its value register $|v\rangle = |v_m \cdots v_2 v_1 v_0\rangle$ initialized to $|0\rangle^{\otimes m}$. In parallel to that, the threshold register, $|\rho\rangle = |\rho_m \cdots \rho_2 \rho_1 \rho_0\rangle$, is set to a specific binary value with which the search will be done. $|v\rangle$ and $|\rho\rangle$ are then taken into the quantum comparator as the input strings b and a respectively (this order choice is unimportant overall, but if changed it also requires modification of the phase kickback). This circuit is interested in values that are higher than ρ_{thresh} and so the output from $|o_2o_1\rangle$ of interest is $|01\rangle$ as this means $a < b$ (see Eq. 20). Seeing as the phase kickback is easiest to do with a simple CX gate, we control the $|-\rangle$ ancillary with the $|o_2\rangle$ state, which switches all the solution states phase to -1. Now what is left to do is to disentangle the register and revert them to their original state by mirroring

the previous steps and finally operate a diffusion operator onto the index register. This whole process is shown in Figure 9 with $n = 5$ and $m = 4$ and represents one Grover iteration. It is of course possible to repeat these steps multiple times before measurement to lower the probability of erroneous answers, or possibly perform quantum counting.

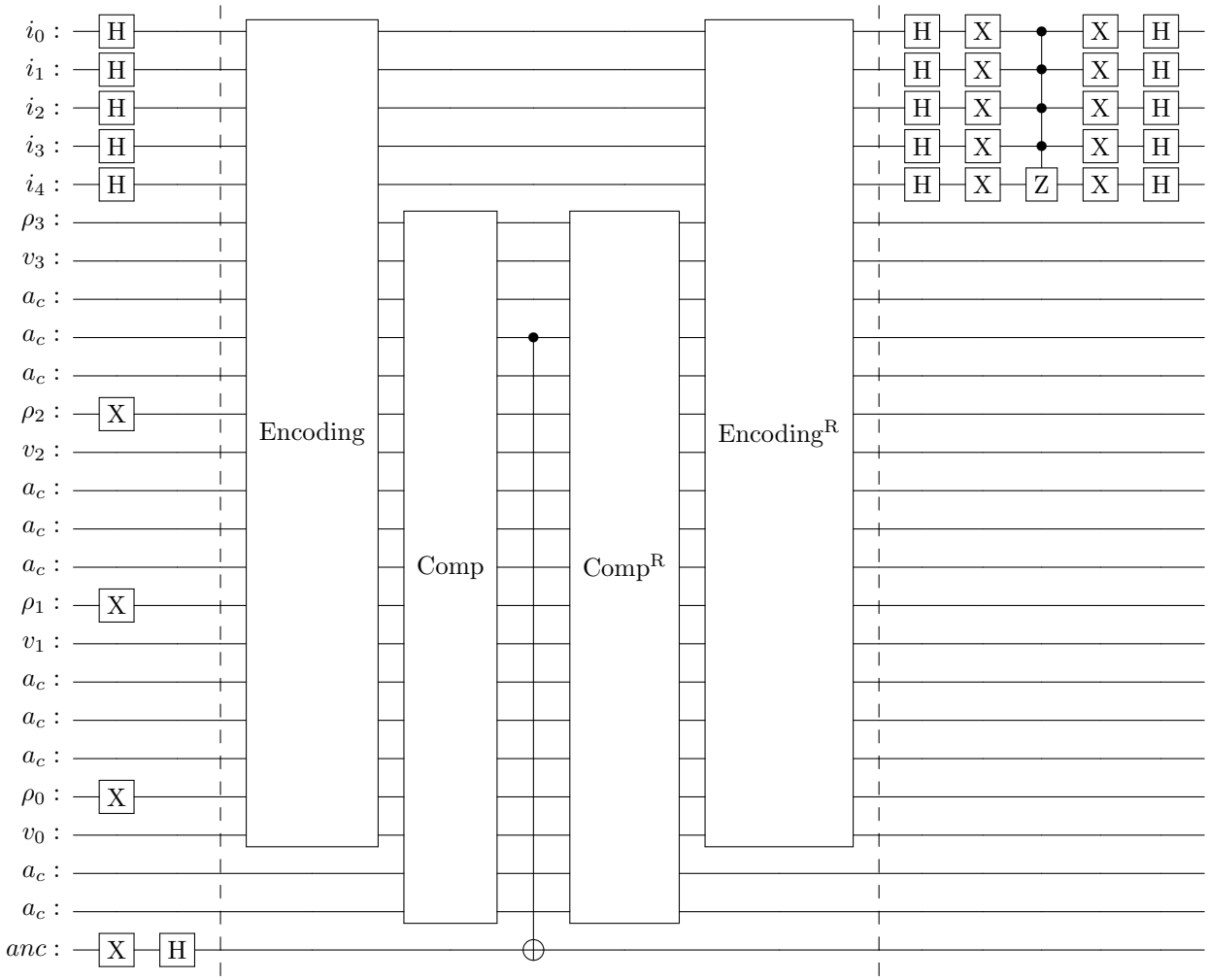


Figure 9: Example of the full SNR retrieval quantum circuit with 5 qbits in $|i\rangle$ and 4 qbits for registers $|v\rangle$ and $|\rho\rangle$ (because of how the comparator work with the extra ancillary qbits a_c this results in 20 qbits in total), allowing for an array length of 32 (2^5) and a maximum value of 16 (2^4). The threshold value in this circuit is 7 (the value is defined by $|\rho_3\rho_2\rho_1\rho_0\rangle$ in this case $(0111)_{bin} = 7$).

3 Methods

While the circuit described in the last section was built fully theoretically from the ground up using knowledge accumulated during the project, the testing of the theory had to be done using preexisting quantum simulation tools. For this purpose, we used the Qiskit Python SDK (Anis et al., 2021). It is an open-source package made for creating quantum circuits and algorithms with the idea of testing them on different quantum computer simulators (local/cloud) as well as actual quantum computer/device prototypes provided by IBM Quantum (cloud). In this project, only the local simulations were used. These have the benefit of easy testing and debugging and, for small sized circuits (between 15-25 qbits), even a standard laptop can run them. Importantly, these simulations can also model realistic noise in quantum system and emulate NISQ (Noise Intermediate-Scale Quantum) devices (Wood, 2018). This feature has not been used in this project, and all simulations were made on a "perfect" quantum device.

Qiskit handles the simulation process by transpiling the theoretical circuit that was provided. This allows the researchers to have only limited knowledge of practical issues like noise and decoherence and still be able to test their theoretical model, which was the case for this project. The transpiling process goes through multiple phases of circuit optimization and transforms the original circuit into a circuit that can theoretically be executed on different present day quantum systems (which the chosen simulation model will try to imitate). More information can be found in their transpiler documentation. In the case of all the results shown in the next section, the simulations were run on the Qiskit Aer simulator, using the QASM back-end without any specific choice of passes (technical term meaning the model with which the simulator transforms the quantum circuit to fit certain requirements, see Figure 10). As mentioned above, this simulator includes configurable noise models, that might be used in the future of this project.

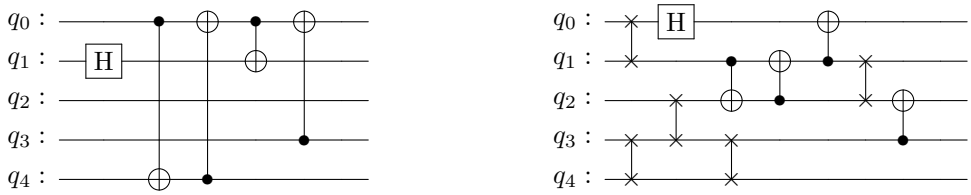


Figure 10: Example of a theoretical quantum circuit (left) and its transpiled version (right) using a pass algorithm called StochasticSwap, which can be used to format a circuit for specific coupling maps based on physical quantum devices. Both will have the same final state, but the right one will be much simpler to implement.

In the process of building the circuit shown in section 2.4.3, each sub-component was tested independently to verify that the logic was sound. Once done, the whole circuit was assembled and every part of it was made into an independent and scalable module so that the only input required for the creation of the circuit would be the array itself and the threshold value. For the moment the code uses classical means to estimate the size and the maximum value of the stored SRN array, but these are variables that will be known (or estimated) when running an actual matched filtering process. The code for the full circuit and code can be found here.

The general way the testing, and more specifically the running of quantum circuits, is made on a classical computer is to repeat the simulation a certain number of times and plot the measurements into a histogram. This allows us to plot a probability density function of the final state collapse and see which computational bases are more likely to come out. In the case of this project, most of the testing was done with 2048 runs (generally called counts), but as the size of the circuit increases the number of counts needed increases as well, as they will be more spread out. A certain basis is considered a solution if the peak is noticeably different from the other bases,

as visible in Figure 11. With this method of measuring the success of the circuit, we tested many arrays of different lengths and values. We also tried quantum counting and the use of multiple Grover operators with the circuit. As the idea is to apply quantum computing to the real world, we then ran the circuit on a spliced version of a massive array of length 2^{17} with actual SNR values (Gao et al., 2021) to investigate the usefulness of this algorithm. The splicing was also investigated theoretically to find out if there exists an optimal break down of a parameter space to minimize the complexity of both Grover's algorithm and quantum counting at the same time and many plots were made to observe the evolution of circuit complexity in function of the length of the array and maximum size of the values stored inside. Additionally for the sake of information, all of the plots shown in the next section were done using the Plotly Python libraries and can be found in their interactive form with the code linked above.

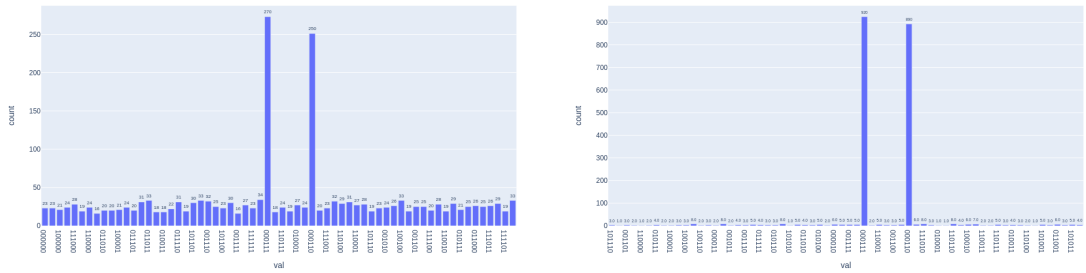


Figure 11: Example of two outputs from a Grover search quantum simulation plotted with Plotly. On the left is a single Grover iteration compared to three in the circuit on the right side. The highest probability bases are given by the visible peaks. These graphs complement the explanations in section 2, as we see that the probability of getting a correct solution is increased dramatically with more Grover iterations. Both have 2048 counts in total.

4 Results and Discussion

As the theoretical aspect of the circuit has been discussed in depth in section 2, we will not address any particular theoretical functionalities in this section even though they are, in themselves, results of this project. We will however see and discuss the circuit outputs, the testing of quantum counting on it and the results of multiple Grover iterations. We will then discuss the application of the algorithm on some real world SNR values as well as possible ways to measure the complexity linked to the general process. As a final part, we will discuss some additions and improvements that could have been made to this project, as well as the possible paths that it opens for future research.

4.1 Circuit behaviour

We will first look at different outputs of the circuit, some of which can be seen in Figure 12. The input array for these was made with random values bellow the threshold ρ_{thresh} and three solutions randomly placed. The plots shown are simply a proof that the circuit works and as we can see, it was possible to run it for medium sized arrays. 4096 values (bottom-right plot) was not the maximum length reached, but the time a run takes increases exponentially with more qbits (as expected from a classical simulation of a quantum circuit) and so we stopped at 12 qbits for this demonstration (the full circuit had 27 qbits, 12 for the array length and 15 for the value comparison as the threshold was set to be less than 8).

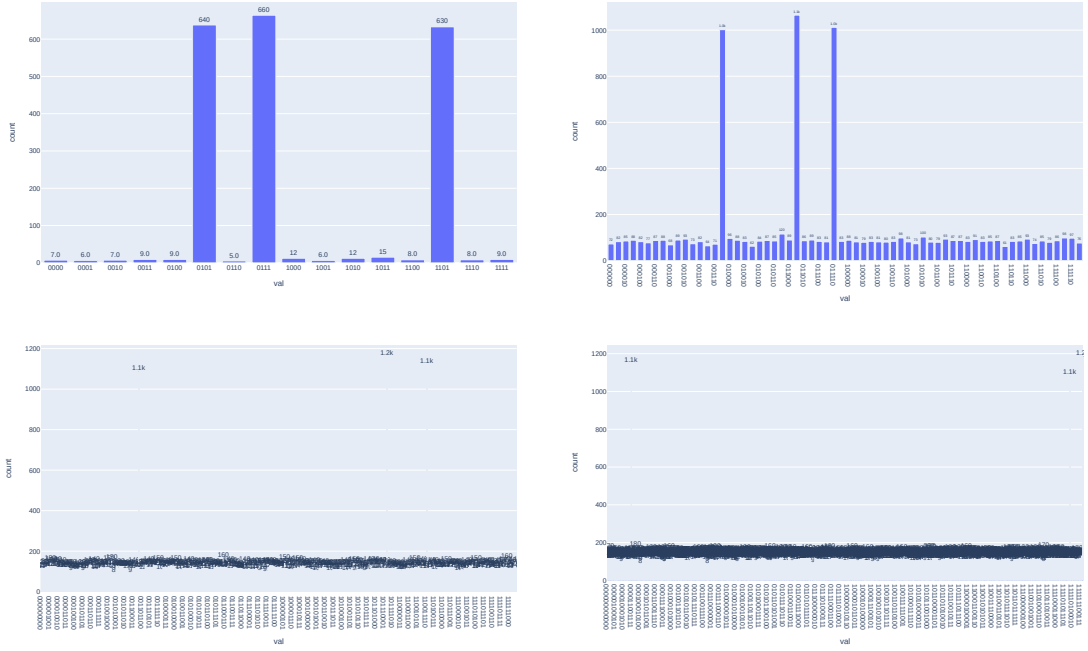


Figure 12: Different examples of results from the normal running of the circuit, with different sized arrays as input. Each of the arrays were randomly generated, but had a fixed number of solutions over the threshold. In this case all of the runs were fixed with three solutions that are visible as peaks in the histograms. Up-left: array length of 16. Up-right: length of 64. Down-left: length of 512. Down-right: length of 4096. Run times in the same order are: 292ms, 1.26s, 33.2s, 28m46s. The PDF compiler has trouble with the thinness of the peaks on the last two plots. You can find their original versions here.

The logical next step was to try quantum counting. We tried with many different inputs and structures, but we never managed to make it predict the correct number of solutions, even approximately. Figure 13 shows two tests we did on the same array with the same threshold (specific values are not important for this analysis). Every test results in the same set of highest solutions. In the plots above, the four highest peaks are for the bases $|00000000\rangle$, $|10000000\rangle$, $|01000000\rangle$ and $|11000000\rangle$. These respectively give angles of 0, π, $3\pi/2$ and $\pi/2$ which in turn result in wildly inaccurate estimations of the number of solutions (they are not given here because the values vary depending on the number of counting qbits). These values don't seem to be random as they are always the same four peaks (or a subset of the four). We have not yet found the source of these errors, but seeing as the circuit manages to detect solutions correctly (as shown above), we should expect quantum counting to work as well. We originally thought that the problem lay in the fact that we weren't able to use more counting qbits, but the current supposition is that there is an implementation issue with the Grover operator in the quantum counting algorithm, likely due to a miscomprehension on how quantum counting should be implemented on a more complicated case like this one. But overall this should not be a very long lasting problem.

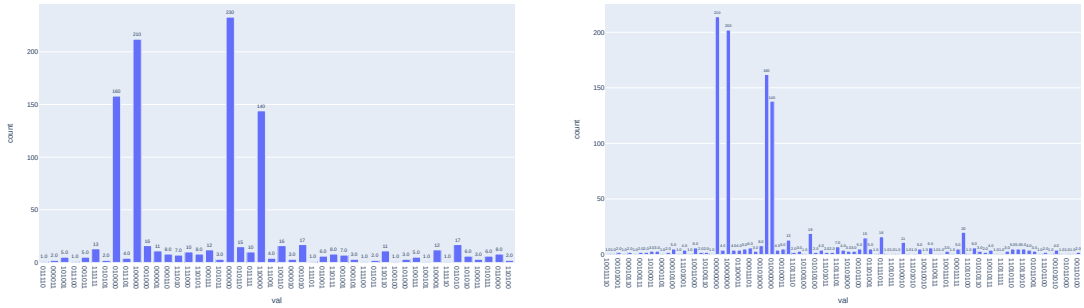


Figure 13: Two results from quantum counting with the circuit. Left: 6 counting qbits; Right: 8 counting qbits. We can see they have the same structure, with four major peaks.

Seeing as we could not calculate the optimal number of Grover iterations it was best to simply test by hand different numbers of them. Figure 14 is one example of the results we got, in this case going from one iteration to 9, with array lengths of 64 and values from 0 to 7. We did those tests on different array inputs and they gave, as expected, differing outcomes but with the same general pattern. The results were quite surprising, as we were generally accustomed to seeing graphs akin to those in Figure 11, in which the solution peaks were always for the same values. However, the probability of measuring them would simply increase until the optimal value of iterations was reached (it is also the theoretical expectation that we saw in section 2). This is not what is visible from these images. In these there is no progressive increase of the probability for solutions, there is only very abrupt changes per extra iteration, and the shape of the probability density function is totally different in each. There is a reoccurring pattern however, as is visible from Figure 16, with the correct spikes appearing every 4 additional Grover iterations. The images 1, 5 and 9 all have the same and correct solution peaks (the ninth is less noticeable). We suppose that this happens because the rotation angle θ at each Grover iteration is larger than $\pi/2$ and as such overshoots the solution basis and changes the outcome. We have yet to confirm this hypothesis and are not sure that this is actually the case. All the tests with different arrays have come to the same kind of pattern with varying periods of the cycle.

In some of the other test we also saw (unlike here) that the probability of getting the solution would still increase when getting to the next period, but this was not a common sight. It is also possible that the reason for this weird behaviour is linked to the failure of the quantum counting

and the resulting phases. We have not really been able to dig that much into these issues due to lack of time, but this is possibly something that a more experienced quantum programmer can help us solve. For now, we have decided to stick with only one Grover iteration for the rest of these tests, as this was the only number of iterations with seemingly consistent correct results.

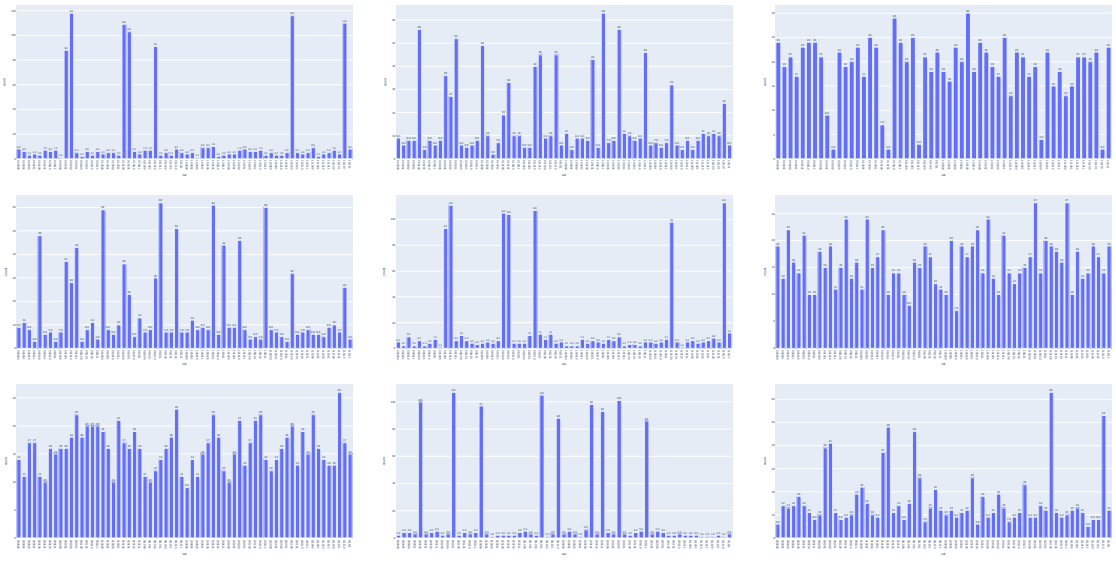


Figure 14: Example of different number of Grover iterations applied on the same array with length 16 and 7 solutions. The number of iterations grows from left to right, up to bottom. As such top-left is 1 iteration, top-right is 3, bottom-left is 7 and bottom-right is 9. The correct solutions are very visible on the first and fifth plots (ninth also has them but less clearly).

Satisfied with the outputs and the multiple tests, we realized that we could directly test the algorithm on some pre-processed SNR values. Seeing that Gao et al. (2021) had some of that type of data stored in their Github repository we decided to try and replicate their results, but with the added bonus of running it on a fully quantum simulation, unlike what they did. Of course the array being 2^{17} (meaning $n = 17$) values long with the maximum SNR of 19 (meaning $m = 5$) meant that we could not try it in a single run, as running a simulation of $n + 5 * m = 42$ qubits is not possible on the hardware we had at our disposal. Instead, we broke down that initial array into smaller and smaller bits until the computer stopped running out of memory. This happened at an array length of 16 values, meaning the original array needed to be broken into 8192 individual pieces. Each of them was run successfully and the solutions collected into a file. They matched expectations exactly, finding all of the solutions required, meaning this was a total success and that we managed to replicate the results of Gao et al. (2021) with a fully quantum circuit (but unfortunately still missing the time-series to SNR algorithm). We were unfortunately unable to produce the same plots, as the tools necessary were not quickly usable, but it will soon follow.

4.2 Complexity measurement

There is also another goal to the splicing of the array, as this by definition splices the the parameter space into smaller pieces. We mentioned in the theory that Grover's algorithm has a $O(\sqrt{N})$ complexity, while quantum counting has a polynomial complexity dependant on the counting qbit number. This complexity is measured in number of applications of the Grover

operator ($\sqrt{N}$ operations), and as such we wanted to investigate the possibility of an optimal splicing of the parameter space, such that the number of operations needed in Grover's algorithm and quantum counting is minimized for both. It would also allow earlier implementation on smaller devices. We are looking at this theoretically as well but haven't yet come up with a satisfactory solution. It is however possible to analyze this problem from another perspective. With the circuit and simulators that we have we are able to see what the implementation will look like on an actual quantum device, and more specifically, we will be able to know what type of gates and how many of them are used.

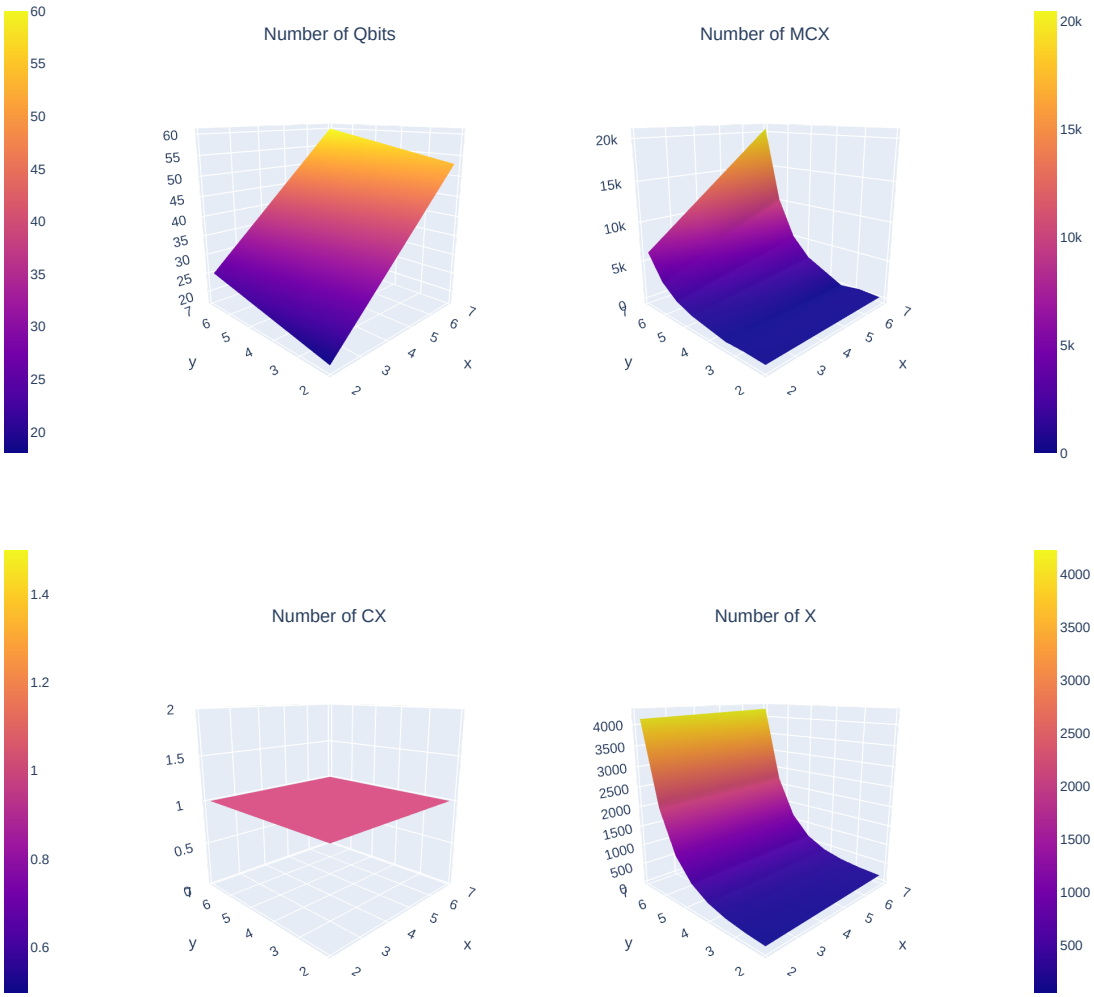


Figure 15: Plots of the evolution of different circuit parameters depending on the characteristics of the input array (length of array on the y axis and maximum value on the x axis). The x and y axes are scaled logarithmically. Top-left: total number of qbits; Top-right: number of MXC gates; Bottom-left: number of CX gates; Bottom-right: number of X gates.

To date, we have not really defined how to measure complexity. Generally computer scientists make use of time complexity, but in our case it is difficult as we don't have any way to test and predict the speed of different quantum gates. Therefore, we have decided to adopt a simpler measure for the complexity of our circuit, namely the number type of gates present (which also give a measure of a circuit's depth), as it seemed like an easily measurable parameter and it would evolve depending on the size of the input array. It would also be a parameter that would change from quantum device to another, possibly allowing us to determine the feasibility of the circuit on different platforms.

Figures 15 and 16 contain plots of that idea, showing us the evolution of the number gates and qbits depending on the length of the array as well as the maximum values inside that array. We can see some expected behaviour, as the total number of qbits (top-left plot) increases linearly with both variables. The slope is steeper on the x axis because the number of qbits needed for the comparator increases by a factor of 5 compared to the index register. This is a pretty big difference, but it is luckily contained to a certain range, as detectors generally have expected SNR value ranges depending on their sensitivity and noise values. In the case of the data we were testing, the SNR was around 19 at maximum, resulting in a bit string of 5 qbits (total of 25 qbits for the comparison) while the maximum SNR value for the ET pathfinder is expected to be around 500. This would mean an increase to a bit string of 9 bits and a final circuit of around 62 qbits (assuming the bank has 2^{17} templates) for the third generation detectors. This number seems promising, seeing that the largest quantum device at the moment has over 100 qbits (Ball, 2021), but the encoding system adds a lot of complexity to the gate structure and, as such, will probably not be feasible until a proper general quantum computer is built. The other plots however seem to indicate the sharpest increase in complexity grows with the array lengths, on the y-axis (except for the CX gate that stays constant, as it represents the only phase kickback control gate). This is most notable on the measure of X gates, which grows polynomially or exponentially with the length and has no variation on the x axis. The number of multi-control-X gates also has a polynomial/exponential increase on the y-axis, with a higher slope when the values are larger. The x-axis seems to have a linear trend, with a larger slope when the array is larger.

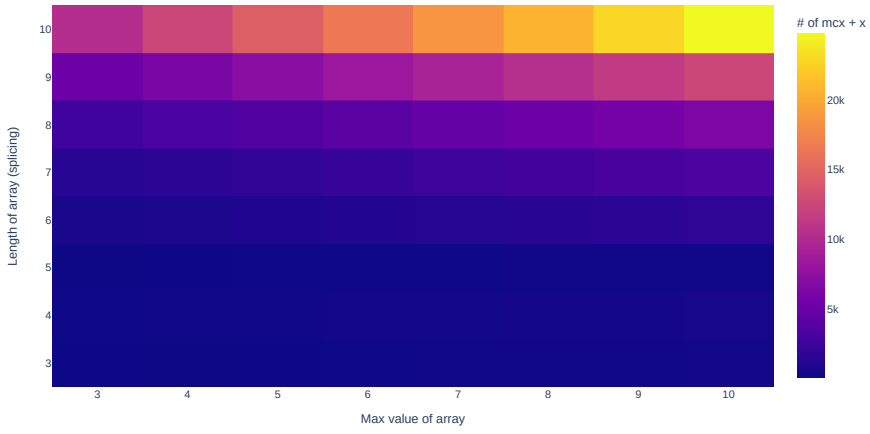


Figure 16: Plot of the summed evolution of MCX and X gates as a measure of circuit complexity

We also thought that a combined form of these plots could possibly give a good measure of

complexity, as it is easier to follow than many different plots for each variable. This is shown in Figure 16, where the numbers of MCX and X gates were combined into a single plot. Unfortunately, with these developments arriving rather late in the project, we were not able to test all of these possibilities fully and these plots should be looked at with caution, as a more in-depth analysis has yet to be done.

It is also to be noted that no such complexity measurement has yet been done on quantum counting, nor on multiple applications of the Grover iteration. This is partly due to the necessity of fixing these particular processes, but also due to time constraints. It is important to keep in mind that all the tests that were done above used the standard transpiling procedure. That means that many other models of complexity could rise simply by changing these settings and running the tests again. It would also point toward a more realistic version of the algorithm and maybe to the possible optimal slicing number that would balance the complexity between the Grover search and the quantum counting cycle.

Additionally, the different transpiling passes would be able to break down the circuits to even smaller parts to have a more in-depth view of the number of gates needed. We did not mention this in the theory section, but complicated multi-qbit gates can always be reduced to a combination of rotation operators, phase operators and CNOTs as they form a universal set of quantum gates (Williams, 2011). A prime example is the CCX, or Toffoli gate, that can be broken down into single qubit gates (phase and rotation) and a minimum of six CNOTs (Barenco et al., 1995). As such, the plots in Figure 15 could be further broken down into the smallest sub-components of the circuit with the goal of constructing a more universal visualization of complexity. Of course it should also be done with different types of transpiler passes to give an interesting perspective on the quantum device upon which the circuit would be implemented.

4.3 Impacts of the project

Overall, the results described above are a promising start for the adaptation of quantum technologies to the study of gravitational waves. The choice of applying Grover's algorithm to matched filtering was made to be an interesting introduction to the field. It was clear from the beginning that the quadratic improvement in operational cost Grover's algorithm promised would not be sufficient to revolutionize the computational cost of GW data-analysis. The results above tend to demonstrate that applying Grover's algorithm to matched filtering will require a universal quantum device that is still far from being built.

Additionally, the issues with the implementation of quantum counting and multiple Grover iterations are worrying. These results have, however, shown that sensible progress was possible even with only a few months of research, which is promising for the future. It has also given this group the necessary tools for further study and experimentation. The investigation of different transpiling methods and complexity models in the future will give insight not only into the Grover process and its possibilities, but also more generally to the application of quantum computing in classical data-analysis techniques. It has also built foundational knowledge about the field of quantum computing in the current research group. As such, this project has proven to be an invaluable start to the larger objectives of the current research programme. Of course, matched filtering and Grover's algorithm are only the beginning of the search for quantum solutions and after the full analysis of their potential, it will be time to look at more complicated problems like parameter estimation with the tool-set that was initially constructed for this project.

5 Conclusion

The principal objective of this work was to investigate the applicability of quantum algorithms in the context of gravitational wave data-analysis. This was done through the study and design of a quantum circuit able to recreate the last steps in the matched filtering procedure, as well as an examination of the general complexity of the process. The task of the circuit was specifically to extract all the SNR values above a specific threshold in a consistent and repeatable manner using Grover's algorithm. This was achieved and thoroughly tested, including with real world data.

While the circuit itself is fully functional, some strange behaviour that derives from it, specifically linked to quantum counting and the use of multiple Grover iterations, has not yet been fully understood and is under further investigation. These issues combined with time limitations have restricted the complexity analysis to a single type of circuit transpilation. Even though the analysis might not be fully accurate, the tools to make a more in-depth complexity study have been created and are ready for use. A specific point of interest for these tools, once quantum counting is solved, is the slicing of the parameter space into smaller parts. This might be a way to optimize the circuit needed for the best processing of the data and at the same time simplify the circuit which might allow for earlier implementation on quantum devices. Another point of interest would be the application of different transpilation passes that would give different circuit complexity profiles as well as more information on the possibility of said implementation.

The general objective of the project was the examination of the possible uses of quantum computing in gravitational research. In this paper we tried to investigate a single path to that objective, namely Grover's algorithm combined with matched filtering, with some success. But there are many more paths to be discovered and explored, which links to our secondary goal. That was to create a foundation on which the research to come can step on, and continue building from. In that, we believe we were very successful. This work has allowed an in-depth study of the mechanics of quantum computing and the gathering of resources and knowledge that will hopefully be useful in the close and more distant future of this larger project.

6 Critical Reflection

I want to start this critical reflection by looking back at the first times we discussed this project with Gideon. I very distinctly remember the call in which he listed what had already been done and by whom at Maastricht University in the field of linking gravitational wave (GW) research and quantum computing. This subject was one that I was deeply attracted to, as I both enjoy General Relativity and Quantum Mechanics. Very suddenly he explained that there was in fact the need for a student to start working on the "birth" of what will become a major research project at FSE in liaison with IBM and other major universities like Utrecht University. He then simply asked me if I wanted to be that student.

At that point my mind was racing with thoughts and emotions. Disbelief, elation, unease, excitement, self doubt and many other feelings were all battling to take over my mind. I just did not know if I would be ready or up to the task that he invited me to take on... But, at last, I took a deep breath and decided that this was going to be an opportunity that I would not forgive myself missing, and so I accepted the job. Of course, at that point it dawned on me that I didn't know much about gravitational waves nor quantum computing. What did I embark on? Well Gideon seemed confident, and so I relaxed, looking forward for the project to begin. And what a project!

Working with everyone involved was both interesting and inspiring. I was able to meet and work with experts in their respective fields and observe how research was done, both by discussing ideas with them and by experiencing it myself. I discovered that the process is not linear at all and that leaps forward can happen when least expected. It was also thrilling that the small group researching this at MSP (Gideon, Sophia and I), as well as the GW group in Utrech, were all quite new to the topic and so every week we were able to make great progress in our understanding of it. Of course, the learning curve was quite sharp to begin with, and reading QC articles is particularly arduous when you are not yet comfortable with the subject. It is also a rather new field, where advances are made very fast and possibly often kept secret, meaning information further than what is found in text books can be really difficult to find. This is probably an area where I could have done more, as just reading more articles that were related to specific topics of interest could have given us a wider perspective on new possibilities to research. But as mentioned, it was very difficult to find information, at least in my limited experience.

I also believe that I could have been more efficient and organized in taking physical and numerical notes of what had been discussed and thought about. This came back biting a few times when I could not properly remember some mathematical derivations that we solved in Gideon's office or forgetting about a meeting. The importance of accurately noting exchanges is definitely something that I have learnt

I very much believe that this project has, overall, been rather successful for the limited time allotted, and this also seems to be the general feeling when I recently presented the state of the project to different research groups at MSP. Many exciting doors have been opened in the last few weeks alone! This makes me happy and hopeful for the project, but I regret that this specific experience is soon over for me... Having been there at such an early stage does make me wish that I could continue working on this for a little bit longer.

But hey, who knows what the future holds?

7 Acknowledgements

I would like to thank Gideon and Sophia for their help and enjoyable company during this project. I would also like to extend my gratitude to the Eindhoven University of Technology and the University of Utrecht in particular for their help and participation during this project. I also acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

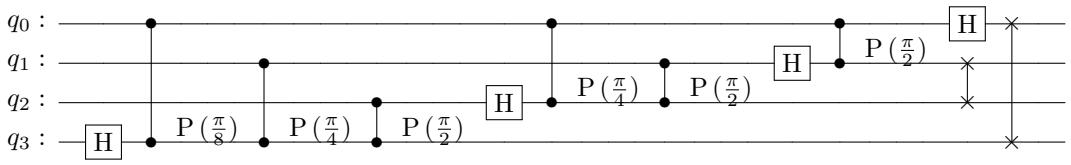
8 Appendix

8.1 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is effectively a switch from the computational basis, $|x\rangle$, to the Fourier basis, $|\tilde{x}\rangle$. Assuming n qbits and $N = 2^n$ basis states, the definition of the QFT is as follows:

$$|\tilde{x}\rangle = \text{QFT}\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N}\,|y\rangle$$

and can easily be constructed as a quantum circuit. The following is an example for 4 qbits:



Implementing this on Python using Qiskit can be done with the following recursive code:

```python
def qft(n):
    c = QuantumCircuit(n)

    def sw_reg(c, n): #produces the bit swap at the end
        for qbit in range(n//2):
            c.swap(qbit, n-qbit-1)
        return c

    def qft_rot(c, n): #recursive function for qft
        if n == 0:
            return c

        n -= 1
        c.h(n)

        for qbit in range(n):
            c.cp(np.pi/2**(n-qbit), qbit, n)

        qft_rot(c, n)

    qft_rot(c, n)
    sw_reg(c, n)

    return c
```

If you wish to learn more about this topic I would recommend either the Qiskit tutorial or the book by Nielsen (2000). These recommendations also hold for any other topic in quantum computing.

9 References

Abbott, B. et al. (LIGO Scientific Collaboration and Virgo Collaboration). (2016). Observation of Gravitational Waves from a Binary Black Hole Merger. Phys. Rev. Lett. 116 (6): 061102. DOI: 10.1103/PhysRevLett.116.061102.

Abbott, R. et al. (2021). Observation of Gravitational Waves from Two Neutron Star–Black Hole Coalescences. The Astrophysical Journal Letters, Volume 915, Number 1.

Alexander, T. et al. (2020). Qiskit Pulse: Programming Quantum Computers Through the Cloud with Pulses. Quantum Sci. Technol. 5 (2020) 044006. DOI: 10.1088/2058-9565/aba404

Anis S., Abraham H. et al. (2021). Qiskit: An Open-source Framework for Quantum Computing. DOI: 10.5281/zenodo.2573505

Ball, P. (2021). First quantum computer to pack 100 qubits enters crowded race. Nature 599, 542. DOI: 10.1038/d41586-021-03476-5

Barish, B. and Weiss, R. (1999). LIGO and the Detection of Gravitational Waves. Physics Today. 52 (10): 44. DOI: 10.1063/1.882861.

Barenco, A. et al. (1995). "Elementary gates for quantum computation". Physical Review A. American Physical Society. 52 (5): 3457–3467. doi:10.1103/PhysRevA.52.3457. PMID 9912645. S2CID 8764584

Barkett, K. (2019). Computational Methods for Gravitational Wave Physics: Spectral Cauchy-Characteristic Extraction and Tidal Splicing. Dissertation (Ph.D.), California Institute of Technology. doi:10.7907/3DH5-7773.

Benioff, P. (1980). The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines. Journal of Statistical Physics, 22, 563.

Bird, I. et al. (2019). Gravitational-Wave Data Analysis: Computing Challenges in the 3G Era, GWIC-3G Subcommittee Reports. Retrieved from: https://gwic.ligo.org/3Gsubcomm/.

Camp, J. and Cornish, N. (2004). Gravitational wave astronomy. Annual Review of Nuclear and Particle Science, Vol. 54:525-577. DOI: 10.1146/annurev.nucl.54.070103.181251.

Castelvecchi, D. (2020). What 50 gravitational-wave events reveal about the Universe. Nature. DOI: https://doi.org/10.1038/d41586-020-03047-0.

Cho, A. (2021). European plan for gigantic new gravitational wave detector passes milestone. Science. DOI: https://doi.org/10.1038/d41586-020-03047-0.

Cho, A.(2020). IBM promises 1000-qubit quantum computer—a milestone—by 2023. Science. DOI: 10.1126/science.abe8122

Einstein, A (1918). "Über Gravitationswellen". Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften Berlin. part 1: 154–167.

Friedman, A. (1922). "Über die Krümmung des Raumes". Zeitschrift für Physik. 10 (1): 377–386. DOI: 10.1007/BF01332580.

Gao, S. et al. (2021). A quantum algorithm for gravitational wave matched filtering. University of Glasgow. arXiv:2109.01535.

Gibney, E. (2019). Hello quantum world! Google publishes landmark quantum supremacy claim. Nature 574, 461-462. DOI: https://doi.org/10.1038/d41586-019-03213-z.

Hild, S., Chelkowski, S., Freise., A. (2008). Pushing towards the ET sensitivity using 'conventional' technology. General Relativity and Quantum Cosmology. arXiv:0810.0604.

Maggiore, M. (2007). Gravitational Waves: Volume 1: Theory and Experiments (Illustrated ed.). Oxford University Press.

Mermin, D. (2007). Quantum Computer Science: An Introduction. Cambridge University Press. ISBN: 9780511813870. DOI: https://doi.org/10.1017/CBO9780511813870.

Naeye, R. (2019). Stellar Mystery Solved, Einstein Safe. MIT Press Release, Sky and Telescope.

Nielsen, M. and Chuang, I. (2000). Quantum Computation and Quantum Information. Cambridge: Cambridge University Press. ISBN 0-521-63503-9.

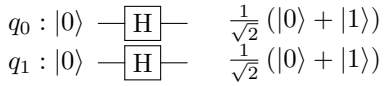
Oliveira, D. and Ramos R. (2007). Quantum bit string comparator: circuits and applications. Quantum Computers and Computing. 7.

Punturo, M. et al. (2010). The Einstein Telescope: a third-generation gravitational wave observatory. Classical and Quantum Gravity, Volume 27, Number 19.

Schwarzschild, K. (1916). "Über das Gravitationsfeld eines Massenpunktes nach der Einsteinschen Theorie". Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften. 7: 189–196.

Williams, C. (2011). "Quantum Gates", Explorations in Quantum Computing, Texts in Computer Science, London: Springer, pp. 51–122, doi:10.1007/978-1-84628-887-6_2

Wood, C. (2018). "Introducing Qiskit Aer: A high performance simulator framework for quantum circuits". Medium.


$q_0 : |0\rangle$
H
$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$q_1 : |0\rangle$
H
$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$