Technische Universität Graz

Bachelorarbeit Technische Physik

Simulation Of The Quantum Grover Algorithm With Matlab

Tobias Kugel

Mat. Nr: 01530702

August 30, 2018

Contents

1 Introduction **3**
1.1 Overview . . . 3
1.2 Qubits . . . 3
1.3 Quantum register . . . 3

2 Quantum Gates **4**
2.1 NOT-Gate . . . 4
2.2 Hadamard-Gate . . . 5
2.3 CNOT-Gate . . . 5
2.4 Toffoli-Gate . . . 5

3 Quantum Algorithms **6**
3.1 Query Operator . . . 6
3.1.1 Quantum Parallelism . . . 7
3.2 Deutsch's Algorithm . . . 7

4 Grover's Algorithm **8**
4.1 Oracle $\hat{O}$ . . . 10
4.1.1 Oracle function f . . . 10
4.1.2 Phonebook example . . . 10
4.2 $\hat{X}$ Operator . . . 10
4.3 Step by step analysis . . . 11
4.4 Geometrical visualization . . . 12
4.5 Iteration and time analysis . . . 13
4.6 Logic circuit . . . 14
4.7 Matlab simulation . . . 15
4.8 Quantum counting . . . 18
4.8.1 Phase estimation . . . 18
4.8.2 Grover quantum counting . . . 19

5 Conclusion **20**

1 Introduction

1.1 Overview

The quantum theory was postulated to explain physical phenomena which could not be explained by classical physics anymore. The new insights enabled scientists to simulate computers which use the quantum mechanical properties to operate. These are the so-called quantum computers. In this thesis I will give a short introduction of the key terms in section 1.2 and 1.3, explain some quantum gates in section 2 and explain a simple algorithm using quantum gates which is in section 3. After that the focus will be on Grover's quantum search algorithm. (Section 4) The idea of this algorithm is to find elements in a database faster than a classical search algorithm. I will explain the concept of Grover's algorithm and introduce the problem when there is an unknown number of search solutions in the database. (Section 4.8) In section 4.7 I present some code output and simulation figures of the algorithm that were programmed using Matlab. For this thesis I have used scripts, papers and books which are listed on the last page.

1.2 Qubits

A Quantumbit, or qubit for short, is represented by 2-dimensional Hilbert space $\mathcal{H}_2$. [1] The states of the bits 0 and 1 of a classical computer correspond to the qubit states $|0\rangle$ and $|1\rangle$. These are quantum mechanical states with only two different outcomes when being measured. There are several opportunities to realize this so-called two state system.[1] Using a quantum mechanical property called superposition principle enables to generate more states than the classical 0 and 1.[2] A superposition is a linear combination of single states.

$$|\Psi\rangle = \sum_{k\in|0\rangle,|1\rangle} \alpha_k |k\rangle \tag{1}$$

A condition for every linear combination is the norm of the state: $\sum_k |\alpha_k|^2 = 1$. When measuring the state $|\Psi\rangle$ the probability to measure the value k is equal to the factor $|\alpha_k|^2$. [1] The qubits $|0\rangle$ and $|1\rangle$ get assigned to the vector form:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

1.3 Quantum register

A quantum register is a 2^n-dimensional Hilbert space out of n-qubits. [4] For example, a quantum register with two qubits would be generated by a tensor product of the two Hilberspaces $\mathcal{H}_2 \otimes \mathcal{H}_2 = \mathcal{H}_4$. There are four new basis vectors which form an orthogonal system: $|00\rangle$,$|01\rangle$,$|10\rangle$ and $|11\rangle$. This is an example of how two combined states would look like:

$$|\Theta\rangle_{\in\mathcal{H}_4} = |\Phi\rangle_{\in\mathcal{H}_2} \otimes |\Psi\rangle_{\in\mathcal{H}_2} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{2}$$

[1] See: Einführung in Quantencomputer [2], P.29
[2] See: Quantum Computation and Quantum Information [3], P.15

The vector form of the new basis states is the tensor product of the single qubits and have the form:

$$|0\rangle\otimes|0\rangle = |00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad |0\rangle\otimes|1\rangle = |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \quad |1\rangle\otimes|0\rangle = |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \quad |1\rangle\otimes|1\rangle = |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Taking this 4-dimensional system and creating an operation with a single qubit operator[1] $\hat{O}$ which only interacts with the second qubit is realized as follows:

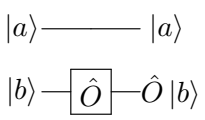


Figure 1: A single qubit operator applied on the second qubit of a two qubit quantum register.

Mathematically, the quantum operation of figure 1 is the tensor product of the states:

$$(\mathbb{1}_2 \otimes \hat{O})\,|ab\rangle = \mathbb{1}_2\,|a\rangle \otimes \hat{O}\,|b\rangle \tag{3}$$

2 Quantum Gates

Quantum gates enable operations with qubits. They can be represented as matrices which have to be unitary. ($MM^\dagger = \mathbb{1}$) That makes the operations always reversible which means that there is never a loss of information throughout the whole computation process. [1]

2.1 NOT-Gate

The NOT-gate is simply turning the qubit state from $|0\rangle$ to $|1\rangle$ and vice versa. This is an illustration of how this gate works:

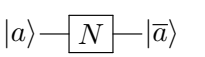


Figure 2: NOT-Gate ($\overline{a}$ is the inverse of a)

The matrix representation of the NOT operation (figure 2) on a single qubit is:

$$N = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \tag{4}$$

[1] A single qubit operator is also known as an unary operator. [5], P.14

2.2 Hadamard-Gate

The Hadamard-gate is a gate which does not exist in classical computing. It changes a qubit into a superposition of two states by rotating it.[1] This gives many new opportunities for computation processes.

$$|0\rangle \text{ —}\boxed{H}\text{— } \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \text{ —}\boxed{H}\text{— } \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Figure 3: Hadamard-Gates when operating a $|0\rangle$ and a $|1\rangle$ state.

The matrix representation of a Hadamard-gate (figure 3) is:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{5}$$

2.3 CNOT-Gate

All the gates before were gates for operating single qubits. The CNOT-Gate needs a quantum register of two qubits. It negates the second qubit only if there is a $|1\rangle$ at the first qubit. This operation is equal to a modulo 2 ($\oplus$) addition. All quantum circuits can be constructed using only CNOT and unary gates. [5]

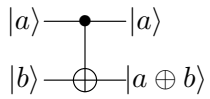


Figure 4: CNOT Gate

The matrix representation of a CNOT gate (figure 4) is:

$$C_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

2.4 Toffoli-Gate

The Toffoli gate is a three qubit gate which is very important for quantum computation. All quantum circuits can be constructed (in some approximated sense) using only Toffoli gates and Hadamard gates. [6]

[1] See: Quantum Computation and Quantum Information [3], P.19

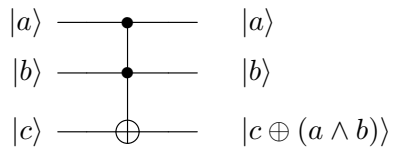
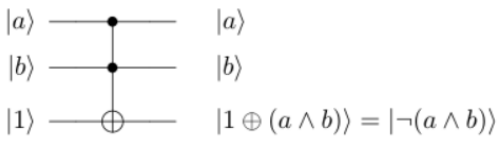
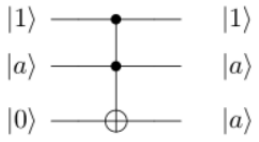


Figure 5: Toffoli-Gate

The Toffoli gate (Figure 5) has three input and output bits. $|a\rangle$ and $|b\rangle$ are the control bits. The third is a target bit that is flipped if both control bits are set to $|1\rangle$, and otherwise is left unchanged.



(a) NAND gate



(b) FANOUT gate

Figure 6: (a) Qubit configuration to perform NAND operations. (b) Qubit configuration to perform FANOUT operations.

In figure 6, two examples of how to use the Toffoli gate can be seen. Having NAND gates (figure 6 (a)) and NOT gates makes it possible to realize all possible logic circuits like in classical computation. The FANOUT gate (figure 6 (b)) is used to duplicate a qubit.
The matrix representation of a Toffoli gate (figure 5) is:

$$Toffoli = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{7}$$

3 Quantum Algorithms

In the following sections $|x\rangle$ will be a decimal repesentation of a set of qubits. Transforming $|x\rangle$ into the binary representation gives the quantum register. For example the state $|3\rangle$ gives the quantum register $|1\rangle \otimes |1\rangle = |11\rangle$.

3.1 Query Operator

The query operator is used to map the value of a function $f(x) : \{0,1\}^n \to \{0,1\}$ onto a qubit whereas the state $|x\rangle$ remains unchanged.

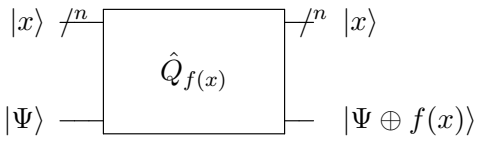


Figure 7: The query operator $\hat{Q}_{f(x)}$ for the function $f(x)$. The '$/^n$' through the wire represents a set of n-qubits.

For example, the query operator acting on the state: $|x\rangle \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{=|\Psi\rangle}$ results in the state:

$$\hat{Q}_{f(x)}(|x\rangle \otimes |\Psi\rangle) = (-1)^{f(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{8}$$

3.1.1 Quantum Parallelism

Quantum parallelism is a fundamental feature that is used by quantum algorithms.

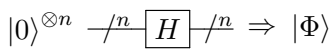


Figure 8: Creation of a parallelized state.

By applying a Hadamard gate for every qubit a superpositioned state is created. (Figure 8) The state can be written as follows:

$$|\Phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{9}$$

Using the state $|\Phi\rangle$ for the query operator gives the result:

$$|\Psi'\rangle = \hat{Q}_{f(x)}(|\Phi\rangle |\Psi\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{10}$$

The superpositioned state $|\Psi'\rangle$ has now information about all function values in its amplitudes although the query operator was applied only once. Thus, the function was queried for more than one value simultaneously. Making use of this is called quantum parallelism. [1]

3.2 Deutsch's Algorithm

Deutsch's Algorithm evaluates if the result of a function $f(x) : \{0,1\} \to \{0,1\}$ is the same for both x or not.

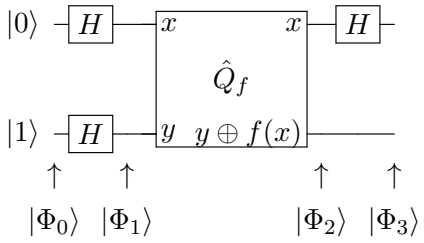


Figure 9: Quantum circuit for Deutsch's algorithm. $\hat{Q}_f$ is the query operator gate. (See figure 7)

Figure 9 shows the quantum circuit for Deutsch's algorithm. The input state

$$|\Phi_0\rangle = |01\rangle \tag{11}$$

is sent through two Hadamard gates and gives

$$|\Phi_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{12}$$

Equation 8 shows that applying $\hat{Q}_f$ to the state $|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ results in $(-1)^{f(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Applying $\hat{Q}_f |\Phi_1\rangle$ gives two possibilities:

$$|\Phi_2\rangle = \begin{cases} \pm \frac{|0\rangle+|1\rangle}{\sqrt{2}} \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{if } f(0) = f(1) \\ \\ \pm \frac{|0\rangle-|1\rangle}{\sqrt{2}} \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{if } f(0) \neq f(1). \end{cases} \tag{13}$$

Whereas the signs of the states in equation 13 are not relevant for the consecutive algorithm. Applying a Hadamard gate on the first qubit again yields:

$$|\Phi_3\rangle = \begin{cases} |0\rangle \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{if } f(0) = f(1) \\ \\ |1\rangle \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{if } f(0) \neq f(1). \end{cases} \tag{14}$$

Measuring the first qubit state always gives $|0\rangle$ or $|1\rangle$ as an outcome. Through this, it is possible to find out whether or not the function is the same for both values of $x = 0$ and $x = 1$. The quantum algorithm for this evaluation is faster than a classical one which would need at least two evaluations of $f(x)$. [3]

4 Grover's Algorithm

A computer's searching process for elements in a database can be very time-consuming. Grover's algorithm is a fast quantum database search algorithm. There is a database of N elements. It searches for the position x of one element in the database which has the value $f(x)$. M is the number of solution elements meaning $f(x)$ gives the wanted value at M positions. When searching a domain the quantum search to find one element requires $\mathcal{O}(\sqrt{\frac{N}{M}})$ steps whereas classical algorithms require in average $\mathcal{O}(\frac{N}{M})$ steps. [7]

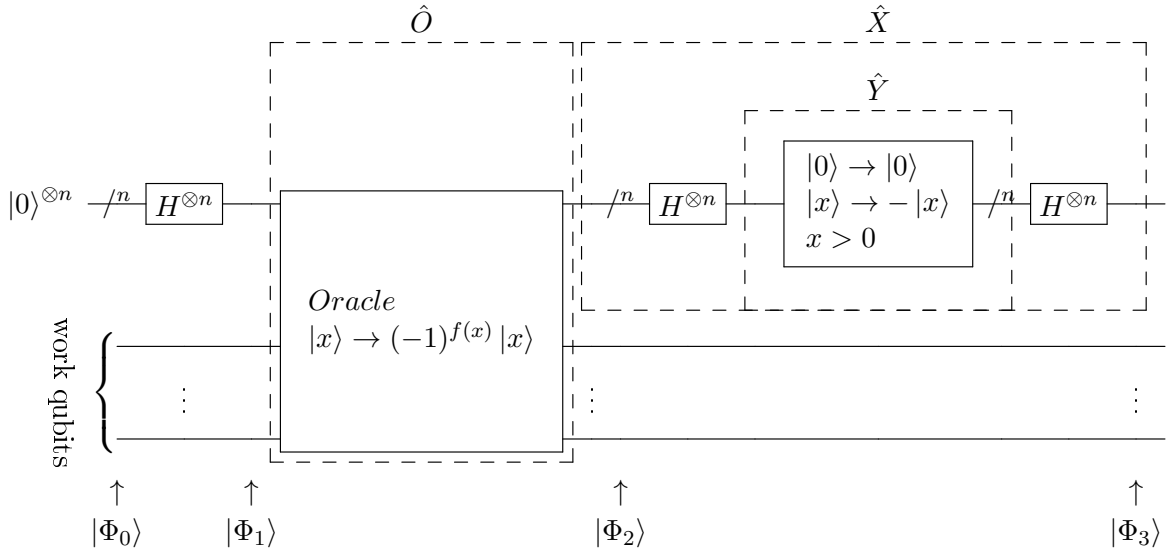


Figure 10: Grover's operator $\hat{G}$ consisting of the $\hat{O}$ and the $\hat{X}$ operator. ($\hat{G} = \hat{X}\hat{O}$)

Grover's algorithm consists of two main operations. The Oracle operator $\hat{O}$ and the 'inversion about the mean' operator $\hat{X}$. Figure 10 shows the connection of these two. Basically, it starts with a wave function with all possible indices of the search domain entries superpositioned. The algorithm consists of single iterations which have to be repeated and after each iteration the amplitudes of the search solutions transform in a way that their probability of getting measured increases whilst the amplitude of all non-solution indices decreases. In section 4.2 and 4.1 the Oracle $\hat{O}$ and the operator $\hat{X}$ get described. One Grover iteration is $\hat{G} = \hat{X}\hat{O}$.
Due to the fact that quantum operations always have to be reversible there is always at least one work qubit for this algorithm. (See section 2) An example of a work qubit would be the $|y\rangle$ qubit in figure 7. This qubit is only needed for the query operator to flip the amplitude for solutions of the function f.

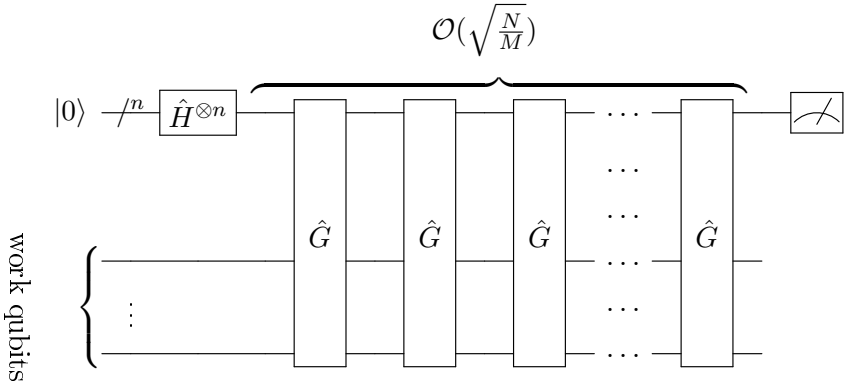


Figure 11: Grover's algorithm.

Figure 11 shows iterations of the Grover operator. The last step is the measurement

of the wave function which ideally gives the solutions. In subsection 4.3 there is a step by step analysis of the operations inside a Grover operator.

4.1 Oracle $\hat{O}$

At first the Oracle $\hat{O}$ is introduced which can be thought as a black box whose only task is to flip the amplitude of the search solutions by 180° using quantum parallelism. (See section 3.1.1) Nevertheless, this section will specify the Oracle and give some simple examples.

4.1.1 Oracle function f

As described in section 3.1, it is possible to query a function simultaneously for more than one value. For Grover's algorithm it is assumed that there is an unsorted list with a domain of 2^n entries. In addition to this, there is a function $f(x)$ which gives the element of the lists entry with the index x whereby $x \in \{x_0, x_1, ..., x_{2^n-1}\}$ is the binary representation of a natural number $\mathcal{N}$. Searching is defined as finding out at which indices x the list entry is y. For this there needs to be another function f_y that queries $f(x)$ and returns 1 if the list entry equals y and 0 if it does not.

$$f_y(x) = \begin{cases} 0 & \text{if } f(x) \neq y \\ 1 & \text{if } f(x) = y \end{cases} \tag{15}$$

These functions implemented are the preconditions for an Oracle. [1]

4.1.2 Phonebook example

M. Charemza gives an example of how the Oracle functions can be seen.[1] He wrote that the unsorted list can be compared to the entries in a phonebook. Each row on the list represents an entry in the phonebook, the telephone number, with the index x_i. The only way to find a certain telephone number is by checking each line and finding out if the telephone number is equal to the wanted one. It is assumed that the telephone number y which we are looking for is at the position $\overline{x}$. The search of the phonebook itself can be represented by a function $f_y(x)$ and checking for each line results in 1 if the line x has the right number, and 0 if it has not.

4.2 $\hat{X}$ Operator

This operator is also called 'inversion about the mean' operator. It consists of the operations

$$\hat{X} = \hat{H}^{\otimes n} \hat{Y} \hat{H}^{\otimes n}. \tag{16}$$

The operator $\hat{Y}$ flips the amplitude for every qubit by 180° except the $|0\rangle$ qubit.

$$\hat{Y} = 2\,|0\rangle\,\langle 0| - \mathbb{1} \tag{17}$$

Applying the $\hat{H}^{\otimes n}$ on both sides gives the $\hat{X}$ operator:

$$\hat{X} = 2\,|\Phi\rangle\,\langle\Phi| - \mathbb{1}. \tag{18}$$

[1] See: An Introduction to Quantum Computing [1], P.46

The state $|\Phi\rangle$ is the same as in equation 9. In section 4.4 the geometrical relevance of this operator gets discussed.

4.3 Step by step analysis

The change of the single states of one Grover iteration $\hat{G}$ will be described using the marked states at the bottom line of figure 10. The function of the single operations will be discussed in the next subsections. The starting state is:

$$|\Phi_0\rangle = |0\rangle^{\otimes n} \otimes |\text{work qubits}\rangle\,. \tag{19}$$

For reasons of convenience the $\otimes\,|\text{work qubits}\rangle$ will be left out for the other parts. After applying the Hadamard gate to all n gates there is a superpositioned state of every index.

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{20}$$

To understand the single steps it is assumed to know that out of $2^n = N$ elements there are M solutions to the search problem. That means that there are M indices of x which return 1 when querying the oracle function $f_y(x)$. Therefore, the wave function is split into two parts. The sum with the $'$ indicates the elements which are not a solution of the problem and the $''$ indicates the solution elements.

$$|\alpha\rangle = \sum\nolimits' |x\rangle \quad \text{and} \quad |\beta\rangle = \sum\nolimits'' |x\rangle \tag{21}$$

The same wave function of equation 20 with the new variables is:

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^n}}(|\alpha\rangle + |\beta\rangle). \tag{22}$$

In the next step the Oracle $\hat{O}$ gets applied and gives:

$$|\Phi_2\rangle = \frac{1}{\sqrt{2^n}}(|\alpha\rangle - |\beta\rangle). \tag{23}$$

At last the result of the $\hat{X}$ operation yields:

$$|\Phi_3\rangle = \frac{1}{\sqrt{2^n}} \left[\left(\frac{2((N-M)-M)}{N} - 1 \right) |\alpha\rangle + \left(\frac{2((N-M)-M)}{N} + 1 \right) |\beta\rangle \right]. \tag{24}$$

By analyzing the prefactors it is already evident that the $|\beta\rangle$ states prefactor increases whilst the other decreases.

4.4 Geometrical visualization

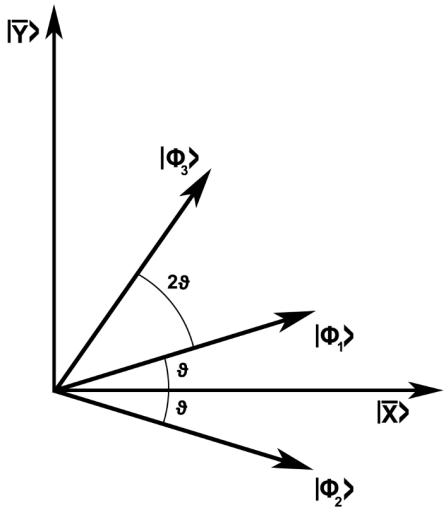


Figure 12: The geometrical visualization of one Grover iteration.

For a graphical visualization two normalized states get defined. ($|\alpha\rangle$ and $|\beta\rangle$ are defined like in equation 21.)

$$|\overline{x}\rangle = \frac{1}{\sqrt{N-M}}\,|\alpha\rangle \quad \text{and} \quad |\overline{y}\rangle = \frac{1}{\sqrt{M}}\,|\beta\rangle \tag{25}$$

Then it is possible to represent $|\Phi_1\rangle$ as:

$$|\Phi_1\rangle = \sqrt{\frac{N-M}{N}}\,|\overline{x}\rangle + \sqrt{\frac{M}{N}}\,|\overline{y}\rangle\,. \tag{26}$$

By applying the Oracle $\hat{O}$ this vector is mirrored around the $|\overline{x}\rangle$ axis.

$$|\Phi_2\rangle = \sqrt{\frac{N-M}{N}}\,|\overline{x}\rangle - \sqrt{\frac{M}{N}}\,|\overline{y}\rangle \tag{27}$$

The $\hat{X}$ gate transforms the vector $|\Phi_2\rangle$ into the following state:

$$|\Phi_3\rangle = \underbrace{\left(\frac{2N-4M}{N}-1\right)\sqrt{\frac{N-M}{N}}}_{=C_1}\,|\overline{x}\rangle + \underbrace{\left(\frac{2N-4M}{N}+1\right)\sqrt{\frac{M}{N}}}_{=C_2}\,|\overline{y}\rangle \tag{28}$$

Because all of the states $|\Phi_i\rangle$ are normalized it is possible to say that

$$\sqrt{\frac{N-M}{N}} = cos(\vartheta) \quad \text{and} \quad \sqrt{\frac{M}{N}} = sin(\vartheta). \tag{29}$$

That transforms $|\Phi_1\rangle$ into

$$|\Phi_1\rangle = cos(\vartheta)\,|\overline{x}\rangle + sin(\vartheta)\,|\overline{y}\rangle \tag{30}$$

and $|\Phi_3\rangle$ into

$$|\Phi_3\rangle = \underbrace{cos(3\vartheta)}_{=C_1} |\overline{x}\rangle + \underbrace{sin(3\vartheta)}_{=C_2} |\overline{y}\rangle \,. \tag{31}$$

This makes it graphically comprehensible that the $\hat{X}$ gate mirrors the $|\Phi_2\rangle$ state around the $|\Phi_1\rangle$ state. (Figure 12 shows the vector states $|\Phi_i\rangle$) Therefore it can be said that:

$$\hat{G}^k |\Phi_1\rangle = cos((2k+1)\vartheta) |\overline{x}\rangle + sin((2k+1)\vartheta) |\overline{y}\rangle \,. \tag{32}$$

In this base one Grover iteration equals the matrix:

$$\hat{G} = \begin{pmatrix} cos(2\vartheta) & -sin(2\vartheta) \\ sin(2\vartheta) & cos(2\vartheta) \end{pmatrix} \tag{33}$$

The probability to measure one of the solutions after k iterations is:

$$P_{sol} = |sin((2k+1)\vartheta)|^2 \tag{34}$$

4.5 Iteration and time analysis

With equation 32 the number of necessary iterations for reasonable measurement results can be found out. It is the goal to find k the number of iterations where the vectors projection on the $|\overline{y}\rangle$ axis is maximized. Therefore, the angle $(2k+1)\vartheta$ needs to be as close to $\frac{\pi}{2}$ $(90°)$ as possible. For that the following approximation is used:

$$\vartheta = sin^{-1}\left(\sqrt{\frac{M}{N}}\right) \approx \sqrt{\frac{M}{N}}. \tag{35}$$

Setting $(2k+1)\sqrt{\frac{M}{N}} = \frac{\pi}{2}$ gives:

$$k = \frac{\pi}{4}\sqrt{\frac{N}{M}} - \frac{1}{2} \tag{36}$$

As k is always an integer the closest integer to k gives the number of iterations needed for the optimal outcome. This shows that Grover's algorithm needs to perform $\mathcal{O}(\sqrt{\frac{N}{M}})$ operations to yield a solution of the search with high probability. [8]
If $M \geq \frac{N}{2}$ the number of iterations needed by the search algorithm increases with M. There are search tasks where it is not known whether the number of solutions M is greater or smaller than $\frac{N}{2}$. Although this is not a realistic situation in database searches there is an approach which helps to prevent that in a case like this the iterations increase. In this case one can double the number of elements in the search space by adding an extra qubit. $(2N = 2^{n+1})$ As a consequence of that, always less than half of the elements are solutions. [3] Running the search algorithm with that augmented input register gives a new estimate for the iterations k:

$$k = \frac{\pi}{4}\sqrt{\frac{2N}{M}} - \frac{1}{2} \tag{37}$$

This approach still increases the number of iterations but helps keeping track of how many iterations are needed.

4.6 Logic circuit

A boolean function of a logic circuit has the same constraints and properties as the Oracle function f. A boolean function maps: $f_B : B^n \to B$, where $B = \{0, 1\}$. When implementing a boolean function as the Oracle function, the truth table of the boolean function can be seen as the list with all the 2^n binary combinations as an index. The list entry would be 0 if the index is not a solution of the boolean function and 1 if the index is a solution. Due to the outcome which is only 1 and 0 the boolean function does not need a further constraint like f_y. It always searches for the entry 1. As an example, suppose to have the boolean function $f_B = (a \wedge b) \wedge c$.

Table 1: The truth table of the boolean function $f_B = (a \wedge b) \wedge c$.

a	b	c	f_B
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Implementing this function into an Oracle to fulfill all the constraints looks like that:

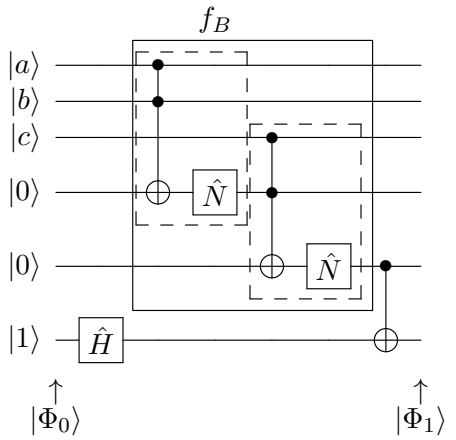


Figure 13: Boolean function f_B implemented into the Oracle operator. The dashed line boxes are AND-gates. (See figure 6 a)

The starting state is:

$$|\Phi_0\rangle = |abc\rangle \otimes \underbrace{|001\rangle}_{\text{work qubits} = |x\rangle} \tag{38}$$

whereas the last three qubits are work qubits. After the application of the new Oracle gate the state is:

$$|\Phi_1\rangle = \underbrace{(-1)^{f_B(a,b,c)}\,|a,b,c\rangle}_{\text{query qubits}}\,|x'\rangle\,. \tag{39}$$

This is a valid Oracle wavefunction, whereas only the query qubits are relevant for the consecutive algorithm. A big disadvantage of using boolean functions in quantum algorithms is that the number of total qubits grows very fast. For example, the simple circuit of figure 13 needs already two work qubits to be realized. The more qubits there are in a quantum register the harder it is to realize it in an experiment. In the next subsection the application of the Grover's algorithm with this Oracle will get simulated with Matlab.

4.7 Matlab simulation

The practical part of my bachelor thesis was to simulate Grover's algorithm with Matlab. The Oracle as it is in figure 13 was implemented into the code and is used for this Grover's algorithm simulation.[1] In this case it is known in prior that there is a search space of 8 elements with one solution. (See truth table 1) Through this it is possible to predict the number of Grover iterations needed to find a solution with the help of equation 36 which was derived in section 4.5.

$$k = \frac{\pi}{4}\sqrt{\frac{N}{M}} - \frac{1}{2} = 1.72 \Rightarrow 2 \text{ iterations} \tag{40}$$

[1] E-Mail tobiaskugel@student.tugraz.at for the source code and simulation graphs

There will be listings of the code output for each iteration. The starting state is equivalent to $|\Phi_1\rangle$ of figure 10 and the last three qubits are the work qubits.

```
|phi1> = 0.35355|000001> + 0.35355|001001> + 0.35355|010001>
+ 0.35355|011001> + 0.35355|100001> + 0.35355|101001>
+ 0.35355|110001> + 0.35355|111001>
```

Applying the Grover operator gives:

```
G*|phi1> = 0.17678|000001> + 0.17678|001001> + 0.17678|010001>
+ 0.17678|011001> + 0.17678|100001> + 0.17678|101001>
+ 0.17678|110001> + 0.88388|111001>
```

The probability to measure the already known solution $|111\rangle|x'\rangle$ is already $P_{sol} = |0.88388|^2 \approx 0.78$ after one operation. The second operation gives:

```
G*G*|phi1> = - 0.088388|000001> - 0.088388|001001> - 0.088388|010001>
- 0.088388|011001> - 0.088388|100001> - 0.088388|101001>
- 0.088388|110001> + 0.97227|111001>
```

The probability of measuring the solution $|111\rangle|x'\rangle$ is now $P_{sol} = |0.97227|^2 \approx 0.95$ which is already a reasonable value for the predicted two operations. Applying the operator one more time gives:

```
G*G*G*|phi1> = -0.30936|000001> - 0.30936|001001> - 0.30936|010001>
- 0.30936|011001> - 0.30936|100001> - 0.30936|101001>
- 0.30936|110001> + 0.57452|111001>
```

The probability to measure the solution $P_{sol} = |0.57452|^2 \approx 0.33$ decreases again which means that measuring the value of the wavefunction after two iterations would already give a reasonable result.

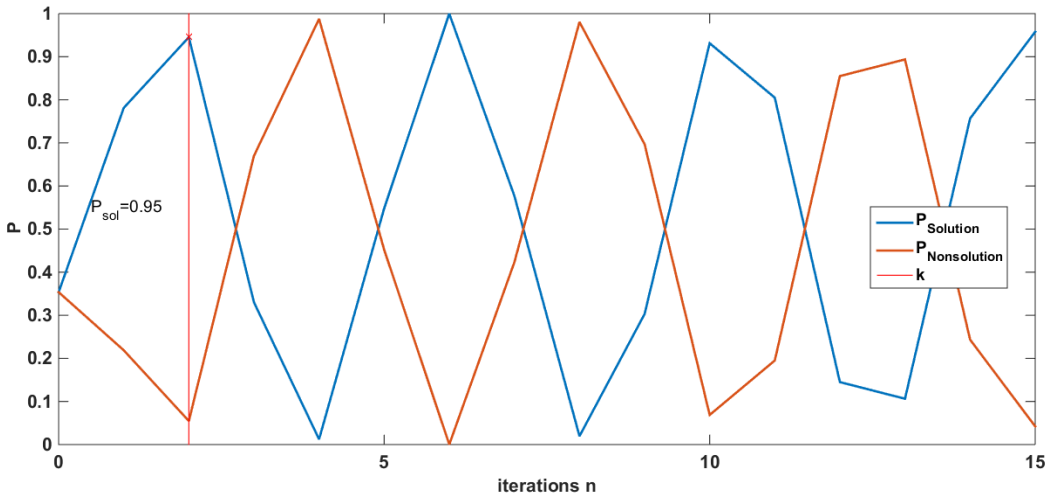


Figure 14: The probability $P_{Solution}$ to measure a solution in dependence of the number of Grover iterations n. k is the calculated value for the number of iterations needed. (See equation 40)

Figure 14 shows the alternation of the probabilities to measure a solution in dependence of the number of Grover iterations. The vertical red line marks the calculated number of needed solutions which corresponds to the first probability peak of the simulation.

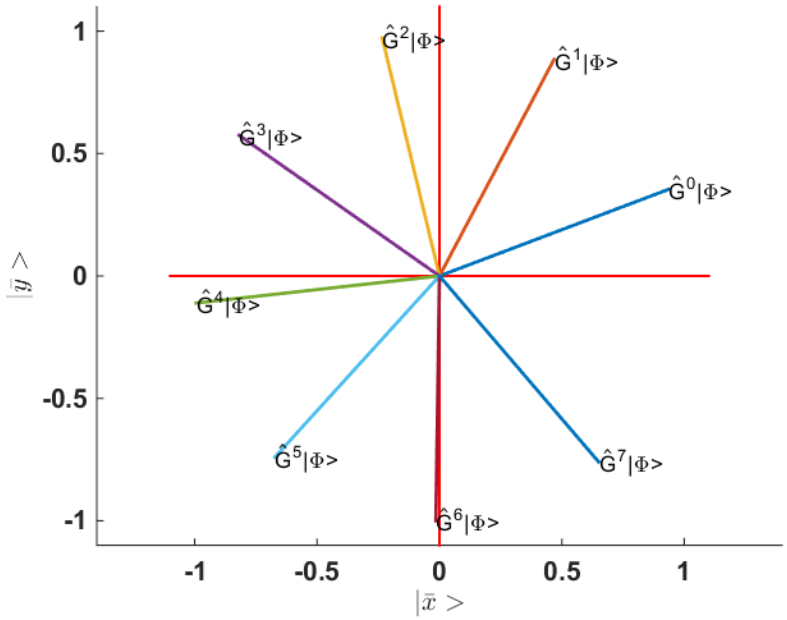


Figure 15: The Grover iterations represented in the same base as in the geometrical visualization of figure 12.

Figure 15 shows the simulation of the Grover's algorithm as a form of rotation with the rotational matrix. (Equation 33) The new wavefunction in this base is equal to equation 32. The projection of the vectors onto the vertical axis is proportional to the probability

to measure a solution.

4.8 Quantum counting

There are searching tasks where the number of solutions M is not known in advance. Combining Grover's algorithm with the phase estimation technique, which is based upon the quantum Fourier transformation, gives us the possibility to estimate the number of solutions M.

4.8.1 Phase estimation

Assume there is an unitary operator $\hat{U}$ with the eigenvector $|u\rangle$ and the eigenvalue $e^{2\pi i\varphi}$. The goal of the phase estimation algorithm is to estimate φ. The algorithm uses two registers. The first contains t qubits with the initial state $|0\rangle$. The size of this register depends on the number of digits of accuracy we want to estimate φ and the probability we wish the phase estimation to be successful. The second register needs as many qubits as the operator $\hat{U}$ needs to operate. The phase estimation is performed in three stages. The first contains out of these operations:

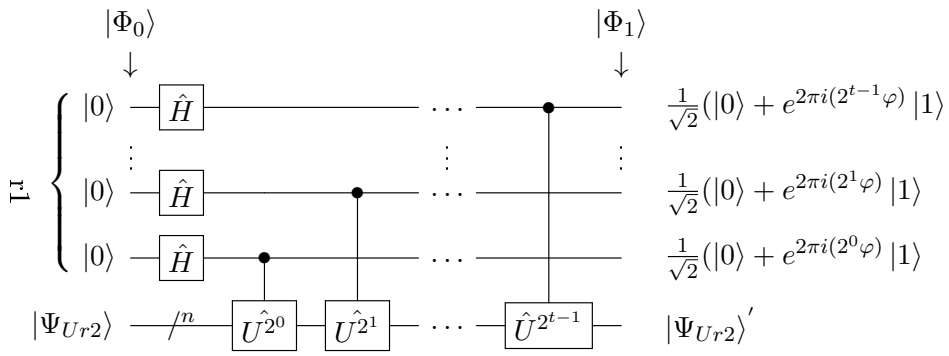


Figure 16: The first stage of the phase estimation circuit. $r1$ marks the first register with t qubits. The state $|\Psi_{Ur2}\rangle$ is the state of the second register.

As seen in section 4.4, it is possible to write every state of the second register of figure 16 as a superpositioned state of the eigenvectors of the operator $\hat{U}$. The first stage of this algorithm maps the phasefactors of the new state onto the qubits of the first register. The first state $|\Phi_0\rangle$ of figure 16 is:

$$|\Phi_0\rangle = |000\rangle \otimes |\Psi_{Ur2}\rangle \tag{41}$$

and after applying the gates we yield:

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} e^{2\pi i\varphi k} |x\rangle \otimes |\Psi_{Ur2}\rangle' \tag{42}$$

$|\Psi_{Ur2}\rangle'$ is the state of the second register after the application of the gates. The second stage is an application of an inverse Fourier transform on the first register. The third and final stage is to measure the state of the first register and evaluate φ. [3]

4.8.2 Grover quantum counting

One Grover iteration in the basis of $|\overline{x}\rangle$ and $|\overline{y}\rangle$ which was used for the geometrical visualization in section 4.4 is equal to the matrix (equation 33):

$$\hat{G} = \begin{pmatrix} cos(2\vartheta) & -sin(2\vartheta) \\ sin(2\vartheta) & cos(2\vartheta) \end{pmatrix}$$

The eigenvalues of this matrix are

$$\lambda_1 = e^{i2\vartheta} \quad \text{and} \quad \lambda_2 = e^{i(2\pi - 2\vartheta)}. \tag{43}$$

For an easier analysis of the next calculations the elements of the second register are doubled by adding one qubit. Using equation 29: $sin^2(\vartheta) = \frac{M}{2N}$ gives the correlation between the angle and the search elements M and N. Applying the phase estimation circuit of figure 16 gives an estimate of ϑ to an accuracy of m bits, with a probability of success of at least $1 - \epsilon$. To achieve this there need to be $t = m + [log(2 + \frac{1}{2\epsilon})]$ qubits in the first register.

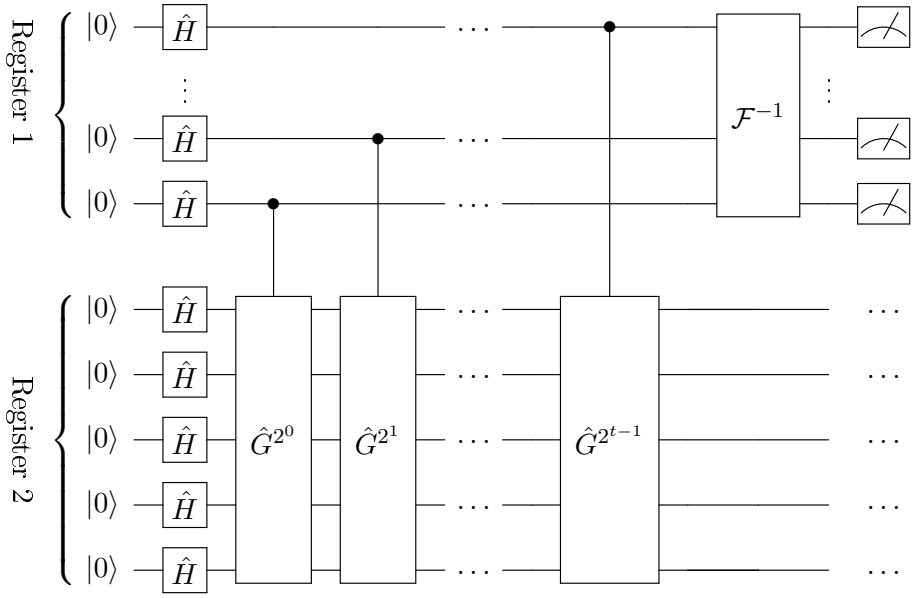


Figure 17: The circuit for finding the number of solutions of a search problem. *Register* 1 marks the first register with t qubits. *Register* 2 marks the Grover algorithm register with n+1 qubits. $\mathcal{F}^{-1}$ is the inverse Fourier transform.

Figure 17 shows the use of Grover operations in a phase estimation algorithm like figure 16. The measurement gives an estimate of the angle ϑ. Using the correlation $sin^2(\vartheta) = \frac{M}{2N}$ the number of answers is:

$$M = sin^2(\vartheta)2N. \tag{44}$$

The absolute error of that value has the upper bound

$$|\Delta M| < \left(\sqrt{2MN} + \frac{N}{2^{m+1}}\right) 2^{-m}. \text{ [3]} \tag{45}$$

5 Conclusion

The goal of this thesis was to get to know quantum algorithms in general and learn about how to simulate them with Matlab. It especially focused on the Grover's algorithm in the case of an arbitrary number of search solution elements. The more search solution elements there are the less iterations it takes to find one of the elements. (See equation 36) But to find all the elements the algorithm needs to get applied more often. This means that the computation time increases with the number of search solution elements. As it was described in section 4.8 it is possible to find out the number of solution elements of a search problem which is unkown apriori. Bigger quantum registers are needed for increasing the measurement accuracy. The Matlab simulation of the Grover's algorithm shows the behavior of the probability to measure a reasonable solution. (Figure 14 and 15) If this quantum search algorithm could get applied on conventional databases it wouldn't only speed up searching processes but also solve scientific numerical tasks. For example solving 3-SAT[1] problems. [9] An interesting paper about the experimental realization of the Grover's algorithm was written by J. Jones, M. Mosca and R. Hansen. They used an NMR quantum computer to realize a two qubit register.[2] All in all, understanding Grover's algorithm and its different applications gives a good overview of quantum computing and shows that there is a lot of potential in quantum computers for the future.

[1] A complex boolean function. SAT is an abbreviation for satisfiability.

[2] See: Implementation of a Quantum Search Algorithm on a Nuclear Magnetic Resonance Quantum Computer [7]

References

[1] M. Charemza. *Projectscript: An introduction to quantum computing* (2005) https://warwick.ac.uk/fac/sci/physics/research/cfsa/people/pastmembers/charemzam/ (20.08.2018).

[2] E. Arrigoni. *Vorlesungsskript: Einführung in Quantencomputer.* - (2010) https://itp.tugraz.at/ arrigoni/ (20.08.2018).

[3] M. A. Nielsen, I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010. doi:10.1017/CBO9780511976667

[4] R. B. Griffiths. *Hilbert Space Quantum Mechanics.* - (2014) http://quantum.phys.cmu.edu/QCQI/qmd111.pdf (20.08.2018).

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, H. Weinfurter. *Elementary gates for quantum computation.* Phys. Rev. A **52** (1995) 3457. doi:10.1103/PhysRevA.52.3457

[6] Y. Shi. *Both Toffoli and controlled-NOT Need Little Help to Do Universal Quantum Computing.* Quantum Info. Comput. **3** (2003) 84.

[7] J. A. Jones, M. Mosca, R. H. Hansen. *Implementation of a quantum search algorithm on a quantum computer.* Nature **393** (1998) 344. doi:10.1038/30687

[8] M. Boyer, G. Brassard, P. Høyer, A. Tapp. *Tight Bounds on Quantum Searching.* Fortschritte der Physik **46** (1998) 493. doi:10.1002/(SICI)1521-3978(199806)46:4

[9] S.-T. Cheng, M.-H. Tao. *Quantum cooperative search algorithm for 3-SAT.* Journal of Computer and System Sciences **73** (2007) 123 . doi:10.1016/j.jcss.2006.09.003