Quantum Computation: Shor's algorithm

A.J. Cornelissen

July 5, 2016

Bachelor Thesis
Assessment committee: Prof. Dr. J.M.A.M. van Neerven
Dr. M. Blaauboer
Dr. J.L.A. Dubbeldam
Dr. A.F. Otte
Dr. J.A.M. de Groot
Research institute: Delft University of Technology
Faculties: Electrical Engineering, Mathematics and Computer Science
Applied Sciences
Contact: a.j.cornelissen@student.tudelft.nl

Abstract

In recent years, the field of quantum computation has evolved to a promising research area, with the capacity to become as important as classical programming is today.

This text serves as an introduction into the field of quantum computation. The main results are a proof that any quantum circuit can be implemented using a finite set of quantum gates, and a description of the principle and implementation of Shor's algorithm. Furthermore, Shor's algorithm is simulated on a classical computer to check if the procedure yields the results that are expected. This turns out to be the case.

There are some aspects of this research that could be improved. The lower bound on the probability that Shor's algorithm succeeds could be improved, and the classical simulation of Shor's algorithm could be sped up as well by using a better implementation of the inverse Fourier transform.

Contents

Abstract **i**

1 Introduction **1**

2 Postulates of quantum mechanics **2**
2.1 Mathematical background . . . 2
2.1.1 Hilbert spaces . . . 2
2.1.2 Tensor products . . . 3
2.2 Postulates . . . 6

3 Quantum circuits **10**
3.1 Qubits . . . 10
3.1.1 Bits vs. qubits . . . 10
3.1.2 Multiple qubits . . . 11
3.2 Quantum gates . . . 13
3.2.1 Logic gates vs. quantum gates . . . 13
3.2.2 Single qubit gates . . . 14
3.2.3 Multiple qubit gates . . . 14
3.3 Quantum circuits . . . 16
3.3.1 Swap circuit . . . 16
3.3.2 Registers . . . 17
3.4 Example: Deutsch's algorithm . . . 17

4 Universality of the Controlled NOT, Hadamard and $\pi/8$ gate **20**
4.1 Reduction to two-level unitary operators . . . 20
4.1.1 Two-level unitary matrices and operators . . . 20
4.1.2 Reduction of a quantum gate to a product of two-level unitary operators . . . 21
4.2 Reduction to (controlled) single qubit gates . . . 24
4.2.1 Swapping . . . 24
4.2.2 Implementing conditional operations with multiple control qubits . . . 24
4.3 Approximation of (controlled) single qubit gates by Hadamard and $\pi/8$ gates . . . 27
4.3.1 Bloch sphere . . . 27
4.3.1.1 Representation of the state of a single qubit using the Bloch sphere . . . 27
4.3.1.2 The effect of single qubit operations on the Bloch sphere . . . 28
4.3.2 Implementation of controlled single qubit gates . . . 36
4.3.3 Distance between unitary matrices . . . 38
4.3.4 Approximation of single qubit gates by Hadamard and $\pi/8$ gates . . . 42

5 Shor's algorithm **45**
5.1 Quantum Fourier transform . . . 45
5.2 Phase estimation algorithm . . . 50
5.3 Order-finding algorithm . . . 58
5.3.1 Principle behind the quantum circuit . . . 59
5.3.2 Implementation of the quantum circuit . . . 60
5.3.3 Continued fraction expansion . . . 64
5.3.4 Finding the order from its divisors . . . 73
5.4 Shor's algorithm . . . 74

6 Conclusion **78**

References **79**

A Proof that λ defined by the λ-equation is irrational **80**
A.1 Required knowledge from abstract algebra . . . 80
A.1.1 Definitions . . . 80
A.1.2 Euclidean division . . . 83
A.1.3 Unique factorization theorem . . . 85
A.1.4 Gauss's lemma . . . 87
A.2 Cyclotomic polynomials . . . 88
A.2.1 Definition . . . 88
A.2.2 Integer coefficients . . . 90
A.2.3 Irreducibility over $\mathbb{Q}$ . . . 90
A.3 The λ-polynomial . . . 94
A.3.1 $e^{2\pi i\lambda}$ is a root . . . 94
A.3.2 Irreducibility over $\mathbb{Q}$ . . . 95
A.3.3 Completion of the proof . . . 96

B Simulation of Shor's algorithm on a classical computer **97**

C Experimental realization of Deutsch's algorithm **106**

1 Introduction

In recent decades, the world, as it entered the period collectively referred to as *the information age*, has undergone dramatic changes, under the influence of rapid developments of computerized technologies. These technological developments manifest in a myriad of applications, ranging from the microprocessors that are at the core of the smartphone one uses on an hourly basis, to the intricate systems that allow for spacecrafts to connect and dock in the midsts of space. This digital revolution has irrevocably changed the world, and technology nowadays takes a prominent place in one's everyday life.

The capacity to fabricate increasingly smaller devices was one of the developments that contributed immensely to the digital revolution. As devices got smaller, they became easier to manage, and they became suited for a larger variety of applications. This development in particular has gone very rapidly, as can be seen from the growth of the number of transistors that are present in a processor.

This development, though, is reaching its physical limits. As the size of transistors is approaching the order of nanometers, unwanted effects like *quantum tunneling* disrupt the normal operation of these components, yielding undesirable results. So, to keep up with the demands of society, researchers are faced with the question of finding other ways to improve technologies.

One of the most radical changes that is being pursued in the last 20 years, is a change in the way that calculations are implemented physically. Rather than trying to avoid the quantum mechanical effects that manufacturers of processors are faced with, researchers in the field of *quantum computing* try to harness these effects. Computers that are based on these quantum mechanical laws are collectively referred to as *quantum computers*, as opposed to *classical computers* that rely on the classical laws of physics.

There are, roughly speaking, two difficulties that have to be overcome in order to realize a functional quantum computer. First of all, one needs to be able to construct a physical system that can be manipulated according to these quantum mechanical laws. Recently, headway has been made in this direction by researchers from IBM [1], as they have provided the scientific community with a first functional quantum computer. The experimental realization of the quantum computer, though, will not be the topic of this text.

The other challenge that needs to be overcome, is harnessing the full potential of such a quantum computer, when it is constructed. To this end, powerful *quantum algorithms*, that can be used to build programs that can be run on quantum computers, need to be developed. Researchers that take up the challenge of devising these quantum algorithms will find that it is eminent to adopt the logic of quantum mechanics as an alternative form of reasoning, in contrast to the classical one we use on a daily basis.

This text will focus on covering the basis of quantum mechanics, and giving the reader insight in the alternative form of reasoning described above. These observations will be used to introduce quantum circuits. To this end, two exemplary results will be covered. The first shows that any quantum algorithm can be implemented using a finite set of components, which will be the subject of chapter 4. Thereafter, the exemplary algorithm provided by Shor to factorize integers is covered in chapter 5. This algorithm in particular will provide a glimpse of the full potential of the quantum computer, and how it can be used to accomplish tasks more efficiently than a classical computer.

The approach taken in this text will be largely based on the work of Nielsen and Chuang [2], but with more emphasis on the theoretical mathematical process.

This research was part of the double degree Bachelor program Applied Mathematics and Applied Physics, provided by the faculties Applied Sciences and Electrical Engineering, Mathematics and Computer Science, at the Technical University in Delft. It constitutes a Bachelor Thesis of both studies simultaneously.

2 Postulates of quantum mechanics

In order to understand how a quantum computer functions, obviously one needs to understand the very basics of quantum mechanics, and how these can be put to good use. The basics of quantum mechanics are comprised of four very general propositions, collectively referred to as the postulates of quantum mechanics. They can be seen as the axioms on which the entire field of quantum mechanics is built.

In the first subsection, 2.1, an overview of some required mathematical background will be given. Thereafter, in subsection 2.2, the postulates will be covered one by one.

2.1 Mathematical background

In order to fully comprehend the postulates of quantum mechanics, one is required to be familiar with a few mathematical definitions. These will be summarized below, and are mainly included for convenience. First of all, we will have a look at Hilbert spaces and some of their properties that are relevant for this text. Next, we will highlight some of the relevant properties of tensor products.

2.1.1 Hilbert spaces

Hilbert spaces are of profound importance in quantum mechanics, as they completely describe all the states a system can be in. The reader is assumed to be familiar with vector spaces, inner products, norms, complete metric spaces and fields. Definitions of these can be found in [3], and a definition of fields is also present in appendix A.

Definition 2.1: Hilbert space
Let V be a vector space over a field F, on which an inner product $\langle \cdot, \cdot \rangle$ is defined. Then we define a norm such that for $v \in V$, $|v| = \sqrt{\langle v, v \rangle}$. Then $d : V \times V \to \mathbb{R}$ defined by $d(v, w) = |v - w|$ for $v, w \in V$ is a metric. If (V, d) forms a complete metric space, then V is a Hilbert space over F.

For example, $\mathbb{R}^2$ is a vector space over $\mathbb{R}$. If we take the standard inner product, we have for $x = (x_1, x_2), y = (y_1, y_2) \in \mathbb{R}^2$ that $\langle x, y \rangle = x_1 y_1 + x_2 y_2$. So, we define the norm by $|x| = \sqrt{x_1^2 + x_2^2}$. Hence, the distance of two vectors in $\mathbb{R}^2$ is just the Euclidean distance, and it is well known that $\mathbb{R}^2$ with the Euclidean metric forms a complete metric space. Hence, $\mathbb{R}^2$ is a Hilbert space over $\mathbb{R}$.

Definition 2.2: Complex Hilbert space
A Hilbert space over the complex numbers $\mathbb{C}$ is called a complex Hilbert space.

For example, one can easily check using the procedure outlined above that $\mathbb{C}^n$ is a complex Hilbert space for any $n \in \mathbb{N}$. This is the Hilbert space that will be of most use to us. Intuitively, it suffices during the remainder of this text to think about Hilbert spaces as if they are identical to $\mathbb{C}^n$, for some $n \in \mathbb{N}$. The following theorem justifies this claim.

Theorem 2.3: Isomorphism between a complex Hilbert space and $\mathbb{C}^n$
Let H be a complex Hilbert space, with dimension n. Then it is isomorphic with the complex Hilbert space $\mathbb{C}^n$.

Proof: As H has dimension n, we can find a linearly independent set of n vectors in H. Using the Gram-Schmidt orthogonalization process, we can turn this set into an orthonormal basis $\{h_1, ..., h_n\}$ of H. Now, we can create a linear map $\psi : H \to \mathbb{C}^n$, with the property that $\psi(h_i) = e_i$, where e_i denotes the ith unit standard basic vector in $\mathbb{C}^n$. As we now map two orthonormal bases bijectively, we have found a bijective

map between H and $\mathbb{C}^n$. Additionally, this map is linear, so it is an isomorphism. Hence, H and $\mathbb{C}^n$ are isomorphic. $\square$

2.1.2 Tensor products

Another very useful mathematical concept that one often encounters in quantum mechanics is the tensor product. Before turning our attention to this concept, we provide some auxiliary definitions.

Definition 2.4: Dual of a vector space
Let V be a vector space over a field F. Then the set of all linear functions from V to F is called the dual of V, and is denoted by V^*.

Note that there is a natural way to define addition and scalar multiplication on elements of the dual of a vector space. Suppose that V is vector space over F, and let $\psi, \psi \in V^*$, then, for any $v \in V$ and $f \in F$, define:

$$(\phi + \psi)(v) = \phi(v) + \psi(v) \qquad \text{and} \qquad (f\phi)(v) = f \cdot \phi(v)$$

Now, it is easily checked that V^* is a vector space over F itself.

Definition 2.5: Bilinear functions of vector spaces
Suppose V and W are vector spaces over some scalar field F. Now suppose that ϕ is a function that maps $V \times W$ to F. Then, ϕ is said to be bilinear if it satisfies the following criteria:

1. *For all $v_1, v_2 \in V$ and $w \in W$, $\phi(v_1 + v_2, w) = \phi(v_1, w) + \phi(v_2, w)$.*
2. *For all $v \in V$ and $w_1, w_2 \in W$, $\phi(v, w_1 + w_2) = \phi(v, w_1) + \phi(v, w_2)$.*
3. *For all $v \in V$, $w \in W$ and $f \in F$, $\phi(fv, w) = \phi(v, fw) = f\phi(v, w)$.*

The set of all such functions is denoted by $\mathcal{B}(V, W)$.

It is also easily checked that $\mathcal{B}(V, W)$ is a vector space over F, by checking that it is closed under addition and multiplication by a scalar in F.

Now, we turn our attention to the tensor product of two vector spaces. [4]

Definition 2.6: Tensor product of two vector spaces
Suppose V and W are two vector spaces over a scalar field F. We define the following function, which is referred to as the tensor product:

$$\cdot \otimes \cdot : V \times W \to \mathcal{B}(V, W)^*$$

where evaluating this function for any $v \in V$, $w \in W$ and $B \in \mathcal{B}(V, W)$ yields:

$$(v \otimes w)(B) = B(v, w)$$

We refer to the tensor product of the vector spaces V and W as the linear subspace of $\mathcal{B}(V, W)^*$ that is spanned by the tensor product of vectors in V and W, as such:

$$V \otimes W = \text{Span}\{v \otimes w : v \in V, w \in W\}$$

An element of $V \otimes W$ is called a pure tensor if it can be written as $v \otimes w$, for some $v \in V$ and $w \in W$.

As $V \otimes W$ is defined as the linear subspace of $\mathcal{B}(V, W)^*$, we find trivially that $V \otimes W$ is a vector space over F as well. Do note the tensor product is itself bilinear, as we can check all properties from definition 2.5:

1. Take $v_1, v_2 \in V$ and $w \in W$ at random. Then, for any $B \in \mathcal{B}(V, W)$, we have:
   $((v_1 + v_2) \otimes w)(B) = B(v_1 + v_2, w) = B(v_1, w) + B(v_2, w) = (v_1 \otimes w)(B) + (v_2 \otimes w)(B)$

2. Take $v \in V$ and $w_1, w_2 \in W$ at random. Then, for any $B \in \mathcal{B}(V,W)$, we find:
   $(v \otimes (w_1 + w_2))(B) = B(v, w_1 + w_2) = B(v, w_1) + B(v, w_2) = (v \otimes w_1)(B) + (v \otimes w_2)(B)$
3. Take $v \in V$, $w \in W$ and $f \in F$ at random. Then, for any $B \in \mathcal{B}(V,W)$, we have:
   $((fv) \otimes w)(B) = B(fv, w) = fB(v, w) = f(v \otimes w)(B) = B(v, fw) = (v \otimes (fw))(B)$

As $V \otimes W$ is a vector space, it makes sense to determine its dimension. This is the subject of the next theorem:

Theorem 2.7: Dimension of the tensor product of two vector spaces
Suppose V and W are two vector spaces over F, with finite dimensions n and m, respectively. Let $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w_m\}$ be bases of V and W, respectively. Then the dimension of the tensor product of V and W is given by $\dim(V \otimes W) = nm$ and a basis for $V \otimes W$ is given by $\{v_i \otimes w_j : 1 \leq i \leq n, 1 \leq j \leq m\}$.

Proof: From the definition of the tensor product of V and W, we find that the set $\{v \otimes w : v \in V, w \in W\}$ is a set that spans $V \otimes W$. Take a vector $v \otimes w$ in this set at random. Then we can write v and w as linear combinations of the bases of V and W that we defined above, so we find that there exist $c_1, \ldots, c_n$ and $d_1, \ldots, d_m$ in F such that:

$$v = \sum_{i=1}^{n} c_i v_i \qquad \text{and} \qquad w = \sum_{i=1}^{m} d_i w_i$$

From the bilinearity of the tensor product, we now find:

$$v \otimes w = \sum_{i=1}^{n} \sum_{j=1}^{m} c_i d_j (v_i \otimes w_j)$$

So, if we define the following set:

$$B = \{v_i \otimes w_j : 1 \leq i \leq n, 1 \leq j \leq m\}$$

Then we find $v \otimes w \in \text{Span}(B)$. As we chose $v \otimes w$ in $\{v \otimes w : v \in V, w \in W\}$ at random, we now find:

$$\{v \otimes w : v \in V, w \in W\} \subseteq \text{Span}(B)$$

Hence, as from their definitions it is clear that $\text{Span}(B)$ is a subset of $V \otimes W$, we find: $V \otimes W = \text{Span}\{v \otimes w : v \in V, w \in W\} = \text{Span}(B)$. As B contains nm vectors, we find $\dim(V \otimes W) \leq nm$. If we now prove that B is a set of independent vectors, we find that B is a basis for $V \otimes W$, from which it easily follows that $\dim(V \otimes W) = nm$.

To this end, suppose we have $\lambda_{i,j} \in F$ with $1 \leq i \leq n$ and $1 \leq j \leq m$ and suppose:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} (v_i \otimes w_j) = 0$$

The 0 on the right hand side is now the additive identity of $\mathcal{B}(V,W)^*$. Hence, for every element A in $\mathcal{B}(V,W)$, we find:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} A(v_i, w_j) = \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} (v_i \otimes w_j)(A) = 0(A) = 0$$

Now, the 0 on the right hand side is the additive identity of F. This holds for any choice of A in $\mathcal{B}(V,W)$. So, if we require A to be a function $\phi \cdot \psi$, with $\phi : V \to F$ and $\psi : W \to F$ both linear functions, we find by plugging in:

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} \phi(v_i) \psi(w_j) = \psi \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} \phi(v_i) w_j \right)$$

As this holds for any choice of ϕ, we find:

$$0 = \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{i,j} \phi(v_i) w_j$$

As the set $\{w_1, \ldots, w_m\}$ forms a basis of W, it is an independent set of vectors, so we find, for all $1 \leq j \leq m$:

$$0 = \sum_{i=1}^{n} \lambda_{i,j} \phi(v_i) = \phi\left(\sum_{i=1}^{n} \lambda_{i,j} v_i \right)$$

Again, as this must hold for any linear function ϕ, we find for all $1 \leq j \leq m$:

$$0 = \sum_{i=1}^{n} \lambda_{i,j} v_i$$

But, similar to the argument before, the set $\{v_1, \ldots, v_n\}$ forms a basis for V, so it is independent, hence we find for all $1 \leq i \leq n$ and $1 \leq j \leq m$:

$$\lambda_{i,j} = 0$$

So, B is indeed linearly independent, hence it forms a basis for $V \otimes W$. This in turn implies $\dim(V \otimes W) = nm$, as required. $\square$

Suppose now that V and W have standard bases, denoted by $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w_m\}$, respectively. Then, we will define the basis $\{v_1 \otimes w_1, \ldots, v_1 \otimes w_m, v_2 \otimes w_1, \ldots v_n \otimes w_m\}$ to be the standard basis of $V \otimes W$.

Note that a direct consequence of this theorem is that with $n, m \in \mathbb{N}$, $\mathbb{C}^n \otimes \mathbb{C}^m$ is isomorphic with $\mathbb{C}^{nm}$. If we define the standard basis vectors of $\mathbb{C}^k$ by $e_1^{(k)}, \ldots, e_k^{(k)}$, with $k \in \mathbb{N}$, then we can associate $e_i^{(n)} \otimes e_j^{(m)}$ with $e_{n(i-1)+j}^{(nm)}$. This gives rise to a bijective mapping between the bases of $\mathbb{C}^n \otimes \mathbb{C}^m$ and $\mathbb{C}^{nm}$, implying that they are isomorphic.

We will now turn our attention to linear operators on $V \otimes W$. Suppose V and W are again vector spaces over some scalar field F, and L_V and L_W are linear operators on V and W, respectively. Then, these linear operators combine naturally into a linear operator on $V \otimes W$, denoted by $L_V \otimes L_W$. It has the following action on the pure tensor $v \otimes w$ for any $v \in V$ and $w \in W$:

$$(L_V \otimes L_W)(v \otimes w) = L_V(v) \otimes L_W(w)$$

Its action on $V \otimes W$ is now uniquely determined by the linearity of $L_V \otimes L_W$.

There are some interesting phenomena that occur when we have a look at the matrix representation of these operator on tensor products of vector spaces.

Definition 2.8: Kronecker product
Suppose A and B are two square complex matrices with dimensions $n \times n$ and $m \times m$. Then the Kronecker product of these matrices is denoted by $A \otimes B$ and defined as the following complex $nm \times nm$ matrix:

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix}$$

Theorem 2.9: Matrix representation of linear operators on the tensor product of vector spaces
Suppose V and W are vector spaces of dimensions n and m, respectively. Denote the standard bases of V and W by $\{v_1, \ldots, v_n\}$ and $\{w_1, \ldots, w_m\}$, respectively. Let L_V and L_W be linear operators that act on V and W, and have matrix representations A and B with respect to these standard bases, respectively. Then, the matrix representation of $L_V \otimes L_W$, with respect to the standard basis of $V \otimes W$, is given by $A \otimes B$.

Proof: Take a vector from the standard basis of $V \otimes W$ at random. Then we note that there exist $1 \leq i \leq n$ and $1 \leq j \leq m$ such that this basis vector is $v_i \otimes w_j$. We find: $(L_V \otimes L_W)(v_i \otimes w_j) = L_V(v_i) \otimes L_W(w_j)$. We can find $L_V(v_i)$ and $L_W(w_j)$ from the matrix representations of L_V and L_W:

$$L_V(v_i) = \sum_{k=1}^{n} A_{ki} v_k \qquad \text{and} \qquad L_W(w_j) = \sum_{l=1}^{m} B_{lj} w_l$$

Hence, by the bilinear property of the tensor product, we find:

$$(L_V \otimes L_W)(v_i \otimes w_j) = L_V(v_i) \otimes L_W(w_j) = \left(\sum_{k=1}^{n} A_{ki} v_k \right) \otimes \left(\sum_{l=1}^{m} B_{lj} w_l \right) = \sum_{k=1}^{n} \sum_{l=1}^{m} A_{ki} B_{lj} (v_k \otimes w_l)$$

So, in column $n(i-1)+j$ of the matrix representation of $L_V \otimes L_W$, we find that the index at row $n(k-1)+l$ is equal to $A_{ki}B_{lj}$. This is exactly what is prescribed by the theorem. Hence, this completes the proof. □

Finally, we introduce some shorthand notation. Suppose that we have some expression E. Then, with $n \in \mathbb{N}$, $E^{\otimes n}$ is a shorthand notation for $E \otimes \cdots \otimes E$, where E is repeated n times. Furthermore, suppose $n, m \in \mathbb{Z}$ with $n > m$, and suppose $E_m, \ldots, E_n$ are expressions. Then we introduce the following shorthand notations:

$$\bigotimes_{i=m}^{n} E_i = E_m \otimes E_{m+1} \otimes \cdots \otimes E_n \qquad \text{and} \qquad \bigotimes_{i=n}^{m} E_i = E_n \otimes E_{n-1} \otimes \cdots \otimes E_m$$

2.2 Postulates

There are four postulates that are collectively referred to as the postulates of quantum mechanics. These are covered one by one below, similar to their treatment in [2].

The first postulate of quantum mechanics describes the states an isolated physical system can be in.

Postulate 1: State space
Consider an isolated physical system. Associated with this system is a complex Hilbert space, referred to as the state space of the system. The state of the system is completely described by a unit vector in this Hilbert space, referred to as the state vector.

Do note that this postulate does not specify which Hilbert space should be associated with which physical system. All it does is it provides the framework in which the physical system needs to be described.

Frequently, the state vector is written in the *ket*-notation: $|\cdot\rangle$. The dual vector is written in the *bra*-notation: $\langle\cdot|$. By consequence, the inner product of two state vectors $|\phi\rangle$ and $|\psi\rangle$ is notated as $\langle\phi|\psi\rangle$.

Intuitively, one choice of orthonormal basis vectors of the state space of a physical system can be regarded as the "normal" states a system can be in. If we consider a couch that can be either red or blue, then we can intuitively identify the state space of this couch with a two-dimensional Hilbert space, in which the basis vectors are identified with the "red" and "blue" states, which, for now, we will respectively denote by the vectors $|r\rangle$ and $|b\rangle$. If the state vector of the couch is one of these state vectors or a multiple thereof, then we say that the couch is in a basis state. The postulate above, though, prescribes that the state vector of the couch can also be of the form $\alpha_1|r\rangle + \alpha_2|b\rangle$, where $\alpha_1, \alpha_2 \in \mathbb{C}$ and $|\alpha_1|^2 + |\alpha_2|^2 = 1$. So, if we take the state

vector of the couch to be $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|b\rangle$, then we could say that the couch is red and blue simultaneously. We then say that the couch is in a superposition state. In postulate three, we will see that performing a measurement of the color of the couch will force the couch to choose between the red and blue color.

Postulate 2: Evolution of the state of an isolated physical system
Consider an isolated physical system with state space H. The time evolution of the state vector of the system is described by a unitary operator acting on H. So, if we consider the evolution of the system between two time instances t_1 and t_2, then there exists a unitary operator U acting that H which solely depends on t_1 and t_2, such that the state vector at time instant t_1, denoted by $|\psi_1\rangle$, and the state vector at time instant t_2, denoted by $|\psi_2\rangle$, are related by $|\psi_2\rangle = U|\psi_1\rangle$.

Again, note that this postulate does not give us any information about this unitary operator. It therefore only describes the mathematical framework in which the evolution of a physical system needs to be described, but it does not specify this evolution itself.

Postulate 3: Measurement of the state of a physical system
Consider an isolated physical system. Associated with a measurement of this system is a set of measurement outcomes, denoted by O, and a set of corresponding measurement operators, $\{M_m : m \in O\}$, which are linear operators that act on the state space of the system, such that they satisfy the completeness relation:

$$\sum_{m \in O} M_m^* M_m = I$$

Let $|\psi\rangle$ be the state vector of the physical system. The probability that the measurement yields the outcome $m \in O$ is given by $\langle\psi|M_m^* M_m|\psi\rangle$. If the measurement yielded $m \in O$, the state after the measurement becomes:

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^* M_m|\psi\rangle}}$$

First of all, note that the probabilities of all outcomes sum to one, as a direct result of the completeness relation:

$$\sum_{m \in O} \langle\psi|M_m^* M_m|\psi\rangle = \left\langle \psi \middle| \sum_{m \in O} M_m^* M_m \middle| \psi \right\rangle = \langle\psi|I|\psi\rangle = \langle\psi|\psi\rangle = 1$$

Let's return to the example in which a couch can be described by a state vector $|\psi\rangle$ of the form $\alpha_1|r\rangle + \alpha_2|b\rangle$. Suppose that we want to measure the color of this couch. There are two possible outcomes, red and blue, so $O = \{r, b\}$. We define the measurement operators M_r and M_b to have the following effect for every $\alpha_1, \alpha_2 \in \mathbb{C}$:

$$M_r(\alpha_1|r\rangle + \alpha_2|b\rangle) = \alpha_1|r\rangle \qquad \text{and} \qquad M_b(\alpha_1|r\rangle + \alpha_2|b\rangle) = \alpha_2|b\rangle$$

Note that these measurement operators satisfy the completeness relation, because for every state vector $|\psi\rangle = \alpha_1|r\rangle + \alpha_2|b\rangle$, we have:

$$(M_r + M_b)|\psi\rangle = (M_r + M_b)(\alpha_1|r\rangle + \alpha_2|b\rangle) = M_r(\alpha_1|r\rangle + \alpha_2|b\rangle) + M_b(\alpha_1|r\rangle + \alpha_2|b\rangle) = \alpha_1|r\rangle + \alpha_2|b\rangle = |\psi\rangle$$

So, as this holds for any state vector $|\psi\rangle$, we have $M_r + M_b = I$. The probability of measuring the color red or blue is now given by:

$$\begin{aligned} P(r) &= \langle\psi|M_r^* M_r|\psi\rangle = \langle r|\alpha_1^*\alpha_1|r\rangle = \alpha_1^*\alpha_1\langle r|r\rangle = |\alpha_1|^2 \\ P(b) &= \langle\psi|M_b^* M_b|\psi\rangle = \langle b|\alpha_2^*\alpha_2|b\rangle = \alpha_2^*\alpha_2\langle b|b\rangle = |\alpha_2|^2 \end{aligned}$$

Hence, if the state vector of the couch is given by $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|b\rangle$, then the probability of measuring the color red equals $\frac{1}{2}$, as does the probability of measuring that the color of the couch is blue. Hence, the color of the couch prior to the measurement is ill-defined, as identical measurements possibly yield different results.

According to the postulate, the state after the measurement of the color of the couch depends on the outcome of the measurement. Suppose that the outcome was red, then the new state is given by:

$$\frac{M_r|\psi\rangle}{\sqrt{P(r)}} = \sqrt{2} \cdot \frac{1}{\sqrt{2}}|r\rangle = |r\rangle$$

Similarly, if the measurement yielded that the couch is blue, then the state of the couch after the measurement becomes $|b\rangle$. Hence, after the measurement, the state of the system becomes a basis state. We say that the state of the system collapses to a basis state upon performing a measurement.

There is some dispute about the status of this postulate. Arguably, the universe is an isolated physical system, hence its state vector evolves with a unitary operation, according to the second postulate of quantum mechanics. But this state vector completely describes the state of the observer and the observed system. Hence, one should be able to derive what happens to the observed system upon a measurement from postulate 2, but no considerable progress has been made in that direction. For the purpose of this text, however, this is not at all an issue, since we will always make a clear distinction between the observed system and the observer, and assume that the results of this postulate hold nevertheless.

Postulate 4: Composite systems
Consider n isolated physical systems, with state spaces V_1 through V_n. Then the state space of the composite system is given by $V_1 \otimes V_2 \otimes \cdots \otimes V_n$. If the systems individually have state vectors $v_i \in V_i$, then the state vector of the composite system is $v_1 \otimes v_2 \otimes \cdots \otimes v_n$.

Note that if, for example, we have two isolated physical systems with state spaces V and W, and we have $v_1, v_2 \in V$ and $w_1, w_2 \in W$ pairwise orthogonal state vectors, then we find that $\frac{1}{\sqrt{2}}(v_1 \otimes w_1 + v_2 \otimes w_2)$ is a state vector of the composite system. But, it cannot be written in the form $v \otimes w$ for some $v \in V$ and $w \in W$. If this is the case, we call the state of the composite system an *entangled state*. In this case, we can no longer refer to the state vector of the individual constituents of the composite system.

Suppose now that U_V is an operation on the state space V of some isolated physical system, and that it forms a composite system with another system that has state space W. Then the operation on the first system naturally leaves the second system unaltered, so the resulting operation on the composite system is given by $U_V \otimes I_W$, where I_W is the identity operation on W.

We can apply this new postulate to the example of the couch given above. Suppose that we have two couches that can both be red or blue. Then we denote the state space of such a couch by $V = \{\alpha_1|r\rangle + \alpha_2|b\rangle : \alpha_1, \alpha_2 \in \mathbb{C}\}$. According to postulate 4, the state space of the composite system of both couches is now given by $V^{\otimes 2} = V \otimes V$. Hence, a possible state of the couches is:

$$|\psi_1\rangle = \frac{1}{2}|r\rangle \otimes |r\rangle + \frac{1}{2}|r\rangle \otimes |b\rangle + \frac{1}{2}|b\rangle \otimes |r\rangle + \frac{1}{2}|b\rangle \otimes |b\rangle$$

Do note that this is not an entangled state, as according to the rules supplied in subsection 2.1.2, this state can be rewritten as:

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|b\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|b\rangle\right)$$

On the other hand, the following state is an entangled state:

$$|\psi_2\rangle = \frac{1}{\sqrt{3}}|r\rangle \otimes |r\rangle + \frac{1}{\sqrt{3}}|r\rangle \otimes |b\rangle + \frac{1}{\sqrt{3}}|b\rangle \otimes |b\rangle$$

Recall that if a composite system is in an entangled state, we cannot speak of the state of their individual constituents. So, if the two-couch system is in the state $|\psi_2\rangle$, we cannot speak of the state of the first couch, or the state of the second. We can only denote the state of the composite system.

By performing a measurement of the color on the first couch similar to the one we described above, the measurement operators M_r and M_b now generalize to operators on the state space of the composite system: $M_r \otimes I_V$ and $M_b \otimes I_V$. Hence, if the two-couch system is initially in the state $|\psi_2\rangle$, then we can calculate the following probability that the first couch is red. Remember that $|r\rangle \otimes |b\rangle$ and $|r\rangle \otimes |r\rangle$ are orthogonal.

$$P(\text{red}) = \left(\frac{1}{\sqrt{3}}\langle r| \otimes \langle r| + \frac{1}{\sqrt{3}}\langle r| \otimes \langle b|\right)\left(\frac{1}{\sqrt{3}}|r\rangle \otimes |r\rangle + \frac{1}{\sqrt{3}}|r\rangle \otimes |b\rangle\right) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

Similarly, we find that the probability of finding that the first couch is blue is equal to $\frac{1}{3}$. The resulting state of the composite system of couches after a measurement with outcome r can also be determined, in a similar way as before:

$$\begin{aligned}\frac{M_r \otimes I_V|\psi_2\rangle}{\sqrt{P(\text{red})}} &= \sqrt{\frac{3}{2}} \cdot \left(\frac{1}{\sqrt{3}}|r\rangle \otimes |r\rangle + \frac{1}{\sqrt{3}}|r\rangle \otimes |b\rangle\right) = \frac{1}{\sqrt{2}}|r\rangle \otimes |r\rangle + \frac{1}{\sqrt{2}}|r\rangle \otimes |b\rangle \\ &= |r\rangle \otimes \left(\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|b\rangle\right)\end{aligned}$$

Do note that this state is no longer entangled, as the state vector of the first couch has collapsed to a basis state.

3 Quantum circuits

The previous chapter presented a very general introduction into the realm of quantum mechanics. In this chapter, we will have a more specialized look into how the principles of quantum mechanics can be used to develop a quantum computer.

In the first section, we will introduce the concept of the qubit, which, like the bit in a classical computer, forms the basic building block of the quantum computer. Afterwards, we will have a look at how these qubits can be manipulated, using devices called quantum gates. Next, we will investigate how these quantum gates can be combined to form quantum circuits, which are comparable to programs on a classical computer. Finally, we will consider Deutsch's algorithm, which is an instructive exemplary quantum algorithm.

3.1 Qubits

In this section, we will have a look at the concept of a qubit. We will start with the notion of a classical bit, and then compare this concept to its quantum mechanical counterpart.

3.1.1 Bits vs. qubits

A classical computer manipulates bits, which is a shorthand for binary digits. As it is a very basic building block of computers, it makes sense to provide a formal definition.

Definition 3.1: Bit
A bit is a unit of information, that can hold one of two values.

Typically, the value of a bit is represented by the state of a physical system that can be in one of two distinct states. For example, one could take a door as a physical system. It can be in one of two distinct states, namely opened and closed. Therefore, a door can be used to represent a bit. Other examples include a bicycle that can be locked or unlocked, or a lamp that can be either on or off. A computer typically manipulates a large amount of small physical systems that can be either charged or uncharged.

Typically, the two values that the bit can have are labeled 0 and 1. Each of the states of the physical system is associated with one of these values. Do note that a binary digit is also either a 0 or a 1, hence the name.

We can now generalize this notion of a bit to the quantum realm. This brings about the following definition of a quantum bit.

Definition 3.2: Quantum bit
A quantum bit is a unit of information, that can hold a unit vector of a two-dimensional complex Hilbert space.

As we know from the first postulate of quantum mechanics, every isolated physical system has an associated Hilbert space, called the state space of the system. All systems that are associated with a two-dimensional Hilbert space, are capable of representing a quantum bit, or qubit for short.

An example of such a system is a photon, which consists of an oscillating electric and magnetic field. The polarization of the electric field can be represented by a Jones vector, which is a vector in $\mathbb{C}^2$. Hence, the polarization of a photon is associated with a vector from a two-dimensional Hilbert space, so a photon can be used to represent a qubit.

Photons might not be the best physical systems suited for developing a quantum computer. This naturally raises the question which physical systems are. This is a whole different topic — one we are not concerned

with in this text. From this point onwards, we will assume that we can always find the physical systems needed and manipulate these in the way we want.

The two standard basis vectors of the two-dimensional complex Hilbert space are often denoted by $|0\rangle$ and $|1\rangle$. This implies that the state of every qubit can be written as $\alpha|0\rangle+\beta|1\rangle$, with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2+|\beta|^2 = 1$. Moreover, from chapter 2 we know that every two-dimensional complex Hilbert space is isomorphic with $\mathbb{C}^2$, hence we can associate the two basis vectors $|0\rangle$ and $|1\rangle$ with the two standard basis vectors in $\mathbb{C}^2$. We now find that we can associate every state of a quantum system that represents a qubit with a vector $(\alpha\,\beta)^T$, with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Both notations will be used interchangeably in this text.

If a measurement is performed on a single qubit, generally the following measurement operators are taken (where we assume that the states are written as column vectors in this case).

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{and} \qquad M_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

These measurement operators obviously satisfy the completeness relation, $M_0 + M_1 = I$, and if a system is in the state $\psi = (\alpha\,\beta)^T$ prior to the measurement, the probability that the outcome of the measurement is m, is given by $P(m)$:

$$\begin{aligned} P(0) &= \psi^* M_0^* M_0 \psi = \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha^*\alpha = |\alpha|^2 \\ P(1) &= \psi^* M_1^* M_1 \psi = \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \beta^*\beta = |\beta|^2 \end{aligned}$$

Hence, we find that the probabilities of the outcomes sum to one, meaning that our measurement will return a value for sure. Moreover, we observe that if we multiply our initial state with a constant of modulus 1, it does not influence the results of the measurement. Therefore, we say that for example the states $|0\rangle$ and $-|0\rangle$ are equal, up to an unimportant constant.

Furthermore, if we look at the resulting state of the qubit after the measurement, then we see that it depends on the outcome of the measurement. If the measurement yielded a value of 0, then the resulting state vector becomes:

$$|\overline{\psi}_0\rangle = \frac{M_0\psi}{\sqrt{\psi^* M_0^* M_0 \psi}} = \frac{M_0\psi}{\sqrt{P(0)}} = \frac{\alpha}{|\alpha|}|0\rangle$$

Similarly, if the measurement yielded a value of 1, then the resulting state vector becomes:

$$|\overline{\psi}_1\rangle = \frac{M_1\psi}{\sqrt{\psi^* M_1^* M_1 \psi}} = \frac{M_1\psi}{\sqrt{P(1)}} = \frac{\beta}{|\beta|}|0\rangle$$

Hence, if the result of the measurement was m, then the resulting state is equal to $|m\rangle$, up to an unimportant constant. This complies with the collapse of the superposition states to basis states that we found in chapter 2.

3.1.2 Multiple qubits

In a classical computer, scaling up is very straightforward. Scaling up is done by using multiple bits separately, hence a 2-bit system is nothing more than two 1-bit systems packed together into one system.

In the quantum realm, there turns out to be a more interesting way of scaling up the amount of qubits. A 2-qubit system is a system that consists of two qubits that can be measured separately, but that can also hold the information stored in one unit vector in a *four*-dimensional Hilbert space. Similarly, an n-qubit

system is a system that contains the information worth a vector in a 2^n-dimensional Hilbert space, but where each qubit can be measured separately.

We now know from the previous chapter that in general, an n-qubit state space contains 2^n basis vectors, given by the tensor product of the basis states of a 1-qubit system: $|b_{n-1}\rangle \otimes |b_{n-2}\rangle \otimes \cdots \otimes |b_0\rangle$, where b_0 through b_{n-1} take values 0 or 1. There exist various ways to express these basis vectors. Suppose, namely, that $N = (b_{n-1} \dots b_0)_2$. Then the following notation all refer to the same computational basis state:

$$|b_{n-1}\rangle \otimes |b_{n-2}\rangle \otimes \cdots \otimes |b_0\rangle = |b_{n-1}\rangle|b_{n-2}\rangle \cdots |b_0\rangle = |b_{n-1}b_{n-2} \dots b_0\rangle = |N\rangle$$

So, for example, in a 3-qubit system, the states $|1\rangle \otimes |1\rangle \otimes |0\rangle$, $|1\rangle|1\rangle|0\rangle$, $|110\rangle$ and $|6\rangle$ all refer to the same basis vector of the state space of the 3-qubit system. The basis $\{|0\rangle, \dots, |2^n - 1\rangle\}$ of the state space of an n-qubit system is referred to as the computational basis.

As multiple qubit systems have a state space that is a Hilbert space with 2^n dimensions, its state space is isomorphic to $\mathbb{C}^{2^n}$. Generally, the computational basis states are associated with the standard basis vectors of $\mathbb{C}^{2^n}$. Hence, in a two-qubit system, a state $(0\,0\,1\,0)^T$ is associated with the state $|2\rangle = |10\rangle$, and the state vector $(1\,0\,0\,1)^T/\sqrt{2}$ is associated with the state $(|0\rangle + |3\rangle)/\sqrt{2} = (|00\rangle + |11\rangle)/\sqrt{2}$.

Do note that the state we just saw, $(|00\rangle + |11\rangle)/\sqrt{2}$, is an entangled state, as it cannot be written as a pure tensor of state vectors of the individual quantum bits. Hence we can no longer speak of the state vector of the first qubit, or the state vector of the second qubit. We can only talk about the state vector of the two-qubit system. This is a very fundamental difference between multiple qubit systems and systems that are comprised of multiple classical bits, and it is what lies at the heart of the speedup that can be achieved with quantum computers, if it is used in an ingenious way.

Finally, let's have a look at what happens when a measurement is performed on a 2-qubit system. Suppose a system starts out in the state $(|00\rangle + |01\rangle + |11\rangle)/\sqrt{3}$ and that the first qubit is measured. The column vector representation is in this case somewhat easier to work with, so the state of the multiple qubit system is $(1\,1\,0\,1)^T/\sqrt{3}$. The measurement operators corresponding to a measurement of the first qubit are now augmented with an identity matrix, such that the new measurement operators that act on the state space of the composite system become (where I is the identity operator that acts on the state space of the second qubit):

$$\begin{aligned} M_0^{(1)} &= M_0 \otimes I \\ M_1^{(1)} &= M_1 \otimes I \end{aligned}$$

Hence, by checking its action on the computational basis states, the matrix representations of these operators with respect to the computational basis can be determined. (Strictly speaking, it is incorrect to denote the matrix representation of an operator by the same symbol, as the operator itself, but it is common practice in this field.)

$$M_0^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_1^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calculating the probabilities now yields $P(0) = 2/3$ and $P(1) = 1/3$. The resulting state after a measurement of 0 is $(|00\rangle + |01\rangle)/\sqrt{2}$, and similarly after a measurement of 1, the system collapses to the state $|11\rangle$, which the reader is encouraged to check for him-/herself.

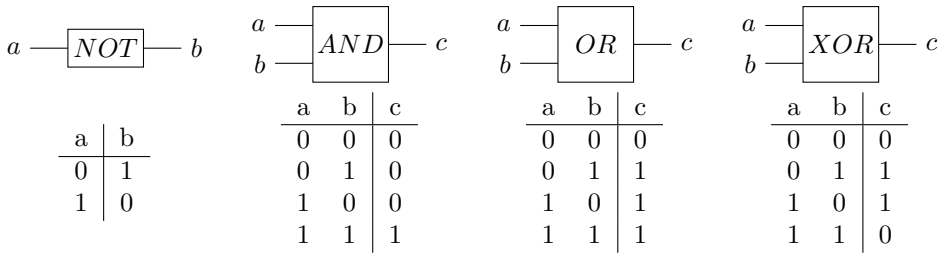
3.2 Quantum gates

In the previous section, we have seen in what states single and multiple qubit systems can be. This section will be covering how we can manipulate the states of these systems. First of all, we will have a look at how single qubit states can be manipulated, and afterwards, we will look at how the state of a multiple qubit system can be altered.

3.2.1 Logic gates vs. quantum gates

Before we investigate the manipulation of single qubit states, let's first have a look at the operations we can perform on classical bits. On one classical bit, there is really one operation we can perform, apart from doing nothing, and that action is flipping the bit, i.e. mapping a 0 to a 1, and a 1 to a 0. This is accomplished by a NOT operation, which can be viewed as a gate as it requires a bit to pass through, and flips its value in the process.

Such gates, that require classical input bits and determines the output according to some procedure, are called logic gates. Other well known-logic gates are the AND, OR and XOR gates. These all require two input bits and have only one classical bit as output. They can be graphically represented by indicating the input and output bits as horizontal lines that are attached to the gate on the left and right side, respectively. A few examples are provided in figure 3.1.



a	b
0	1
1	0

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

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

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

Figure 3.1: Some commonly used logic gates. The horizontal lines denote individual bits. The ones shown on the left of the gates are input bits, whereas the ones on the right represent the output of the gate. Their behavior is described by the truth tables supplied below the gates themselves.

Quantum gates are similar, in the sense that they require a number of input qubits and return a number of output qubits. The second postulate of quantum mechanics now tells us, though, that we can only perform unitary operations on any physical system. This means that every operation we do must be invertible, hence the number of qubits entering the quantum gate must equal the number of qubits exiting the quantum gate. Moreover, the matrix representation of the quantum gate with respect to the computational basis must be a unitary matrix.

At first sight, this constrains the amount of possible quantum gates significantly. For example, the classical AND gate, which requires two input bits and only one output bit, cannot be directly simulated using a quantum gate. From this, we already see that developing quantum algorithms is a totally different business from finding classical algorithms.

Graphically, quantum gates are depicted in a similar way as logic gates. The matrix representation of the operator is indicated at the center of the graphical representation. On the left and right hand side, horizontal lines enter and exit this gate, which represent the input and output qubits.

3.2.2 Single qubit gates

Quantum gates that only manipulate the state of one qubit are called single qubit gates. Some important single qubit gates are listed in table 3.1, where their matrix representation is relative to the computational basis.

X	$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Y	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Z	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
H	$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
S	$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
T	$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$

Table 3.1: Commonly used single qubit gates. Note that in the graphical representation of the quantum gate, the letter that denotes the matrix representation with respect to the computational basis is used. The T gate is also referred to as the $\pi/8$ gate, for historical reasons.

All single qubit gates listed in this table have their own special properties. For example, X is somewhat like the quantum mechanical counterpart of the NOT gate, as it maps the state $|0\rangle$ to $|1\rangle$ and vice versa. Moreover, the matrix representations of the X, Y and Z gates are called the Pauli matrices, as we will see in chapter 4. H, also known as the *Hadamard gate*, creates superpositions of states from basis states: $|0\rangle$ is mapped to $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle$ is mapped to $(|0\rangle - |1\rangle)/\sqrt{2}$. As $H^2 = I$, the converse is also true: the superposition states referred to above are mapped back to the basis states. Finally, the T gate is also called the $\pi/8$ gate, for historical reasons.

3.2.3 Multiple qubit gates

Multiple qubit gates are quantum gates that alter the state of multiple qubit systems. As noted before, a quantum gate must perform a unitary operation on a set of qubits, and the number of input qubits must equal the number of output qubits, so in general, every multiple qubit gate can be represented similar to the ones depicted in figure 3.2.

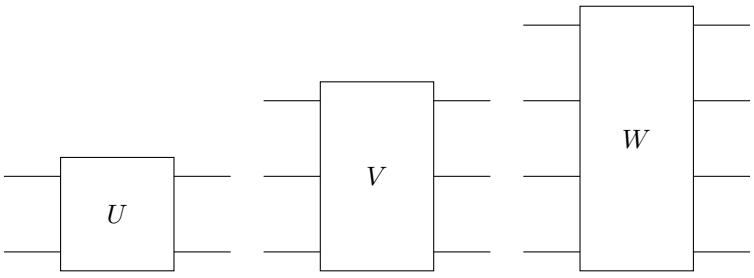


Figure 3.2: Most general representation of 2- 3- and 4-qubit quantum gates, in this case denoted by U, V and W respectively. Here, U, V and W generally denote the matrix representation of the operation with respect to the computational basis.

There are a number of multiple qubit gates, though, that are used in such common practice that they deserve special mention. They have also been given a slightly different graphical representation.

First of all, suppose we have a single qubit gate, with matrix representation U and suppose that we only want to execute the corresponding operation if the state of another qubit is $|1\rangle$. This last qubit is called the control qubit, and the qubit which U acts on is referred to as the target qubit. This 2-qubit operation is called a conditional single qubit operation. Its graphical representation can be found in figure 3.3 on the left hand side. Its counterpart, where the operation U is only performed when the state of the control qubit is $|0\rangle$, is displayed in figure 3.3 on the right.

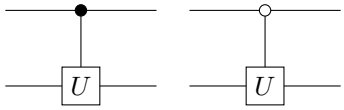


Figure 3.3: Controlled single qubit operations. The top qubit is referred to as the control qubit, and the bottom one as the target qubit. In the left figure, the operation U is only executed when the top qubit is in state $|1\rangle$, and the target qubit is left unaltered otherwize. Similarly, in the right quantum gate, the operation U is only performed when the control qubit is in state $|0\rangle$.

This notion of conditional operation can easily be expanded to quantum gates with more control and target qubits. An example is given in figure 3.4.

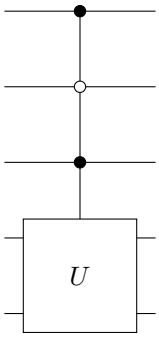


Figure 3.4: A conditional quantum gate with 3 control qubits and 2 target qubits. The 2-qubit operation U is only executed when the first three qubits are in state $|1\rangle \otimes |0\rangle \otimes |1\rangle$.

But, we have yet to consider the most important conditional operations. One of these is the controlled NOT operations, or CNOT. It operates on one control qubit and one target qubit, and performs the X operation on the target qubit if the control qubit is set. Its graphical representation can be found in figure 3.5.

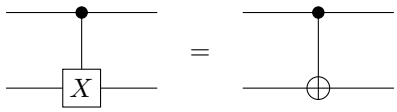


Figure 3.5: The graphical representation of the CNOT gate. On the right, the special notation is used, whereas the representation on the left side is completely identical.

Similarly, one can of course apply the X gates on one target qubit using more than one control qubits. Then, one obtains the Toffoli gate, as shown in figure 3.6.

The importance of the CNOT and Toffoli gates is its use in the implementation of conditional executions, just like an if statement in an average classical programming language. The target qubit is only flipped *if*

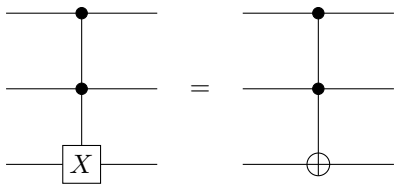


Figure 3.6: The graphical representation of the Toffoli gate. On the right, the special notation is shown, which is identical to the more standard notation on the left hand side.

the control qubits are in state $|1\rangle$. We will see a number of applications in the next chapter, as well as in chapter 5.

As a final remark, we have a look at the matrix representation of the CNOT gate. We observe that the states $|00\rangle$ and $|01\rangle$ are left unaltered, as the control qubit is in state $|0\rangle$ in that case. The other two basis states, $|10\rangle$ and $|11\rangle$ are swapped by the CNOT gate. Hence, we obtain the following matrix representation. Recall that this matrix representation is relative to the computational basis, which in a 2-qubit system is equal to $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Obtaining the matrix representation of the other multiple qubit gates is done in a similar way. The matrix representation of the CNOT gate in particular will become very important in chapter 4.

3.3 Quantum circuits

In general, a quantum circuit is nothing more than a collection of quantum gates, executed in a prescribed order. A program is nothing more than a sequence of instructions that the classical computer needs to execute, so a quantum circuit is in some respects comparable to a program on a classical computer.

The graphical representation of a quantum circuit shows all qubits that are affected as horizontal lines, and the quantum gates are placed on these lines, not entirely unlike musical notes on a stave. The order in which the quantum gates are executed is from left to right. Optionally, the state of the qubits before the quantum circuit is executed is shown on the left side of the horizontal lines, and the state the qubits are in after execution is shown on the right side.

3.3.1 Swap circuit

As an example, we consider the swap circuit. Suppose we have a 2-qubit system, and we want to swap the state of the first qubit with the state of the second qubit. Then we want to have a unitary operation that takes the basis state $|01\rangle$ to $|10\rangle$ and $|10\rangle$ to $|01\rangle$, and leaves $|00\rangle$ and $|11\rangle$ unaltered. The corresponding matrix representation is given below on the left hand side:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

On the right hand side, we see that we can obtain the matrix representation needed to swap the state of two qubits by multiplying the matrix representation of three CNOT gates, where the first and the last take the

second qubit as the target qubit, and the second CNOT gate takes the first qubit as the target qubit. The resulting circuit is shown in figure 3.7.

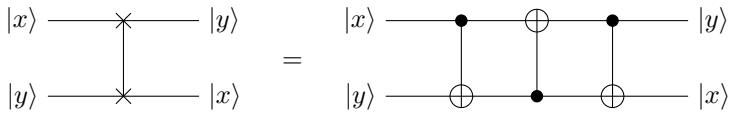


Figure 3.7: The swap circuit. The state of the first and second qubit are swapped by applying three CNOT gates, as shown in the right figure. In the left figure, the shorthand notation of swapping is shown.

3.3.2 Registers

In a classical computer, generally bits are grouped to form bytes, which consist of 8 bits. These bytes are then grouped again to form up to 64-bit systems, in order to improve the capacity to store large integers. Such a collection of bits is called a register.

In quantum computing, qubits can be grouped in a similar way as well. For example, n qubits can be grouped to form a register. In the graphical representation, this is denoted by an oblique line intersecting the horizontal lines that denote the qubits. The number of qubits that make up the register is placed above this oblique line. Sometimes it is necessary to refer to the individual qubits of the register, then the horizontal line that denotes the register splits into its constituent qubits.

The circuit that swaps the contents of two equally sized registers of n qubits is shown in figure 3.8. This circuit will be useful in chapter 5.

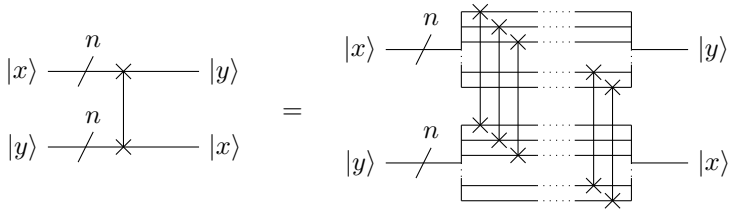


Figure 3.8: Implementation of the swap operation of two n-qubit registers. The swap operation that swaps the state of two qubits is applied n times.

3.4 Example: Deutsch's algorithm

As a final remark to quantum circuits, we consider an instructive example, known as Deutsch's algorithm. It is particularly interesting because it is currently experimentally feasible to actually execute this algorithm on a real quantum computer. The details of this experiment can be found in appendix C.

Suppose that one has a function $f : \{0, 1\} \to \{0, 1\}$. There are four such possible functions, two of which are constant (i.e. $f(0) = f(1)$), and two of which are not constant. Suppose now that one would like to determine whether f is constant. On a classical computer, one would first calculate $f(0)$ and $f(1)$, and then compare the results using an XOR gate. Thus, the classical algorithm might be graphically represented like in figure 3.9. This implementation requires two evaluations of the function f.

We will now try to find a quantum circuit that accomplishes the same task as the classical program. For this, we first of all specify the way f is evaluated. Note that as we do not in advance know the specific properties of f, we cannot ensure that its operation is reversible. So, we cannot implement f using a single

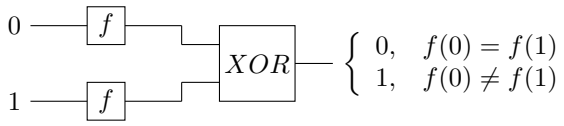


Figure 3.9: Classical implementation of a program to determine whether the function f is constant. This program requires two evaluations of f.

qubit gate, hence we have to resort to a 2-qubit gate U_f at the very least. The gate U_f is shown in figure 3.10.

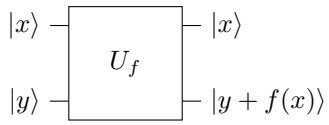


Figure 3.10: Implementation of the evaluation of f. We see that if y is 0, then the second qubit will hold the value of $f(x)$ after execution of this gate. Otherwize, it will hold the inverse of $f(x)$, as the addition is done mudulo 2. This is to ensure that the gate is unitary for all functions f.

Now, we implement this gate U_f into the quantum circuit shown in figure 3.11.

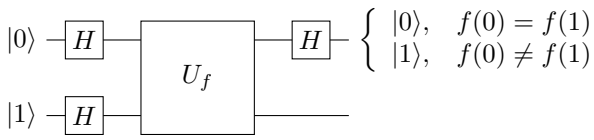


Figure 3.11: Implementation of the Deutsch algorithm. Note that measuring the first qubit at the conclusion of the execution of the algorithm will reveal if f is constant. Also note that only one execution of U_f is needed.

If we trace the steps of the algorithm, we observe that the qubits start out in the state $|0\rangle \otimes |1\rangle$. Applying the Hadamard gates yields the following state (recall that $|a\rangle \otimes |b\rangle$ is denoted by $|ab\rangle$, with $a, b \in \{0, 1\}$):

$$\frac{1}{2}\left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle - |1\rangle\right) = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$$

Applying the U_f gate yields:

$$\frac{1}{2}|0\ f(0)\rangle - \frac{1}{2}|0\ 1{+}f(0)\rangle + \frac{1}{2}|1\ f(1)\rangle - \frac{1}{2}|1\ 1{+}f(1)\rangle$$

Now, suppose that f is constant. Then we can write this state as:

$$\frac{1}{2}\left(|0\rangle + |1\rangle\right) \otimes \left(|f(0)\rangle - |1{+}f(0)\rangle\right)$$

So, applying the Hadamard gate on the first qubit now yields $|0\rangle$. On the other hand, if the f is not constant, we find that the state after U_f can be rewritten as:

$$\frac{1}{2}\left(|0\rangle - |1\rangle\right) \otimes \left(|f(0)\rangle - |f(1)\rangle\right)$$

Hence, applying the Hadamard gate on the first qubit yields a state $|1\rangle$. So, we find that indeed, the circuit behaves as expected.

Interestingly, the gate U_f was only executed once in the execution of Deutsch's algorithm. So, if the evaluation of f turns out to be a very computationally heavy procedure, then Deutsch's algorithm will provide a way to determine if f is constant that is approximately twice as fast.

To summarize, the idea of this algorithm is to first load the system in a superposition state, and then to apply the U_f gate to calculate both $f(0)$ and $f(1)$ at the same time. Then, using some clever techniques, one hopes to be able to retrieve some of the global properties of the output of f. This technique is called *quantum parallellism*. It is also used in Shor's algorithm, which is covered in chapter 5. The experimental realization of this quantum circuit is covered in appendix C.

4 Universality of the Controlled NOT, Hadamard and $\pi/8$ gate

In the previous chapter, we have seen that every unitary matrix acting on a Hilbert space with 2^n dimensions, for some $n \in \mathbb{N}$, is associated with a unique quantum gate acting on the state space of n qubits with respect to the computational basis. Thus, there are an infinite number of gates that could be used to create quantum circuits with. This, though, is not experimentally realizable. It would be much more convenient to have a discrete set of quantum gates that, successively, can approximate any quantum circuit to within arbitrary accuracy. A set of quantum gates with that property is called a universal set. An example of such a set consists of the Controlled NOT, the Hadamard and the $\pi/8$ gate, and the universal property of this set is proven in this chapter.

In order to construct this proof, the problem is first of all reduced to showing that the Hadamard and $\pi/8$ gates are universal for single qubit gates. This is done in two steps, in section 4.1 and 4.2, respectively. The last step is proven using the Bloch sphere and its properties, in section 4.3.

4.1 Reduction to two-level unitary operators

In this section, we will prove that every unitary operator can be written as the product of two-level unitary operators. First of all, the definition of this two-level unitary operator will be given (4.1.1), after which the actual proof will be delivered (4.1.2).

4.1.1 Two-level unitary matrices and operators

Central to the first part of the proof is the concept of two-level unitary matrices and two-level unitary operators. Therefore, we first provide their definitions.

Definition 4.1: Two-level unitary matrix
A two-level unitary matrix is a unitary matrix that is identical to the identity matrix, up to at most a 2×2 submatrix.

For example, the following matrix is unitary. Moreover, it is identical to the identity matrix, except for the 2×2 submatrix formed by taking the 2nd and 4th rows and 2nd and 4th columns. Hence, this matrix is two-level unitary.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Note that every 2×2 unitary matrix is trivially two-level unitary.

Definition 4.2: Two-level operator with respect to the computational basis
A two-level unitary operator with respect to the computational basis is an operator whose matrix representation with respect to the computational basis is given by a two-level unitary matrix.

Do note that the matrix representation of the CNOT gate with respect to the computational basis is two-level unitary, so the CNOT gate is a two-level unitary operator. From now on, all matrices are understood to be with respect to the computational basis, and this will not be explicitly noted on every occasion.

The following lemma will prove to be very useful shortly.

Lemma 4.3: Inverse of two-level unitary matrices
Suppose U is a two-level unitary matrix. Then its inverse is two-level unitary as well.

Proof: Let U be an arbitrary two-level unitary matrix with dimensions $n \times n$. First of all, note that as U is unitary, $U^{-1} = U^*$, so its inverse is unitary as well. Moreover, if U differs from the identity matrix by only a 2×2 submatrix, then so does $U^* = U^{-1}$, hence we find that U^{-1} is a two-level unitary matrix as well. $\square$

This lemma implies trivially that the inverse of every two-level unitary operator is a two-level unitary operator as well.

4.1.2 Reduction of a quantum gate to a product of two-level unitary operators

In this subsection, we will prove that every quantum gate can be written as a product of two-level unitary operators. First of all, it will be proven that one can zero out the first column of the matrix representation, up to one entry using two-level unitary matrices. Next, it will be proven that this enables us to apply induction to the size of the matrix representation of the quantum gate.

Lemma 4.4:
Suppose U is a unitary matrix with dimensions $n \times n$. Then one can find two-level unitary matrices $U_1, \ldots, U_m$ such that $(\prod_{i=1}^{m} U_i)\, U$ has only one non-zero entry in the first column.

Proof: Suppose all entries in the first column of U are zero. Then $\det(U) = 0$, hence U is a not a unitary matrix, so we reach a contradiction right away. Thus, we can safely assume that U has at least one non-zero entry in the first column.

Suppose now that U has precisely one non-zero entry in the first column. Then there is nothing to prove, hence this provides the basis for mathematical induction.

Now suppose that for a unitary matrix U with k non-zero entries in the first column, we can find two-level unitary matrices U_1 through U_m such that $U_1 U_2 \cdots U_m U$ has only one non-zero entry in the first column. This is our induction hypothesis. If we then have a unitary matrix V with $k+1$ non-zero entries in the first column, we can find two indices i and j, such that $i \neq j$ and $V_{i1}, V_{j1} \neq 0$. Now, define the following $n \times n$ matrix W, which is equal to the identity matrix, except for the following entries:

$$W_{ii} = \frac{V_{i1}}{\sqrt{|V_{i1}|^2 + |V_{j1}|^2}} \qquad W_{ij} = \frac{V_{j1}}{\sqrt{|V_{i1}|^2 + |V_{j1}|^2}}$$
$$W_{ji} = \frac{-V_{j1}}{\sqrt{|V_{i1}|^2 + |V_{j1}|^2}} \qquad W_{jj} = \frac{V_{i1}}{\sqrt{|V_{i1}|^2 + |V_{j1}|^2}}$$

Note that W is a two-level unitary matrix. Also note that if we look at WV, we zero out one entry in the first column:

$$(WV)_{j1} = W_{ji} V_{i1} + W_{jj} V_{j1} = \frac{-V_{j1} V_{i1} + V_{i1} V_{j1}}{\sqrt{|V_{i1}|^2 + |V_{j1}|^2}} = 0$$

So, now we are left with a matrix, WV, that has k non-zero elements in the first column. But then, by our induction hypothesis, we can find matrices U_1 through U_m such that: $U_1 U_2 \cdots U_m WV$ has only 1 non-zero entry in the first column. Hence, by induction to the number of non-zero elements in the first column of the matrix U, we find that for any $n \times n$ matrix U, we can find two-level unitary matrices U_1 through U_m such that $(\prod_{i=1}^{m} U_i)\, U$ has only one non-zero element in the first column. $\square$

Lemma 4.5:
Suppose U is a unitary matrix with dimensions $n \times n$, where $n \geq 2$. Then we can find two-level unitary matrices U_1 through U_m such that:

$$\left(\prod_{i=1}^{m} U_i\right) U = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & W & \\ 0 & & & \end{array}\right]$$

where W is a unitary matrix with dimensions $(n-1) \times (n-1)$.

Proof: Let U be an arbitrary unitary matrix with dimensions $n \times n$, where $n \geq 2$. Then, by the previous lemma, we can find matrices V_1 through V_l such that $V = \left(\prod_{i=1}^{l} V_i\right) U$ has only one non-zero entry in the first column. If this non-zero entry is in the first row, then we define $m = l$ and $U_i = V_i$, for all i.

If the non-zero element, though, is in the jth row of V, where $j \neq 1$, then we define the following $n \times n$ matrix, A, that is identical to the identity matrix up to the following elements:

$$\begin{array}{ll} A_{11} = 0 & A_{1j} = 1 \\ A_{j1} = 1 & A_{jj} = 0 \end{array}$$

Now, A is two-level unitary and one observes:

$$\begin{array}{rcl} (AV)_{11} & = & A_{11}V_{11} + A_{1j}V_{j1} = V_{j1} \neq 0 \\ (AV)_{j1} & = & A_{j1}V_{11} + A_{jj}V_{j1} = V_{11} = 0 \end{array}$$

The 1st and jth rows are swapped, hence AV has a non-zero matrix element in the top left corner. Now, we find that $A\left(\prod_{i=1}^{l} V_i\right) U$ has one non-zero element in the first column, and it is in the first row as well. So, we define $m = l + 1$, $U_{i+1} = V_i$ and $U_1 = A$. So, in general, we have now found matrices U_1 through U_m such that $\left(\prod_{i=1}^{m} U_i\right) U$ has a non-zero element in the top left corner, and zeros in the rest of the first column.

Let's define $T = \left(\prod_{i=1}^{m} U_i\right) U$. Then T is a product of unitary matrices, hence T itself is unitary. Therefore, all columns of T are unit vectors. As all matrix elements of the first column of T are 0, except for the first entry, we find $|T_{11}|^2 = 1$. But as T is unitary, also all row vectors are unit vectors, hence $\sum_{i=1}^{n} |T_{1i}|^2 = 1$. Applying $|T_{11}|^2 = 1$, we obtain:

$$\sum_{i=2}^{n} |T_{1i}|^2 = 0 \Rightarrow \forall 2 \leq i \leq n, T_{1i} = 0$$

Hence, the only non-zero element of T in the first row is located in the top left corner as well.

Now, let's define X as the identity matrix, except for the top-left matrix element: $X_{11} = T_{11}^{-1}$. Then as $|T_{11}^{-1}| = |T_{11}|^{-1} = 1^{-1} = 1$, we find that X is a two-level unitary matrix. So now, we can find a unitary $(n-1) \times (n-1)$ matrix W such that:

$$X\left(\prod_{i=1}^{m} U_i\right) U = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & W & \\ 0 & & & \end{array}\right]$$

By noting that X can easily be appended to the list of two-level unitary matrices U_1 through U_m that are denoted inside the parentheses, we find that we have proven this lemma. □

Theorem 4.6: Decomposition of unitary matrices into two-level unitary matrices
Every unitary matrix U can be written as a product of two-level unitary matrices.

Proof: Let U be an arbitrary unitary matrix with dimensions $n \times n$. If $n = 1$ or $n = 2$, then U is itself two-level unitary, so there is nothing to prove. This provides the basis of induction.

Now suppose that every unitary matrix of dimension $k \times k$ can be written as a product of two-level unitary matrices U_1 through U_m with dimensions $k \times k$. This is our induction hypothesis. Now suppose that U has dimensions $(k+1) \times (k+1)$. Then by the preceding lemma, we can find two-level unitary operators V_1 through V_l such that:

$$\left(\prod_{i=1}^{l} V_i\right) U = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & W & \\ 0 & & & \end{array}\right]$$

where W is a unitary $k \times k$ matrix. Hence, by the induction hypothesis, we can now find $k \times k$ unitary matrices U_1 through U_m such that:

$$W = \prod_{i=1}^{m} U_i$$

Now, we define the following notation. For every unitary matrix A, we define:

$$A^{\dagger} = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & A & \\ 0 & & & \end{array}\right]$$

Note that $A^{\dagger}$ is again unitary. We find:

$$\left(\prod_{i=1}^{l} V_i\right) U = W^{\dagger} = \prod_{i=1}^{m} U_i^{\dagger}$$

So, moving all V_i's to the other side, we obtain:

$$U = V_l^{-1} V_{l-1}^{-1} \cdots V_2^{-1} V_1^{-1} \prod_{i=1}^{m} U_i^{\dagger}$$

As all V_i's and $U_i^{\dagger}$'s are two-level unitary operators and inverses of two-level unitary matrices are again two-level unitary matrices, we have written U as a product of two-level unitary matrices. Hence, we have proven the theorem. □

We are now finally able to prove what is the goal of this section:

Theorem 4.7: Decomposition of quantum gates into two-level unitary operators
Every quantum gate can be written as a product of two-level unitary operators.

Proof: Take an arbitrary quantum gate, and let U be its matrix representation with respect to the computational basis. Then we can find two-level unitary matrices U_1 through U_m such that

$$U = \prod_{i=1}^{m} U_i$$

Associated to every two-level unitary matrix U_i is a two-level unitary operator, hence we have found a way to write every quantum gate as the product of two-level unitary operators. □

So, in this section, we have proven that we can write any quantum gate as a product of two-level unitary operators. Hence, it is sufficient to find a way to implement two-level unitary operators using the CNOT, Hadamard and $\pi/8$ gates, in order to approximate any quantum circuit.

4.2 Reduction to (controlled) single qubit gates

In the previous section, we have seen that we can write every quantum gate as a product of two-level unitary operators. In this section, we will exploit this result to show that we can write every quantum gate as a product of single qubit gates and controlled single qubit gates. To this end, we will systematically substitute complicated quantum gates by circuits that consist of less intricate quantum gates.

4.2.1 Swapping

First of all, we will have a better look at what happens when a two-level unitary operator is used. As the matrix representation with respect to the computational basis is equal to the identity matrix, except for a 2×2 submatrix, we find that only two computational basis states are affected by the operator.

Suppose that these basis states are $|001\rangle$ and $|011\rangle$. Then one can apply the single qubit operator U given by the 2×2 submatrix to the second qubit, conditional on the first qubit being 0 and the third qubit being 1, as shown in figure 4.1.

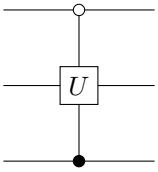


Figure 4.1: This operation only affects the basis states $|001\rangle$ and $|011\rangle$.

Let's now consider a more complicated example, where the basis states $|000\rangle$ and $|011\rangle$ are affected. The idea is to first swap $|000\rangle$ with $|001\rangle$, such that we obtain the same problem as in the previous paragraph, and then undo the swapping. This in indicated in figure 4.2.

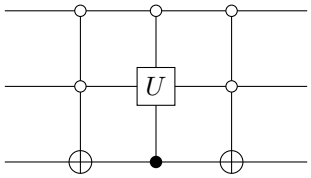


Figure 4.2: This sequence of operations only affects the basis states $|000\rangle$ and $|011\rangle$.

In general, the idea is to first use successive swapping of basis states so that the two basis states that are affected only differ by 1 bit. Then, the 2×2 submatrix can be executed by a single qubit operation, conditional on the value of all other qubits. After that, the swapping is undone.

4.2.2 Implementing conditional operations with multiple control qubits

According to the previous arguments, all that is left to do is find a way to implement single qubit operations that are controlled by an arbitrary number of qubits, using the CNOT, Hadamard and $\pi/8$ gate. In this subsection, this problem will be reduced even further.

First of all, we note that we can restrict ourselves to quantum gates that check if all control qubits are set, as the ones that check for a cleared control qubit can be converted to these by adding two X-gates, as is shown in figure 4.3.

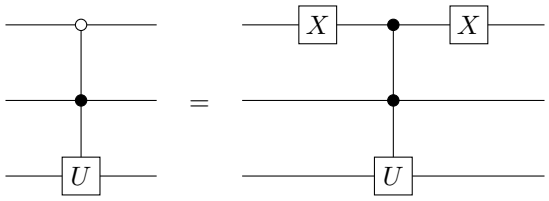


Figure 4.3: All controlled gates that check for a cleared qubit can be replaced by ones that check for set qubits, with the addition of two X single qubit gates.

So, all that is left to do is finding a way to implement quantum gates like the one shown in figure 4.4, where the number of control qubits is arbitrary. In order to do so, we will need to use the spectral decomposition of a unitary matrix. For completeness, the statement of the theorem is provided below.

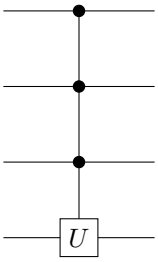


Figure 4.4: A single qubit gate with a number of control qubits.

Theorem 4.8: Spectral decomposition of unitary matrices
Suppose U is a unitary matrix with eigenvalues λ_1 through λ_n, and proj_{λ_1} through proj_{λ_n} are the projection matrices on the eigenspaces associated with these eigenvalues. Then, U can be written as:

$$U = \sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}$$

This allows us to define the square root of a unitary matrix:

Definition 4.9: Square root of a unitary matrix
Let U be a unitary matrix, with the following spectral decomposition:

$$U = \sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}$$

Then the square root of U is defined by:

$$\sqrt{U} = \sum_{i=1}^{n} \sqrt{\lambda_i} \text{proj}_{\lambda_i}$$

Here the branch cut used to calculate the square root of the complex λ_i is the negative real axis in the complex plane, such that the resulting value is the principle square root.

Note that as the eigenspaces of different eigenvalues of U are orthogonal, we find that $(\sqrt{U})^2 = U$, as we would expect. Furthermore, as $|\sqrt{\lambda_i}| = 1$ for all λ_i, we find $(\sqrt{\lambda_i})^* = \frac{1}{\sqrt{\lambda_i}}$. We use that a projection matrix is Hermitian, the eigenspaces of different eigenvalues are orthogonal and the square of projection matrices yield the original projection matrix, to find:

$$\begin{aligned}
\sqrt{U}\sqrt{U}^* &= \left(\sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}\right)\left(\sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}\right)^* = \sum_{i=1}^{n}\sum_{j=1}^{n} \sqrt{\lambda_i}\sqrt{\lambda_j}^* \text{proj}_{\lambda_i}\text{proj}^*_{\lambda_j} \\
&= \sum_{i=1}^{n} \sqrt{\frac{\lambda_i}{\lambda_i}} \text{proj}^2_{\lambda_i} = \sum_{i=1}^{n} \text{proj}_{\lambda_i} = I \\
\sqrt{U}^*\sqrt{U} &= \left(\sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}\right)^*\left(\sum_{i=1}^{n} \lambda_i \text{proj}_{\lambda_i}\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \sqrt{\lambda_i}^* \sqrt{\lambda_j} \text{proj}^*_{\lambda_i}\text{proj}_{\lambda_j} \\
&= \sum_{i=1}^{n} \sqrt{\frac{\lambda_i}{\lambda_i}} \text{proj}^2_{\lambda_i} = \sum_{i=1}^{n} \text{proj}_{\lambda_i} = I
\end{aligned}$$

So, we find that $\sqrt{U}$ is again a unitary matrix, for any unitary matrix U.

Now, suppose we would like to implement a single qubit operation with matrix representation U, controlled by two control qubits, like the one shown on the left hand side in figure 4.5. Then, we define $V = \sqrt{U}$, and implement it with the circuit shown on the right in figure 4.5.

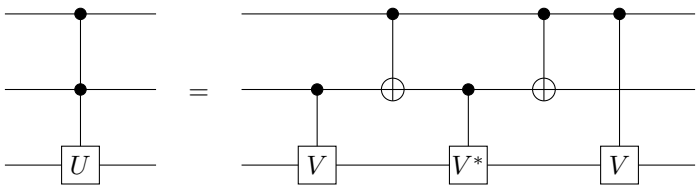


Figure 4.5: A single qubit gate with two control qubits is shown on the left, and an implementation using only single qubit gates with one control qubit is shown on the right. Here, $V = \sqrt{U}$.

By checking all four basis states of the control qubits, one can easily verify that this implementation executes an operation V^2 on the target qubit, only if both control qubits are set to 1. Otherwize, the target qubit is left unaltered.

This approach can be scaled up easily. Consider, for example, the quantum gate shown on the left side in figure 4.6. This gate can be implemented using the an analogous circuit used in figure 4.5, as can be seen on the right hand side of figure 4.6, where V is again defined as $\sqrt{U}$.

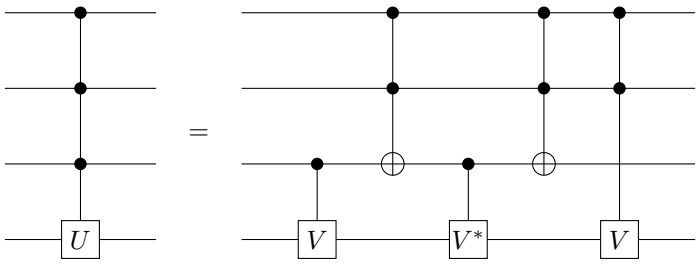


Figure 4.6: A single qubit gate with three control qubits is shown on the left, and an implementation using qubit gates with at most 2 control qubits is shown on the right. Here, $V = \sqrt{U}$.

We see now that the single qubit operation that is conditional under 3 control qubits, can be implemented using 3 single qubit quantum operations that have 2 control qubits. Inductively, one can now implement

any single qubit operation that is conditional under an arbitrary number of control qubits, using only single qubit operations that are conditional under 1 qubit, which we refer to as controlled single qubit gates.

So, the problem of showing that the CNOT, Hadamard and $\pi/8$ gates are universal, has been reduced to showing that they can be used to implement single qubit gates and controlled single qubit gates.

4.3 Approximation of (controlled) single qubit gates by Hadamard and $\pi/8$ gates

In the previous sections, we have seen that we can implement any arbitrary quantum gate, using only gates of the form shown in figure 4.7.

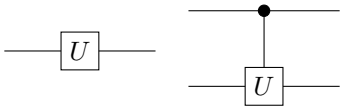


Figure 4.7: A single qubit gate and a controlled single qubit gate, with one control qubit.

This section will be devoted to showing that these quantum gates can be implemented to arbitrary accuracy, using only the CNOT, Hadamard and $\pi/8$ gates.

In order to achieve this, we will first need to introduce a very important concept to help us visualize what is happening when a single qubit operation is applied to one qubit. This concept is called the Bloch sphere, and is the main subject of subsection 4.3.1. After that, we will show in section 4.3.2 that any controlled single qubit gate can be implemented using the CNOT gate and other single qubit gates. Next, in section 4.3.3, we will introduce the concept of approximating a quantum gate, which we will use in section 4.3.4 to prove that the Hadamard and $\pi/8$ gates can be used to approximate any arbitrary single qubit gate.

4.3.1 Bloch sphere

4.3.1.1 Representation of the state of a single qubit using the Bloch sphere

Suppose $|\psi\rangle$ is the state of a single qubit. Then, we have seen in chapter 3 that we can write $|\psi\rangle$ as $a|0\rangle + b|1\rangle$, where $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$. Using the polar notation of complex numbers, this can be rewritten as:

$$|\psi\rangle = e^{i\alpha} r_a |0\rangle + e^{i\beta} r_b |1\rangle$$

where $r_a, r_b \geq 0$ and $\alpha, \beta \in [0, 2\pi)$. The condition $|a|^2 + |b|^2 = 1$ now reduces to $r_a^2 + r_b^2 = 1$, hence the point (r_a, r_b) lies on the unit circle in the first quadrant, as shown in figure 4.8.

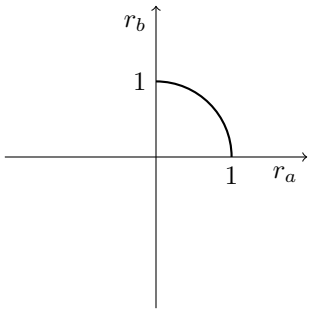


Figure 4.8: The possible values of r_a and r_b are shown. They form a quarter of the unit circle.

Thus, we can find a $\theta' \in [0, \pi/2]$ such that $r_a = \cos\theta'$ and $r_b = \sin\theta'$. For later convenience, though, we will use $\theta = 2\theta'$, so that $r_a = \cos\frac{\theta}{2}$ and $r_b = \sin\frac{\theta}{2}$, and $\theta \in [0, \pi]$. By defining $\phi = \beta - \alpha + 2k\pi$, where $k \in \mathbb{Z}$ is chosen such that $\phi \in [0, 2\pi)$, we now obtain:

$$|\psi\rangle = e^{i\alpha}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right)$$

From chapter 3, we know that this phase factor, $e^{i\alpha}$, is not of importance when doing measurements. This factor will be referred to as an "unimportant factor". Hence, the state is completely determined by the values of θ and ϕ, where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$. Thus, any state vector of a single qubit can be visualized as a point on a unit sphere in 3-dimensions, where spherical coordinates are used:

$$\begin{aligned} x &= \sin\theta\cos\phi \\ y &= \sin\theta\sin\phi \\ z &= \cos\theta \end{aligned}$$

This unit sphere is called the Bloch sphere, and will prove to be a very helpful tool in visualizing the effect of single qubit operations, as we will show soon.

For example, for the state $|0\rangle$, we find $\theta = 0$, hence it is located on the north pole of the Bloch sphere, in $(0, 0, 1)$. For $|1\rangle$, on the other hand, we find $\theta = \pi$, hence it is located on the south pole of the Bloch sphere, in $(0, 0, -1)$. Superposition states are located somewhere in between, for example, for $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, we find $\theta = \pi/2$ and $\phi = 0$, hence it is located on the equator, in $(1, 0, 0)$. See also figure 4.9.

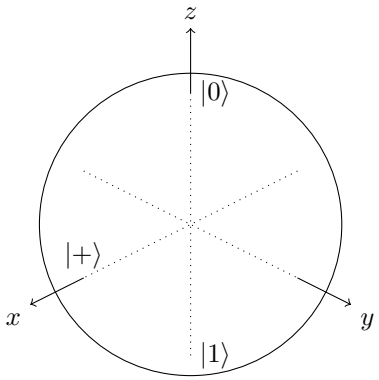


Figure 4.9: The Bloch sphere, with the position of three states indicated: the two basis states, $|0\rangle$ and $|1\rangle$, and the superposition state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

4.3.1.2 The effect of single qubit operations on the Bloch sphere

One could ask oneself how the single qubit operations affect the representation of the states on the Bloch sphere. It turns out that every single qubit operation is associated with a rotation of the Bloch sphere, which is what we will prove now.

Central to this proof will be the so-called Pauli matrices. In fact, they are the matrix representations of some single qubit operations we have already seen in chapter 3:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Note that all Pauli matrices are Hermitian. Some relationships between these Pauli matrices turn out to be particularly useful.

$$\begin{array}{ccc} X^2 = I & XY = iZ & XZ = -iY \\ YX = -iZ & Y^2 = I & YZ = iX \\ ZX = iY & ZY = -iX & Z^2 = I \end{array}$$

The following theorem shows a first application of Pauli matrices.

Theorem 4.10: Decomposition of matrix representation of single qubit gate into Pauli matrices *Take an arbitrary single qubit gate, with corresponding matrix representation U. Then we can rewrite U in the following form:*

$$U = e^{i\alpha}\left(\cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}\left(\sin\theta\cos\phi X + \sin\theta\sin\phi Y + \cos\theta Z\right)\right)$$

where $\alpha \in [0, 2\pi)$, $\chi \in [0, 2\pi]$, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$. Conversely, for any choice of α, χ, θ and ϕ, this matrix U is unitary.

Proof: Suppose that we have an arbitrary single qubit gate, with matrix representation U. Then this matrix U can be written as:

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where $a, b, c, d \in \mathbb{C}$. An arbitrary number $x \in \mathbb{C}$ can be written in polar form: $x = r_x e^{i\phi_x}$. Now, as all rows and columns of U should be of unit length, we obtain:

$$|a|^2 + |b|^2 = 1 \Rightarrow r_a^2 + r_b^2 = 1$$
$$|c|^2 + |d|^2 = 1 \Rightarrow r_c^2 + r_d^2 = 1$$
$$|a|^2 + |c|^2 = 1 \Rightarrow r_a^2 + r_c^2 = 1$$
$$|b|^2 + |d|^2 = 1 \Rightarrow r_b^2 + r_d^2 = 1$$

Hence, we obtain: $r_b = r_c$ and $r_a = r_d$. Furthermore, the two column vectors should be orthogonal:

$$r_a r_b e^{-i\phi_a + i\phi_b} + r_c r_d e^{-i\phi_c + i\phi_d} = 0$$

Using $r_a = r_d$ and $r_b = r_c$, we obtain:

$$e^{i(\phi_b - \phi_a - \phi_d + \phi_c + \pi)} = 0 \Rightarrow \phi_b - \phi_a - \phi_d + \phi_c + \pi \equiv 0 \mod 2\pi$$

Now define $\alpha = (\phi_a + \phi_d)/2$. Then, we can write U as:

$$U = e^{i\alpha}\begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix}$$

where $a' = e^{-i\alpha}a$, and similarly for the other matrix elements. As we are multiplying by a number that has a modulus of 1, the radii are left unaltered, hence for example $r_{a'} = r_a$. The arguments, however, are not. Particularly:

$$\begin{aligned} \phi_{a'} &= \phi_a - \alpha = \phi_a - \frac{\phi_a + \phi_d}{2} = \frac{\phi_a - \phi_d}{2} \\ \phi_{d'} &= \phi_d - \alpha = \phi_d - \frac{\phi_a + \phi_d}{2} = \frac{\phi_d - \phi_a}{2} \end{aligned}$$

Hence, we find $\phi_{a'} = -\phi_{d'}$. Substitution in the equation of arguments derived above, we obtain:

$$\phi_{b'} + \alpha - \phi_{a'} - \alpha - \phi_{d'} - \alpha + \phi_c + \alpha + \pi \equiv 0 \mod 2\pi$$

All α's cancel, and using $\phi_{a'} = -\phi_{d'}$, we obtain:

$$\phi_{b'} + \phi_{c'} + \pi \equiv 0 \mod 2\pi$$

Hence: $\phi_{c'} \equiv -\phi_{b'} + \pi \mod 2\pi$. So, we find an interesting set of relations:

$$\begin{aligned}(a')^* &= (r_a e^{i\phi_{a'}})^* = r_a e^{-i\phi_{a'}} = r_d e^{i\phi_{d'}} = d' \\ (c')^* &= (r_c e^{i\phi_{c'}})^* = r_c e^{-i\phi_{c'}} = r_b e^{i(\phi_{b'}-\pi)} = -r_b e^{i\phi_{b'}} = -b'\end{aligned}$$

Hence, we can find $k, l, m, n \in \mathbb{R}$ such that:

$$U = e^{i\alpha} \begin{bmatrix} k+li & -m+ni \\ m+ni & k-li \end{bmatrix}$$

This can be rewritten using the Pauli matrices, as follows:

$$U = e^{i\alpha}\left(kI + liZ + miY + niX\right)$$

As U is unitary, we know $UU^* = I$. Using that the Pauli matrices are Hermitian and all relations of Pauli matrices listed above, this yields:

$$\begin{aligned}UU^* &= e^{i\alpha}(kI + liZ + miY + niX) \cdot e^{-i\alpha}(kI - liZ - miY - niX) \\ &= k^2I^2 - kliIZ - kmiIY - kniIX + kliZI + l^2Z^2 + lmZY + lnZX \\ &\quad + kmiYI + lmYZ + m^2Y^2 + mnYX + kniXI + lnXZ + mnXY + n^2X^2 \\ &= k^2I - kliZ - kmiY - kniX + kliZ + l^2I - lmiX + lniY \\ &\quad + kmiY + lmiX + m^2I - mniZ + kniX - lniY + mniZ + n^2I \\ &= (k^2 + l^2 + m^2 + n^2)I\end{aligned}$$

Hence, we find $k^2 + l^2 + m^2 + n^2 = 1$. So, the vector (k, l, m, n) lies on the four dimensional unit sphere, hence we can write it in four dimensional spherical coordinates, where $\chi' \in [0, \pi]$, $\theta \in [0, \pi]$, and $\phi \in [0, 2\pi)$:

$$\begin{aligned}k &= \cos\chi' \\ l &= \sin\chi'\cos\theta \\ m &= \sin\chi'\sin\theta\sin\phi \\ n &= \sin\chi'\sin\theta\cos\phi\end{aligned}$$

As before, we define $\chi = 2\chi'$ for future convenience, so we find $\chi \in [0, 2\pi]$. Thus, we obtain the following expression for U:

$$U = e^{i\alpha}\left(\cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}\left(\sin\theta\cos\phi X + \sin\theta\sin\phi Y + \cos\theta Z\right)\right)$$

Finally, we see that any choice of χ, θ and ϕ yields $k^2 + l^2 + m^2 + n^2 = 1$ and so $UU^* = I$ for any choice of α, χ, θ and ϕ. Hence U is unitary for any choice of α, χ, θ and ϕ. So, we have proven what we set out to prove. □

The use of this theorem is not directly obvious at this stage. The following definition and theorem, though, will change this.

Definition 4.11: Rotation matrices
Let $\hat{n}$ be a unit vector in three dimensions. It is written in spherical coordinates as $\hat{n} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. Then the rotation matrix $R_{\hat{n}}(\chi)$, with $\chi \in \mathbb{R}$, is defined by:

$$R_{\hat{n}}(\chi) = \cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}(\sin\theta\cos\phi X + \sin\theta\sin\phi Y + \cos\theta Z)$$

We will first have a closer look at $R_{\hat{z}}(\chi)$ and $R_{\hat{y}}(\chi)$.

Lemma 4.12: Rotation about the z-axis
Suppose $\chi \in \mathbb{R}$. In the Bloch sphere, $R_{\hat{z}}(\chi)$ can be visualized as a rotation about the z-axis, over an angle of χ.

Proof: Suppose we have a qubit in the state associated with the Bloch vector represented in polar coordinates by $(r_x, r_y, r_z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. Then its state vector with respect to the computational basis is, up to an unimportant constant with modulus 1:

$$\psi = \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix}$$

To reduce the complexity of the calculations, we introduce the so-called density operator of this state:

$$\rho = \psi\psi^* = \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} & e^{-i\phi}\sin\frac{\theta}{2} \end{bmatrix}$$

Rewriting this expression yields:

$$\rho = \begin{bmatrix} \cos^2\frac{\theta}{2} & e^{-i\phi}\cos\frac{\theta}{2}\sin\frac{\theta}{2} \\ e^{i\phi}\cos\frac{\theta}{2}\sin\frac{\theta}{2} & \sin^2\frac{\theta}{2} \end{bmatrix}$$

Applying double angle formulas $\cos^2\alpha = \frac{1}{2} + \frac{1}{2}\cos(2\alpha)$, $\sin^2\alpha = \frac{1}{2} - \frac{1}{2}\cos(2\alpha)$ and $\cos\alpha\sin\alpha = \frac{1}{2}\sin(2\alpha)$ leads to:

$$\rho = \frac{1}{2}\begin{bmatrix} 1+\cos\theta & e^{-i\phi}\sin\theta \\ e^{i\phi}\sin\theta & 1-\cos\theta \end{bmatrix}$$

Expanding the exponentials yields:

$$\rho = \frac{1}{2}\begin{bmatrix} 1+\cos\theta & \sin\theta\cos\phi - i\sin\theta\sin\phi \\ \sin\theta\cos\phi + i\sin\theta\sin\phi & 1-\cos\theta \end{bmatrix}$$

So, rewriting this in the form of Pauli matrices, we obtain:

$$\rho = \frac{1}{2}(I + \sin\theta\cos\phi X + \sin\theta\sin\phi Y + \cos\theta Z) = \frac{1}{2}(I + r_x X + r_y Y + r_z Z)$$

Applying the operation $R_{\hat{z}}(\chi)$ yields the new vector ψ':

$$\psi' = R_{\hat{z}}(\chi)\psi$$

So, the new density operator is given by:

$$\rho' = R_{\hat{z}}(\chi)\psi(R_{\hat{z}}(\chi)\psi)^* = R_{\hat{z}}(\chi)\rho R_{\hat{z}}(\chi)^*$$

Substituting for ρ, we obtain:

$$\begin{aligned} \rho' &= \frac{1}{2}R_{\hat{z}}(\chi)(I + r_x X + r_y Y + r_z Z)R_{\hat{z}}(\chi)^* \\ &= \frac{1}{2}R_{\hat{z}}(\chi)IR_{\hat{z}}(\chi)^* + \frac{1}{2}r_x R_{\hat{z}}(\chi)XR_{\hat{z}}(\chi)^* + \frac{1}{2}r_y R_{\hat{z}}(\chi)YR_{\hat{z}}(\chi)^* + \frac{1}{2}r_z R_{\hat{z}}(\chi)ZR_{\hat{z}}(\chi)^* \end{aligned}$$

We now evaluate every term individually, where we use the abbreviations $c_\chi = \cos\frac{\chi}{2}$ and $s_\chi = \sin\frac{\chi}{2}$:

$$\begin{aligned} R_{\hat{z}}(\chi)IR_{\hat{z}}(\chi)^* &= (c_\chi I - is_\chi Z)\, I\, (c_\chi I + is_\chi Z) = \left(c_\chi^2 I - ic_\chi s_\chi Z + ic_\chi s_\chi Z + s_\chi^2 Z^2\right) = I \\ R_{\hat{z}}(\chi)XR_{\hat{z}}(\chi)^* &= (c_\chi I - is_\chi Z)\, X\, (c_\chi I + is_\chi Z) = \left(c_\chi^2 X + c_\chi s_\chi Y + c_\chi s_\chi Y - s_\chi^2 X\right) \\ &= (c_\chi^2 - s_\chi^2)X + 2c_\chi s_\chi Y = \cos\chi X + \sin\chi Y \\ R_{\hat{z}}(\chi)YR_{\hat{z}}(\chi)^* &= (c_\chi I - is_\chi Z)\, Y\, (c_\chi I + is_\chi Z) = \left(c_\chi^2 Y - c_\chi s_\chi X - c_\chi s_\chi X - s_\chi^2 Y\right) \\ &= (c_\chi^2 - s_\chi^2)Y - 2c_\chi s_\chi X = -\sin\chi X + \cos\chi Y \\ R_{\hat{z}}(\chi)ZR_{\hat{z}}(\chi)^* &= (c_\chi I - is_\chi Z)\, Z\, (c_\chi I + is_\chi Z) = \left(c_\chi^2 Z - ic_\chi s_\chi I + ic_\chi s_\chi I + s_\chi^2 Z\right) = Z \end{aligned}$$

Hence, we find:

$$\begin{aligned}\rho' &= \frac{1}{2}I + \frac{1}{2}r_x(\cos\chi X + \sin\chi Y) + \frac{1}{2}r_y(-\sin\chi X + \cos\chi Y) + \frac{1}{2}r_z Z \\ &= \frac{1}{2}(I + (r_x\cos\chi - r_y\sin\chi)X + (r_x\sin\chi + r_y\cos\chi)Y + r_z Z)\end{aligned}$$

So, we find the following new coordinates in the Bloch sphere:

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \end{bmatrix} = \begin{bmatrix} \cos\chi & -\sin\chi & 0 \\ \sin\chi & \cos\chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

Hence, the coordinates of the vector in the Bloch sphere are rotated about the z-axis by an angle of χ. This completes the proof. □

Lemma 4.13: Rotation about the y-axis
Let $\chi \in \mathbb{R}$. In the Bloch sphere, $R_{\hat{y}}(\chi)$ can be visualized as a rotation about the y-axis, over an angle χ.

Proof: This proof is very similar to the previous proof, so we will skip right ahead to the new density operator. All steps before this are identical to the previous proof.

$$\begin{aligned}\rho' &= R_{\hat{y}}(\chi)\rho R_{\hat{y}}(\chi)^* = \frac{1}{2}R_{\hat{y}}(\chi)(I + r_x X + r_y Y + r_z Z)R_{\hat{y}}(\chi)^* \\ &= \frac{1}{2}R_{\hat{y}}(\chi)IR_{\hat{y}}(\chi)^* + \frac{1}{2}r_x R_{\hat{y}}(\chi)XR_{\hat{y}}(\chi)^* + \frac{1}{2}r_y R_{\hat{y}}(\chi)YR_{\hat{y}}(\chi)^* + \frac{1}{2}r_z R_{\hat{y}}(\chi)ZR_{\hat{y}}(\chi)^*\end{aligned}$$

Again, evaluating this expression term by term and using $c_\chi = \cos\frac{\chi}{2}$ and $s_\chi = \sin\frac{\chi}{2}$:

$$\begin{aligned}R_{\hat{y}}(\chi)IR_{\hat{y}}(\chi)^* &= (c_\chi I - is_\chi Y)I(c_\chi I + is_\chi Y) = (c_\chi^2 I - ic_\chi s_\chi Y + ic_\chi s_\chi Y + s_\chi^2 I) = I \\ R_{\hat{y}}(\chi)XR_{\hat{y}}(\chi)^* &= (c_\chi I - is_\chi Y)X(c_\chi I + is_\chi Y) = (c_\chi^2 X - c_\chi s_\chi Z - c_\chi s_\chi Z - s_\chi^2 X) \\ &= (c_\chi^2 - s_\chi^2)X - 2c_\chi s_\chi Z = \cos\chi X - \sin\chi Z \\ R_{\hat{y}}(\chi)YR_{\hat{y}}(\chi)^* &= (c_\chi I - is_\chi Y)Y(c_\chi I + is_\chi Y) = (c_\chi^2 Y - ic_\chi s_\chi I + ic_\chi s_\chi I + s_\chi^2 Y) = Y \\ R_{\hat{y}}(\chi)ZR_{\hat{y}}(\chi)^* &= (c_\chi I - is_\chi Y)Z(c_\chi I + is_\chi Y) = (c_\chi^2 Z + c_\chi s_\chi X + c_\chi s_\chi X - s_\chi^2 Z) \\ &= (c_\chi^2 - s_\chi^2)Z + 2c_\chi s_\chi X = \sin\chi X + \cos\chi Z\end{aligned}$$

Hence, substituting these results back, we find:

$$\begin{aligned}\rho' &= \frac{1}{2}I + \frac{1}{2}r_x(\cos\chi X - \sin\chi Z) + \frac{1}{2}Yr_y + \frac{1}{2}r_z(\sin\chi X + \cos\chi Z) \\ &= \frac{1}{2}(I + (r_x\cos\chi + r_z\sin\chi)X + r_y Y + (-r_x\sin\chi + r_z\cos\chi)Z)\end{aligned}$$

So, we find for the new coordinates in the Bloch sphere:

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \end{bmatrix} = \begin{bmatrix} \cos\chi & 0 & \sin\chi \\ 0 & 1 & 0 \\ -\sin\chi & 0 & \cos\chi \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

Hence, the effect of $R_{\hat{y}}(\chi)$ is indeed a rotation about the y-axis over an angle of χ in the Bloch sphere. So, that completes the proof. □

Theorem 4.14: Rotation about an arbitrary axis
Let $\hat{n}$ be an arbitrary three-dimensional unit vector. In the Bloch sphere, $R_{\hat{n}}(\chi)$ can be visualized as a rotation about $\hat{n}$ over an angle χ, where $\chi \in \mathbb{R}$.

Proof: As $\hat{n}$ is a unit vector, we can write it in spherical coordinates in the following way: $(n_x, n_y, n_z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. Note that a rotation about the $\hat{n}$ axis over an angle χ can be achieved by executing the following rotations consecutively, as is also shown in figure 4.10.

1. Rotate about the z-axis by $-\phi$.
2. Rotate about the y-axis by $-\theta$.
3. Rotate about the z-axis by χ.
4. Rotate about the y-axis by θ.
5. Rotate about the z-axis by ϕ.

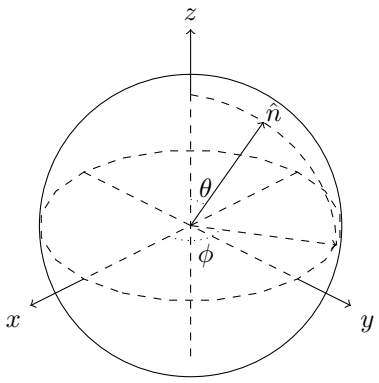


Figure 4.10: A rotation about the axis $\hat{n}$ can be composed of rotations about the $\hat{z}$ and $\hat{y}$ axes. One first rotates $\hat{n}$ towards the xz-plane, after which $\hat{n}$ is rotated up towards the north pole. Then the actual rotation around $\hat{n}$ is performed, after which the preliminary steps are undone.

So, we calculate the operation that results when these operations are performed consecutively. Note therefore that $R_{\hat{y}}(\theta)^*$ is $R_{\hat{y}}(-\theta)$, as we are just turning the Bloch sphere in the other direction. The same holds for rotations about the z-axis. We obtain, using the relations derived in the previous proofs:

$$\begin{aligned}
& R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)R_{\hat{z}}(\chi)R_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi) \\
= \;& R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)\left(\cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}Z\right)(\chi)R_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi) \\
= \;& \cos\frac{\chi}{2}R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)IR_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi) - i\sin\frac{\chi}{2}R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)ZR_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi) \\
= \;& \cos\frac{\chi}{2}R_{\hat{z}}(\phi)IR_{\hat{z}}(-\phi) - i\sin\frac{\chi}{2}R_{\hat{z}}(\phi)(\sin\theta X + \cos\theta Z)R_{\hat{z}}(-\phi) \\
= \;& \cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}(\sin\theta R_{\hat{z}}(\phi)XR_{\hat{z}}(-\phi) + \cos\theta R_{\hat{z}}(\phi)ZR_{\hat{z}}(-\phi)) \\
= \;& \cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}(\sin\theta(\cos\phi X + \sin\phi Y) + \cos\theta Z) \\
= \;& \cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}(\sin\theta\cos\phi X + \sin\theta\sin\phi Y + \cos\theta Z) \\
= \;& \cos\frac{\chi}{2}I - i\sin\frac{\chi}{2}(n_xX + n_yY + n_zZ) \\
= \;& R_{\hat{n}}(\chi)
\end{aligned}$$

Hence, we find that $R_{\hat{n}}(\chi)$ is indeed a rotation about the axis $\hat{n}$ by an angle of χ, as required. □

This brings us to the most important result of the Bloch sphere:

Corollary 4.15: Single qubit gates and rotations in the Bloch sphere
Every single qubit gate is associated with a single rotation in the Bloch sphere, up to an unimportant constant. On the other hand, every rotation in the Bloch sphere resembles a single qubit gate.

Proof: Let's first take an arbitrary single qubit gate. Then we have seen that it can be decomposed into a sum of Pauli matrices. This decomposition yields that it is equal up to an unimportant constant to $R_{\hat{n}}(\chi)$, where $\hat{n}$ is a unit vector in 3D, and $\chi \in \mathbb{R}$. The last theorem now yields that this is a rotation in the Bloch sphere about the axis defined by $\hat{n}$ over an angle χ.

On the other hand, if we take a random rotation in the Bloch sphere, then there is a unit vector on its axis, which we call $\hat{n}$. Then this rotation can be expressed as the operation $R_{\hat{n}}(\chi)$, according to the preceding theorem. But this rotation is equal to a single qubit gate, where the unimportant constant in the decomposition can be randomly chosen. □

We have now developed an intuitive way of thinking about single qubit gates. This turns out to be of great value when coming up with proofs for theorems that concern these gates. An excellent example is the following theorem, that we will need later on in our proof of the universality of the CNOT, Hadamard and $\pi/8$ gate.

Theorem 4.16: Consecutive rotations around two fixed axes
Suppose $\hat{n}$ and $\hat{m}$ are unit vectors on two fixed axes in the Bloch sphere. Let θ be the angle between these axes. If $\theta \neq 0$ and $\theta \neq \pi$, that is, the axes are non-parallel, then any single qubit gate with matrix representation U can be performed by a sequence of consecutive rotations about $\hat{n}$ and $\hat{m}$ of the following form:

$$U = e^{i\alpha} R_{\hat{n}}(\alpha_{m+1}) \left(\prod_{i=1}^{m} R_{\hat{m}}(\beta_i) R_{\hat{n}}(\alpha_i) \right)$$

where $\alpha, \alpha_1, \ldots, \alpha_{m+1}, \beta_1, \ldots, \beta_m \in [0, 2\pi)$, and:

$$m = \begin{cases} \left\lceil \frac{\pi}{2\theta} \right\rceil, & 0 < \theta \leq \frac{\pi}{2} \\ \left\lceil \frac{\pi}{2(\pi-\theta)} \right\rceil, & \frac{\pi}{2} \leq \theta < \pi \end{cases}$$

Proof: Take an arbitrary single qubit gate, with matrix representation U. Then we have already seen that we write it like $e^{i\alpha} R_{\hat{s}}(\chi)$, with $\alpha \in [0, 2\pi)$, $\hat{s}$ some 3D unit vector, and χ some angle. Hence, the problem reduces to showing that the rotation $R_{\hat{s}}(\chi)$ of the Bloch sphere can be achieved by the sequence of rotations given in matrix representation by:

$$R_{\hat{s}}(\chi) = R_{\hat{n}}(\alpha_{m+1}) \left(\prod_{i=1}^{m} R_{\hat{m}}(\beta_i) R_{\hat{n}}(\alpha_i) \right)$$

for any axis $\hat{s}$ and angle χ.

To prove this, we first have a closer look at what happens when we perform a rotation in 3D space. Suppose one has an orthonormal basis, and wonders what happens to that basis upon rotation. Then this basis remains orthonormal, and its orientation is preserved as well. Hence, if one finds out what happens to two vectors of this orthonormal basis upon rotation, then one can deduce what happens to the third one. So, it is sufficient to take an orthonormal basis of the three-dimensional space, and check whether two of its vectors are mapped to their desired location, to conclude that the desired rotation is achieved.

We are going to use this in the following way. As the rotation $R_{\hat{s}}(\chi)$ is invertible, we can easily find the vector $\hat{v}$ that is mapped to $\hat{n}$. Take $\hat{w}$ another unit vector at random, that is orthogonal to $\hat{v}$.

Now picture the sphere with $\hat{n}$ pointing up, and define spherical coordinates accordingly. Define the azimuthal angle, ϕ, in such a way that the azimuthal angle of $\hat{m}$ is 0. We will first assume that $\theta \leq \pi/2$. See also figure

4.11.

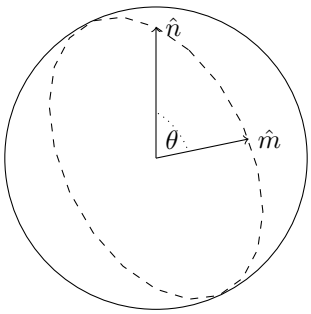


Figure 4.11: Bloch sphere with the axes $\hat{n}$ and $\hat{m}$, and their angle θ indicated.

Now, the vector $\hat{v}$ has polar angle θ_v and azimuthal angle ϕ_v. Depending on the value of θ_v, we decide what to do next.

If $\theta_v \leq 2\theta$ (where θ denotes the angle between $\hat{n}$ and $\hat{m}$), then we can first rotate the Bloch sphere about $\hat{n}$ and then about $\hat{m}$, such that $\hat{v}$ is mapped to the north pole. This is because the path of the north pole upon rotation around $\hat{m}$ forms a circle that covers all polar angles between 0 and 2θ, as is shown in figure 4.12. The first rotation about $\hat{n}$ serves to rotate $\hat{v}$ onto this circle, and the rotation about $\hat{m}$ afterwards completes the mapping to the north pole.

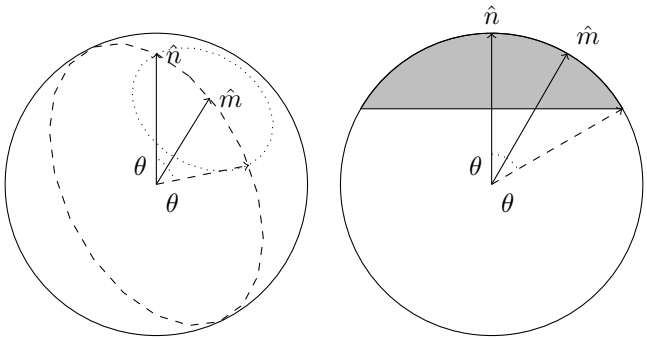


Figure 4.12: Figure illustrating the last step of moving the vector $\hat{v}$ towards the north pole. One can move the vector $\hat{v}$ into the north pole from the circle around $\hat{m}$ indicated as the dotted line in the left picture. If $\theta_v \leq 2\theta$, then $\hat{v}$ is located somewhere in the gray area in the right figure, from where it is possible to rotate the vector onto the circle by a rotation about $\hat{n}$.

If, on the other hand, $\theta_v > 2\theta$, then we must first bring $\hat{v}$ up in the sphere, such that it has a polar angle smaller than or equal to 2θ, after which we can apply the above technique. This is done in the following way. First rotate the sphere about the $\hat{n}$ axis, such that the vector $\hat{v}$ is mapped to a position directly below $\hat{m}$. Then rotate about $\hat{m}$ over an angle of π. This brings the vector $\hat{v}$ up by a polar angle of 2θ, as can be seen in figure 4.13.

This process can be repeated until the polar coordinate reduces to a value smaller than or equal to 2θ. As θ_v starts out at a value below π, this process must be repeated a maximum number of $\left\lceil \frac{\pi}{2\theta} \right\rceil - 1$ times. Hence, with the finalizing step of putting $\hat{v}$ on the north pole, after the following $2m$ rotations the vector $\hat{v}$ is located on the north pole.

$$\prod_{i=1}^{m} R_{\hat{m}}(\beta_i) R_{\hat{n}}(\alpha_i)$$

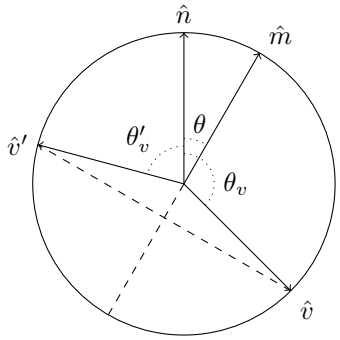


Figure 4.13: The vector $\hat{v}$ can be brought up by a rotation about $\hat{m}$. The vector $\hat{v}$ must first be placed directly below $\hat{m}$. Then, rotating about $\hat{m}$ over an angle π yields: $\theta'_v + \theta = \theta_v - \theta$, hence $\theta'_v = \theta_v - 2\theta$.

Now, the vector $\hat{w}$ must be put in place. As it was orthogonal to $\hat{v}$ in the beginning and we have only done rotations since, it must still be orthogonal to $\hat{v}$. Hence it is located somewhere on the equator of the sphere, and so it can be put into place by a single rotation about $\hat{n}$. With two orthogonal vectors in place, we can now conclude that we have successfully executed the desired rotation, and that we can write the matrix representation of every single qubit gate in the following manner:

$$U = e^{i\alpha} R_{\hat{n}}(\alpha_{m+1}) \left(\prod_{i=1}^{m} R_{\hat{m}}(\beta_i) R_{\hat{n}}(\alpha_i) \right)$$

Finally, suppose that $\theta > \pi/2$. Then one can take the axis $-\hat{m}$, and perform all rotations referred to above with negative angle. The angle between $\hat{n}$ and $-\hat{m}$ is $\pi - \theta$, hence the difference in the formulae for m. □

The above theorem is very general. A more specific corollary is given below:

Corollary 4.17: Consecutive rotations about z and y axes
Take an arbitrary single qubit gate with matrix representation U. Then it can be written as:

$$U = e^{i\alpha} R_{\hat{z}}(\beta) R_{\hat{y}}(\gamma) R_{\hat{z}}(\delta)$$

where $\alpha, \beta, \gamma, \delta \in [0, 2\pi)$.

Proof: The angle between $\hat{z}$ and $\hat{y}$ is $\frac{\pi}{2}$, so when we apply the above theorem, we obtain $m = 1$. Hence, the result follows directly by substituting $m = 1$ in the expression for U. □

We now have all the information we need about the Bloch sphere, in order to continue our proof that CNOT, Hadamard and $\pi/8$ gates are universal.

4.3.2 Implementation of controlled single qubit gates

So far, we have seen that we can implement any quantum circuit by using single qubit gates, and controlled single qubit gates with one control qubit. First of all, we focus on the controlled single qubit gate. To that end, we need the following lemma:

Lemma 4.18: ABC-decomposition of single qubit gates
Take an arbitrary single qubit gate, and denote its matrix representation by U. Then we can find unitary matrices A, B and C, and $\alpha \in [0, 2\pi)$, such that:

$$U = e^{i\alpha} AXBXC \qquad \text{and} \qquad ABC = I$$

Proof: From the previous section, we have found that we can write U in the following form.

$$U = e^{i\alpha} R_{\hat{z}}(\beta) R_{\hat{y}}(\gamma) R_{\hat{z}}(\delta)$$

Now define the following matrices:

$$\begin{aligned} A &= R_{\hat{z}}(\beta) R_{\hat{y}}(\gamma/2) \\ B &= R_{\hat{y}}(-\gamma/2) R_{\hat{z}}(-(\delta+\beta)/2) \\ C &= R_{\hat{z}}((\delta-\beta)/2) \end{aligned}$$

Then we find:

$$ABC = R_{\hat{z}}(\beta) R_{\hat{y}}\left(\frac{\gamma}{2}\right) \cdot R_{\hat{y}}\left(-\frac{\gamma}{2}\right) R_{\hat{z}}\left(-\frac{\delta+\beta}{2}\right) \cdot R_{\hat{z}}\left(\frac{\delta-\beta}{2}\right) = R_{\hat{z}}(\beta) R_{\hat{z}}(-\beta) = I$$

Furthermore, we have $XYX = iZX = i^2 Y = -Y$ and $X^2 = I$, hence:

$$X R_{\hat{y}}(\chi) X = X\left(\cos\frac{\chi}{2} I - i \sin\frac{\chi}{2} Y\right) X = \cos\frac{\chi}{2} I + i \sin\frac{\chi}{2} Y = R_{\hat{y}}(-\chi)$$

Similarly, with $XZX = iXY = i^2 Z = -Z$, and $X^2 = I$, we find:

$$X R_{\hat{z}}(\chi) X = X\left(\cos\frac{\chi}{2} I - i \sin\frac{\chi}{2} Z\right) X = \cos\frac{\chi}{2} I + i \sin\frac{\chi}{2} Z = R_{\hat{z}}(-\chi)$$

Hence:

$$\begin{aligned} XBX &= X R_{\hat{y}}\left(-\frac{\gamma}{2}\right) R_{\hat{z}}\left(-\frac{\delta+\beta}{2}\right) X = X R_{\hat{y}}\left(-\frac{\gamma}{2}\right) XX R_{\hat{z}}\left(-\frac{\delta+\beta}{2}\right) X \\ &= R_{\hat{y}}\left(\frac{\gamma}{2}\right) R_{\hat{z}}\left(\frac{\delta+\beta}{2}\right) \end{aligned}$$

So, substitution yields:

$$\begin{aligned} e^{i\alpha} AXBXC &= e^{i\alpha} R_{\hat{z}}(\beta) R_{\hat{y}}\left(\frac{\gamma}{2}\right) R_{\hat{y}}\left(\frac{\gamma}{2}\right) R_{\hat{z}}\left(\frac{\delta+\beta}{2}\right) R_{\hat{z}}\left(\frac{\delta-\beta}{2}\right) \\ &= e^{i\alpha} R_{\hat{z}}(\beta) R_{\hat{y}}(\gamma) R_{\hat{z}}(\delta) = U \end{aligned}$$

So indeed, we found matrices A, B and C such that $e^{i\alpha} AXBXC = U$ and $ABC = I$. □

The usefulness of this lemma becomes apparent if we look at the following implementation of a controlled single qubit gate. Suppose we have a controlled single qubit gate, with matrix representation U and matrices A, B and C as in the lemma above. Then we can implement the controlled single qubit gate as shown in figure 4.14.

Note that this last controlled qubit gate can be implemented by the gate shown in figure 4.15.

So, we have now found that we can implement any quantum circuit by making use of the CNOT and single qubit gates.

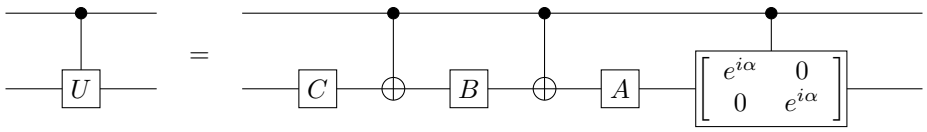


Figure 4.14: Implementation of the single qubit gate controlled by one control qubit.

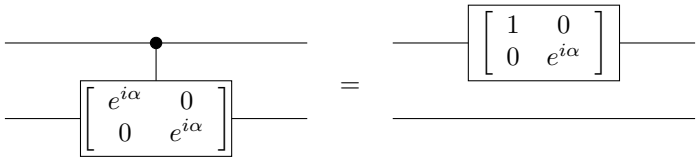


Figure 4.15: Implementation of the controlled multiplication of a phase factor.

4.3.3 Distance between unitary matrices

Now all that is left to do is showing that any single qubit gate can be approximated up to arbitrary accuracy by using the Hadamard and $\pi/8$ gates. To do so, though, we must first develop an idea when an approximation of a quantum gate is close. To quantify this, we introduce the following metric on the space of 2×2 unitary matrices.

Definition 4.19: Distance in the space of 2×2 unitary matrices
Let Q be the collection of all unitary 2×2 matrices. Then we define a function $E : Q \times Q \to \mathbb{R}$. For any two 2×2 unitary matrices U and V:

$$E(U, V) = \max_{\mathbf{v}} ||(U - V)\mathbf{v}||$$

where $\mathbf{v}$ can be any unit vector in $\mathbb{C}^2$.

Theorem 4.20: Metric on the space of 2×2 unitary matrices
The function E defined above is a metric on Q.

Proof: Suppose U and V are two unitary matrices with dimensions 2×2. We check all properties of metrics systematically below.

Obviously $E(U, V) \geq 0$. Moreover, if $E(U, V) = 0$, we have that for any vector $\mathbf{v}$, $(U - V)\mathbf{v} = \mathbf{0}$, hence $U\mathbf{v} = V\mathbf{v}$. But then if we fill in the standard basic vectors for $\mathbf{v}$, we obtain $U = V$. Similarly, if $U = V$, we find $(U - V)\mathbf{v} = 0$ for all $\mathbf{v}$, hence $E(U, V) = 0$. So, $E(U, V) = 0 \Leftrightarrow U = V$.

Trivially, $E(U, V) = E(V, U)$ also holds, because $||(U - V)\mathbf{v}|| = || - (V - U)\mathbf{v}|| = ||(V - U)\mathbf{v}||$, for all $\mathbf{v}$.

Now, suppose W is also a 2×2 unitary matrix. Then:

$$\begin{aligned} E(U, V) + E(V, W) &= \max_{\mathbf{v}} ||(U - V)\mathbf{v}|| + \max_{\mathbf{v}} ||(V - W)\mathbf{v}|| \\ &\geq \max_{\mathbf{v}} ||(U - V)\mathbf{v} + (V - W)\mathbf{v}|| \\ &= \max_{\mathbf{v}} ||(U - W)\mathbf{v}|| = E(U, W) \end{aligned}$$

So, the triangle inequality is satisfied as well, hence (Q, E) is a metric space. □

Now, we have a way to talk about the distance between two matrices. The following theorem justifies this notion of distance.

Theorem 4.21: Effect of the distance on measurements
Suppose a single qubit system starts out in the state with corresponding vector $\mathbf{v}$*. Then one of two quantum operations is performed, with matrix representation* U *and* V*. Then, the qubit is measured, with corresponding measurement matrices* $\{M_m\}$*. The difference in probability that the measurement yields outcome* m *is bounded by the distance of* U *and* V *in the following way:*

$$|P_U(m) - P_V(m)| \leq 2E(U,V)$$

where $P_U(m)$ *and* $P_V(m)$ *denote the probability that the measurement outcome is* m*, after the application of* U *or* V *respectively.*

Proof: The probabilities are given by the measurement postulate of quantum mechanics, as we have seen in chapter 2. Their matrix form yields:

$$\begin{aligned} P_U(m) &= \mathbf{v}^* U^* M_m^* M_m U \mathbf{v} \\ P_V(m) &= \mathbf{v}^* V^* M_m^* M_m V \mathbf{v} \end{aligned}$$

Defining $\mathbf{d} = (U - V)\mathbf{v}$, we obtain:

$$\begin{aligned} |P_U(m) - P_V(m)| &= |\mathbf{v}^* U^* M_m^* M_m U \mathbf{v} - \mathbf{v}^* V^* M_m^* M_m V \mathbf{v}| \\ &= |\mathbf{v}^* U^* M_m^* M_m (\mathbf{d} + V\mathbf{v}) - \mathbf{v}^* V^* M_m^* M_m V \mathbf{v}| \\ &= |\mathbf{v}^* U^* M_m^* M_m \mathbf{d} + \mathbf{v}^* U^* M_m^* M_m V \mathbf{v} - \mathbf{v}^* V^* M_m^* M_m V \mathbf{v}| \\ &= |\mathbf{v}^* U^* M_m^* M_m \mathbf{d} + (\mathbf{v}^* U^* - \mathbf{v}^* V^*) M_m^* M_m V \mathbf{v}| \\ &= |\mathbf{v}^* U^* M_m^* M_m \mathbf{d} + \mathbf{d}^* M_m^* M_m V \mathbf{v}| \\ &\leq |\mathbf{v}^* U^* M_m^* M_m \mathbf{d}| + |\mathbf{d}^* M_m^* M_m V \mathbf{v}| \end{aligned}$$

Now, we invoke the Cauchy-Schwarz inequality, which states that for any $\mathbf{v}$ and $\mathbf{w}$, the inequality $|\mathbf{v}^*\mathbf{w}| \leq ||\mathbf{v}|| \cdot ||\mathbf{w}||$ holds. Hence:

$$\begin{aligned} |P_U(m) - P_V(m)| &\leq |\mathbf{v}^* U^* M_m^* M_m \mathbf{d}| + |\mathbf{d}^* M_m^* M_m V \mathbf{v}| \\ &\leq ||M_m^* M_m U \mathbf{v}|| \cdot ||\mathbf{d}|| + ||\mathbf{d}|| \cdot ||M_m^* M_m U \mathbf{v}|| \end{aligned}$$

Now, we will want to find an upper bound for $||M_m^* M_m U \mathbf{v}||$. Recall from chapter 2 that the measurement operators satisfy the completeness relation: $\sum_i M_i^* M_i = I$. As $M_i^* M_i$ is a Hermitian matrix, we find that its eigenvalues are real. Moreover, $M_i^* M_i$ is a positive semidefinite matrix, as for any vector $\mathbf{w} \in \mathbb{C}^2$, we have $\mathbf{w}^* M_i^* M_i \mathbf{w} = (M_i \mathbf{w})^* M_i \mathbf{w} = ||M_i \mathbf{w}||^2 \geq 0$. Hence, the eigenvalues of $M_i^* M_i$ are all non-negative.

Suppose now that $\mathbf{w}$ is an eigenvector of $M_m^* M_m$, with corresponding eigenvalue λ. Then, we find:

$$\mathbf{w}^*\mathbf{w} = \mathbf{w}^* I \mathbf{w} = \mathbf{w}^* \sum_i M_i^* M_i \mathbf{w} = \mathbf{w}^* M_m^* M_m \mathbf{w} + \sum_{i \neq m} \mathbf{w}^* M_i^* M_i \mathbf{w}$$

Working out the first term using $M_m^* M_m \mathbf{w} = \lambda \mathbf{w}$ yields:

$$\mathbf{w}^* M_m^* M_m \mathbf{w} = \mathbf{w}^* \lambda \mathbf{w} = \lambda \mathbf{w}^* \mathbf{w}$$

The other terms can be lower bounded by 0, as every term is greater than or equal to 0. Hence:

$$\mathbf{w}^*\mathbf{w} \geq \lambda \mathbf{w}^*\mathbf{w} \Rightarrow \lambda \leq 1$$

We already know that λ is non-negative, so we find $0 \leq \lambda \leq 1$.

Now, as $M_m^* M_m$ is a Hermitian matrix, its eigenvectors span $\mathbb{C}^2$, so $\mathbf{w} \in \mathbb{C}^2$ can be written as a linear combination of eigenvectors of $M_m^* M_m$ of unit length: $\mathbf{w} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2$. Moreover, these eigenvectors are orthogonal, so:

$$||\mathbf{w}||^2 = \mathbf{w}^*\mathbf{w} = (\alpha_1^* \mathbf{v}_1^* + \alpha_2^* \mathbf{v}_2^*)(\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2) = |\alpha_1|^2 + |\alpha_2|^2$$

Plugging this in yields:

$$\begin{aligned} ||M_m^* M_m \mathbf{w}||^2 &= ||\alpha_1 M_m^* M_m \mathbf{v}_1 + \alpha_2 M_m^* M_m \mathbf{v}_2||^2 = ||\alpha_1 \lambda_1 \mathbf{v}_1 + \alpha_2 \lambda_2 \mathbf{v}_2||^2 \\ &\leq |\alpha_1|^2 |\lambda_1|^2 ||\mathbf{v}_1||^2 + |\alpha_2|^2 |\lambda_2|^2 ||\mathbf{v}_2||^2 \leq |\alpha_1|^2 + |\alpha_2|^2 = ||\mathbf{w}||^2 \end{aligned}$$

Using this in our upper bound for $|P_U(m) - P_V(m)|$, we find:

$$\begin{aligned} |P_U(m) - P_V(m)| &\leq ||M_m^* M_m U \mathbf{v}|| \cdot ||\mathbf{d}|| + ||\mathbf{d}|| \cdot ||M_m^* M_m U \mathbf{v}|| \\ &\leq ||U\mathbf{v}|| \cdot ||\mathbf{d}|| + ||\mathbf{d}|| \cdot ||U\mathbf{v}|| \\ &= ||\mathbf{v}|| \cdot ||\mathbf{d}|| + ||\mathbf{d}|| \cdot ||\mathbf{v}|| \\ &= ||\mathbf{d}|| + ||\mathbf{d}|| = 2||\mathbf{d}|| \\ &= 2||(U - V)\mathbf{v}|| \\ &\leq 2E(U, V) \end{aligned}$$

Hence, indeed, we found what needed to be proven. □

Intuitively, this last theorem makes sense. If the distance between two matrices is very small, the probability that the outcome of the measurement is affected should be very slim. So, we can now safely conclude that if we want V to approximate U, it is sufficient to require that $E(U, V)$ is small.

The following theorem is very useful when dealing with compositions of rotations.

Theorem 4.22: Distance of the product of matrices
Suppose we have four 2×2 unitary matrices, denoted by U, V, W and X. Then the following relation holds:

$$E(UV, WX) \leq E(U, W) + E(V, X)$$

Proof: The proof makes use of the triangle inequality:

$$\begin{aligned} E(UV, WX) &= \max_{\mathbf{v}} ||(UV - WX)\mathbf{v}|| \\ &= \max_{\mathbf{v}} ||(UV - UX + UX - WX)\mathbf{v}|| \\ &\leq \max_{\mathbf{v}} ||(UV - UX)\mathbf{v}|| + \max_{\mathbf{v}} ||(UX - WX)\mathbf{v}|| \\ &= \max_{\mathbf{v}} ||U(V - X)\mathbf{v}|| + \max_{\mathbf{v}} ||(U - W)X\mathbf{v}|| \\ &= \max_{\mathbf{v}} ||(V - X)\mathbf{v}|| + \max_{\mathbf{v}} ||(U - W)\mathbf{v}|| \\ &= E(V, X) + E(U, W) \end{aligned}$$

as required. □

Finally, the following theorem is important to approximate rotations around a fixed axis.

Theorem 4.23: Approximation of rotations around a fixed axis
Suppose $\hat{n}$ is a unit vector, and $\alpha, \beta \in [0, 2\pi)$. Then:

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\beta)) \leq \frac{|\beta - \alpha|}{2}$$

Proof: As before, we can rewrite $\hat{n}$ in its spherical coordinates: $\hat{n} = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. Then we note that the following relation holds for any $\chi \in [0, 2\pi)$:

$$R_{\hat{n}}(\chi) = R_{\hat{z}}(\phi) R_{\hat{y}}(\theta) R_{\hat{z}}(\chi) R_{\hat{y}}(-\theta) R_{\hat{z}}(-\phi)$$

Combining this with the previous theorem, we obtain:

$$\begin{aligned}
E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\beta)) &= E(R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)R_{\hat{z}}(\alpha)R_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi), R_{\hat{z}}(\phi)R_{\hat{y}}(\theta)R_{\hat{z}}(\beta)R_{\hat{y}}(-\theta)R_{\hat{z}}(-\phi)) \\
&\leq E(R_{\hat{z}}(\phi), R_{\hat{z}}(\phi)) + E(R_{\hat{y}}(\theta), R_{\hat{y}}(\theta)) + E(R_{\hat{z}}(\alpha), R_{\hat{z}}(\beta)) \\
&\quad + E(R_{\hat{y}}(-\theta), R_{\hat{y}}(-\theta)) + E(R_{\hat{z}}(-\phi), R_{\hat{z}}(-\phi)) \\
&= E(R_{\hat{z}}(\alpha), R_{\hat{z}}(\beta)) \\
&= E(R_{\hat{z}}(\alpha), R_{\hat{z}}(\alpha)R_{\hat{z}}(\beta - \alpha)) \\
&\leq E(R_{\hat{z}}(\alpha), R_{\hat{z}}(\alpha)) + E(I, R_{\hat{z}}(\beta - \alpha)) \\
&= E(I, R_{\hat{z}}(\beta - \alpha))
\end{aligned}$$

Now, take a unit vector $\mathbf{v} \in \mathbb{C}^2$ at random. It can be written as:

$$\mathbf{v} = e^{i\gamma} \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix}$$

Recall that the rotation about the z-axis can be written as:

$$R_{\hat{z}}(\chi) = \cos\left(\frac{\chi}{2}\right) I - i\sin\left(\frac{\chi}{2}\right) Z = \begin{bmatrix} e^{-i\frac{\chi}{2}} & 0 \\ 0 & e^{i\frac{\chi}{2}} \end{bmatrix}$$

Applying the rotation about the z-axis to the vector $\mathbf{v}$ yields:

$$R_{\hat{z}}(\beta - \alpha)\mathbf{v} = e^{i\gamma} \begin{bmatrix} e^{-i\frac{\beta-\alpha}{2}} & 0 \\ 0 & e^{i\frac{\beta-\alpha}{2}} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix} = e^{i\gamma} \begin{bmatrix} e^{-i\frac{\beta-\alpha}{2}}\cos\frac{\theta}{2} \\ e^{i\left(\phi+\frac{\beta-\alpha}{2}\right)}\sin\frac{\theta}{2} \end{bmatrix}$$

So, the error in the vector $\mathbf{v}$, induced by the rotation, is given by:

$$\begin{aligned}
\mathbf{v} - R_{\hat{z}}(\beta - \alpha)\mathbf{v} &= e^{i\gamma}\left(\begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix} - \begin{bmatrix} e^{-i\frac{\beta-\alpha}{2}}\cos\frac{\theta}{2} \\ e^{i\left(\phi+\frac{\beta-\alpha}{2}\right)}\sin\frac{\theta}{2} \end{bmatrix}\right) \\
&= e^{i\gamma} \begin{bmatrix} \cos\frac{\theta}{2}\left(1 - e^{-i\frac{\beta-\alpha}{2}}\right) \\ \sin\frac{\theta}{2}e^{i\phi}\left(1 - e^{i\frac{\beta-\alpha}{2}}\right) \end{bmatrix}
\end{aligned}$$

The length of this vector is given by:

$$\begin{aligned}
||\mathbf{v} - R_{\hat{z}}(\beta - \alpha)\mathbf{v}||^2 &= \left|\cos\frac{\theta}{2}\left(1 - e^{-i\frac{\beta-\alpha}{2}}\right)\right|^2 + \left|\sin\frac{\theta}{2}e^{i\phi}\left(1 - e^{i\frac{\beta-\alpha}{2}}\right)\right|^2 \\
&= \cos^2\frac{\theta}{2}\left|1 - e^{-i\frac{\beta-\alpha}{2}}\right|^2 + \sin^2\frac{\theta}{2}\left|1 - e^{i\frac{\beta-\alpha}{2}}\right|^2 \\
&= \left|1 - e^{i\frac{\beta-\alpha}{2}}\right|^2 = \left|1 - \cos\frac{\beta-\alpha}{2} - i\sin\frac{\beta-\alpha}{2}\right|^2 \\
&= \left(1 - \cos\frac{\beta-\alpha}{2}\right)^2 + \sin^2\frac{\beta-\alpha}{2} \\
&= 1 - 2\cos\frac{\beta-\alpha}{2} + \cos^2\frac{\beta-\alpha}{2} + \sin^2\frac{\beta-\alpha}{2} \\
&= 2 - 2\cos\frac{\beta-\alpha}{2} = 4\sin^2\frac{\beta-\alpha}{4}
\end{aligned}$$

So, we find that the error is given by:

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\beta)) = 2\left|\sin\frac{\beta-\alpha}{4}\right|$$

Using the approximation $|\sin(x)| \leq |x|$, we obtain:

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\beta)) \leq 2\left|\frac{\beta-\alpha}{4}\right| = \frac{|\beta-\alpha|}{2}$$

Hence, we have proven the theorem. □

4.3.4 Approximation of single qubit gates by Hadamard and $\pi/8$ gates

Now we have collected all ingredients to show that any single qubit gate can be approximated by Hadamard and $\pi/8$ gates up to arbitrary accuracy. Recall that the matrix representation of the $\pi/8$ gate is denoted by T. We note:

$$\begin{aligned} H &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}}(X+Z) \\ T &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = e^{i\frac{\pi}{8}}\begin{bmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{bmatrix} = e^{i\frac{\pi}{8}}\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right) \end{aligned}$$

We can now use the relations of Pauli matrices, $XY = iZ$, $YZ = iX$, $ZX = iY$, $YX = -iZ$, $ZY = -iX$ and $XZ = -iY$ to rewrite HTH:

$$\begin{aligned} HTH &= e^{i\frac{\pi}{8}} \cdot \frac{1}{2}(X+Z)\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right)(X+Z) \\ &= e^{i\frac{\pi}{8}} \cdot \frac{1}{2}\left(\cos\left(\frac{\pi}{8}\right)X - \sin\left(\frac{\pi}{8}\right)Y + \cos\left(\frac{\pi}{8}\right)Z - i\sin\left(\frac{\pi}{8}\right)I\right)(X+Z) \\ &= e^{i\frac{\pi}{8}} \cdot \frac{1}{2}\left(\cos\left(\frac{\pi}{8}\right)I + i\sin\left(\frac{\pi}{8}\right)Z + i\cos\left(\frac{\pi}{8}\right)Y - i\sin\left(\frac{\pi}{8}\right)X\right. \\ &\qquad \left. -i\cos\left(\frac{\pi}{8}\right)Y - i\sin\left(\frac{\pi}{8}\right)X + \cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right) \\ &= e^{i\frac{\pi}{8}}\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)X\right) \end{aligned}$$

We can use this result to rewrite $HTHT$ and $THTH$.

$$\begin{aligned} HTHT &= e^{i\frac{\pi}{4}}\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)X\right)\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right) \\ &= e^{i\frac{\pi}{4}}\left(\cos^2\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)X - i\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)Z + i\sin^2\left(\frac{\pi}{8}\right)Y\right) \\ &= e^{i\frac{\pi}{4}}\left(\cos^2\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)\left[\cos\left(\frac{\pi}{8}\right)X - \sin\left(\frac{\pi}{8}\right)Y + \cos\left(\frac{\pi}{8}\right)Z\right]\right) \\ THTH &= e^{i\frac{\pi}{4}}\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right)\left(\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)X\right) \\ &= e^{i\frac{\pi}{4}}\left(\cos^2\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)Z - i\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)X - i\sin^2\left(\frac{\pi}{8}\right)Y\right) \\ &= e^{i\frac{\pi}{4}}\left(\cos^2\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)\left[\cos\left(\frac{\pi}{8}\right)X + \sin\left(\frac{\pi}{8}\right)Y + \cos\left(\frac{\pi}{8}\right)Z\right]\right) \end{aligned}$$

We define the vectors $\mathbf{n} = (\cos(\pi/8), -\sin(\pi/8), \cos(\pi/8))$ and $\mathbf{m} = (\cos(\pi/8), \sin(\pi/8), \cos(\pi/8))$. The corresponding unit vectors are denoted by $\hat{n}$ and $\hat{m}$. We define χ by the following equation:

$$\cos\left(\frac{\chi}{2}\right) = \cos^2\left(\frac{\pi}{8}\right)$$

Note that $\frac{\chi}{2\pi}$ is an irrational number. This is proven in appendix A. Now, we find the following relations:

$$HTHT = e^{i\frac{\pi}{4}}R_{\hat{n}}(\chi) \qquad \text{and} \qquad THTH = e^{i\frac{\pi}{4}}R_{\hat{m}}(\chi)$$

The angle between $\hat{n}$ and $\hat{m}$ is equal to the angle between $\mathbf{n}$ and $\mathbf{m}$. If we denote this angle by θ, we find:

$$\theta = \text{acos}\left(\frac{\mathbf{n}\cdot\mathbf{m}}{||\mathbf{n}||\cdot||\mathbf{m}||}\right) \approx 32.6^\circ$$

As $\theta < \frac{\pi}{2}$, we find when we apply the decomposition theorem of rotations about arbitrary axes:

$$m = \left\lceil \frac{\pi}{2\theta} \right\rceil = 3$$

Hence, for any single qubit gate, we can write its matrix representation U in the following form:

$$U = e^{i\alpha} R_{\hat{n}}(\alpha_4) R_{\hat{m}}(\beta_3) R_{\hat{n}}(\alpha_3) R_{\hat{m}}(\beta_2) R_{\hat{n}}(\alpha_2) R_{\hat{m}}(\beta_1) R_{\hat{n}}(\alpha_1)$$

Take $\varepsilon > 0$. Then if we can approximate $R_{\hat{n}}(\alpha)$ and $R_{\hat{m}}(\beta)$ for arbitrary α and β up to a distance of $\varepsilon/7$, we find, if we denote the approximation with an overline:

$$\begin{aligned}
& E\left(\overline{R_{\hat{n}}(\alpha_4)} \cdot \prod_{i=1}^{3} \overline{R_{\hat{m}}(\beta_i)} \cdot \overline{R_{\hat{n}}(\alpha_i)}, R_{\hat{n}}(\alpha_4) \prod_{i=1}^{3} R_{\hat{m}}(\beta_i) R_{\hat{n}}(\alpha_i)\right) \\
= \;& E(\overline{R_{\hat{n}}(\alpha_4)}, R_{\hat{n}}(\alpha_4)) + \sum_{i=1}^{3} \left(E(\overline{R_{\hat{m}}(\beta_i)}, R_{\hat{m}}(\beta_i)) + E(\overline{R_{\hat{n}}(\alpha_i)}, R_{\hat{n}}(\alpha_i)) \right) \\
< \;& \frac{\varepsilon}{7} + 3 \cdot \left(\frac{\varepsilon}{7} + \frac{\varepsilon}{7} \right) = \varepsilon
\end{aligned}$$

Hence, if we are able to find approximations for $R_{\hat{n}}(\alpha)$ and $R_{\hat{m}}(\beta)$, with arbitrary accuracy, then we can approximate U with arbitrary accuracy as well, up to an unimportant constant. The following lemma covers just that.

Lemma 4.24: Density of multiples of χ in $[0, 2\pi)$
We consider χ defined by the equation

$$\cos\left(\frac{\chi}{2}\right) = \cos^2\left(\frac{\pi}{8}\right)$$

Then we define Q the set of all positive multiples of χ modulo 2π. Hence:

$$Q = \{k\chi \mod 2\pi : k \in \mathbb{N}_0\}$$

Then Q is dense in $[0, 2\pi)$.

Proof: First of all, note that $\frac{\chi}{2\pi} \notin \mathbb{Q}$. This is proven in appendix A. This implies that a function given by $k \mapsto k\chi \mod 2\pi$ is injective. Suppose, namely, that it is not. Then there would exist a $q \in Q$ such that $k\chi \equiv l\chi \mod 2\pi$, for $k, l \in \mathbb{N}_0$ and $k \neq l$. Hence, $(k-l)\chi \equiv 0 \mod 2\pi$, which implies that $(k-l)\chi = 2m\pi$, with $m \in \mathbb{Z}$. But then $\frac{\chi}{2\pi} = \frac{m}{k-l}$, which would imply that $\frac{\chi}{2\pi} \in \mathbb{Q}$. This is a contradiction.

Now, take $m \in \mathbb{N}$ at random. We divide $[0, 2\pi)$ into m intervals: $[0, \frac{2\pi}{m})$, $[\frac{2\pi}{m}, \frac{4\pi}{m})$, etc. Now consider the set $Q_m = \{k\chi \mod 2\pi : 0 \leq k \leq m, k \in \mathbb{N}_0\}$. Then $Q_m \subseteq Q$ and the number of elements in Q_m is $m+1$. Hence, there is at least one interval that at least contains two elements of Q_m (this is called the pigeonhole principle). So, there exists a $0 \leq k \leq m-1$, and distinct $i, j \in \{0, \ldots, m\}$ such that $\frac{2k\pi}{m} \leq i\chi, j\chi < \frac{2(k+1)\pi}{m} \mod 2\pi$. Hence, we find $|i\chi - j\chi| < \frac{2\pi}{m} \mod 2\pi$.

Now we consider two cases. Suppose that $i\chi > j\chi \mod 2\pi$ and $i > j$. Then $(i-j)\chi \mod 2\pi \in Q$ and hence $\{k(i-j)\chi \mod 2\pi : k \in \mathbb{N}_0\} \subseteq Q$. Hence, a random point $a \in [0, 2\pi)$ is never further than $\frac{2\pi}{m}$ away from a point in Q.

Now suppose $i\chi > j\chi \mod 2\pi$ and $i < j$. Then $-\frac{2\pi}{m} < (j-i)\chi < 0 \mod 2\pi$. Furthermore, $\{k(j-i)\chi \mod 2\pi : k \in \mathbb{N}_0\} \subseteq Q$. Hence, again, a random point $a \in [0, 2\pi)$ is never further than $\frac{2\pi}{m}$ away from a point in Q.

So, in both cases, we find that we can approximate $a \in [0, 2\pi)$ at a distance no larger than $\frac{2\pi}{m}$. As we chose m at random, we now find that we can approximate a arbitrarily close. So Q is dense in $[0, 2\pi)$. $\square$

This theorem tells us that for any $\alpha \in [0, 2\pi)$ and $\varepsilon > 0$, we can find a $N \in \mathbb{N}_0$ such that $|\alpha - (N\chi \mod 2\pi)| < 2\varepsilon$. Hence:

$$E\left(R_{\hat{n}}(\alpha), (HTHT)^N\right) = E(R_{\hat{n}}(\alpha), R_{\hat{n}}(N\chi \mod 2\pi)) \leq \frac{|\alpha - (N\chi \mod 2\pi)|}{2} < \varepsilon$$

Similarly, for any $\beta \in [0, 2\pi)$, we can find a $N \in \mathbb{N}_0$ such that $|\beta - (N\chi \mod 2\pi)| < 2\varepsilon$. This yields:

$$E\left(R_{\hat{m}}(\beta), (THTH)^N\right) = E(R_{\hat{m}}(\beta), R_{\hat{m}}(N\chi \mod 2\pi)) \leq \frac{|\beta - (N\chi \mod 2\pi)|}{2} < \varepsilon$$

Hence, we can approximate $R_{\hat{n}}(\alpha)$ and $R_{\hat{m}}(\beta)$ arbitrarily close. This completes the proof that the CNOT, Hadamard and $\pi/8$ gates are universal.

In this chapter, we have proven that we can in principle implement any quantum circuit using only a finite set of quantum gates. It must be mentioned, though, that it is hard to predict how many times the quantum gates of this finite set need to be used. It turns out that this is not a very efficient way of implementing arbitrary quantum gates, so if one wants to invent an efficient algorithm for a quantum computer, then this person should look for other solutions to implement the circuit.

5 Shor's algorithm

In chapters 2 and 3, we have had a brief introduction into the realm of quantum computing. In chapter 4, we used this to prove the universality of a discrete set of quantum gates. We saw that a construction using this set of gates does not necessarily yield an efficient implementation.

In this chapter, we will look at the factorization problem, and particularly at how Shor's algorithm can be used to tackle this problem. Moreover, we will prove that with a quantum computer, Shor's algorithm provides an efficient way of factoring integers, which is believed to be impossible on a classical computer.

In order to understand how Shor's algorithm works, we must first elaborate on the techniques that are used in this algorithm. These will be covered one by one in the following sections. The first section will cover the quantum Fourier transform (5.1), which is followed by the phase estimation algorithm (5.2). Then we will continue with the order finding algorithm (5.3), which will finally be implemented in the factorization algorithm (5.4), invented by Shor.

5.1 Quantum Fourier transform

In physics, the discrete Fourier transform is one of the most frequently used tools to analyze data. Suppose that this data is given by N complex numbers, denoted by $x_0, \ldots, x_{N-1} \in \mathbb{C}$. The discrete Fourier transform maps these numbers to N different complex numbers $y_0, \ldots, y_{N-1} \in \mathbb{C}$, by the following formula:

$$y_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{\frac{2\pi i j k}{N}}$$

In quantum computation, this transformation can be used in the following way. Suppose we have an n-qubit system. Then the state vector of this system can be written as a superposition of the computational basis states $|0\rangle$ through $|2^n - 1\rangle$. Now interpret all the coefficients of these computational basis states as the data, and apply the Fourier transform to these coefficients. This operation is referred to as the quantum Fourier transform. Hence, the quantum Fourier transform has the following effect on an arbitrary state vector, where y_k is defined as before, with $N = 2^n$.

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \mapsto \sum_{k=0}^{2^n-1} y_k |k\rangle$$

This operation is only allowed if it is unitary, which is proven in the lemma and theorem below.

Lemma 5.1: Sum of the roots of unity
Suppose $N \in \mathbb{N}$ and $\zeta_k = e^{\frac{2\pi i k}{N}}$, with $0 \leq k \leq N-1$. Then all ζ_k's are referred to as the Nth roots of unity, and they sum to 0:

$$\sum_{k=0}^{N-1} e^{\frac{2\pi i k}{N}} = 0$$

Proof: Consider the polynomial $x^N - 1$. The roots of this polynomial are precisely the Nth roots of unity. Hence, according to the fundamental theorem of algebra, one can rewrite $x^N - 1$ in the following way:

$$x^N - 1 = \prod_{k=0}^{N-1} \left(x - e^{\frac{2\pi i k}{N}} \right)$$

Expanding the brackets on the right side of the equation yields a polynomial of degree N. The linear

coefficient of this polynomial equals to the following:

$$-\sum_{k=0}^{N-1} e^{\frac{2\pi ik}{N}}$$

But the polynomial on the left and right hand side of the equation are equal, hence their coefficients are equal as well. So, we find that:

$$\sum_{k=0}^{N-1} e^{\frac{2\pi ik}{N}} = 0$$

as required. □

Theorem 5.2: Unitarity of the quantum Fourier transform
The quantum Fourier transform, applied to $n \in \mathbb{N}$ qubits, is a unitary operation.

Proof: Define $\omega = e^{\frac{2\pi i}{2^n}}$. Then the quantum Fourier transform has the following effect on an arbitrary quantum state:

$$\sum_{j=0}^{2^n-1} x_j|j\rangle \mapsto \sum_{k=0}^{2^n-1} y_k|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left(\sum_{j=0}^{2^n-1} x_j \omega^{jk} \right) |k\rangle$$

Hence, if we denote the matrix representation of the quantum Fourier transform by U, then U is a $2^n \times 2^n$ matrix, with the following entries. (For convenience, we let the indices of the entries run from 0 to $2^n - 1$.)

$$U_{kj} = \frac{1}{\sqrt{2^n}} \omega^{jk} \qquad (0 \leq j, k \leq 2^n - 1)$$

In order to prove that the quantum Fourier transform is a unitary operation, we must prove that U is a unitary matrix. In other words, we must prove that $UU^* = U^*U = I$. Let's first consider the entries of U^*. We will use the following relation: $\omega^* = (e^{\frac{2\pi i}{2^n}})^* = e^{-\frac{2\pi i}{2^n}} = \omega^{-1}$. We find:

$$(U^*)_{jk} = U^*_{kj} = \frac{1}{\sqrt{2^n}} (\omega^*)^{jk} = \frac{1}{\sqrt{2^n}} \omega^{-jk} \qquad (0 \leq j, k \leq 2^n - 1)$$

Hence, we find the following formula for the entries of UU^*:

$$(UU^*)_{jk} = \sum_{l=0}^{2^n-1} U_{jl}(U^*)_{lk} = \frac{1}{2^n} \sum_{l=0}^{2^n-1} \omega^{jl}\omega^{-kl} = \frac{1}{2^n} \sum_{l=0}^{2^n-1} \left(\omega^{j-k}\right)^l \qquad (0 \leq j, k \leq 2^n - 1)$$

So, if $j = k$, we find:

$$(UU^*)_{jk} = \frac{1}{2^n} \sum_{l=0}^{2^n-1} 1^l = \frac{1}{2^n} \cdot 2^n = 1$$

If, on the other hand, $j \neq k$, we define $1 \leq m \leq 2^n - 1$ such that $j - k \equiv m \mod 2^n$. Then we have:

$$(UU^*)_{jk} = \frac{1}{2^n} \sum_{l=0}^{2^n-1} (\omega^m)^l$$

Note that the additive group generated by m modulo 2^n has a least non-zero element, which is the greatest common divisor of m and 2^n. The order of this group is denoted by O:

$$O = \frac{2^n}{\gcd(2^n, m)}$$

Hence, we find:

$$(UU^*)_{kl} = \frac{1}{2^n} \cdot \gcd(2^n, m) \sum_{l=0}^{O-1} \omega^{\gcd(2^n,m)l} = \frac{1}{O} \sum_{l=0}^{O-1} e^{\frac{2\pi i l}{O}} = 0$$

This last summation equals 0 according to the previous lemma. Hence we find that all the elements on the diagonal of UU^* are 1 and all the others equal 0. So $UU^* = I$. An identical argument shows that $U^*U = I$ as well. So, U is indeed unitary. $\square$

It is now interesting to see how a computational basis state is affected by the quantum Fourier transform. For this, we introduce the following notation. For $m, n \in \mathbb{N}_0$ and $j_{-m}, \ldots, j_n \in \{0, 1\}$ we define:

$$(j_n \ldots j_1 j_0 . j_{-1} j_{-2} \ldots j_{-m})_2 = \sum_{k=-m}^{n} j_k 2^k$$

The left hand side is called the binary representation of $j \in \mathbb{R}$ if the sum on the right hand side equals j. If we use this notation to express the computational basis states, we observe that for example the following holds, as we would expect:

$$|101\rangle = |5\rangle = |(101)_2\rangle$$

The result of the quantum Fourier transform, when it is applied to a computational basis state, is investigated in the following theorem.

Theorem 5.3: Qauntum Fourier transform applied to computational basis states
Suppose we have an n-qubit system. Let j be a random integer between 0 and $2^n - 1$, and let $j_{n-1} \ldots j_0$ be its binary representation. When the quantum Fourier transform is applied to $|j\rangle$, the result is given by:

$$\frac{\left(|0\rangle + e^{2\pi i (0.j_0)_2} |1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i (0.j_1 j_0)_2} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i (0.j_{n-1} \ldots j_1 j_0)_2} |1\rangle\right)}{\sqrt{2^n}}$$

Before giving the actual proof, let's consider an example. Specifically, let's have a look at what happens to $|2\rangle = |10\rangle$ in a 2-qubit system when the quantum Fourier transform is applied.

We denote the initial state of the system by $\sum_{j=0}^{3} x_j |j\rangle$ and the final state by $\sum_{k=0}^{3} y_k |k\rangle$. We find that $x_j = 1$ if $j = 2$ and $x_j = 0$ otherwise. This yields the following values for y_k:

$$y_k = \sum_{l=0}^{3} x_l e^{\frac{2\pi i k l}{4}} = e^{\frac{2\pi i \cdot 2 \cdot k}{4}}$$

Hence, the new state $|\psi\rangle$ is given by the following expression, where $k_1 k_0$ denotes the binary representation of k:

$$|\psi\rangle = \sum_{k=0}^{3} y_k |k\rangle = \frac{1}{2} \sum_{k=0}^{3} e^{\frac{2\pi i \cdot 2 \cdot k}{4}} |k_1 k_0\rangle$$

Expanding the summation now yields:

$$|\psi\rangle = \frac{1}{2} \left(|00\rangle + e^{\frac{2\pi i \cdot 2 \cdot 1}{4}} |01\rangle + e^{\frac{2\pi i \cdot 2 \cdot 2}{4}} |10\rangle + e^{\frac{2\pi i \cdot 2 \cdot 3}{4}} |11\rangle \right)$$

The key idea is to split the computational basis states into their single qubit tensor products, and associate the sign factors with the individual qubit states that have value $|1\rangle$, as follows:

$$|\psi\rangle = \frac{1}{2} \left[|0\rangle \otimes |0\rangle + |0\rangle \otimes \left(e^{\frac{2\pi i \cdot 2 \cdot 1}{4}} |1\rangle \right) + \left(e^{\frac{2\pi i \cdot 2 \cdot 2}{4}} |1\rangle \right) \otimes |0\rangle + \left(e^{\frac{2\pi i \cdot 2 \cdot 2}{4}} |1\rangle \right) \otimes \left(e^{\frac{2\pi i \cdot 2 \cdot 1}{4}} |1\rangle \right) \right]$$

Now, notice that this can be written in product form, as follows:

$$|\psi\rangle = \frac{1}{2}\left(|0\rangle + e^{\frac{2\pi i \cdot 2 \cdot 2}{4}}|1\rangle\right) \otimes \left(|0\rangle + e^{\frac{2\pi i \cdot 2 \cdot 1}{4}}|1\rangle\right)$$

Writing $\frac{2}{4}$ in binary notation in the exponent yields $(0.10)_2$. Recall that this 10 comes from the original state to which we are applying the quantum Fourier transform.

$$|\psi\rangle = \frac{1}{2}\left(|0\rangle + e^{2\pi i \cdot (0.10)_2 \cdot 2}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \cdot (0.10)_2 \cdot 1}|1\rangle\right)$$

Now in the first factor, the exponent is multiplied by 2, hence the binary representation is shifted to the left by one digit. Only the fractional part remains, as $e^{2\pi i}$ equals 1 and vanishes accordingly. Thus:

$$|\psi\rangle = \frac{1}{2}\left(|0\rangle + e^{2\pi i \cdot (0.0)_2}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \cdot (0.10)_2}|1\rangle\right)$$

This is exactly the form as predicted by the theorem. The proof below uses the exact same steps, but the notation is somewhat cumbersome.

Proof: The result follows directly from the definition of the quantum Fourier transform. We denote the initial state of the system by $\sum_{k=0}^{2^n-1} x_k|k\rangle$ and the final state by $\sum_{k=0}^{2^n-1} y_k|k\rangle$. If the initial state is $|j\rangle$, then x_k equals 1 if $k = j$ and 0 otherwise. Hence, the resulting coefficients y_k are:

$$y_k = \sum_{l=0}^{2^n-1} x_l e^{\frac{2\pi i k l}{2^n}} = e^{\frac{2\pi i j k}{2^n}}$$

Hence, the resulting state can be written as follows. In the repeated tensor product, l runs from $n-1$ to 0, with steps of -1.

$$\begin{aligned}
\sum_{k=0}^{2^n-1} y_k|k\rangle &= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}}|k\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} e^{\frac{2\pi i j (k_{n-1}\ldots k_0)_2}{2^n}}|k_{n-1}\ldots k_0\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{k_{n-1}=0}^{1} \sum_{k_0=0}^{1} \bigotimes_{l=n-1}^{0} e^{2\pi i j k_l \cdot 2^{l-n}}|k_l\rangle \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{l=n-1}^{0} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l \cdot 2^{l-n}}|k_l\rangle\right] \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{l=n-1}^{0} \left[|0\rangle + e^{2\pi i j \cdot 2^{l-n}}|k_l\rangle\right] \\
&= \frac{1}{\sqrt{2^n}} \bigotimes_{l=n-1}^{0} \left[|0\rangle + e^{2\pi i (0.j_{n-1}\ldots j_1 j_0)_2 \cdot 2^l}|k_l\rangle\right] \\
&= \frac{\left(|0\rangle + e^{2\pi i (0.j_0)_2}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i (0.j_1 j_0)_2}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i (0.j_{n-1}\ldots j_1 j_0)_2}|1\rangle\right)}{\sqrt{2^n}}
\end{aligned}$$

as required. □

The above representation of the quantum Fourier transform of a computational basis state is referred to as the product form. As we can see, taking the Fourier transform of a computational basis state does not yield an entangled state, but it does leave all qubits in superposition.

Using the previous theorem, we can now come up with an efficient implementation of the quantum Fourier transform performed on n qubits. The idea will be to manipulate the qubits one by one to construct the product form in steps. The final state of the last qubit, though, will be prepared in the first qubit. Similarly, the final state of the second to last qubit will be constructed in the second qubit. As a last step, the states of all qubits are swapped (except perhaps for the middle one if the number of qubits is odd).

We will need the Hadamard gate, and the following set of n gates, with matrix notations denoted by $R_1, \ldots, R_n$:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix} \qquad (1 \leq k \leq n)$$

Note that $R_1 = Z$, and $R_2 = S$ and $R_3 = T$.

The construction of the quantum Fourier transform, then, is done as follows. We suppose that the system of n qubits is initially in one of the compuational basis states, represented by $|j\rangle = |j_{n-1} \ldots j_1 j_0\rangle$. As the operation is linear and the computational basis states span the state space, checking the validity of the operation on the computational basis vectors is sufficient for checking the validity of the operation.

First of all, we apply a Hadamard gate on the first qubit. This yields the following state:

$$\Big(|0\rangle + e^{2\pi i (0.j_{n-1})_2}|1\rangle\Big) \otimes |j_{n-2} \ldots j_1 j_0\rangle$$

Applying the R_2 gate on the first qubit, conditional on the second qubit, yields the following state:

$$\Big(|0\rangle + e^{2\pi i (0.j_{n-1} j_{n-2})_2}|1\rangle\Big) \otimes |j_{n-2} \ldots j_1 j_0\rangle$$

Repeating this process by applying R_3 to the first qubit, conditional on the third qubit, will add another $2\pi i(0.00 j_{n-3})$ to the exponent of the $|1\rangle$-state of the first qubit. In a similar manner, the state can be brought into the following form:

$$\Big(|0\rangle + e^{2\pi i (0.j_{n-1} \ldots j_1 j_0)_2}|1\rangle\Big) \otimes |j_{n-2} \ldots j_1 j_0\rangle$$

Note that from the product form, we see that the state of the first qubit is now equal to the state the last qubit will have to be in at the end of the quantum Fourier transform. We will leave the first qubit like this until the very end, when we will swap the first and last qubits.

Continuing in a similar manner for the second qubit, we can bring the state to the following form:

$$\Big(|0\rangle + e^{2\pi i (0.j_{n-1} \ldots j_1 j_0)_2}|1\rangle\Big) \otimes \Big(|0\rangle + e^{2\pi i (0.j_{n-2} \ldots j_1 j_0)_2}|1\rangle\Big) \otimes |j_{n-3} \cdots j_1 j_0\rangle$$

This process can be repeated on all qubits, leaving us with the following state:

$$\Big(|0\rangle + e^{2\pi i (0.j_{n-1} \ldots j_1 j_0)_2}|1\rangle\Big) \otimes \cdots \otimes \Big(|0\rangle + e^{2\pi i (0.j_1 j_0)_2}|1\rangle\Big) \otimes \Big(|0\rangle + e^{2\pi i (0.j_0)_2}|1\rangle\Big)$$

All that is left to do is swapping the first, second, third, etc. qubits with the last, second to last, third to last, etc. qubits, respectively. We have already seen a way to implement this at the end of chapter 3, using three CNOT gates. Hence, we have found a way to implement the quantum Fourier transform. A quantum circuit showing the process described above is included in figure 5.1.

As we would like to implement this circuit efficiently, it makes sense to have a look at the complexity of this algorithm. Specifically, if every quantum gate is an operation that takes a certain amount of time, money, ingenuity, or resource in general, then it makes sense to find the number of gates as a function of the number of qubits that need to be transformed.

For this implementation of the quantum Fourier transform, we need n gates to prepare the state of the first qubit. Then we need $n - 1$ more gates to manipulate the state of the second qubit. In total we need

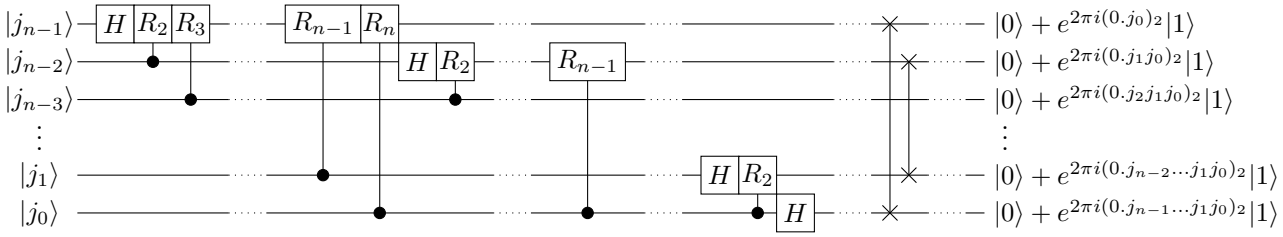


Figure 5.1: Implementation of the quantum Fourier transform, performed on n qubits. The time complexity of this implementation is $\mathcal{O}(n^2)$.

$n+(n-1)+\cdots+1 = n(n+1)/2$ gates for the first part. Next, we need to perform at most $n/2$ swaps, each of which costs 3 CNOT gates. Hence, the complexity of this implementation of the quantum Fourier transform is given by $\mathcal{O}(n^2)$. So, it is an efficient algorithm, as it scales polynomially with the number of qubits. If we take into account that there are $N = 2^n$ data points that are transformed, then we see that the complexity is $\mathcal{O}(\log^2(N))$. The problem, though, is that the output of the quantum Fourier transform is not readily accessible. It is "hidden" as the coefficients of the computational basis states, and so the full potential of the quantum Fourier transform can only be harnessed if that data is used in a smart way afterwards.

As a final remark, note that we can use this result to find an equally efficient algorithm to implement the inverse quantum Fourier transform. After all, all quantum gates are unitary, and can therefore easily be reversed. Hence, applying the inverse of all quantum gates above in opposite order yields an efficient implementation of the inverse quantum Fourier transform.

5.2 Phase estimation algorithm

In this section, we will find the first application of the quantum Fourier transform. In fact, it will be the most important application for our purposes. The quantum Fourier transform, namely, is used in the phase estimation algorithm, as is explained in this section.

Suppose we have a system with $n+t$ qubits. We refer to the first t qubits as the first register, and to the last n qubits as the second register. There is also an operation with matrix representation U available that acts on the second register. The precise implementation of this operation is unknown, but we do know that the vector representation of the state $|u\rangle$, which is a state of n qubits, is an eigenvector of U. As U is unitary, we know that the eigenvalue associated with $|u\rangle$ can be written as $e^{2\pi i\phi}$ with $0 \leq \phi < 1$. The goal of the phase estimation algorithm is to find the value of ϕ. See also figure 5.2.

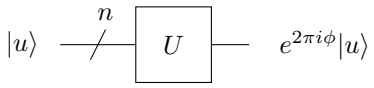


Figure 5.2: If an eigenstate of U, $|u\rangle$, is manipulated by the operation U itself, then the resulting state will only differ by a phase constant, written as $e^{2\pi i\phi}$. The goal of the phase estimation algorithm is to determine ϕ.

The idea is to approximate the value of $2^t\phi$ in the t qubits of the first register. First of all, these first t qubits are put in the computational basis state $|0\rangle$, and the second register is prepared in the state $|u\rangle$. Then the first register is put into superposition, using Hadamard gates. Afterwards, the operation with matrix representation U is performed 2^j times, conditional on the $(t-j)$th qubit. Afterwards, the inverse Fourier transform is applied to the first register. Measuring the first register now yields an approximation of $2^t\phi$. See also figure 5.3.

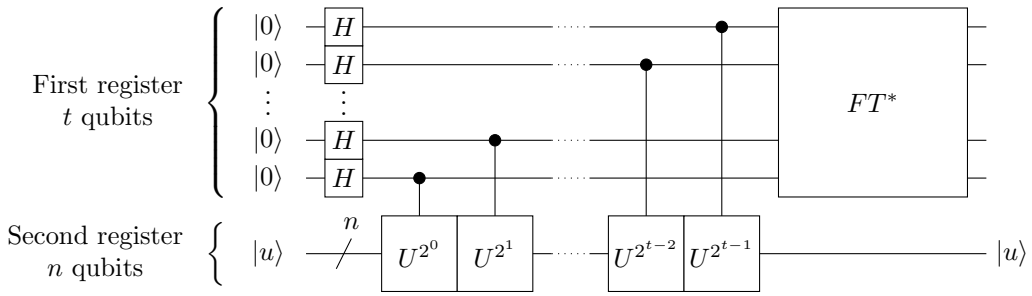


Figure 5.3: Implementation of the phase estimation algorithm. The output of the first register is too complicated to be shown in this figure, but according to theorem 5.5, a measurement in the computational basis will yield an accurate estimate of $2^t\phi$ with high probability.

The initial state of the system is: $|0\rangle^{\otimes t} \otimes |u\rangle$. After performing the Hadamard gates, all qubits of the first register are in superposition states:

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes t} \otimes |u\rangle = \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t} \otimes |u\rangle$$

Now, applying the first conditional operation yields the following state:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes (t-1)} \otimes (|0\rangle + e^{2\pi i\phi}|1\rangle) \otimes |u\rangle$$

Combining this with all the other conditional operations yields the following state:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i\phi \cdot 2^{t-1}}|1\rangle) \otimes (|0\rangle + e^{2\pi i\phi \cdot 2^{t-2}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i\phi}|1\rangle) \otimes |u\rangle$$

Now suppose that ϕ can be written exactly as $(0.\phi_{t-1}\phi_{t-2}\ldots\phi_1\phi_0)_2$. Then the state before the inverse Fourier transform can be written as:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i(0.\phi_0)_2}|1\rangle) \otimes (|0\rangle + e^{2\pi i(0.\phi_1\phi_0)_2}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i(0.\phi_{t-1}\ldots\phi_1\phi_0)_2}|1\rangle) \otimes |u\rangle$$

The state of the first register is exactly the product form of theorem 3, hence applying the inverse Fourier transform yields a computational basis state for the first register. So, the final state becomes:

$$|\phi_{t-1}\phi_{t-2}\ldots\phi_1\phi_0\rangle \otimes |u\rangle$$

Hence, measuring the first register and dividing the resulting value by 2^t yields the exact value of ϕ with probability 1. So, if ϕ can be written exactly as $(0.\phi_{t-1}\phi_{t-2}\ldots\phi_1\phi_0)_2$, then the phase estimation algorithm behaves as expected.

The interesting thing is what happens when the binary form of ϕ does not terminate after t digits. Then one would hope that measuring the first register would yield some approximating value of $2^t\phi$, just like in the exact case. An example is given in figure 5.4, where $\phi = 0.33$ is approximated with $t = 4$ qubits.

The argument can be made rigurous by bounding the total probability of the tails of the probability distribution. This is done in the following theorem. First of all, though, we must determine when an approximating value is close. This is done according to the following definition.

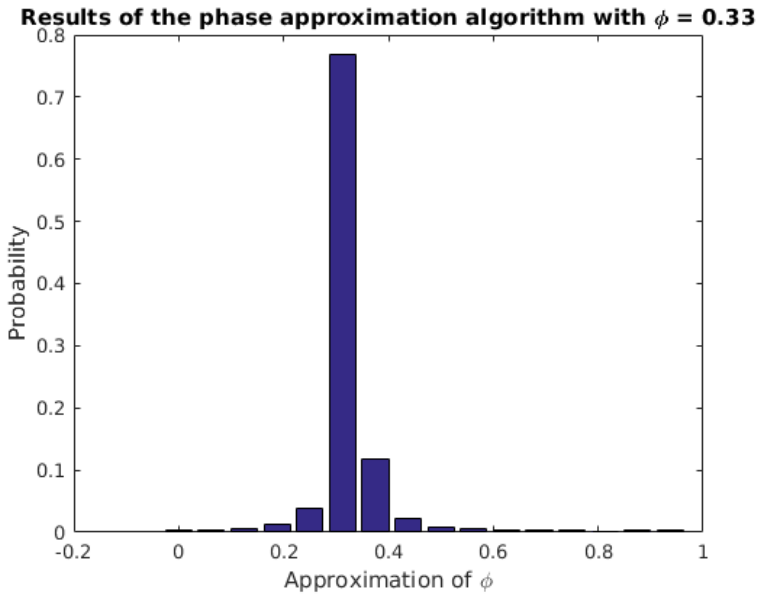


Figure 5.4: Histogram of the approximation of ϕ after completion of the phase estimation algorithm. There is a high probability that the approximation of ϕ will be close to its original value of $\phi = 0.33$. The peak is located at 0.3125.

Definition 5.4: Modulus modulo N
Suppose $N \in \mathbb{N}$. Then for any integer $n \in \mathbb{Z}$, we define its modulus modulo N as follows:

$$||n||_N = \min_{m \in \mathbb{Z}} \{|m| : n \equiv m \mod N\}$$

Furthermore, if we have two integers $n_1, n_2 \in \mathbb{Z}$, then the distance between n_1 and n_2 modulo N is defined by $||n_1 - n_2||_N$.

For example, we find that $||7 - 1||_8 = 2$. This can be visualized by drawing 1 and 7 on a cyclic number line that runs from 0 to 7 periodically. The smallest distance between 1 and 7 is then equal to 2, as predicted by the expression $||7 - 1||_8 = 2$.

It follows trivially that for any $N \in \mathbb{N}$ and $n \in \mathbb{Z}$, we have $||n||_N \leq |n|$. Note also that the triangle inequality holds for this notion of a distance. Namely, suppose that we have $N \in \mathbb{N}$ and $n_1, n_2, n_3 \in \mathbb{Z}$. Then:

$$\begin{aligned}
||n_1 - n_3||_N &= \min_{m_1 \in \mathbb{Z}} \{|m_1| : n_1 - n_3 \equiv m_1 \mod N\} \\
&= \min_{m_1 \in \mathbb{Z}} \{|m_1| : n_1 - n_2 + n_2 - n_3 \equiv m_1 \mod N\} \\
&= \min_{m_2, m_3 \in \mathbb{Z}} \{|m_2 + m_3| : n_1 - n_2 \equiv m_2 \mod N \wedge n_2 - n_3 \equiv m_3 \mod N\} \\
&\leq \min_{m_2 \in \mathbb{Z}} \{|m_2| : n_1 - n_2 \equiv m_2 \mod N\} + \min_{m_3 \in \mathbb{Z}} \{|m_3| : n_2 - n_3 \equiv m_3 \mod N\} \\
&= ||n_1 - n_2||_N + ||n_2 - n_3||_N
\end{aligned}$$

The following theorem, now, uses this distance to characterize the proximity of the approximation of ϕ to the real value of ϕ.

Theorem 5.5: Executing the phase estimation algorithm with a fixed accuracy
Let $0 \leq \phi < 1$, $m, n \in \mathbb{N}$ and $0 < \varepsilon < 1$. Suppose one has an n-qubit quantum gate U, with eigenvalue $e^{2\pi i\phi}$. To approximate ϕ to within a range of 2^{-m} modulo 1 with a probability of at least $1-\varepsilon$, it suffices to execute the phase estimation algorithm with at least

$$t = m + \left\lceil \log_2 \left(2 + \frac{1}{2\varepsilon}\right) \right\rceil$$

qubits in the first register.

Proof: The case where ϕ can be exactly written as $(0.\phi_{t-1}\dots\phi_1\phi_0)_2$ has already been dealt with above. Therefore, we will focus on the case where the binary representation of ϕ does not terminate after t digits.

We start with the last exact result that we obtained above, which is the state before the inverse Fourier transform:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i\phi\cdot 2^{t-1}}|1\rangle) \otimes (|0\rangle + e^{2\pi i\phi\cdot 2^{t-2}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i\phi}|1\rangle) \otimes |u\rangle$$

Expanding the parentheses yields:

$$\frac{1}{\sqrt{2^t}} \left(\sum_{k=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle \right) \otimes |u\rangle$$

Note that the inverse Fourier transform of a computational basis state is given by a formula similar to the ones derived in the previous section. It can be derived easily by looking at the matrix elements of U^* in the proof of theorem 5.2.

$$|k\rangle \mapsto \frac{1}{\sqrt{2^t}} \sum_{l=0}^{2^t-1} e^{-\frac{2\pi ikl}{2^t}}|l\rangle$$

Substituting this yields the following final state of the system:

$$\frac{1}{2^t} \left(\sum_{k=0}^{2^t-1} e^{2\pi i\phi k} \sum_{l=0}^{2^t-1} e^{-\frac{2\pi ikl}{2^t}}|l\rangle \right) \otimes |u\rangle$$

As both sums are finite, they can be interchanged, yielding the following representation of the final state of the system:

$$\frac{1}{2^t} \left(\sum_{l=0}^{2^t-1} \sum_{k=0}^{2^t-1} e^{2\pi i\phi k} e^{-\frac{2\pi ikl}{2^t}}|l\rangle \right) \otimes |u\rangle$$

Let $(0.\phi_{t-1}\dots\phi_1\phi_0\phi_{-1}\dots)_2$ be the binary representation of ϕ. Then define $b = (\phi_{t-1}\dots\phi_1\phi_0)_2$. Hence, b is the largest integer below $2^t\phi$. So, $0 \leq 2^t\phi - b < 1$, which implies $0 \leq \phi - 2^{-t}b < 2^{-t}$.

We expect that the state $|b\rangle$ has the largest coefficient in the summation above, as b is a good approximation of $2^t\phi$. Let's define the coefficients of the computational basis states in the final state of the system relative to b, so we define α_l as the coefficient of $|m\rangle$ with $b + l \equiv m \mod 2^t$. Hence, the final state of the system can be written as:

$$\frac{1}{2^t} \left(\sum_{l=0}^{2^t-1} \sum_{k=0}^{2^t-1} e^{2\pi i\phi k} e^{-\frac{2\pi ik(b+l)}{2^t}}|(b+l) \mod 2^t\rangle \right) \otimes |u\rangle = \sum_{l=0}^{2^t-1} \alpha_l |(b+l) \mod 2^t\rangle \otimes |u\rangle$$

where α_l can be rewritten using the formula $\sum_{n=0}^{m} r^n = \frac{1-r^{m+1}}{1-r}$ for $r \neq 1$. Note that this criterion is met because $2^t\phi \neq b$, as we are only investigating the case where ϕ does not have an exact binary representation

in t digits.

$$\alpha_l = \frac{1}{2^t}\sum_{k=0}^{2^t-1} e^{2\pi i\phi k} e^{-\frac{2\pi i k(b+l)}{2^t}} = \frac{1}{2^t}\sum_{k=0}^{2^t-1}\left(e^{\frac{2\pi i(2^t\phi-(b+l))}{2^t}}\right)^k = \frac{1-e^{2\pi i(2^t\phi-(b+l))}}{2^t\left(1-e^{\frac{2\pi i(2^t\phi-(b+l))}{2^t}}\right)}$$

Note that we can bound the modulus of the numerator of α_l quite easily using the triangle inequality:

$$|1-e^{2\pi i(2^t\phi-(b+l))}| \leq 1+|e^{2\pi i(2^t\phi-(b+l))}| = 1+1 = 2$$

Hence, we obtain:

$$|\alpha_l| \leq \frac{2}{2^t\left|1-e^{\frac{2\pi i(2^t\phi-(b+l))}{2^t}}\right|}$$

For $\theta \in [-\pi, \pi]$, we find:

$$|1-e^{i\theta}|^2 = (1-\cos\theta)^2 + \sin^2\theta = 2-2\cos\theta = \sin^2\frac{\theta}{2}$$

Taking the square root on both sides:

$$|1-e^{i\theta}| = \left|\sin\frac{\theta}{2}\right| \geq \frac{2|\theta|}{\pi}$$

As we have $0 \leq 2^t\phi - b < 1$, we find that for $-2^{t-1}+1 \leq l \leq 2^{t-1}$, the following holds:

$$-\pi = \frac{2\pi\cdot(0-2^{t-1})}{2^t} \leq \frac{2\pi((2^t\phi-b)-l)}{2^t} \leq \frac{2\pi(1-(-2^{t-1}+1))}{2^t} = \pi$$

So, for $-2^{t-1}+1 \leq l \leq 2^{t-1}$, we find:

$$|\alpha_l| \leq \frac{2}{2^t\cdot\frac{2\left|\frac{2\pi((2^t\phi-b)-l)}{2^t}\right|}{\pi}} = \frac{2\pi}{2\cdot 2\pi|(2^t\phi-b)-l|} = \frac{1}{2|(2^t\phi-b)-l|}$$

Now, we can find a bound on the probability that a measurement of the first register yields a value M, such that $||M-b||_{2^t} > e$, where $e > 1$. This is done by considering both tails of the distribution separately.

$$\begin{aligned} P(||M-b||_{2^t} > e) &= \sum_{l=-2^{t-1}+1}^{-(e+1)} |\alpha_l|^2 + \sum_{l=e+1}^{2^{t-1}} |\alpha_l|^2 \\ &\leq \frac{1}{4}\left[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{((2^t\phi-b)-l)^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{((2^t\phi-b)-l)^2}\right] \end{aligned}$$

Using $0 \leq 2^t\phi - b < 1$, we find that:

$$\begin{aligned}
P(||M-b||_{2^t} > e) &\leq \frac{1}{4}\left[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{(0-l)^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{(1-l)^2}\right] \\
&\leq \frac{1}{4}\left[\sum_{l=-2^{t-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{l=e+1}^{2^{t-1}} \frac{1}{(l-1)^2}\right] \\
&\leq \frac{1}{4}\left[\sum_{l=e}^{2^{t-1}-2} \frac{1}{l^2} + \sum_{l=e}^{2^{t-1}-1} \frac{1}{l^2}\right] \leq \frac{1}{2}\sum_{l=e}^{2^{t-1}-1} \frac{1}{l^2} \\
&\leq \frac{1}{2}\int_{e-1}^{2^{t-1}-1} \frac{1}{l^2}\,\mathrm{d}l \leq \frac{1}{2}\int_{e-1}^{\infty} \frac{1}{l^2}\,\mathrm{d}l = \frac{1}{2}\left[-\frac{1}{l}\right]_{e-1}^{\infty} \\
&= \frac{1}{2}\cdot\left[0 + \frac{1}{e-1}\right] = \frac{1}{2(e-1)}
\end{aligned}$$

Hence, if we calculate $\overline{\phi} = 2^{-t}M$, then $\overline{\phi}$ is an approximation of ϕ, such that with a probability of at least $1 - \frac{1}{2(e-1)}$:

$$||\overline{\phi} - \phi||_1 = 2^{-t}||2^t\overline{\phi} - 2^t\phi||_{2^t} \leq 2^{-t}(||M-b||_{2^t} + ||b - 2^t\phi||_{2^t}) < 2^{-t}(e+1)$$

Suppose now that we want to approximate ϕ up to an accuracy of 2^{-m} modulo 1, with a probability of success of at least $1-\varepsilon$. Then the probability of failure must not exceed ε, hence $\varepsilon \geq \frac{1}{2(e-1)}$, or rewriting:

$$e \geq 1 + \frac{1}{2\varepsilon}$$

On the other hand, the approximation must be good enough, which can only be ensured if $2^{-t}(e+1) \leq 2^{-m}$, or:

$$e \leq 2^{t-m} - 1$$

Hence, it is only possible to find a suitable value for e if:

$$1 + \frac{1}{2\varepsilon} \leq 2^{t-m} - 1$$

Namely, if this relation holds, then the left hand side is greater than 1, and thus the value of the right hand side exceeds 1 as well. Hence, $t > m$, so the right hand side is an integer. So, if the above inequality holds, $e = 2^{t-m} - 1$ is a suitable choice.

Rewriting the inequality yields:

$$2^{t-m} \geq 2 + \frac{1}{2\varepsilon} \Leftrightarrow t \geq m + \log_2\left(2 + \frac{1}{2\varepsilon}\right)$$

Hence, it is sufficient to choose t according to the following formula:

$$t = m + \left\lceil \log_2\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$$

This completes the proof. □

We have now found a way to approximate ϕ with arbitrary accuracy and arbitrary small probability of failure. This does require, though, that we are able to prepare the eigenstate $|u\rangle$ of the operation denoted with matrix representation U. In practice, however, we are generally not able to construct such an eigenstate easily, so

it is a good idea to have a look at what happens when we apply the phase estimation to a superposition of eigenstates of U. This is done in the following theorem.

Theorem 5.6: Applying the phase estimation algorithm to a superposition of eigenstates
Let $m, n \in \mathbb{N}$ and $0 < \varepsilon < 1$. Suppose one has an n-qubit operator U, with at least k eigenstates $|u_1\rangle$ through $|u_k\rangle$ and corresponding eigenvalues $e^{2\pi i\phi_1}$ through $e^{2\pi i\phi_k}$ with $\phi_1, \ldots, \phi_k \in [0,1)$. Define p as the smallest distance between ϕ_1 and ϕ_j when calculating modulo 1, so:

$$p = \min_{2 \leq j \leq k} ||\phi_1 - \phi_j||_1$$

Also, suppose $p \geq 2^{-m+1}$. Furthermore, suppose that the phase estimation algorithm is applied with the number of qubits in the first register defined as follows:

$$t = m + \left\lceil \log_2 \left(2 + \frac{1}{2\varepsilon} \right) \right\rceil$$

Suppose also that the system starts out in the following state:

$$|0\rangle \otimes (c_1|u_1\rangle + \cdots + c_k|u_k\rangle)$$

Then the probability of finding an approximation of ϕ_1, denoted by $\overline{\phi_1}$ within a proximity of 2^{-m} modulo 1 is lower bounded by the following formula:

$$P(||\phi_1 - \overline{\phi_1}||_1 \leq 2^{-m}) \geq |c_1|^2(1-\varepsilon) - \frac{(4 + \ln(2^{t-m} - 2))(|c_2| + \cdots + |c_k|) \cdot |c_1|}{2^t(p - 2^{-m}) - 1}$$

Proof: For every integer j between 1 and n, we define b_j as the largest integer below $2^t\phi_j$, as before. Furthermore, for all integers j and l between 1 and n and 0 and $2^t - 1$, respectively, we define $\alpha_l^{(j)}$ as the coefficient of the basis state $|b_j + l \mod 2^t\rangle$, had $|u_j\rangle$ been the original state of the system. Hence, we find that the final state of the first register before measurement is given by:

$$c_1 \left(\alpha_{-b_1}^{(1)}|0\rangle + \cdots + \alpha_{2^t-1-b_1}^{(1)}|2^t - 1\rangle \right) + \cdots + c_k \left(\alpha_{-b_k}^{(k)}|0\rangle + \cdots + \alpha_{2^t-1-b_k}^{(k)}|2^t - 1\rangle \right)$$

Recall that for any $-2^{t-1} + 1 \leq l \leq 2^{t-1}$ and j, $\alpha_l^{(j)}$ can be bounded in the following way, as long as we require $|l| > 1$:

$$\left|\alpha_l^{(j)}\right| \leq \frac{1}{2|2^t\phi - b_j - l|} \leq \frac{1}{2(|l| - |2^t\phi - b_j|)} \leq \frac{1}{2(|l| - 1)}$$

We also borrow the choice of e from the previous theorem: $e = 2^{t-m} - 1$. Now, define the following constant:

$$\mu = \min_{2 \leq j \leq k} ||b_1 - b_j||_{2^t} - e$$

Do note that the value of b_i is related to the one of ϕ_i by $0 \leq 2^t\phi_i - b_i < 1$. A similar relation for b_j and ϕ_j holds. So, we find:

$$\begin{aligned}
2^t||\phi_i - \phi_j||_1 &= ||2^t\phi_i - 2^t\phi_j||_{2^t} \\
&\leq \left|\left|2^t\phi_i - \left(b_i + \frac{1}{2}\right)\right|\right|_{2^t} + \left|\left|\left(b_i + \frac{1}{2}\right) - \left(b_j + \frac{1}{2}\right)\right|\right|_{2^t} + \left|\left|2^t\phi_j - \left(b_j + \frac{1}{2}\right)\right|\right|_{2^t} \\
&\leq \left|2^t\phi_i - \left(b_i + \frac{1}{2}\right)\right| + ||b_i - b_j||_{2^t} + \left|2^t\phi_j - \left(b_j + \frac{1}{2}\right)\right| \\
&\leq \frac{1}{2} + ||b_i - b_j||_{2^t} + \frac{1}{2} = ||b_i - b_j||_{2^t} + 1
\end{aligned}$$

Rewriting this, we obtain $||b_i - b_j||_{2^t} \geq 2^t ||\phi_i - \phi_j||_1 - 1$. This yields:

$$\mu \geq 2^t \min_{2 \leq j \leq k} ||\phi_1 - \phi_j||_1 - 1 - e = 2^t p - 1 - 2^{t-m} + 1 = 2^t (p - 2^{-m})$$

By now applying $p \geq 2^{-m+1}$, we can make rigorous that $\mu > 1$, which we will need later on. We use that our choice of t yields $t > m$.

$$\mu \geq 2^t (p - 2^{-m}) \geq 2^t (2^{-m+1} - 2^{-m}) = 2^{t-m} > 1$$

We now derive a lower bound for the probability of finding ϕ_1 within a proximity of 2^{-m} modulo 1. Again we define M to be the result of the measurement of the first register. The inequality $(a-b)^2 \geq a^2 - 2ab$ for $a, b > 0$ is used, among some results from the previous theorem.

$$\begin{aligned}
P(||b_1 - M||_{2^t} \leq e) &= \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left| c_1 \alpha^{(1)}_{l-b_1} + \cdots + c_k \alpha^{(k)}_{l-b_k} \right|^2 \\
&\geq \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left(\left| c_1 \alpha^{(1)}_{l-b_1} \right| - \left(\left| c_2 \alpha^{(2)}_{l-b_2} \right| + \cdots + \left| c_k \alpha^{(k)}_{l-b_k} \right| \right) \right)^2 \\
&\geq \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left(\left| c_1 \alpha^{(1)}_{l-b_1} \right|^2 - 2 \left| c_1 \alpha^{(1)}_{l-b_1} \right| \left(\left| c_2 \alpha^{(2)}_{l-b_2} \right| + \cdots + \left| c_k \alpha^{(k)}_{l-b_k} \right| \right) \right) \\
&\geq \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left(\left| c_1 \alpha^{(1)}_{l-b_1} \right|^2 - 2 \left| c_1 \alpha^{(1)}_{l-b_1} \right| \left(\frac{|c_2|}{2(||l - b_2||_{2^t} - 1)} + \cdots + \frac{|c_k|}{2(||l - b_k||_{2^t} - 1)} \right) \right) \\
&\geq \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left(\left| c_1 \alpha^{(1)}_{l-b_1} \right|^2 - 2 \left| c_1 \alpha^{(1)}_{l-b_1} \right| \cdot \frac{|c_2| + \cdots + |c_k|}{2(\mu - 1)} \right) \\
&\geq \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left(|c_1|^2 \left| \alpha^{(1)}_{l-b_1} \right|^2 - \frac{|c_2| + \cdots + |c_k|}{\mu - 1} \cdot |c_1| \cdot \left| \alpha^{(1)}_{l-b_1} \right| \right) \\
&\geq |c_1|^2 (1 - \varepsilon) - \frac{(|c_2| + \cdots + |c_k|) \cdot |c_1|}{\mu - 1} \cdot \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left| \alpha^{(1)}_{l-b_1} \right| \\
&\geq |c_1|^2 (1 - \varepsilon) - \frac{(|c_2| + \cdots + |c_k|) \cdot |c_1|}{2^t (p - 2^{-m}) - 1} \cdot \sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left| \alpha^{(1)}_{l-b_1} \right|
\end{aligned}$$

We can now bound the sum that remains in the following way. We use that $|\alpha_l(1)| \leq 1$, as $\alpha^{(1)}_l$ would be a coefficient of the final state of the system, if $|u_1\rangle$ were the input of the algorithm.

$$\begin{aligned}
\sum_{l \in \{l : ||b_1 - l||_{2^t} \leq e\}} \left| \alpha^{(1)}_{l-b_1} \right| &= \sum_{l=-e}^{e} \left| \alpha^{(1)}_l \right| \leq 3 + \sum_{l=-e}^{-2} \left| \alpha^{(1)}_l \right| + \sum_{l=2}^{e} \left| \alpha^{(1)}_l \right| \\
&\leq 3 + 2 \sum_{l=2}^{e} \frac{1}{2(|l| - 1)} = 3 + \sum_{l=2}^{e} \frac{1}{l - 1} \\
&= 3 + \sum_{l=1}^{e-1} \frac{1}{l} \leq 3 + 1 + \int_1^{e-1} \frac{1}{l} \, \mathrm{d}l = 4 + [\ln(l)]_1^{e-1} \\
&= 4 + \ln(e - 1) = 4 + \ln\left(2^{t-m} - 2\right)
\end{aligned}$$

So, we can now complete our lower bound:

$$P(||b_1 - M||_{2^t} \leq e) \geq |c_1|^2 (1 - \varepsilon) - \frac{(4 + \ln(2^{t-m} - 2))(|c_2| + \cdots + |c_k|) \cdot |c_1|}{2^t (p - 2^{-m}) - 1}$$

Now define our approximation $\overline{\phi_1}$ by M divided by 2^t. If we now find $||b_1 - M||_{2^t} \leq e$, we have:

$$\begin{aligned} ||\overline{\phi_1} - \phi_1||_1 &= 2^{-t}||2^t\overline{\phi} - 2^t\phi||_{2^t} = 2^{-t}||M - 2^t\phi||_{2^t} \leq 2^{-t}(||M - b_1||_{2^t} + ||b_1 - 2^t\phi||_{2^t}) \\ &\leq 2^{-t}(||M - b_1||_{2^t} + |b_1 - 2^t\phi|) < 2^{-t}(e+1) = 2^{-t} \cdot 2^{t-m} = 2^{-m} \end{aligned}$$

So, we find:

$$P(||\overline{\phi_1} - \phi_1||_1 \leq 2^{-m}) \geq |c_1|^2(1-\varepsilon) - \frac{(4 + \ln(2^{t-m} - 2))(|c_2| + \cdots + |c_k|) \cdot |c_1|}{2^t(p - 2^{-m}) - 1}$$

Hence, we have proven the theorem. □

The bound given in the previous theorem need not be very strict. Frequently, one will find that a few qubits less than the proposed amount will still yield a probability of success in the vicinity of $|c_1|^2(1-\varepsilon)$.

5.3 Order-finding algorithm

Suppose that we have two coprime positive integers, $x, N \in \mathbb{N}$, with $x < N$. Then $\gcd(x, N) = 1$. The *order* of x modulo N is defined by the smallest positive integer r such that $x^r \equiv 1 \mod N$. For example, the order of 5 modulo 7 is equal to 6, as $5^6 = 15625 \equiv 1 \mod 7$ and $5^n \not\equiv 1 \mod 7$ for $n = 1, \ldots, 5$.

First of all, we determine that such an r always exists as long as $\gcd(x, N) = 1$. Therefore, we consider the group of numbers coprime to N.

Theorem 5.7: Group structure of coprime numbers
Take $N \in \mathbb{N}$, and define the set C_N as the set of numbers strictly smaller than N that are coprime to N:

$$C_N = \{n \in \mathbb{N} : n < N \wedge \gcd(n, N) = 1\}$$

Then C_N is a group under multiplication modulo N.

Proof: The identity element of C_N is 1, and modular multiplication is also trivially associative. Suppose that $a, b \in C_N$. Then $\gcd(a, N) = \gcd(b, N) = 1$, hence $\gcd(ab, N) = 1$. So, by Bézout's identity, we find that there exist $u, v \in \mathbb{Z}$, such that $abu + vN = 1$. Hence, $abu \equiv 1 \mod N$. Suppose now $ab \notin C_N$. Then ab and N have a common divisor, hence abu also has a common divisor with N. But then $abu \not\equiv 1 \mod N$, hence we find a contradiction. So $ab \in C_N$. Thus, C_N is closed under multiplication modulo N.

Now, it suffices to show that every element $n \in C_N$ has a multiplicative inverse. By Bézout's identity, we know that $\gcd(n, N) = 1$ implies that $\exists u, v \in \mathbb{Z}$ such that $un + vN = 1$. Hence, $un \equiv 1 \mod N$. But then $u \mod N$ is a multiplicative inverse of n modulo N. If u were not coprime to N, then we would have $\gcd(u, N) = p > 1$, hence $\gcd(un, N) \geq p$. But then $un \not\equiv 1 \mod N$, hence u must be coprime to N. So $u \in C_N$. This completes the proof. □

It now follows directly that as C_N is finite, its subgroup generated by x is also finite. Hence, the order of this subgroup, denoted by r, is finite, yielding $x^r \equiv 1 \mod N$. Do also note that there exists a multiplicative inverse of x modulo N, conveniently denoted by $x^{r-1} \mod N$.

For future notational purposes, the number of bits needed to represent N is defined by L: $L = \lfloor \log_2(N) \rfloor + 1$.

Finding the order of a number is not a very straightforward task. Rather, it is believed to be a hard problem on classical computers, as an algorithm polynomial in L has not been discovered. However, a quantum circuit performing this task in polynomial time has been found, which seems to indicate that quantum computers are conceptually more powerful than classical computers. The order-finding algorithm for quantum computers is the subject of this section.

First of all, we will give an overview of how the phase estimation algorithm, covered in the previous section, can be used to find orders and we will also consider the probability that the algorithm succeeds. Next, we

will have a closer look at the precise implementation of the algorithm as a quantum circuit. Then, we will elaborate on the mathematical process of finalizing the result.

5.3.1 Principle behind the quantum circuit

To find the order of x modulo N, we define the operator U that acts on L qubits. For any state $|j\rangle$ with $0 \leq j \leq N-1$, we want the operation to have the following effect:

$$U|j\rangle = |xj \mod N\rangle \qquad (0 \leq j \leq N-1)$$

Note that the effect of U on the basis states $|j\rangle$ where $N \leq j \leq 2^L - 1$ is not specified. Hence it does not matter where these states are being mapped to by U, as long as U is a unitary operation. One could choose to map these states to themselves, so $U|j\rangle = |j\rangle$ for $N \leq j \leq 2^L - 1$, but for our purposes, it suffices to assume that the mapping is unitary.

The operation U is invertible, as we have seen in theorem 7 that x has a multiplicative inverse modulo N, namely $x^{r-1} \mod N$. Hence, applying the operation $|j\rangle \mapsto |x^{r-1}j \mod N\rangle$ for any $0 \leq j \leq N-1$ would lead to an inverse operation of U. As U only cycles computational basis states, the length of any state vector remains unaltered, hence the operation is indeed unitary, so it can be implemented in a quantum circuit.

Even though we are trying to find the order r of x modulo N, we know that it exists and that its value is smaller than or equal to N. Now we define the states $|u_0\rangle, \ldots, |u_{r-1}\rangle$:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k s}{r}} |x^k \mod N\rangle \qquad (s = 0, \ldots, r-1)$$

These r states are all eigenstates of U, because for any integer s that obeys $0 \leq s \leq r-1$, we find:

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k s}{r}} |x^{k+1} \mod N\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i (k-1) s}{r}} |x^k \mod N\rangle \\ &= e^{\frac{2\pi i s}{r}} \cdot \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k s}{r}} |x^k \mod N\rangle = e^{\frac{2\pi i s}{r}} |u_s\rangle \end{aligned}$$

The real use of these states follows from the following relation. In its derivation, lemma 5.1 is used.

$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle &= \frac{1}{r} \sum_{s=0}^{r-1} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k s}{r}} |x^k \mod N\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} \left(\sum_{s=0}^{r-1} e^{-\frac{2\pi i k s}{r}} \right) |x^k \mod N\rangle \\ &= \frac{1}{r} \cdot r |x^0 \mod N\rangle = |1\rangle \end{aligned}$$

So, the state $|1\rangle$ is a superposition of r eigenstates of U:

$$|1\rangle = \frac{1}{\sqrt{r}} (|u_0\rangle + \cdots + |u_{r-1}\rangle)$$

Hence, we can now apply the phase estimation algorithm on $|1\rangle$. We expect to obtain a value close to $\frac{s}{r}$ for some integer s between 0 and $r-1$. Using the results from the previous section, we can find a lower bound on the probability that we find such a value within a given proximity. Suppose we use t qubits in the first

register, where t is given by the following formula and $0 < \varepsilon < 1$ can be chosen in order to obtain the desired accuracy.

$$t = 2L + 1 + \left\lceil \log_2 \left(2 + \frac{1}{2\varepsilon} \right) \right\rceil$$

We know that all the values of ϕ are evenly spaced, so when we apply theorem 6, we find $p = \frac{1}{r}$. This yields an approximation $\overline{\phi}$ close to $\frac{0}{r}$ with at least the following probability. Here, the inequality $r \leq N \leq 2^L$ is used.

$$\begin{aligned}
P\left(||\overline{\phi} - 0||_1 \leq 2^{-2L-1}\right) &\geq \frac{1}{r}(1-\varepsilon) - \frac{(4 + \ln(2^{t-2L-1} - 2)) \cdot \frac{r-1}{\sqrt{r}} \cdot \frac{1}{\sqrt{r}}}{2^t(\frac{1}{r} - 2^{-2L-1}) - 1} \\
&\geq \frac{1}{r}(1-\varepsilon) - \frac{(4 + \ln(2^{t-2L-1})) \cdot \frac{r-1}{r}}{2^t(2^{-L} - 2^{-2L-1}) - 1} \\
&\geq \frac{1}{r}(1-\varepsilon) - \frac{4 + (t - 2L - 1)\ln(2)}{2^{t-L} - 2^{t-2L-1} - 1} \\
&\geq \frac{1}{r}(1-\varepsilon) - \frac{4 + (t - 2L - 1)\ln(2)}{2^{t-L-1}} \\
&\geq \frac{1}{r}(1-\varepsilon) - \frac{4 + \left\lceil \log_2\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil \ln(2)}{2^L \cdot 2^{\left\lceil \log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}} \\
&\geq \frac{1}{r}\left(1 - \varepsilon - \frac{4 + \left\lceil \log_2\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil \ln(2)}{2^{\left\lceil \log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}}\right)
\end{aligned}$$

The probability that the approximation is equally close to one of the other values $\frac{s}{r}$, with s integer, is identical. So, we find that the probability that the phase estimation algorithm successfully returns an approximation of $\frac{s}{r}$ within a range of 2^{-2L-1} is lower bounded by:

$$P(\text{success}) \geq 1 - \varepsilon - \frac{4 + \left\lceil \log_2\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil \ln(2)}{2^{\left\lceil \log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}}$$

Numerical analysis of this bound yields that it is positive for $\varepsilon < 0.24$. For values of ε larger than this, the probability of success is probably still quite large, but the bounds applied above are not strict enough for us to be able to say anything about this probability.

Notice in addition that this bound can be chosen arbitrary close to 1, for ε sufficiently small. Hence, the more qubits are used in the first register, the greater the probability of successfully finding an approximation $\overline{\phi}$ that approximates a value of $\frac{s}{r}$ for some integer s between 0 and $r - 1$, within the vicinity of 2^{-2L-1}.

From the approximation of $\frac{s}{r}$ with $0 \leq s \leq r - 1$, we can try to determine the value of r. This can be accomplished using the continued fraction expansion, and will be covered in section 5.3.3.

5.3.2 Implementation of the quantum circuit

In the previous subsection, we used the phase estimation algorithm to find approximations of the eigenvalues of the operator U. To implement this phase estimation algorithm, though, recall from figure 5.3 that we need to implement controlled U^{2^j} quantum gates, where $0 \leq j \leq t - 1$. The efficiency of the order finding algorithm will eventually hinge partially on an efficient implementation of these quantum gates, which we set out to find in this section. The approach is largely based on [5].

We will first consider addition with quantum gates. Next, we will investigate modulo addition and multiplication. Finally, we will derive an implementation for the exponentiation that is needed.

First of all, let's have a look at what happens when we add two binary numbers. Suppose we have two n-bit integers a and b, with binary representation $(a_{n-1} \dots a_0)_2$ and $(b_{n-1} \dots b_0)_2$. Suppose their sum is the integer d. Then d is at most $n+1$ bits in length, so it has binary representation $(d_n \dots d_0)_2$. In calculating d, we typically start with its last digit. It is given by: $d_0 = a_0 + b_0 \mod 2$. Next, we determine whether an overflow occurred, hence we define a carry bit, c_1, as follows:

$$c_1 = \begin{cases} 0, & a_0 + b_0 < 2 \\ 1, & a_0 + b_0 \geq 2 \end{cases}$$

Then, we calculate the following digit: $d_1 = a_1 + b_1 + c_1 \mod 2$. Next, we calculate the following carry bit, c_2, by checking if $a_1 + b_1 + c_1$ equaled or exceeded 2. Continuing in this manner, we calculate all results, until we find that $d_{n+1} = c_{n+1}$. An example of this algorithm is shown in figure 5.5.

Bit	5	4	3	2	1	0	
c	*1*	*1*	*1*	*0*	*0*	*0*	
a		1	0	1	1	0	
b		0	1	1	0	1	+
d	1	0	0	0	1	1	

Figure 5.5: Example of the addition algorithm. The 5-bit numbers a and b are added and their result is denoted by the 6-bit number d. In the row denoted by c, the carry bits are shown.

Implementing a quantum circuit to perform the addition of two n-bit numbers a and b is done in a similar way. We first introduce the carry and sum gates depicted in figure 5.6. Note that the matrix representations of these gates are Hermitian, hence the inverse equals the operation itself.

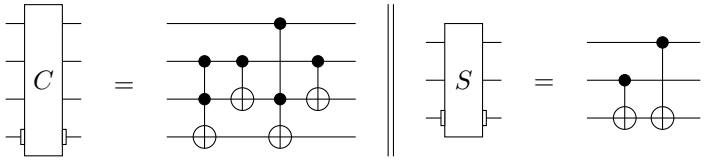


Figure 5.6: This figure shows the implementation of the carry and sum gates. The carry gate takes the top three qubits and checks if two or more are 1. If so, then the bottom one is flipped. The sum gate sums the top two qubits, and checks if the result is 1 $\mod 2$. If so, then the bottom qubit is flipped.

The addition algorithm is now implemented using these gates, as shown in figure 5.7. Three registers are used, with n, $n+1$ and n qubits respectively. The first and second registers contain the values of a and b, respectively. The third register is set to all zeros. Then, first of all, all carry bits are calculated and stored in the third register. Then, the actual summation is performed, and simultaneously, all carry bits are reset to 0. In summary, the addition algorithm maps the state $|a\rangle \otimes |b\rangle \otimes |0\rangle$ to the state $|a\rangle \otimes |a+b\rangle \otimes |0\rangle$.

The addition algorithm takes $2n-1$ C-gates, and n S-gates. As the carry and sum gates only take a finite number of gates to implement, we find that the total time complexity of the addition algorithm is $\mathcal{O}(n)$.

Finally, note that the reverse of this algorithm provides an implementation for substraction, so rather than $+^*$, we will denote the inverse of this sequence of quantum gates by $-$. Hence, the substraction operation will map the state $|a\rangle \otimes |b\rangle \otimes |0\rangle$ to $|a\rangle \otimes |a-b \mod 2^{n+1}\rangle \otimes |0\rangle$, as long as a and b are n-bit numbers. If $b > a$, then we find $-2^n < a - b < 0$, so the most significant qubit of the second register will be set upon completion of the algorithm. If we now use this qubit as a control qubit for other operations, we can implement operations conditional on $b > a$, which will prove to be useful in the algorithms to come.

Now suppose we want to perform addition modulo N of two numbers a and b, which are both smaller than N. Recall that N has L bits, so a and b are also two L-bit numbers. If N is a power of 2, then we can simply discard the most significant bit. If not, then we will need a more ingenious solution, which is presented in

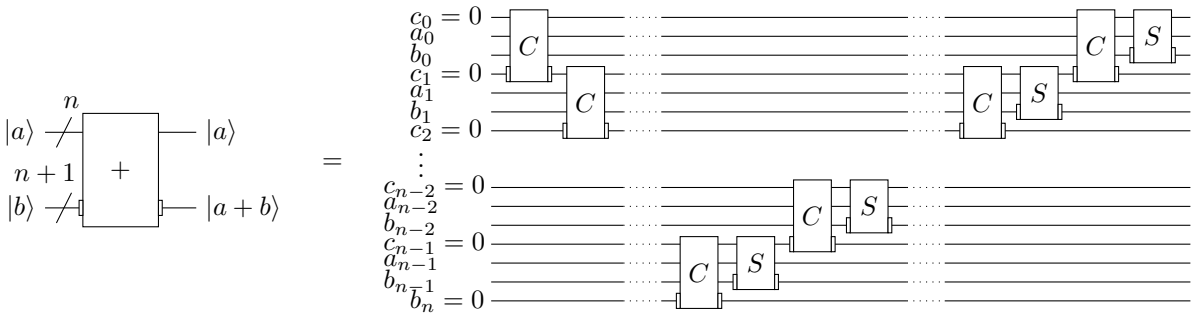


Figure 5.7: Implementation of the quantum circuit that performs addition. The qubits that comprise the three registers are shown in alternating order in the right figure. The circuit takes $|a\rangle \otimes |b\rangle \otimes |0\rangle$ to $|a\rangle \otimes |a+b\rangle \otimes |0\rangle$, where a and b are two n-bit numbers.

figure 5.8. We use a gate N, which maps the state $|0\rangle$ to $|N\rangle$ and vice versa. It can be constructed using CNOT gates only. In this way, we see that N is hardcoded in our quantum algorithm.

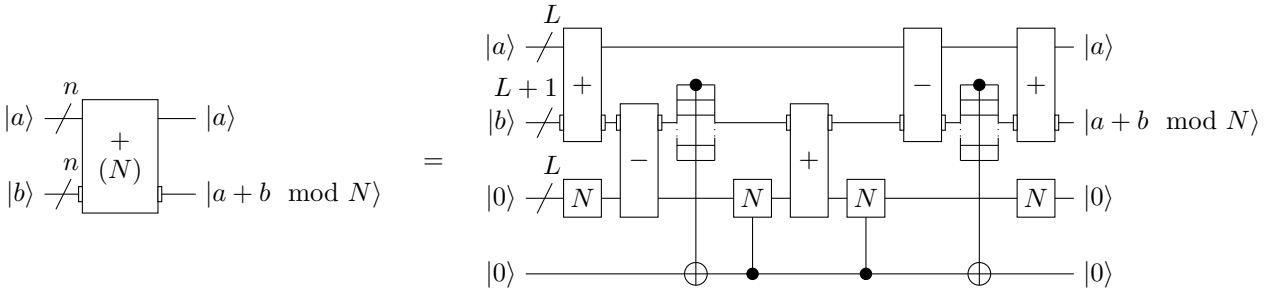


Figure 5.8: The implementation of modular addition. The n-bit integers a and b are added modulo N, and the result is stored in the second register. Note that the actual algorithm appends a qubit to the second register, but as both the input and the output can be represented by n bits, it is not shown in the abbreviated form. The N gate maps the state $|0\rangle$ to $|N\rangle$ and back. The CNOT gates are controlled using the most significant qubit of the second register.

The basic idea of the algorithm is the following. Suppose we have two n-bit numbers, a and b. We first add both numbers using the quantum addition circuit. Then we substract N, and check if the result $a + b - N$ is greater than or equal to 0. If this is not the case, we set an external qubit indicating that an overflow occurred. Only if this external qubit is set, we add N back again. Finally, we reset the external qubit so that we can use it again in future steps. In summary, the algorithm maps the state $|a\rangle \otimes |b\rangle$ to the state $|a\rangle \otimes |a + b \mod N\rangle$, as long as a and b are smaller than N themselves.

The algorithm uses 6 additions, four N-operations, and a constant amount of other operations. Like the addition algorithm, the N-gate also has time complexity $\mathcal{O}(L)$, so we find that the total time complexity of the modulo addition algorithm is $\mathcal{O}(L)$.

This quantum circuit obviously also has a reverse implementation, which maps the state $|a\rangle \otimes |b\rangle$ to the state $|a\rangle \otimes |a - b \mod N\rangle$. This algorithm will conveniently be referred to as substraction modulo N, and the $+$ sign will be changed to $-$ accordingly.

We now turn our attention to controlled multiplying modulo N by a fixed constant, a. More precisely, we are looking for a function that, if the control qubit is set, takes the state $|k\rangle \otimes |0\rangle$ to the state $|k\rangle \otimes |ak \mod N\rangle$, where $0 \leq k \leq N-1$. If, on the other hand, the control qubit is cleared, then the state $|k\rangle \otimes |0\rangle$ should be mapped to $|k\rangle \otimes |k\rangle$ for $0 \leq k \leq N-1$.

We first define the copy operator, as shown in figure 5.9, as we will need it later on in the algorithm.

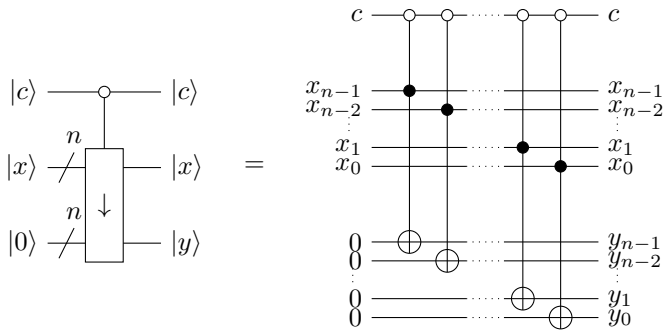


Figure 5.9: Implementation of the copy operation. If the control bit is cleared, then the contents of the second register are copied to the third register. Otherwise, the third register is left at $|0\rangle$. So, if c is set, $y = 0$, and if c is cleared, $y = x$.

Next, all values $2^j a \mod N$ are calculated for $0 \leq j \leq L-1$. This can be done efficiently on a classical computer, using the method of modular multiplication.

Now, we can give an implementation of the controlled multiplication modulo N circuit, as shown in figure 5.10. The idea is to calculate ak using repetitive addition, as in this formula:

$$ak \mod N = (k_{L-1} \dots k_0)_2 a \mod N = k_0 a + k_1 \cdot 2a + \cdots + k_{L-1} 2^{L-1} a \mod N$$

Note that the values of $2^j a \mod N$ we just calculated, are hardcoded into the program. The algorithm performs L modular additions, and $2L$ hardcoded gates, so the time complexity is given by $\mathcal{O}(L^2)$.

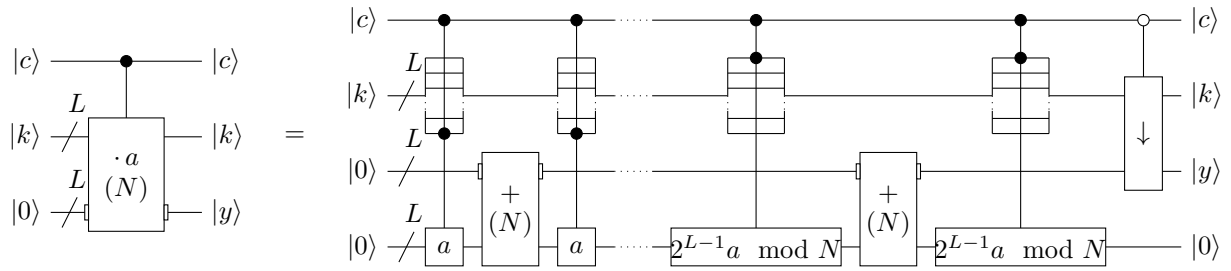


Figure 5.10: Implementation of modular multiplication by a fixed constant a modulo N. The quantum circuit takes the state $|1\rangle \otimes |k\rangle \otimes |0\rangle$ to the state $|1\rangle \otimes |k\rangle \otimes |ak \mod N\rangle$, and the state $|0\rangle \otimes |k\rangle \otimes |0\rangle$ is mapped to $|0\rangle \otimes |k\rangle \otimes |k\rangle$. Hence, y is equal to $ak \mod N$ if c is set, and equal to k if c is cleared.

This operation also has an inverse, which will be denoted by division modulo N. All multiplication symbols, $\cdot$, will be replaced by division symbols, $/$, if the reversed implementation is referred to. Do note, though, that the operation itself does not exactly resemble division of the numerical values of the registers. Rather, if a state $|k\rangle \otimes |ak \mod N\rangle$ is divided by a modulo N, then we obtain $|k\rangle \otimes |0\rangle$.

We now arrive at the point that we can give an implementation of the controlled U^{2^j} operation. First of all, we determine the value of $x^{2^j} \mod N$. This can again be done in an efficient way using a classical computer, with the method of modular exponentiation. Moreover, we are going to need the inverse, denoted by $x^{-2^j} \mod N$. This be efficiently found using the extended Euclid's method.

An implementation of the controlled U^{2^j} gate is given in figure 5.11. The idea is that first the state $|1\rangle \otimes |k\rangle \otimes |0\rangle$ is mapped to $|1\rangle \otimes |k\rangle \otimes |x^{2^j} k \mod N\rangle$. Then, the second and third registers are swapped, yielding

the state $|1\rangle \otimes |x^{2^j} k \mod N\rangle \otimes |k\rangle$. Note that this can also be written as $|1\rangle \otimes |x^{2^j} k \mod N\rangle \otimes |x^{-2^j} x^{2^j} k \mod N\rangle$. Applying division by $x^{-2^j} \mod N$ now yields the state $|1\rangle \otimes |x^{2^j} k \mod N\rangle \otimes |0\rangle$.

On the other hand, if the control qubit is cleared, then the state $|0\rangle \otimes |k\rangle \otimes |0\rangle$ is mapped to $|0\rangle \otimes |k\rangle \otimes |k\rangle$. The swap operation then has no effect, whereupon the division by $x^{-2^j} \mod N$ maps the state back to $|0\rangle \otimes |k\rangle \otimes |0\rangle$.

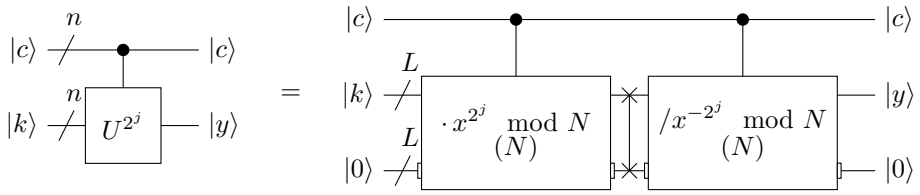


Figure 5.11: Implementation of the controlled U^{2^j} gate, with $0 \leq j \leq L-1$. The state $|0\rangle \otimes |k\rangle$ is mapped to $|0\rangle \otimes |k\rangle$, whereas the state $|1\rangle \otimes |k\rangle$ is mapped to the state $|1\rangle \otimes |x^{2^j} k\rangle$. Hence, if c is set, then y equals $x^{2^j} k$. Otherwise, y equals k.

Note that the time complexity of the controlled U^{2^j} gate is equal to the complexity of the controlled modular multiplication gate. Hence, the complexity is equal to $\mathcal{O}(L^2)$. Note that t of these gates are needed in figure 5.3. Also, $t \equiv \mathcal{O}(L)$ according to our choice of the number of qubits in the first register. Furthermore, the quantum Fourier transform has time complexity $\mathcal{O}(L^2)$ as well, so we find that the phase estimation algorithm has a time complexity of $\mathcal{O}(L^3)$. Hence, this is polynomial in the number of bits needed to represent N, so this algorithm is efficient.

5.3.3 Continued fraction expansion

We have now found an efficient way to approximate a value of $\frac{s}{r}$, where s is random between 0 and $r-1$, within a proximity of 2^{-2L-1}. The question, now, is how to determine r from this approximation. This is done using the continued fraction expansion, which will be the subject of this subsection.

First of all, we define the notion of a continued fraction.

Definition 5.8: Continued fraction
Let $N \in \mathbb{N}$ and $\{a_1, \ldots, a_N\} \subseteq \mathbb{R}_{>0}$ a set of positive real numbers. Then we define the following shorthand notation for what is referred to as a continued fraction:

$$[a_1, a_2, \ldots, a_N] = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_N}}}}$$

So, for example, the number $[1,2,2,3]$ can be calculated easily:

$$\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{2+\frac{1}{3}}}} = \cfrac{1}{1+\cfrac{1}{2+\frac{3}{7}}} = \cfrac{1}{1+\frac{7}{17}} = \frac{17}{24}$$

We would like to be able to invert the process displayed above. That is, we would like to have an algorithm that enables us to construct a continued fraction composed of natural numbers, such that it equals an arbitrarily chosen number. Such a constructed continued fraction of a number q is referred to as the continued

fraction expansion of q. To develop a method to construct these continued fraction expansions, we prove a lemma and theorem below.

Lemma 5.9: Bound on the continued fraction of natural numbers
Suppose that $N \in \mathbb{N}$ and $\{a_1, \ldots, a_N\} \in \mathbb{N}$. Then $0 < [a_1, a_2, \ldots, a_N] \leq 1$.

Proof: We prove this with induction to the length of the continued fraction. The case $N = 1$ is easily checked. In that case, we have $a_1 \geq 1$, so $0 < \frac{1}{a_1} \leq 1$. Hence $0 < [a_1] \leq 1$.

Now suppose that for some $N \in \mathbb{N}$, we have that for all sets $\{a_1, \ldots, a_N\} \subseteq \mathbb{N}$, $0 < [a_1, \ldots, a_N] \leq 1$. This is our induction hypothesis. Now for any set $\{a_1, \ldots, a_{N+1}\} \subseteq \mathbb{N}$, we find:

$$[a_1, \ldots, a_{N+1}] = \frac{1}{a_1 + [a_2, \ldots, a_{N+1}]}$$

This last expression can now be bounded easily using the induction hypothesis:

$$0 < \frac{1}{a_1 + [a_2, \ldots, a_{N+1}]} < \frac{1}{a_1} \leq 1$$

Hence, by substituting back, we find that the induction hypothesis also holds for $N + 1$:

$$0 < [a_1, \ldots, a_{N+1}] \leq 1$$

Hence, by the principle of mathematical induction, we have proven what we set out to prove. □

Theorem 5.10: Continued fraction expansion
Suppose $q \in \mathbb{Q}$ and $0 < q < 1$. Then there exists exactly 1 finite continued fraction with natural numbers that equals q and does not have a 1 as final number.

Proof: As q is a positive rational number, we can write it as a fraction of two natural numbers: $q = \frac{p_0}{q_0}$, where p_0 and q_0 are coprime. Moreover, as $q < 1$, we have $p_0 < q_0$. We set out to find some $N \in \mathbb{N}$ and $\{a_1, \ldots, a_N\} \subseteq \mathbb{N}$ such that $[a_1, \ldots, a_N] = q$.

As a first step, we can rewrite q into the following form:

$$q = \frac{1}{\frac{q_0}{p_0}}$$

We know that $\frac{q_0}{p_0} > 1$. There are now two cases that we need to consider.

First of all, suppose $p_0 = 1$. Then $\frac{q_0}{p_0} = q_0 \in \mathbb{N}$. Now suppose that we choose $N \neq 1$. Then our choice of N and $\{a_1, \ldots, a_N\}$ has to obey:

$$\frac{q_0}{p_0} = a_1 + [a_2, \ldots, a_N]$$

But from the preceding lemma, we found that $0 < [a_2, \ldots, a_N] \leq 1$. As $\frac{q_0}{p_0}$ and a_1 are both natural numbers, so must be $[a_2, \ldots, a_N]$, hence we must have $[a_2, \ldots, a_N] = 1$. But then if $N > 2$, we have $a_2 + [a_3, \ldots, a_N] = 1$, which yields $[a_3, \ldots, a_N] \leq 0$, which is a contradiction with the previous lemma. Choosing $N = 2$, on the other hand, yields $a_2 = 1$, from which we see that the last number of the continued fraction expansion is 1. This was not allowed according to the statement of the theorem. So, we must choose $N = 1$, which implies that we must choose a_1 such that $[a_1] = \frac{1}{a_1} = q = \frac{p_0}{q_0}$. As $p_0 = 1$, we must choose $a_1 = q_0$. As $q_0 > p_0 = 1$, we have $a_1 > 1$, so a_1 is a valid final number in the continued fraction.

On the other hand, suppose $p_0 > 1$. Then as q_0 and p_0 are coprime, we find that $\frac{q_0}{p_0} \notin \mathbb{N}$. So, we cannot write the reciprocal of q as a natural number, so N must be chosen greater than 1. Hence, our choice of N and $\{a_1, \ldots, a_N\}$ must satisfy the following relation.

$$\frac{q_0}{p_0} = a_1 + [a_2, \ldots, a_N]$$

From the previous lemma, we know that $0 < [a_2, \ldots, a_N] \leq 1$, regardless of our choice of N and $\{a_2, \ldots, a_N\}$. This yields that we must choose a_1 in the following interval:

$$\frac{q_0}{p_0} - 1 \leq a_1 < \frac{q_0}{p_0}$$

As a_1 has to be a natural number, there is exactly one choice for a_1 satisfying these criteria:

$$a_1 = \left\lceil \frac{q_0}{p_0} - 1 \right\rceil$$

We now define $p_1 = q_0 - a_1 p_0$ and $q_1 = p_0$. Then we find:

$$[a_2, \ldots, a_N] = \frac{q_0}{p_0} - a_1 = \frac{q_0 - a_1 p_0}{p_0} = \frac{p_1}{q_1}$$

Also, we have that $\gcd(p_0, q_0) = 1$, so by Bézout's identity, we find that there exist $u, v \in \mathbb{Z}$ such that $up_0 + vq_0 = 1$. Rewriting this using the way we defined p_1 and q_1 yields:

$$1 = up_0 + vq_0 = up_0 + v(p_1 + a_1 p_0) = uq_1 + vp_1 + va_1 q_1 = (u + va_1)q_1 + vp_1$$

Hence, also $\gcd(p_1, q_1) = 1$. Furthermore, we have $a_1 < \frac{q_0}{p_0}$, so $p_0 a_1 < q_0$ and hence $p_1 = q_0 - a_1 p_0 > 0$. Finally, we also have $a_1 \geq \frac{q_0}{p_0} - 1$. Moreover, $a_1 \in \mathbb{N}$ and $\frac{q_0}{p_0} \notin \mathbb{N}$, hence $a_1 > \frac{q_0}{p_0} - 1$. This leads to:

$$p_1 = q_0 - a_1 p_0 < q_0 - \left(\frac{q_0}{p_0} - 1 \right) p_0 = q_0 - q_0 + p_0 = p_0$$

So, we have now transformed the problem to finding a continued fraction $[a_2, \ldots, a_N]$ which equals $\frac{p_1}{q_1}$, where $\gcd(p_1, q_1) = 1$. Moreover, we find that $p_1 < p_0$. We can now apply the same procedure as before again, by taking $N = 2$ if $p_1 = 1$, and defining p_2 and q_2 otherwise. We again note that $p_2 < p_1$. In this way, we obtain a sequence p_k that is strictly decreasing, hence there exists an $n \in \mathbb{N}$ such that $p_n = 1$. Hence, we choose $N = n + 1$ and find that $[a_1, \ldots, a_N]$, chosen via the process outlined above, equals q. Moreover, all choices have above been shown to be the only possibilities, so we can conclude that there exists exactly one continued fraction that satisfies the criteria posed in the statement of the theorem. Hence, we have proven the theorem. $\square$

We can now apply the calculation given before in reversed order to find the continued fraction expansion of $\frac{17}{24}$:

$$\frac{17}{24} = \frac{1}{\frac{24}{17}} = \frac{1}{1 + \frac{7}{17}} = \frac{1}{1 + \frac{1}{\frac{17}{7}}} = \frac{1}{1 + \frac{1}{2 + \frac{3}{7}}} = \frac{1}{1 + \frac{1}{2 + \frac{1}{\frac{7}{3}}}} = \frac{1}{1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{3}}}} = [1, 2, 2, 3]$$

The theorem above now tells us that this is the only finite continued fraction that equals $\frac{17}{24}$ and does not have a final number of 1.

At this point, the idea behind the requirement that the last number of the continued fraction may not equal 1, is not at all clear. As a first result concerning this requirement, we note that every rational number can also be written as a continued fraction that does end in a 1.

Theorem 5.11: Continued fraction expansion ending in a 1
Suppose that $q \in \mathbb{Q}$ is a rational number within the range $0 < q < 1$. Let $N \in \mathbb{N}$ and $\{a_1, \ldots, a_N\} \subseteq N$ such that $[a_1, \ldots, a_N] = q$ and $a_N \neq 1$. Then there exists another continued fraction that equals q, ending in a 1 and given by:

$$[a_1, \ldots, a_N - 1, 1] = q$$

Proof: We prove that $[a_1, \ldots, a_N] = [a_1, \ldots, a_N - 1, 1]$. It suffices to show that $[a_N] = [a_N - 1, 1]$, as the rest of the fractional expressions are identical. Indeed, we find:

$$[a_N - 1, 1] = \frac{1}{a_N - 1 + \frac{1}{1}} = \frac{1}{a_N} = [a_N]$$

Hence, we have proven the theorem. □

We also introduce the definition of convergents, which we will need shortly.

Definition 5.12: Convergents
Let $N \in \mathbb{N}$ and $[a_1, \ldots, a_N]$ be a continued fraction with $a_1, \ldots, a_N \in \mathbb{N}$. Then for all $n \in \mathbb{N}$ such that $n \leq N$, we define the nth convergent of this continued fraction by $[a_1, \ldots, a_n]$.

Intuitively, the nth convergent of the continued fraction expansion of a number q provides an estimate of q, the sequence of which converges to q when n grows larger. For example, the convergents of $\frac{17}{24}$ are given in table 5.1, and plotted in figure 5.12. Particularly, notice the oscillatory behavior of the convergents.

n	nth convergent	Approximate value
1	1	1
2	$\frac{2}{3}$	0.6667
3	$\frac{5}{7}$	0.7143
4	$\frac{17}{24}$	0.7083

Table 5.1: A list of the convergents of $\frac{17}{24}$. Even in this small example, one can see that the nth convergent provides a better approximation when n grows larger.

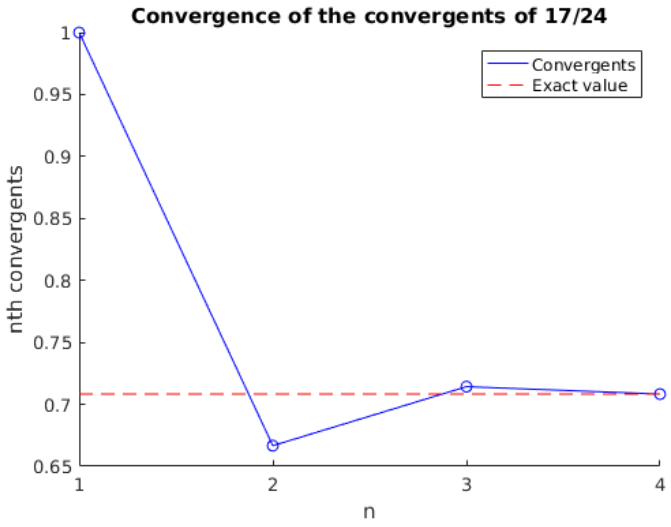


Figure 5.12: A plot of the convergents of $\frac{17}{24}$. Note the oscillatory behavior of the approximations to the exact value.

We will now develop a bound on the speed of the convergence, using the following theorems.

Theorem 5.13: Recursive formula for convergents
Let $N \in \mathbb{N}$ and $[a_1, \ldots, a_N]$ be a continued fraction. Define $b_1 = 1$, $c_1 = a_1$, $b_2 = a_2$ and $c_2 = 1 + a_1 a_2$. Define the next values of $(b_n)_{n=1}^N$ and $(c_n)_{n=1}^N$ in $\mathbb{N}$ by the following recursion formulas:

$$\begin{aligned} b_n &= a_n b_{n-1} + b_{n-2} \quad (3 \leq n \leq N) \\ c_n &= a_n c_{n-1} + c_{n-2} \quad (3 \leq n \leq N) \end{aligned}$$

Then for all natural n that satisfy $n \leq N$, the nth convergent can be expressed by:

$$[a_1, \ldots, a_n] = \frac{b_n}{c_n}$$

Furthermore, if in addition $\{a_1, \ldots, a_N\} \in \mathbb{N}$, then for all $2 \leq n \leq N$, we have the following relation:

$$c_n b_{n-1} - b_n c_{n-1} = (-1)^n$$

Moreover, for all $1 \leq n \leq N$, b_n and c_n are coprime. Finally, the difference between two successive convergents decreases.

Proof: We prove this theorem by induction to n. The cases for $n = 1$, $n = 2$ and $n = 3$ are easily checked, as is done below:

$$\begin{aligned} [a_1] &= \frac{1}{a_1} = \frac{b_1}{c_1} \\ [a_1, a_2] &= \frac{1}{a_1 + \frac{1}{a_2}} = \frac{a_2}{a_1 a_2 + 1} = \frac{b_2}{c_2} \\ [a_1, a_2, a_3] &= \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3}}} = \frac{1}{a_1 + \frac{a_3}{a_2 a_3 + 1}} = \frac{a_2 a_3 + 1}{a_1 a_2 a_3 + a_1 + a_3} = \frac{a_3 b_2 + b_1}{a_3 c_2 + c_1} = \frac{b_3}{c_3} \end{aligned}$$

From the above relations, it is obvious that the recursion relations successfully provide us with the numerator and denominator of the 3rd convergent.

Now, suppose that up to some natural n that satisfies $3 \leq n \leq N-1$, the recursion relations from the question hold. This we refer to as the induction hypothesis. We introduce the continued fraction $[a_1, \ldots, a_n + \frac{1}{a_{n+1}}]$. Then, from the induction hypothesis, we know that there are rational numbers $\overline{b}_1, \ldots, \overline{b}_n$ and $\overline{c}_1, \ldots, \overline{c}_n$ such that the following relations hold:

$$\begin{aligned} [a_1, \ldots, a_m] &= \tfrac{\overline{b}_m}{\overline{c}_m} \qquad (1 \leq m \leq n-1) \\ [a_1, \ldots, a_n + \tfrac{1}{a_n}] &= \tfrac{\overline{b}_n}{\overline{c}_n} \\ \overline{b}_m &= a_m \overline{b}_{m-1} + \overline{b}_{m-2} \qquad (3 \leq m \leq n-1) \\ \overline{c}_m &= a_m \overline{c}_{m-1} + \overline{c}_{m-2} \qquad (3 \leq m \leq n-1) \\ \overline{b}_n &= \left(a_n + \tfrac{1}{a_{n+1}}\right) \overline{b}_{n-1} + \overline{b}_{n-2} \\ \overline{c}_n &= \left(a_n + \tfrac{1}{a_{n+1}}\right) \overline{c}_{n-1} + \overline{c}_{n-2} \end{aligned}$$

As the sequences $\overline{b}$ and $\overline{c}$ are defined by the exact same recursive formula as the one in the theorem, up to $n-1$, we find that for any $1 \leq m \leq n-1$, $\overline{b}_m = b_m$ and $\overline{c}_m = c_m$. We can now calculate the next convergent,

$[a_1, \ldots, a_{n+1}]$, as follows:

$$\begin{aligned}
[a_1, \ldots, a_{n+1}] &= \left[a_1, \ldots, a_n + \frac{1}{a_{n+1}}\right] = \frac{\overline{b}_n}{\overline{c}_n} = \frac{\left(a_n + \frac{1}{a_{n+1}}\right)\overline{b}_{n-1} + \overline{b}_{n-2}}{\left(a_n + \frac{1}{a_{n+1}}\right)\overline{c}_{n-1} + \overline{c}_{n-2}} \\
&= \frac{\left(a_n + \frac{1}{a_{n+1}}\right)b_{n-1} + b_{n-2}}{\left(a_n + \frac{1}{a_{n+1}}\right)c_{n-1} + c_{n-2}} = \frac{a_n b_{n-1} + b_{n-2} + \frac{b_{n-1}}{a_{n+1}}}{a_n c_{n-1} + c_{n-2} + \frac{c_{n-1}}{a_{n+1}}} \\
&= \frac{b_n + \frac{b_{n-1}}{a_{n+1}}}{c_n + \frac{c_{n-1}}{a_{n+1}}} = \frac{a_{n+1}b_n + b_{n-1}}{a_{n+1}c_n + c_{n-1}} = \frac{b_{n+1}}{c_{n+1}}
\end{aligned}$$

Hence, we find by induction to n that the formulas from the question are recursive formulas with which the numerator and denominator of the nth convergents can be determined.

From this point onwards, we assume that $\{a_1, \ldots, a_N\} \subseteq \mathbb{N}$. Now, for the next claim, we need to prove that for any $2 \leq n \leq N$, the following relation holds:

$$c_n b_{n-1} - b_n c_{n-1} = (-1)^n$$

Again, we use induction to n. The case where $n = 2$ is easily checked:

$$c_2 b_1 - b_2 c_1 = (1 + a_1 a_2) \cdot 1 - a_2 a_1 = 1 = (-1)^2$$

Now suppose that it holds for some natural $n \in \mathbb{N}$ that satisfies $2 \leq n \leq N - 1$. Then:

$$\begin{aligned}
c_{n+1}b_n - b_{n+1}c_n &= (a_{n+1}c_n + c_{n-1})b_n - (a_{n+1}b_n + b_{n-1})c_n \\
&= a_{n+1}c_n b_n + b_n c_{n-1} - a_{n+1}b_n c_n - b_{n-1}c_n \\
&= b_n c_{n-1} - b_{n-1}c_n = -(-1)^n = (-1)^{n+1}
\end{aligned}$$

So, by the principle of mathematical induction, we now find that for any $2 \leq n \leq N$, we have $c_n b_{n-1} - b_n c_{n-1} = (-1)^n$.

As for the next claim, we need to prove that for all $1 \leq n \leq N$, b_n and c_n are coprime. Suppose n is even. Then: $c_n b_{n-1} - b_n c_{n-1} = 1$, so by Bézout's identity, we find that $\gcd(b_n, c_n) = 1$.

Now suppose n is odd and $n \neq 1$. Then we have $-c_n b_{n-1} + b_n c_{n-1} = 1$, which also yields that $\gcd(b_n, c_n) = 1$. In addition, if $n = 1$, we find: $\gcd(b_1, c_1) = \gcd(1, a_1) = 1$.

Finally, we can now prove that the sequence of differences between two successive converges decreases. We rewrite the equation $c_n b_{n-1} - b_n c_{n-1} = (-1)^n$ in the following manner:

$$\frac{b_{n-1}}{c_{n-1}} - \frac{b_n}{c_n} = \frac{(-1)^n}{c_{n-1}c_n}$$

As the sequence (c_n) is strictly increasing, we find that the difference between two successive convergents is decreasing:

$$\left|\frac{b_n}{c_n} - \frac{b_{n-1}}{c_{n-1}}\right| = \frac{1}{c_{n-1}c_n}$$

Hence, this completes the proof. $\square$

The above theorem has a few interesting implications. For example, for even n, we find that $q_n p_{n-1} - p_n q_{n-1} = 1 > 0$, hence:

$$[a_1, \ldots, a_n] = \frac{q_n}{p_n} > \frac{p_{n-1}}{q_{n-1}} = [a_1, \ldots, a_{n-1}]$$

For odd n that are not equal to 1, though, we find: $q_n p_{n-1} - p_n q_{n-1} = -1 < 0$ and so:

$$[a_1, \ldots, a_n] = \frac{q_n}{p_n} < \frac{p_{n-1}}{q_{n-1}} = [a_1, \ldots, a_{n-1}]$$

Hence, the oscillatory behavior is reflected by these results. Moreover, we can now find an intuitive explanation for the apparent ambiguity in the choice of the final number of the continued fraction. Apparently, we can choose between estimating the exact number from above or below in the final step.

Finally, now, we obtain the theorem that the application in the order finding routine hinges on.

Theorem 5.14: Bounds on convergents
Let $q \in \mathbb{Q}$ with $0 < q < 1$. Suppose $b, c \in \mathbb{N}$ such that b and c are coprime, $0 < \frac{b}{c} < 1$ and $\left|\frac{b}{c} - q\right| \leq \frac{1}{2c^2}$. Then $\frac{b}{c}$ is a convergent of q.

Proof: First of all, we note that according to theorem 5.10, $\frac{b}{c}$ is equal to a continued fraction $[a_1, \ldots, a_N]$, where $N \in \mathbb{N}$, $a_1, \ldots, a_N \in \mathbb{N}$ and $a_N \neq 1$. Note also that according to theorem 5.11, if we define $M = N+1$, $\alpha_1 = a_n$ for every $1 \leq n \leq N-1$, $\alpha_{M-1} = a_N - 1$ and $\alpha_M = 1$, we find:

$$\frac{b}{c} = [\alpha_1, \ldots, \alpha_M]$$

Note that we have that either N is odd and M is even, or vice versa. We will use this later on. For the continued fraction $[a_1, \ldots, a_N]$, we define the sequences $(b_n)_{n=1}^N$ and $(c_n)_{n=1}^N$ according to theorem 13. Similarly, we define $(\beta_n)_{n=1}^N$ and $(\gamma_n)_{n=1}^N$ for the continued fraction $[\alpha_1, \ldots, \alpha_M]$. Note that $\frac{b}{c} = \frac{b_N}{c_N} = \frac{\beta_M}{\gamma_M}$.

Now, we define $\delta = 2c^2\left(q - \frac{b}{c}\right)$. From the statement of the theorem, we now deduce that $|\delta| < 1$. Moreover, we can solve for q:

$$q = \frac{b}{c} + \frac{\delta}{2c^2}$$

Note that if $\delta = 0$, there is nothing to prove, because then we have $\frac{b}{c} = q$, and as q is rational, it is always the final convergent of itself.

We now make a distinction between two cases. First of all, suppose N is even and $\delta > 0$, or N is odd and $\delta < 0$. We are going to append some numbers to the continued fraction $[a_1, \ldots, a_N]$. More precisely, we want to find the continued fraction expansion of λ that satisfies $[a_1, \ldots, a_N + \lambda] = q$. Note that we can rewrite this as $[a_1, \ldots, a_N, \frac{1}{\lambda}] = q$. Hence, invoking the recursive formulas from theorem 5.13:

$$q = \frac{\frac{1}{\lambda} b_N + b_{N-1}}{\frac{1}{\lambda} c_N + c_{N-1}}$$

Isolating $\frac{1}{\lambda}$ yields:

$$\frac{1}{\lambda} c_N q + c_{N-1} q = \frac{1}{\lambda} b_N + b_{N-1} \Leftrightarrow \frac{1}{\lambda} = \frac{b_{N-1} - c_{N-1} q}{c_N q - b_N}$$

Expanding the fraction yields:

$$\begin{aligned} \frac{1}{\lambda} &= \frac{b_{N-1} - \frac{c_{N-1}}{c_N}(c_N q - b_N) - \frac{c_{N-1}}{c_N} b_N}{c_N q - b_N} = \frac{b_{N-1} - \frac{c_{N-1}}{c_N} b_N}{c_N q - b_N} - \frac{c_{N-1}}{c_N} \\ &= \frac{c_N b_{N-1} - c_{N-1} b_N}{c_N^2 q - c_N b_N} - \frac{c_{N-1}}{c_N} \end{aligned}$$

Using the relation for q derived above, we find $c_N^2 q - c_N b_N = \frac{\delta}{2}$. Hence, we find:

$$\frac{1}{\lambda} = 2\left(\frac{c_N b_{N-1} - c_{N-1} b_N}{\delta}\right) - \frac{c_{N-1}}{c_N}$$

From the previous theorem, we can rewrite the fraction with the relation $c_N b_{N-1} - c_{N-1} b_N = (-1)^N$. Hence, we obtain:

$$\frac{1}{\lambda} = \frac{2(-1)^N}{\delta} - \frac{c_{N-1}}{c_N}$$

We assumed that either N is even and $\delta > 0$, so in that case $\delta = |\delta| = (-1)^N |\delta|$, or N is odd and $\delta < 0$, in which case also $\delta = -|\delta| = (-1)^N |\delta|$. Thus, we find:

$$\frac{1}{\lambda} = \frac{2}{|\delta|} - \frac{c_{N-1}}{c_N}$$

By the recursive formulas for c_N in theorem 5.13, we find that c_n is a strictly increasing sequence. Combining this with $|\delta| < 1$, we obtain:

$$\frac{1}{\lambda} > \frac{2}{1} - 1 = 1$$

So $0 < \lambda < 1$, hence by theorem 5.10 there exists a finite continued fraction that equals λ with natural numbers, such that the last number does not equal 1. Hence, we find $\lambda = [z_1, \ldots, z_P]$, where $z_1, \ldots, z_P \in \mathbb{N}$ and $z_P \neq 1$. So, q can be written as:

$$q = [a_1, \ldots, a_N, z_1, \ldots, z_P]$$

But now, according to theorem 5.10, this is the only such continued fraction that equals q. Hence, $[a_1, \ldots, a_N] = \frac{b}{c}$ is a convergent of q.

The other case is fairly similar. We assume either N is even and $\delta < 0$ or N is odd and $\delta > 0$. Note that we then find that M is even and $\delta > 0$ or M is odd and $\delta < 0$. We now look for λ defined by the following equation:

$$q = [\alpha_1, \ldots, \alpha_M + \lambda]$$

Using the exact same method as above, we find:

$$\frac{1}{\lambda} = \frac{2}{|\delta|} - \frac{\gamma_{M-1}}{\gamma_M} > 2 - 1 = 1$$

Hence, $0 < \lambda < 1$, which again implies in the exact same way that $\frac{b}{c}$ is a convergent of q. □

Now, we apply the newly developed techniques to the problem of finding the order of x modulo N.

Corollary 5.15:
Let $0 < \overline{\phi} < 1$ be a rational number. Let $L, s, r \in \mathbb{N}$ be such that $1 \leq s \leq r - 1 < r \leq N \leq 2^L$. Suppose $|\frac{s}{r} - \overline{\phi}| \leq 2^{-2L-1}$. Let b_n and c_n be the sequences denoting the numerators and denominators of the convergents of $\overline{\phi}$ as in theorem 12. Then there exists an $M \in \mathbb{N}$ such that $\frac{s}{r} = \frac{b_M}{c_M}$ and $c_{M+1} \geq 2^L$.

Proof: First of all, we write $\frac{s}{r}$ as a fraction where the numerator and denominator do not have a common factor: $\frac{s'}{r'}$. So, $\frac{s}{r} = \frac{s'}{r'}$ and $\gcd(s', r') = 1$. Hence, we find:

$$\left| \frac{s'}{r'} - \overline{\phi} \right| = \left| \frac{s}{r} - \overline{\phi} \right| \leq 2^{-2L-1} \leq \frac{1}{2r^2} \leq \frac{1}{2(r')^2}$$

So, according to the previous theorem, $\frac{s'}{r'}$ is a convergent of $\overline{\phi}$, hence there exists an $M \in \mathbb{N}$ such that:

$$\frac{s}{r} = \frac{s'}{r'} = \frac{b_M}{c_M}$$

Also, we define $\delta = 2(r')^2\left(\frac{s'}{r'} - \overline{\phi}\right)$, in accordance with the previous theorem. In addition, we define $\varepsilon = 2^{2L+1}\left(\frac{s}{r} - \overline{\phi}\right)$. We find:

$$\begin{aligned}\overline{\phi} &= \frac{s'}{r'} + \frac{\delta}{2(r')^2} \\ \overline{\phi} &= \frac{s'}{r'} + \frac{\varepsilon}{2^{2L+1}}\end{aligned}$$

According to the statement of the theorem, we have $|\varepsilon| \leq 1$. Equating both equations yields:

$$\frac{\delta}{2(r')^2} = \frac{\varepsilon}{2^{2L+1}} \Leftrightarrow \delta = \frac{(r')^2\varepsilon}{2^{2L}}$$

We can choose the continued fraction such that M is odd and $\delta < 0$, or M is even and $\delta > 0$. This can be done by leaving out or appending the final 1. Hence, as we saw in the previous lemma, if we define λ by the following equation:

$$\overline{\phi} = [c_1, \ldots, c_M + \lambda]$$

Then we find the following expression for λ:

$$\frac{1}{\lambda} = \frac{2}{|\delta|} - \frac{c_{M-1}}{c_M} = \frac{2^{2L+1}}{(r')^2|\varepsilon|} - \frac{c_{M-1}}{c_M}$$

Using $|\varepsilon| \leq 1$, and $c_M = r'$:

$$\frac{1}{\lambda} \geq \frac{2^{2L+1}}{(r')^2} - \frac{c_{M-1}}{r'}$$

Hence, the value a_{M+1} in the continuous fraction expansion of $\overline{\phi}$ is lower bounded by:

$$a_{M+1} \geq \left\lfloor \frac{2^{2L+1}}{(r')^2} - \frac{c_{M-1}}{r'} \right\rfloor > \frac{2^{2L+1}}{(r')^2} - \frac{c_{M-1}}{r'} - 1$$

So, we find, using the recursive formula for the denominators of the approximations:

$$\begin{aligned}c_{M+1} &= a_{M+1}c_M + c_{M-1} > \left(\frac{2^{2L+1}}{(r')^2} - \frac{c_{M-1}}{r'} - 1\right) \cdot r' + c_{M-1} \\ &= \frac{2^{2L+1}}{r'} - r' = \frac{2^{2L}}{r'} + \frac{2^{2L}}{r'} - r' \geq \frac{2^L \cdot r'}{r'} + \frac{(r')^2}{r'} - r' = 2^L\end{aligned}$$

Hence, we have proven the theorem. □

Recall that the order finding algorithm returned a rational number $\overline{\phi}$, such that $0 \leq \overline{\phi} < 1$ and there exists a natural $0 \leq s \leq r-1$ such that $||\overline{\phi} - \frac{s}{r}||_1 \leq 2^{-2L-1}$. If we now require s to be unequal to 0, we find $|\frac{s}{r} - \overline{\phi}| \leq 2^{-2L-1}$, so we can apply the above theorem. As this corresponds to just one eigenvalue of the operator U that is invalid, this adds a factor $\frac{r-1}{r}$ to the total success probability, where we note that $\frac{r-1}{r} \geq \frac{1}{2}$ for $r \geq 2$.

So, in summary, if our order r is greater than 1, we can apply the order finding algorithm to find a rational estimation $\overline{\phi}$ of some $\frac{s}{r}$, where s is between 1 and $r-1$, with at least a fixed probability. Then, we apply the continued fraction expansion and select the last convergent that has a denominator smaller than 2^L. This convergent then equals $\frac{s}{r}$. Then, the denominator of the resulting expression, r', divides r. The following section will handle the finalizing step of finding r from r'.

Finally, let's have a look at the time complexity of the continued fraction expansion. To do this, we consider the denominators of the convergents, c_n, as defined in theorem 12. We find the following inequality for all $n \geq 3$, using the recursive relation defined in theorem 12 and the increasing property of c_n:

$$c_n = a_1c_{n-1} + c_{n-2} \geq c_{n-1} + c_{n-2} \geq 2c_{n-2}$$

As $c_1, c_2 \geq 1$, we find by induction:

$$c_n \geq 2^{\lfloor \frac{n}{2} \rfloor}$$

We also know that the denominator of $\overline{\phi}$ is smaller than 2^L. So, an upper bound for the number of convergents N of $\overline{\phi}$ is given by:

$$2^L \geq c_N \geq 2^{\lfloor \frac{N}{2} \rfloor} \geq 2^{\frac{N}{2}-1}$$

Hence:

$$L \geq \frac{N}{2} - 1 \Leftrightarrow N \leq 2L + 2 = \mathcal{O}(L)$$

The other operations that are involved in finding the continued fractions and calculating the convergents are simple calculations with a maximum of L-bit numbers, hence the time complexity of these operations is $\mathcal{O}(L^2)$ [6]. So, the total time complexity is given by $\mathcal{O}(L^3)$.

5.3.4 Finding the order from its divisors

We have now found a way of efficiently producing numbers that divide the order r of x modulo N. We now want to use these divisors of r to find the value of r itself.

The idea is to execute the process outlined above twice, in order to obtain two values r'_1 and r'_2, that both divide r. Our hope is now that we can find r by investigating the lowest common multiple of these two numbers. In this section, we will derive the probability that this yields the correct value of r.

Executing the order finding algorithm can be modeled as taking a natural number s uniformly in the interval $[1, r-1]$, and calculating $r' = r/\gcd(s, r)$ accordingly. So, if we perform the algorithm twice, we obtain two values for s and r', which we denote by s_1 and s_2 and r'_1 and r'_2, respectively. Our guess for r is now given by $\overline{r}$, defined as:

$$\overline{r} = \text{lcm}(r'_1, r'_2) = \text{lcm}(r/\gcd(s_1, r), r/\gcd(s_2, r))$$

Now, we denote the prime factorization of r by $p_1^{\alpha_1} \cdots p_n^{\alpha_n}$. We find that $\overline{r}$ does not equal r is equivalent to saying that there exists a prime factor of r that divides both s_1 and s_2. The probability of a prime factor of r, say p_i, dividing s_1 is given by the number of possible choices of s_1 that are divisible by p_i, divided by the number of total possible choices of s_1, $r-1$. Hence, we find:

$$P(p_i|s_1) = \left\lfloor \frac{r-1}{p_i} \right\rfloor \cdot \frac{1}{r-1} \leq \frac{r-1}{p_i(r-1)} = \frac{1}{p_i}$$

Similarly, we find $P(p_i|s_2) \leq \frac{1}{p_i}$. We only obtain the true value of r if there does not exist a prime factor of r that divides both s_1 and s_2, hence the following formula:

$$P(\overline{r} = r) = 1 - \sum_{i=1}^{n} P(p_i|s_1) \cdot P(p_i|s_2)$$

Applying the bounds we found on $P(p_i|s_1)$ and $P(p_i|s_2)$, we obtain:

$$P(\overline{r} = r) \geq 1 - \sum_{i=1}^{n} \frac{1}{p_i} \cdot \frac{1}{p_i} = 1 - \sum_{i=1}^{n} \frac{1}{p_i^2}$$

A priori, though, we do not know which prime factors r has. We know, though, that p_i can only be 2 or an odd number greater than or equal to 3. Hence, we can lower bound the probability that $\overline{r}$ equals r in the following way:

$$P(\overline{r} = r) \geq 1 - \frac{1}{4} - \sum_{n=1}^{\infty} \frac{1}{(2n+1)^2} = 2 - \frac{1}{4} - \sum_{n=0}^{\infty} \frac{1}{(2n+1)^2} = \frac{7}{4} - \frac{\pi^2}{8} > \frac{1}{2}$$

Here, the following relation was used, which can be proven using methods that find their origin in complex analysis.

$$\sum_{n=0}^{\infty} \frac{1}{(2n+1)^2} = \frac{\pi^2}{8}$$

Hence, we find the correct value of r with a probability of at least $\frac{1}{2}$. Combining this with the bound we found earlier, and the factor $(\frac{1}{2})^2$ from the requirement that both s_1 and s_2 cannot equal 0, we find that as long as $r \geq 2$ and $0 < \varepsilon < 0.24$, the algorithm succeeds with a probability of at least:

$$P(\text{success}) \geq \frac{1}{8}\left(1 - \varepsilon - \frac{4 + \left\lceil \log_2\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil \ln(2)}{2^{\left\lceil \log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}}\right)^2$$

The greatest common divisor and lowest common multiples can be obtained using Euclid's algorithm, which has time complexity $\mathcal{O}(L^3)$ [6]. As the previous steps also had time complexity $\mathcal{O}(L^3)$, we find that the time complexity of the order finding algorithm is $\mathcal{O}(L^3)$.

5.4 Shor's algorithm

Suppose $N > 1$ is a natural number. We would like to find a non-trivial divisor of N, i.e. a divisor of N that is not equal to 1 or N. On classical computers, this turns out to be a hard problem, in the sense that no efficient algorithm has been found that accomplishes this task. Shor's algorithm, though, does provide an efficient method of finding these divisors, using a quantum computer. This section will cover how the order finding algorithm, as outlined above, can be used to find a divisor of N. For notational convenience, we define L as the number of bits that are needed to represent N: $L = \lfloor \log_2(N) \rfloor + 1$.

The idea is to find a number $x \in \mathbb{N}$ such that $x^2 \equiv 1 \mod N$, where $x \not\equiv 1 \mod N$ and $x \not\equiv -1 \mod N$. If we find such an x, we observe that there exists a $k \in \mathbb{N}$ such that:

$$(x+1)(x-1) = x^2 - 1 = kN$$

Moreover, $x+1$ and $x-1$ are not divisors of N. Furthermore, $x+1$ and $x-1$ cannot both be coprime with N, because if this were the case, we would have $\gcd((x+1)(x-1), N) = 1$, which is a contradiction with $\gcd((x+1)(x-1), N) = N$. Hence, we find that either $1 < \gcd(x+1, N) < N$ or $1 < \gcd(x-1, N) < N$. So, either $\gcd(x+1, N)$ or $\gcd(x-1, N)$ is a non-trivial divisor of N, which is what we set out to do. Note that using Euclid's algorithm, these greatest common divisors can be calculated with $\mathcal{O}(L^3)$ operations.

So, the problem of finding a non-trivial divisor of N reduces to finding an $x \in \mathbb{N}$ such that $x^2 \equiv 1 \mod N$, $x \not\equiv 1 \mod N$ and $x \not\equiv -1 \mod N$. It turns out that it is not always possible to find x satisfying these criteria. Suppose, namely, that N is odd and that it can be written as a power of a prime, so that $N = p^a$ where p is a prime number and $a \in \mathbb{N}$. Suppose also that we find an $x \in \mathbb{N}$ satisfying the criteria stated above. As we can write $(x+1) - (x-1) = 2$, we find that $\gcd(x+1, x-1) \leq 2$. As N is odd, we find $p > 2$, hence p cannot divide both $x+1$ and $x-1$. So N must divide either $x+1$ or $x-1$, yielding $x \equiv 1 \mod N$ or $x \equiv -1 \mod N$. Thus, there is not a suitable choice for x.

Luckily, though, there is an efficient classical method of determining whether N can be written as a^b, with $a, b \in \mathbb{N}$. Suppose N can be written as such. Then, we first of all calculate $\alpha = \log_2(N)$, which takes $\mathcal{O}(L^2)$ operations [6]. Note that we now have $\alpha = b \log_2(a)$. Next, we guess $c = 2$ and calculate $\beta = \alpha/c$. If c now equals b, then $2^\beta = 2^{\log_2(a)} = a$, which is a natural number. Similarly, we guess $c = 3, 4, \ldots$. If we choose $c > \alpha$, though, we have $\beta = \alpha/c < 1$, which yields $1 < 2^\beta < 2$, so we will not find a natural number. So, we only have to guess at most α values of c, meaning that we have to do $\mathcal{O}(L)$ guesses in total. The complexity of the division operation is $\mathcal{O}(L^2)$, and finding the nearest integer of 2^β also takes $\mathcal{O}(L^2)$ operations. Hence, we can check if N can be written as a power in $\mathcal{O}(L^3)$ operations.

Furthermore, we will see shortly that there will also be a problem when finding a suitable choice for x when N is even. Checking beforehand when N is even is easy, though, and 2 is a straightforward non-trivial divisor.

So, if N is odd and it cannot be written as a power, then we might hope that the method outlined above works. We will now turn our attention to finding a suitable choice for x.

The idea is to choose an integer y in the interval $2 \leq y \leq N-2$ uniformly, and determine its order r modulo N using the order finding algorithm. If we now find that r is even, we can choose $x = y^{\frac{r}{2}}$. We then find $x^2 = y^r \equiv 1 \mod N$, as required. If in addition, it turns out that $x \not\equiv -1 \mod N$, we have found a suitable choice for x, and thus can find a non-trivial divisor of N with the steps outlined above.

In the following theorems, we will show that the probability of finding an r that is even such that $y^{r/2} \not\equiv -1 \mod N$ can be lower bounded for odd N. Therefore, we can find a lower bound for the probability of the algorithm to work. This requires some algebraic number theoretical background to be readable, which can be found in [7].

Lemma 5.16:
Let p be an odd prime, α be a positive integer, and d be the largest positive integer such that 2^d is a divisor of $\phi(p^\alpha)$, where ϕ is the Euler-ϕ function. Choose $x \in (\mathbb{Z}/p^\alpha\mathbb{Z})^*$ uniformly at random. Then the probability of 2^d dividing the order of x is exactly $\frac{1}{2}$.

Proof: We use that the group $(\mathbb{Z}/p^\alpha\mathbb{Z})^*$ is cyclic, as is proven in [7]. In addition, we use $\phi(p^\alpha) = p^{\alpha-1}(p-1)$. We immediately find that as p is odd, $\phi(p^\alpha)$ is even, so $d \geq 1$.

As the group $(\mathbb{Z}/p^\alpha\mathbb{Z})^*$ is cyclic, there exists a generating element, which we will denote by ξ. Hence, we can now write $x = \xi^k$, where $1 \leq k \leq \phi(p^\alpha)$. As $\phi(p^\alpha)$ is even, we find that k is odd with probability $\frac{1}{2}$ and k is even with probability $\frac{1}{2}$.

First of all, Suppose that k is odd. We denote the order of x by r. By Euler's theorem, we obtain:

$$\xi^{\phi(p^\alpha)} \equiv 1 \equiv x^r = \xi^{kr} \mod p^\alpha$$

Hence, as ξ is a generating element and the order of $(\mathbb{Z}/p^\alpha\mathbb{Z})^*$ is equal to $\phi(p^\alpha)$, we find that $\phi(p^\alpha)$ divides kr. As k is odd, 2^d must divide r, hence with probability of at least $\frac{1}{2}$, 2^d divides the order of x.

On the other hand, suppose k is even. Then:

$$x^r \equiv 1 = 1^{\frac{k}{2}} \equiv \left(\xi^{\phi(p^\alpha)}\right)^{\frac{k}{2}} = \xi^{\frac{k\phi(p^\alpha)}{2}} = x^{\frac{\phi(p^\alpha)}{2}} \mod p^\alpha$$

So, r must divide $\frac{\phi(p^\alpha)}{2}$, from which it follows that 2^d cannot divide r. Hence, by a probability of $\frac{1}{2}$, 2^d divides the order of x. $\square$

Theorem 5.17:
Suppose N has the following prime decomposition:

$$N = p_1^{\alpha_1} \dots p_m^{\alpha_m}$$

We choose an integer x uniformly in $(\mathbb{Z}/N\mathbb{Z})^*$ and denote its order by r. Then:

$$P(r \text{ is even} \wedge x^{\frac{r}{2}} \not\equiv -1 \mod N) \geq 1 - \frac{1}{2^{m-1}}$$

Proof: We prove the inverse, namely:

$$P(r \text{ is odd} \vee x^{\frac{r}{2}} \equiv 1 \mod N) \leq \frac{1}{2^{m-1}}$$

According to the Chinese remainder theorem, choosing an x uniformly in $(\mathbb{Z}/N\mathbb{Z})^*$ is identical to choosing integers x_j uniformly in $(\mathbb{Z}/p_j^{\alpha_j}\mathbb{Z})^*$, for $j = 1, \ldots, m$. Now, for every such j, define r_j as the order of x_j modulo $p_j^{\alpha_j}$. We find that r is equal to the lowest common multiple of $r_1, \ldots, r_m$.

Define d to be the largest integer such that 2^d is a divisor of r. Similarly, define d_j to be the largest integer such that 2^{d_j} divides r_j. We first of all prove that all values d_1 through d_n are equal if r is odd, or $x^{\frac{r}{2}} \equiv -1 \mod N$.

Suppose r is odd. Then as all r_j's are divisors of r, all r_j's must be odd as well. Hence, for all j, we find $d_j = 0$.

On the other hand, suppose that $x^{\frac{r}{2}} \equiv -1 \mod N$. Then, according to the Chinese remainder theorem, for all j, we find $x^{\frac{r}{2}} \equiv -1 \mod p_j^{\alpha_j}$. Hence, r_j cannot be a divisor of $\frac{r}{2}$, which is equivalent to saying that $2r_j$ cannot divide r. As r_j is a divisor of r, we find that 2^{d_j} does divide r, and 2^{d_j+1} does not, so d_j equals d, for all j.

So, if r is odd or $x^{\frac{r}{2}} \equiv -1 \mod N$, then all d_j's are equal. Take d_1 to be fixed. Now we know that there exists a maximum natural number D_2 such that 2^{D_2} divides $\phi(p_2^{\alpha_2})$. According to the previous lemma, we know that the probability that d_2 equals D_2, is equal to $\frac{1}{2}$. Hence, the probability that d_2 does not equal D_2, is equal to $\frac{1}{2}$ as well. As d_1 either equals D_2 or not, we conclude that the probability of d_2 equaling d_1 is at most $\frac{1}{2}$.

Identical arguments show that the probability of d_j equaling d_1 are upper bounded by $\frac{1}{2}$. So, we find that the probability that $x^{\frac{r}{2}} \equiv -1 \mod N$ or r is odd, is upper bounded by $\frac{1}{2^{m-1}}$. Hence, this proves the theorem. □

Note, that if N has two distinct prime factors, we find $m \geq 2$, hence $\frac{1}{2^{m-1}} \leq \frac{1}{2}$.

We can now summarize all results for Shor's algorithm in the box shown on the next page.

Algorithm: Shor's algorithm
Input: *A natural number $N > 1$ that is not prime. Let L be the number of bits needed to represent N.*
Parameters: $0 < \varepsilon < 0.24$.
Output: *A non-trivial divisor of N.*
Time complexity: $\mathcal{O}(L^3)$.
Probability of success: *Lower bounded by:*

$$\frac{1}{16}\left(1-\varepsilon-\frac{4+\left\lceil\log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil\ln(2)}{2^{\left\lceil\log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}}\right)^2$$

Procedure:

1. *Check if N is even. If yes, return 2.*
   Time complexity: $\mathcal{O}(1)$.
   Probability of success: 1.
2. *Check if N can be written as a power of two natural numbers. That is, check if there exist $a, b \in \mathbb{N}$ such that $a^b = N$ and $b \geq 2$. If yes, return a.*
   Time complexity: $\mathcal{O}(L^3)$.
   Probability of success: 1.
3. *Choose an integer y uniformly in the interval $[2, N-2]$. If $\gcd(y, N) > 1$, return $\gcd(y, N)$.*
   Time complexity: $\mathcal{O}(L^3)$.
   Probability of success: 1.
4. *Apply the order finding algorithm to find the order r of y modulo N. Use $2L + 1 + \left\lceil\log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil$ qubits in the first register and L qubits in the second register. Check afterwards if the correct value for r was found. If not, fail.*
   Time complexity: $\mathcal{O}(L^3)$.
   Probability of success: *Lower bounded by:*

$$\frac{1}{8}\left(1-\varepsilon-\frac{4+\left\lceil\log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil\ln(2)}{2^{\left\lceil\log_2\left(2+\frac{1}{2\varepsilon}\right)\right\rceil}}\right)^2$$

5. *Check if r is even and $y^{\frac{r}{2}} \not\equiv -1 \mod N$. If not, fail.*
   Time complexity: $\mathcal{O}(L^3)$.
   Probability of success: *Lower bounded by $\frac{1}{2}$.*
6. *Return* $\max\left(\gcd\left(y^{\frac{r}{2}}-1 \mod N, N\right), \gcd\left(y^{\frac{r}{2}}+1 \mod N, N\right)\right)$.
   Time complexity: $\mathcal{O}(L^3)$.
   Probability of success: 1.

The algorithm, as outlined above, can be simulated on a classical computer. It goes without saying that step 4 will no longer have a time complexity of $\mathcal{O}(L^3)$ in the simulation process. The implementation of this simulation can be found in appendix B.

6 Conclusion

This text has reached two main results. First of all, it was shown that any quantum circuit can be implemented using a finite set of components, up to arbitrary accuracy. Secondly, it was shown that the algorithm of Shor provides an efficient way of factorizing integers using a quantum computer. Together, they provide an introduction into the realm of quantum computing.

If one were to do a follow-up study, then it would be interesting to investigate the lower bound on the probability that Shor's algorithm succesfully factorizes the number given. This lower bound can probably be improved by investigating which step introduces the most inaccuracy into the bound.

Furthermore, the classical simulation of Shor's algorithm could be improved significantly by having a look at the implementation of the inverse Fourier transformation. Simple measurements reveal that over 80% of the runtime of the code is spent calculating the inverse Fourier transform, which could be significantly sped up by implementing other algorithms to perform this operation.

Ultimately, though, the field of quantum computing will only become a very exciting field of study when the quantum computer is at one's disposal. Until this is experimentally feasible, devising other quantum algorithms that definitively beat the implementations on classical computers is one of the most important tasks for researchers. More generally, investigating the full potential of the concept of the quantum computer is of vital importance to the field as a whole.

References

[1] IBM. *The IBM Quantum Experience*. <http://www.research.ibm.com/quantum/>, 2016.

[2] Nielsen, M.A. and Chuang, I.L., 2000. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press.

[3] Aliprantis, C.D. and Burkinshaw, O., 1990. *Principles of real analysis*, 3rd edition. San Diego: Acadamic Press.

[4] Ryan, A.R., 2002. *Introduction to tensor products of Banach spaces*. London: Springer.

[5] Vedral, V., Barenco, A. and Ekert, A., 1995. *Quantum networks for Elementary Arithmic Operations*. arXiv:quant-ph/9511018.

[6] Knuth, D.E., 1981. *The Art of Computer Programming*. Second edition. Boston: Addison-Wesley.

[7] Stevenhagen, P., 2016. *Algebra I*. Leiden: Universiteit Leiden.

A Proof that λ defined by the λ-equation is irrational

In this section, we wish to prove that λ is irrational if λ is a real number that satisfies the following equation, referred to as the λ-equation:

$$\cos(\lambda\pi) = \cos^2\left(\frac{\pi}{8}\right)$$

The proof of this statement involves a considerable amount of abstract algebra. However, the reader is only required to be familiar with those contents of algebra that are taught in a mathematics Bachelor. Thus, all non-trivial results from algebra are proven below, with the exception of the following theorems [7].

Theorem A.1: Little theorem of Fermat
Let p be a prime number. Then for any integer a satisfying $0 \leq a \leq p-1$, the following relation holds:

$$a^p \equiv a \mod p$$

Theorem A.2: Fundamental theorem of algebra
Let p be a non-constant polynomial with complex coefficients. Then it has at least one complex root.

Theorem A.3: Finite fields
Let p be a prime number. Then the set $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ is a field under addition and multiplication modulo p.

This appendix will be structured as follows. Section A.1 will introduce some of the definitions frequently used in the field of algebra, and additionally proofs of some very fundamental theorems of algebra will be provided. In section A.2, the focus will be on cyclotomic polynomials and how their properties follow from the theorems in section A.1. Finally, section A.3 will be devoted to showing how the properties of cyclotomic polynomials eventually lead to the statement that λ is irrational.

For additional clarity, figure A.1 contains a graphical overview of the steps that together form the proof that λ is an irrational number.

A.1 Required knowledge from abstract algebra

This section will introduce the concepts from abstract algebra that are needed in the proof that λ is irrational. First of all, the required definitions will be given, and subsequently some important theorems will be proven.

A.1.1 Definitions

The definitions covered below are mainly included for extra clarification and completeness.

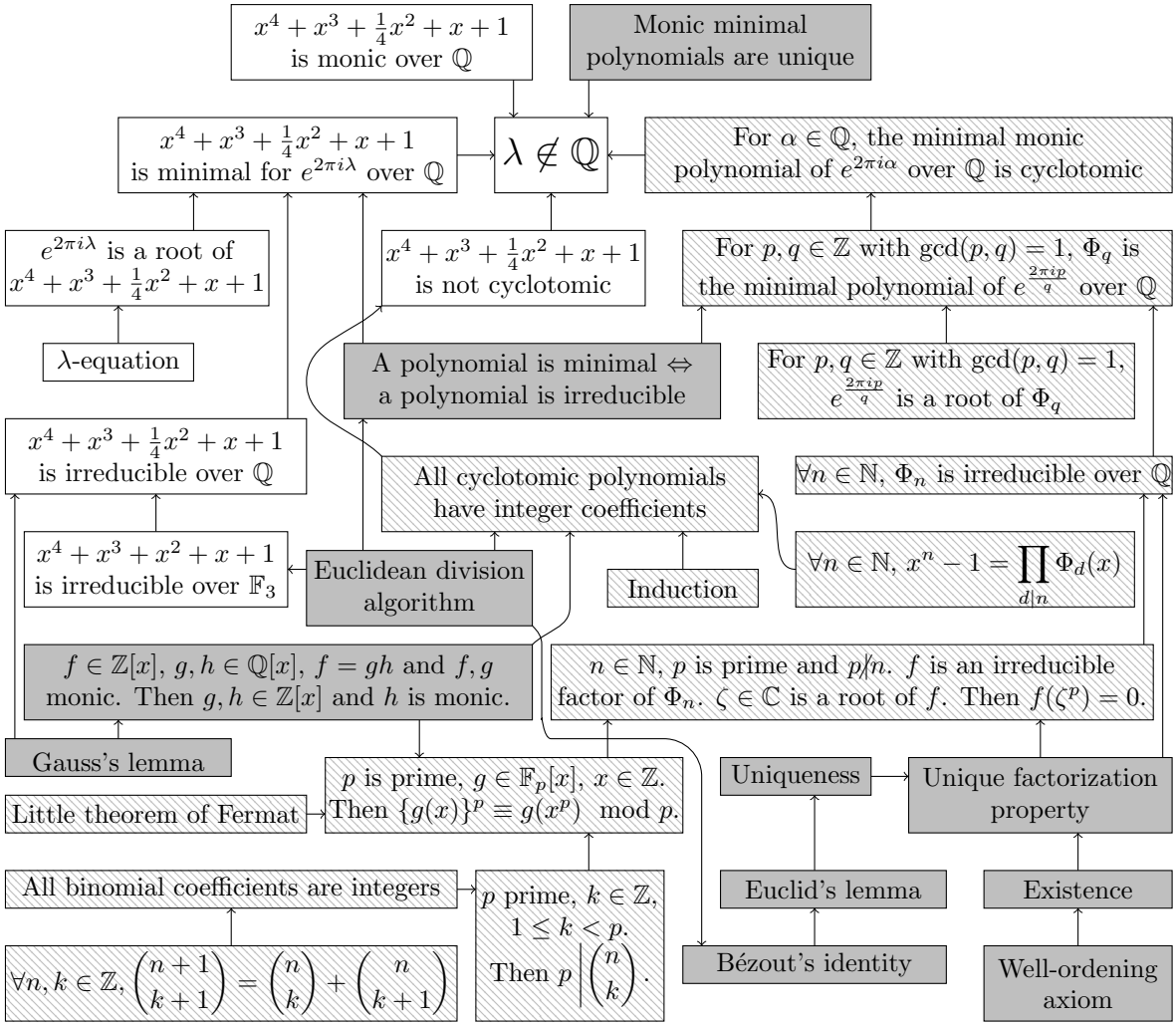


Figure A.1: Flowchart indicating the steps taken in order to prove the irrationality of λ. The gray cells are the steps taken in section A.1. The hatched ones are considered in section A.2. Lastly, the white ones are covered in section A.3.

Definition A.4: Rings
Let A be a set on which an addition and a multiplication operation are defined. Then A is said to be a ring if it satisfies the following ring-axioms:

1. *A is closed under addition and multiplication. So, for every $a, b \in A$, $a + b \in A$ and $ab \in A$.*
2. *The addition and multiplication operations are associative. So, for every $a, b, c \in A$, $(a + b) + c = a + (b + c)$ and $(ab)c = a(bc)$.*
3. *The addition and multiplication operations are commutative. So, for every $a, b \in A$, $a + b = b + a$ and $ab = ba$.*
4. *There exist additive and multiplicative identity elements in A. So, there exist $0_A, 1_A \in A$ such that for every $a \in A$, $a + 0_A = a$ and $a1_A = a$.*
5. *There exist additive inverses. So, for any $a \in A$, there exists a $(-a) \in A$ such that $a + (-a) = 0_A$.*
6. *The multiplication is distributive over addition. So, for any $a, b, c \in A$, $a(b + c) = ab + ac$.*

Definition A.5: Fields
Let A be a ring. Then A is said to be a field if it, in addition, satisfies the following field-axiom:

7. *There exist multiplicative inverses. So, for any $a \in A \setminus \{0_A\}$, there exists an $a^{-1} \in A$ such that $aa^{-1} = 1_A$.*

Note that every field is automatically a ring. One can easily verify that $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{Q}$ are indeed fields. The additive identity is 0 and the multiplicative identity is 1 in all these fields. On the other hand, $\mathbb{Z}$ is merely a ring as for example $2 \in \mathbb{Z}$ does not have a multiplicative inverse in $\mathbb{Z}$. The additive and multiplicative identities of $\mathbb{Z}$ are 0 and 1 as well, respectively.

Definition A.6: Polynomials
Let A be a ring. Let $p(x)$ be an expression of the following form:

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$$

with coefficients $p_0, p_1, \ldots, p_n$ in A. Then p is called a polynomial in x over A. The collection of all such polynomials is denoted by $A[x]$.

Note that polynomials are not functions, they are merely the expressions themselves. Consequently, polynomials are generally not bound to a certain domain. Furthermore, we say that two polynomials are equal if and only if all their coefficients are equal. Adding and multiplying polynomials will be done in the standard way of expanding the parentheses. Without further notice, it will be assumed that the coefficients of a polynomial p over A are denoted by the elements $p_0, p_1, \ldots, p_n$ of A.

Corollary A.7: Polynomial rings
Let A be a ring and $A[x]$ be the collection of all polynomials in x with coefficients in A. Then $A[x]$ is a ring.

The validity of this statement can easily be checked by checking that $A[x]$ satisfies all ring axioms. The additive identity of $A[x]$ is the polynomial of which all coefficients are 0_A. The multiplicative identity is the polynomial for which only the first coefficient is 1_A, and the others are 0_A.

Definition A.8: Degree of polynomials
Let A a be ring, and p be a polynomial over A. Then the maximal index of the non-zero coefficients of p is called the degree of the polynomial p. If all coefficients are zero, then the degree is -1. The degree of p is denoted by $\deg(p) \in \mathbb{N} \cup \{0, -1\}$. *A polynomial is said to be constant if its degree is* 0 *or* -1.

For example, the degree of $x^2 - 1$ is 2, and the degree of 2 is 0 hence it is constant. Furthermore, $\deg(0) = -1$ and so 0 is a constant polynomial as well.

Theorem A.9: Degree of the product of polynomials
Let A be a field and $p, q \in A[x]$ be non-zero polynomials. Then $\deg(pq) = \deg(p) + \deg(q)$.

Proof: By expanding the parentheses, one finds easily that the leading coefficient term of pq is $p_{\deg(p)}q_{\deg(q)}x^{\deg(p)+\deg(q)}$. Following from the definition of the degree of a polynomial, $p_{\deg(p)}$ and $q_{\deg(q)}$ are both non-zero, hence their product is non-zero as well. So, $\deg(pq) = \deg(p) + \deg(q)$. □

For example, we have $2 = \deg(x^2 - 1) = \deg((x - 1)(x + 1)) = \deg(x - 1) + \deg(x + 1) = 1 + 1$.

Definition A.10: Monic polynomials
Let A be a ring. Then $p \in A[x]$ is said to be monic over A if its leading coefficient (that is, $p_{\deg(p)}$) is 1_A.

For example, $x^3 - 4x^2 + 3$ is monic over $\mathbb{Q}$.

Definition A.11: Irreducible polynomials
Let A be a ring and $p \in A[x]$ be a non-constant polynomial. Then p is said to be irreducible over A if there do not exist non-constant polynomials $q, r \in A[x]$ such that $p = qr$. Otherwise, p is said to be reducible over A.

For example, $x^2 - 1$ is reducible over $\mathbb{R}$, as it can be written as $(x - 1)(x + 1)$. The same goes for $x^2 - 2$, as it can be written as $(x - \sqrt{2})(x + \sqrt{2})$. However, $x^2 - 2$ is irreducible over $\mathbb{Q}$.

Definition A.12: Roots
Let A be a ring. Let $\alpha \in A$ be a constant. Let $B \subseteq A$ be a ring such that $0_A = 0_B$ and $1_A = 1_B$. Let $p \in B[x]$ be a polynomial. Then α is said to be a root of p if $p(\alpha) = 0_A$.

For example, $\sqrt{2} \in \mathbb{R}$ is a root of $x^4 - 4 \in \mathbb{Q}[x]$.

Definition A.13: Minimal polynomials
Let A be a ring and $\alpha \in A$ a constant. Let $B \subseteq A$ be a ring such that $0_A = 0_B$ and $1_A = 1_B$. Then a non-zero polynomial in $B[x]$ of least degree that has α as a root is said to be a minimal polynomial of α over B.

For example, $x^2 - 2$ is a minimal polynomial of $\sqrt{2}$ over $\mathbb{Q}$.

Theorem A.14: Uniqueness of monic minimal polynomials
Let A be a ring and $\alpha \in A$ be a constant. Let $B \subseteq A$ be a ring such that $0_A = 0_B$ and $1_A = 1_B$. Then if there exists a monic minimal polynomial of α over B, then this monic minimal polynomial is unique.

Proof: Suppose we have two monic minimal polynomials, $p, q \in B[x]$ such that $p(\alpha) = q(\alpha) = 0$. Then they must have equal degree, as otherwise one of them would not be minimal. But then $(p-q)(\alpha) = p(\alpha)-q(\alpha) = 0 - 0 = 0$, so α is also a root of $p - q$. But the leading terms of p and q are equal, so they cancel in $p - q$, hence $\deg(p - q) < \deg(p) = \deg(q)$. So, we have that p and q cannot be minimal polynomials, which yields a contradiction, unless $p - q = 0$. But then $p = q$. □

A.1.2 Euclidean division

In most standard calculus courses, students are taught the method of partial fraction expansion. This technique is especially useful for integrating rational functions. An example is supplied below.

Suppose one wants to evaluate the following integral:

$$\int_1^2 \frac{2x^4 - x^3 + 4x^2 - 3x + 8}{x + 3}\,\mathrm{d}x$$

Then, using the method of partial fraction expansion, one splits the fraction in the integrand such that the degree of the polynomial in the numerator is less than the degree in the denominator. The integrand is consequently rewritten as follows:

$$\begin{aligned}\frac{2x^4 - x^3 + 4x^2 - 3x + 8}{x+3} &= 2x^3 + \frac{-7x^3 + 4x^2 - 3x + 8}{x+3} \\ &= 2x^3 - 7x^2 + \frac{25x^2 - 3x + 8}{x+3} \\ &= 2x^3 - 7x^2 + 25x + \frac{-78x + 8}{x+3} \\ &= 2x^3 - 7x^2 + 25x - 78 + \frac{242}{x+3}\end{aligned}$$

So, one ends up with four easily integrable terms, and one rational term that has a polynomial of lesser degree in the numerator compared to the degree of the one in the denominator.

The freshman student who did this integration probably wasn't aware that he or she just applied the Euclidean division algorithm. The corresponding theorem states that this method always works and the resulting terms and rational are unique.

Theorem A.15: Euclidean division
Let A be a field and $p, q \in A[x]$ with q non-zero. Then there exist two unique polynomials $r, s \in A[x]$ such that $p = qr + s$ and $\deg(s) < \deg(q)$.

Proof: First we prove the existence of such a $r, s \in A[x]$. Subsequently, we prove the uniqueness of such a choice of r and s.

Suppose $\deg(p) < \deg(q)$. Then take $r = 0$ and $s = p$. We have $p = q0 + s = p$ and $\deg(s) = \deg(p) < \deg(q)$.

Conversely, suppose $\deg(p) \geq \deg(q)$. We define $n = \deg(p)$ and $m = \deg(q)$. Then we define $r_{n-m} = \frac{p_n}{q_m}$. Next, we define $p'_{n-1} = p_{n-1} - q_{m-1}r_{n-m}$, $p'_{n-2} = p_{n-2} - q_{m-2}r_{n-m}$, etc. until we reach q_0. Then, we define $r_{n-m-1} = \frac{p'_{n-1}}{q_m}$. We again define new coefficients: $p''_{n-2} = p'_{n-2} - q_{m-1}r_{n-m-1}$, etc. We continue in this fashion until we have defined all coefficients of r, including r_0. The resulting coefficients $p_{m-1}^{(m-1)}, \ldots, p_0^{(m-1)}$ form the coefficients of s. Then we have $p = qr + s$ and $\deg(s) \leq m - 1 < m = \deg(q)$. Thus we have proven the existence.

Suppose that there are two choices for r and s. So we have $r_1, r_2, s_1, s_2 \in A[x]$ such that $p = qr_1 + s_1 = qr_2 + s_2$, $\deg(s_1) < \deg(q)$ and $\deg(s_2) < \deg(q)$. Then $q(r_1 - r_2) + (s_1 - s_2) = p - p = 0$. So $q(r_1 - r_2) = (s_2 - s_1)$. If r_1 and r_2, and s_1 and s_2 are mutually different, their differences are non-zero polynomials of lesser or equal degree. So:

$$\deg(q) + \deg(r_1 - r_2) = \deg(q(r_1 - r_2)) = \deg(s_2 - s_1) \leq \deg(s_1) < \deg(q)$$

But we know that $\deg(r_1 - r_2) \geq 0$, so we have reached a contradiction. Hence, $s_1 = s_2$ or $r_1 = r_2$, either of which trivially yields the other. So, we have also proven uniqueness. □

The preceding theorem is arguably one of the most important in algebra, as many theorems are based on this observation. This is for example the case for the following theorem, which is needed later in this text.

Theorem A.16: Equivalence of minimal and irreducible polynomials
Let A and $B \subseteq A$ be fields such that $0_A = 0_B$ and $1_A = 1_B$. Let $\alpha \in A$ be a constant. Let $p \in B[x]$ be a polynomial such that $p(\alpha) = 0$. Then p is irreducible over B if and only if p is a minimal polynomial of α over B.

Proof: Suppose that p is irreducible and that it is not a minimal polynomial of α. Then there is a minimal polynomial of α, $q \in B[x]$, such that $q(\alpha) = 0$ and $\deg(q) < \deg(p)$. As q is minimal, it is non-zero, so according to the Euclidean division algorithm, there exist $r, s \in B[x]$ such that $p = qr + s$ and $\deg(s) < \deg(q)$.

But then $0 = p(\alpha) = q(\alpha)r(\alpha) + s(\alpha) = 0r(\alpha) + s(\alpha) = s(\alpha)$. So we find $s(\alpha) = 0$ and $\deg(s) < \deg(q)$. But then we have s is a minimal polynomial of α instead of q, yielding a contradiction, unless $s = 0$. But then we have $p = qr$, hence p is reducible, which is also a contradiction. So, if p is irreducible over B, it is a minimal polynomial of α over B.

Suppose that p is a minimal polynomial of α over B and that it is reducible. Then there exist non-constant $q, r \in B[x]$ such that $p = qr$. This implies $\deg(p) = \deg(q) + \deg(r)$, and as q and r are non-constant, we have $\deg(q) < \deg(p)$ and $\deg(r) < \deg(p)$. Yet we also have $p(\alpha) = 0$ and thus either $q(\alpha) = 0$ or $r(\alpha) = 0$. Hence we have found a polynomial of lesser degree than p which α is a root of, so we find that p is not a minimal polynomial of α. This is a contradiction, so we find that if p is a minimal polynomial of α over B, then it is irreducible over B. □

A.1.3 Unique factorization theorem

In a standard algebra course, one learns the unique factorization theorem for integers. This theorem states that every positive integer can be uniquely written as a finite product of prime factors. The resulting product is called the prime factorization of this number. A similar statement exists for polynomials. The goal of this section is to formulate and prove this theorem. The general idea of this proof is similar to the one in the unique factorization theorem for integers.

First, the notion of divisors and greatest common divisors is generalized to polynomials.

Definition A.17: Divisor
Let A be a ring and $f, g \in A[x]$ be polynomials over A. Then g is said to be a divisor of f over A if there exists a $h \in A[x]$ such that $f = gh$. We denote this by $g|f$.

For example, $x + 1$ is a divisor of $x^2 - 1$ over $\mathbb{Z}$, because $x^2 - 1 = (x + 1)(x - 1)$.

Definition A.18: Greatest common divisor
Let A be a field and $f, g \in A[x]$ be non-zero polynomials over A. Then let $h \in A[x]$ be a monic polynomial of highest degree such that $h|f$ and $h|g$. Then h is said to be a greatest common divisor of f and g. This is denoted by $h = \gcd(f, g)$.

Corollary A.19: Greatest common divisor of irreducible polynomials
Let A be a field and $f, g \in A[x]$ be non-zero irreducible polynomials over A that differ by more than just a constant factor. Then their greatest common divisor is 1_A.

Proof: Suppose c is a greatest common divisor of f and g with $\deg(c) \geq 1$. Then $c|f$ and $c|g$, hence there exist $m, n \in A[x]$ such that $f = mc$ and $g = nc$. But f is irreducible and c is non-constant, hence m must be constant. A similar argument shows that n must be costant. But then $f = \frac{m}{c}g$, hence f and g differ no more than just a constant factor. So, we find a contradiction, and thus the greatest common divisor is a polynomial of degree at most 0. In addition, this polynomial must be monic, hence we are left with the trivial divisor of f and g, 1_A. □

Lemma A.20: Bézout's identity
Let A be a field and $f, g \in A[x]$ be non-zero polynomials over A. Let h be a greatest common divisor of f and g. Then h is uniquely determined, and there exist $p, q \in A[x]$ such that $h = fp + gq$.

Proof: Define S as the set of polynomials given by $fr + gs$ for some $r, s \in A[x]$. Then define $R = \{\deg(s) : 0_A \neq s \in S\}$. This set R has a minimal element, k. Take $p', q' \in A[x]$ such that $\deg(fp' + gq') = k$. Then define d by scaling p' to p and q' to q such that $d = fp + gq$ becomes monic.

According to the Euclidean division algorithm, we can find polynomials $u, v \in A[x]$ such that $f = du + v$

and $\deg(v) < \deg(d)$. We can isolate v and substitute d, as follows:

$$v = f - du = f - (fp + gq)u = f(1 - pu) - gqu$$

Hence we find v is also a linear combination of f and g, so unless $v = 0_A$, $v \in S$. But $\deg(v) < \deg(d)$, so then $k \leq \deg(v) < \deg(d) = k$. Hence, we reached a contradiction, and so we must have $v = 0_A$. Thus, $f = du$, hence d is a divisor of f. Completely analogously, one can prove that d is a divisor of g.

Suppose $c \in A[x]$ is also a divisor of f and g. Then we have $f = mc$ and $g = nc$ for some $m, n \in A[x]$. Then, we find:

$$d = fp + gq = mcp + ncq = c(mp + nq)$$

We know $d \neq 0_A$, so $mp + nq \neq 0_A$ and $c \neq 0_A$. Hence:

$$\deg(d) = \deg(c) + \deg(mp + nq) \geq \deg(c)$$

So, d is indeed a greatest common divisor of f and g. As we previously found that $d = fp + gq$, we have now proven the latter part of the theorem.

Now suppose that C is also a greatest common divisor of f and g. Then, as before, we have $f = MC$ and $g = NC$ for some $M, N \in A[x]$. Again:

$$d = fp + gq = MCp + NCq = C(Mp + Nq)$$

As D and c are both greatest common divisors of f and g, they have equal degree. As we also have $d \neq 0_A$, $C \neq 0_A$ and $Mp + Nq \neq 0_A$, we must have that $\deg(Mp + Nq) = 0$ and so $Mp + Nq$ is constant. But C and d are both monic, so this constant must be 1_A. Hence, $d = C1_A = C$. So, the greatest common divisor of f and q is unique. □

Note that as it is now clear that the greatest common divisor of two polynomials is unique, the notation $\gcd(\cdot, \cdot)$ makes sense.

Lemma A.21: Euclid's lemma
Let A be a field and $p, q \in A[x]$ be two irreducible polynomials over A. If for $r \in A[x]$, $r|pq$, then $r|p$ or $r|q$.

Proof: Suppose that $r|pq$ and r is not a divisor of p. We know that there exists a $n \in A[x]$ such that $rn = pq$. The only divisors of p are 1_A and p, and as r is not a divisor of p, we find that $\gcd(r, p) = 1_A$. Bézout's identity now yields that there exist $a, b \in A[x]$ such that $ra + pb = 1_A$. Multiplying by q yields $q = raq + pbq = raq + brn = r(aq + bn)$. Hence $r|q$. With an identical argument, one can show that r divides p whenever it does not divide q. □

Note that from the above lemma, it is clear that the irreducible polynomials have the same role in polynomial factorization as the prime numbers have in integer factorization.

Now, all groundwork has been laid to provide a proof for the following important theorem, which is the polynomial analogue of the prime factorization.

Theorem A.22: Unique factorization property
Let A be a field and $p \in A[x]$ be a non-constant polynomial. Then p is a product of a finite set of irreducible polynomials over A. Moreover, these polynomials are unique up to a global constant and their overall order.

Proof: First of all, we prove that every polynomial can be written as the product of irreducible polynomials. To this end, suppose that S constitutes the set of all non-constant polynomials that cannot be written as the product of irreducible polynomials. Then take R the set of the degrees of all such polynomials. Then $R \subseteq \mathbb{N}$, hence R has a minimal element k. Denote $s \in S$ a polynomial that cannot be written as a product of irreducible polynomials with $\deg(s) = k$. Then s cannot be irreducible itself, so it has to be reducible. Hence there exist non-constant $q, r \in A[x]$ such that $s = qr$. But then $\deg(q) < \deg(s)$ and $\deg(r) < \deg(s)$, so q and r are not elements of S. Hence they can be written as the product of irreducible

polynomials. But then s can be written as the compound product of these irreducible polynomials. Hence we have reached a contradiction.

Now suppose that p can be written as the product of two sets of irreducible polynomials. So, suppose that $p_1, \ldots, p_n, q_1, \ldots, q_m$ are irreducible polynomials such that:

$$p = \prod_{i=1}^{n} p_i = \prod_{i=1}^{m} q_i$$

We can write $p_1 \ldots p_n = q_1 \ldots q_m$. So we observe $p_1 | q_1 \ldots q_m$. Applying Euclid's lemma repeatedly yields that there exists an i_1 such that $p_1 | q_{i_1}$. As both are irreducible, we have that they are equal up to a constant, so $c_1 p_1 = q_{i_1}$. We now find that $p_2 \ldots p_n = c_1^{-1} q_1 \ldots q_{i_1 - 1} q_{i_1 + 1} \ldots q_m$. Again, we can now find a i_2 such that $c_2 p_2 = q_{i_2}$. We can continue in this manner until one runs out of factors on one side of the equation.

Suppose $n > m$. Then one ends up with $p_{n-m+1} \ldots p_n = c_1^{-1} \ldots c_m^{-1}$. But then the degree on the right side of the equality is 0, and on the left side strictly larger than 0. So, n cannot be strictly greater than m.

Suppose on the other hand $n < m$. Then one ends up with $c_1 \ldots c_n$ on the left side of the equation and a product of $m - n$ irreducible polynomials on the right hand side. Again, the degree of the part on the left side of the equality is 0, whereas the degree of the expression on the right side is strictly greater than 0. So, we find $n = m$.

So, we have found that the number of irreducible polynomials making up the product is fixed, and that the irreducible polynomials themselves have a counterpart which they only differ by a factor with. This completes the proof. □

For example, $x^4 - 1 = (x^2 + 1)(x^2 - 1) = (x^2 + 1)(x + 1)(x - 1)$. The last three factors are all irreducible over $\mathbb{R}$. The unique factorization property of $\mathbb{R}[x]$ now tells us that other products of irreducible polynomials that together form $x^4 - 1$ have the same structure, only a different order and may differ by constants. For example: $x^4 - 1 = (2x - 2)(2x + 2)(\frac{1}{4}x^2 + \frac{1}{4})$.

A.1.4 Gauss's lemma

The next important result that is needed in the proof that λ is an irrational number is Gauss's lemma. After the next definition, the lemma is stated and proven. More general versions of the following definitions and lemmas exist, but they are slightly more complicated and less intuitive. These ones turn out to be sufficiently general for the purpose of this text.

Definition A.23: Primitive polynomials over $\mathbb{Z}$
Let $p \in \mathbb{Z}[x]$. Then p is said to be a primitive polynomial over $\mathbb{Z}$ if the greatest common divisor of all pairs of its coefficients is 1.

For example, $3x^2 - 2$ is a primitive polynomial over $\mathbb{Z}$, as $\gcd(3, -2) = 1$.

Lemma A.24: Gauss's lemma
If $p, q \in \mathbb{Z}[x]$ are primitive polynomials, pq is also a primitive polynomial over $\mathbb{Z}$. Furthermore, if $p \in \mathbb{Z}[x]$ is primitive over $\mathbb{Z}$, then it is irreducible over $\mathbb{Z}$ if and only if it is irreducible over $\mathbb{Q}$.

Proof: Because $\mathbb{Z}[x]$ is a ring of polynomials, $pq \in \mathbb{Z}[x]$. We now prove that pq is primitive. Suppose it is not. Then there exists a prime element $a \in \mathbb{Z}$ such that all coefficients of pq are divisible by a. Now the coefficients of p and q are not all divisible by a, so there exist coefficients p_k and q_l with maximal indeces k and l such that p_k and q_l are not divisible by a. But now $(pq)_{k+l}$ is a sum of terms that are all divisible by a, except for $p_k q_l$. Hence $(pq)_{k+l}$ is not divisible by a, which is a contradiction. Hence, we find that pq is a primitive polynomial over A.

Now, suppose that p is primitive over $\mathbb{Z}$. If p is irreducible over $\mathbb{Q}$, it is clear that it is irreducible over $\mathbb{Z}$ as well, as $\mathbb{Z} \subseteq \mathbb{Q}$. On the other hand, suppose that p is irreducible over $\mathbb{Z}$, but not irreducible over $\mathbb{Q}$. Then there exist a $q, r \in \mathbb{Q}[x]$ such that $p = qr$. All coefficients of q and r can be written as fractions of integers. We now define $c_q \in \mathbb{Z}$ as the lowest common multiple of all denominators of the coefficients of q, and we define $c_r \in \mathbb{Z}$ in a similar manner. We now define $q' = c_q q$ and $r' = c_r r$, such that $p = (c_q c_r)^{-1} q' r'$. Now q' and r' are primitive polynomials over $\mathbb{Z}$, hence $q'r'$ is a primitive polynomial over $\mathbb{Z}$ as well. But then for p to be a primitive polynomial in $\mathbb{Z}[x]$, we must have that $(c_q c_r)^{-1} = 1$. Hence $p = q'r'$, which contradicts the irreducibility of p. Hence p is also irreducible over $\mathbb{Q}$. □

An important corollary is given by the following lemma:

Corollary A.25: Factorization of monic polynomials over $\mathbb{Z}$
Let $f \in \mathbb{Z}[x]$ be a monic polynomial. Suppose furthermore that $g, h \in \mathbb{Q}[x]$ and $f = gh$. If either g or h is monic, then $g, h \in \mathbb{Z}[x]$ and both g and h are monic.

Proof: The coefficient of the leading term of the polynomial gh is given by the product of the coefficients of the leading terms of the individual polynomials. Hence, if f and g are monic, so must be h, and if f and h are monic, g must be monic as well.

We know that $g, h \in \mathbb{Q}[x]$. Suppose one of them is not a member of $\mathbb{Z}[x]$. Without loss of generality, we can assume that this is g, so $g \notin \mathbb{Z}[x]$. That means that the coefficients of g are all fractions of integers. Hence, there exists a lowest common multiple of the denominators of these fractions, $c_g \in \mathbb{Z}$ with $c_g \geq 2$. If $h \notin \mathbb{Z}[x]$, then there also exists a $c_h \in \mathbb{Z}$ defined in a similar manner. If, instead, $h \in \mathbb{Z}[x]$, then at least h is monic, so h must be primitive. In that case we define $c_h = 1 \in \mathbb{Z}$.

Now, we define $g' = c_g g$ and $h' = c_h h$. Then g' and h' are primitive polynomials over A, according to Gauss's lemma. Furthermore, $f = gh = (c_g c_h)^{-1} g'h'$. Again, we observe that f is primitive, hence $(c_g c_h)^{-1} = 1$. But then $c_g c_h = 1$, hence we find $c_g = c_h = 1$ or $c_g = c_h = -1$. Both contradict $c_g \geq 2$. Therefore, both g and h must be members of $\mathbb{Z}[x]$. □

For example, we know that $x+1$ is a divisor of x^2-1. Note that both are monic and have integer coefficients. The preceding lemma now predicts that the polynomial that is obtained by dividing x^2-1 by $x+1$ is again monic and has integer coefficients. Indeed, $\frac{x^2-1}{x+1} = x-1$, so this prediction is correct.

A.2 Cyclotomic polynomials

In this section, the focus will shift from general results from algebra towards the somewhat more specific subject of cyclotomic polynomials. Their definition will be included first, after which some properties will be proven.

A.2.1 Definition

The following definition forms the core of this section.

Definition A.26: Cyclotomic polynomials
Let $n \in \mathbb{N}$. Then define the set of all numbers coprime to n as follows:

$$C_n = \{k \in \mathbb{N} : k \leq n, \gcd(k, n) = 1\}$$

The nth cyclotomic polynomial is the polynomial denoted by Φ_n and defined by:

$$\Phi_n(x) = \prod_{k \in C_n} \left(x - e^{\frac{2\pi i k}{n}}\right)$$

It is instructive to see some examples of cyclotomic polynomials before continuing.
$n = 1$ yields $C_1 = \{1\}$, hence $\Phi_1(x) = x - 1$.
$n = 2$ yields $C_2 = \{1\}$, hence $\Phi_2(x) = x + 1$.
$n = 3$ yields $C_3 = \{1, 2\}$, hence:

$$\Phi_3(x) = \left(x - e^{\frac{2\pi i}{3}}\right)\left(x - e^{\frac{4\pi i}{3}}\right) = x^2 - \left(e^{\frac{2\pi i}{3}} + e^{-\frac{2\pi i}{3}}\right)x + 1 = x^2 + x + 1$$

$n = 4$ yields $C_4 = \{1, 3\}$, hence:

$$\Phi_4(x) = \left(x - e^{\frac{\pi i}{2}}\right)\left(x - e^{\frac{3\pi i}{2}}\right) = (x - i)(x + i) = x^2 + 1$$

$n = 6$ yields $C_6 = \{1, 5\}$, hence:

$$\Phi_6(x) = \left(x - e^{\frac{\pi i}{3}}\right)\left(x - e^{\frac{5\pi i}{3}}\right) = x^2 - \left(e^{\frac{\pi i}{3}} + e^{\frac{5\pi i}{3}}\right) + 1 = x^2 - x + 1$$

Next, a very helpful identity is proven:

Lemma A.27: Product formula for cyclotomic polynomials
Let $n \in \mathbb{N}$. Then define the set of all divisors of n as follows:

$$D_n = \{k \in \mathbb{N} : k | n\}$$

Then, the following relation holds:

$$x^n - 1 = \prod_{d \in D_n} \Phi_d(x)$$

Proof: Let's take $n \in \mathbb{N}$ at random. Note that $x^n - 1$ has exactly n roots, the complex numbers $e^{\frac{2\pi i k}{n}}$, where k runs from 1 to n. These are called the nth roots of unity. From the fundamental theorem of algebra, it is easily observed that:

$$x^n - 1 = \prod_{k=1}^{n} \left(x - e^{\frac{2\pi i k}{n}}\right)$$

For every k in the above sum, we can write the fraction k/n uniquely in a reduced form k'/n', with $\gcd(k', n') = 1$. For $k = n$, it is clear that $k' = n' = 1$, so this yields a factor $x - e^{2\pi i} = x - 1 = \Phi_1(x)$.

Let's now take such an n' at random. Then all fractions k'/n' that can be obtained are the ones in which k' is coprime to n'. On the other hand, suppose k' is coprime to n' and $1 \leq k' \leq n'$. As n' is a divisor of n, we have that $k = k'n/n'$ is an integer. Furthermore, we have $k \geq n/n' \geq 1$ and $k \leq n'n/n' = n$. Hence, we can get the fraction k'/n' by the process prescribed above, by taking $k = k'n/n'$.

Now, all n''s that can be obtained are the divisors of n. Furthermore, for every n' that we can get, all numbers coprime to n' in the interval 1 to n' are available in the numerator. So, we find that the product on

the right hand side of the equation above is equal to the product of all n'th cyclotomic polynomials. Hence we have proven the relation. $\square$

For example, the polynomial $x^6 - 1$ can be rewritten in the following way:

$$\begin{aligned} x^6 - 1 &= \left(x - e^{\frac{2\pi i}{6}}\right)\left(x - e^{\frac{4\pi i}{6}}\right)\left(x - e^{\frac{6\pi i}{6}}\right)\left(x - e^{\frac{8\pi i}{6}}\right)\left(x - e^{\frac{10\pi i}{6}}\right)\left(x - e^{\frac{12\pi i}{6}}\right) \\ &= \left(x - e^{2\pi i\frac{1}{6}}\right)\left(x - e^{2\pi i\frac{1}{3}}\right)\left(x - e^{2\pi i\frac{1}{2}}\right)\left(x - e^{2\pi i\frac{2}{3}}\right)\left(x - e^{2\pi i\frac{5}{6}}\right)(x-1) \\ &= (x-1)\cdot\left(x - e^{2\pi i\frac{1}{2}}\right)\cdot\left(x - e^{2\pi i\frac{1}{3}}\right)\left(x - e^{2\pi i\frac{2}{3}}\right)\cdot\left(x - e^{2\pi i\frac{1}{6}}\right)\left(x - e^{2\pi i\frac{5}{6}}\right) \\ &= \Phi_1(x)\cdot\Phi_2(x)\cdot\Phi_3(x)\cdot\Phi_6(x) \end{aligned}$$

So, once one writes $x^n - 1$ in product form, all that's left to do is change the order of the factors, to obtain the product of cyclotomic polynomials.

A.2.2 Integer coefficients

The product formula of cyclotomic polynomials has some surprising consequences. One of those is that all cyclotomic polynomials have integer coefficients.

Theorem A.28: Cyclotomic polynomials have integer coefficients and are monic
Let $n \in \mathbb{N}$. Then Φ_n is a monic element of $\mathbb{Z}[x]$

Proof: We give a proof by induction. It is easily seen that $\Phi_1(x) = x - 1 \in \mathbb{Z}[x]$ and this polynomial is monic. This provides the basis for induction.

Now take $n \in \mathbb{N}$ at random, and suppose that all cyclotomic polynomials $\Phi_1(x), \ldots, \Phi_{n-1}(x)$ are monic elements of $\mathbb{Z}[x]$. Then we know from the previous lemma:

$$x^n - 1 = \prod_{d\in D_n} \Phi_d(x) = \prod_{d\in D_n\setminus\{n\}} \Phi_d(x)\cdot\Phi_n(x)$$

Let's define $f(x) = \prod_{d\in D_n\setminus\{n\}} \Phi_d(x)$. As $\mathbb{Z}[x]$ is a ring, we have $f \in \mathbb{Z}[x]$. Moreover, f is a product of monic polynomials, so f is monic too. Also, $x^n - 1 = f(x)\cdot\Phi_n(x)$.

Now, we know from the very definition of $\Phi_n(x)$ that it is an element of $\mathbb{C}[x]$. $f(x)$ and $x^n - 1$ are also elements of $\mathbb{C}[x]$. Moreover, $\mathbb{C}$ is a field, so when we apply Euclidean division, we find that if $g, h \in \mathbb{C}[x]$ and $x^n - 1 = f(x)g(x) + h(x)$, then $g(x) = \Phi_n(x)$ and $h(x) = 0$.

We also know that $x^n - 1$ and $f(x)$ are both members of $\mathbb{Q}[x]$. As $\mathbb{Q}$ is a field as well, we can again apply the Euclidean division algorithm to obtain a unique $g, h \in \mathbb{Q}[x]$ such that $x^n - 1 = f(x)g(x) + h(x)$. But then also $g, h \in \mathbb{C}[x]$, so $g(x) = \Phi_n(x)$. Hence, $\Phi_n \in \mathbb{Q}[x]$.

So, now we have $x^n - 1 = f(x)\Phi_n(x)$ with $f, \Phi_n \in \mathbb{Q}[x]$ monic polynomials. Furthermore $x^n - 1 \in \mathbb{Z}[x]$ is monic as well. According to the factorization of monic polynomials over $\mathbb{Z}$ lemma, we now have $\Phi_n \in \mathbb{Z}[x]$ and Φ_n is monic. Therefore, by induction, we have proven that for every $n \in \mathbb{N}$, Φ_n is a monic element of $\mathbb{Z}[x]$. $\square$

A.2.3 Irreducibility over $\mathbb{Q}$

The goal of this subsection is to prove that all cyclotomic polynomials are irreducible over $\mathbb{Q}$. The proof is quite complicated and requires some knowledge from introductory algebra courses. In addition, some properties of binomial coefficients are required. Therefore, the definition of these coefficients is given first.

Definition A.29: Binomial coefficients
For $n, k \in \mathbb{N}_0$ with $k \leq n$, the binomial coefficient n choose k is given by the following equation:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

For any values of $n \in \mathbb{N}_0$ and $k \in \mathbb{Z}$ that do not meet these requirements, the convention that $\binom{n}{k} = 0$ is used.

For example, the binomial coefficient 6 choose 4 is calculated by $\binom{6}{4} = \frac{6!}{4!2!} = \frac{720}{24 \cdot 2} = 15$.

In order to prove some properties of binomial coefficients, the following recursion formula for binomial coefficients is frequently used.

Lemma A.30: Recursion formula for binomial coefficients
Let $n \in \mathbb{N}_0$ and $k \in \mathbb{Z}$. Then the following recursion relation holds:

$$\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$$

Proof: Take $n \in \mathbb{N}_0$ and $k \in \mathbb{Z}$ random. Then we differentiate between four cases.

First of all, suppose $k < -1$ or $k > n$. Then $\binom{n}{k} = 0$, $\binom{n}{k+1} = 0$ and $\binom{n+1}{k+1} = 0$, so the relation holds trivially.

Secondly, suppose $k = -1$. Then we have $\binom{n}{k} = 0$, so we need to prove that $\binom{n+1}{0} = \binom{n}{0}$ holds. Rewriting this in factorials and using $0! = 1$, we must prove $\frac{(n+1)!}{(n+1)!} = \frac{n!}{n!}$. This is equivalent to saying $1 = 1$. Hence the relation holds.

Thirdly, if we have $k = n$, we prove the validity of the relation in a similar manner. We have $\binom{n}{k+1} = 0$, so we must prove $\binom{n+1}{n+1} = \binom{n}{n}$. Again rewriting in factorials yields that proving $\frac{(n+1)!}{(n+1)!} = \frac{n!}{n!}$ is sufficient. But this yields the trivial equality $1 = 1$, hence the relation also holds in this case.

So, we are left with the case in which $0 \leq k < n$. We can now rewrite the right hand side into the left hand side, as follows:

$$\begin{aligned}
\binom{n}{k} + \binom{n}{k+1} &= \frac{n!}{k!(n-k)!} + \frac{n!}{(k+1)!(n-k-1)!} \\
&= \frac{n!(k+1)}{(k+1)!(n-k)!} + \frac{n!(n-k)}{(k+1)!(n-k)!} = \frac{n!(n+1)}{(k+1)!(n-k)!} \\
&= \frac{(n+1)!}{(k+1)!(n-k)!} = \binom{n+1}{k+1}
\end{aligned}$$

Hence in all four cases, the relation holds. □

This recursion relation is what lies at the basis of Pascal's triangle, where the sum of two adjacent tiles determines the value of the tile directly below them. Furthermore, this recursion relation can be used to prove a well-known very fundamental fact about binomial coefficients.

Corollary A.31: Binomial coefficients are integers
All binomial coefficients are members of $\mathbb{Z}$.

Proof: A proof is given by induction. We find that $\binom{0}{0} = \frac{0!}{0! \cdot 0!} = 1 \in \mathbb{Z}$. Furthermore, $\binom{0}{k}$ with $k \neq 0$ is $0 \in \mathbb{Z}$ by definition. This provides the basis for induction.

Now suppose all binomial coefficients with upper number strictly smaller than $n \in \mathbb{N}$ are elements of $\mathbb{Z}$. Then we can take $k \in \mathbb{Z}$ at random and apply the recursion relation from the previous lemma. We know

that $\binom{n-1}{k-1}$ and $\binom{n-1}{k}$ are members of $\mathbb{Z}$, hence so is their sum, $\binom{n}{k}$. Hence we have proven that all binomial coefficients are integer by induction. $\square$

Next, a rather more specific property of binomial coefficients is provided.

Theorem A.32: Divisibility of binomial coefficients by prime factors
Suppose p is a prime number and $1 \leq k \leq p-1$. Then p is a divisor of $\binom{p}{k}$.

Proof: We can rewrite $\binom{p}{k}$ in the following way:

$$\binom{p}{k} = \frac{p!}{k!(p-k)!} = p \cdot \frac{(p-1)!}{k!(p-k)!}$$

Now this last fraction can be written as a fraction $\frac{n}{d}$ with $n, d \in \mathbb{Z}$ such that $\gcd(n,d) = 1$. Then, d is a divisor of $k!(p-k)!$. If we look at the prime factorization of $k!(p-k)!$, though, we do not have p as a factor, because all numbers that are being multiplied are smaller than p. The prime factorization of d is equal to the one of $k!(p-k)!$ with fewer factors, so p is also not a factor in the prime factorization of d. Hence, $\gcd(p,d) = 1$. Combining with $\gcd(n,d) = 1$ and using Euclid's lemma for integers, we find that $\gcd(pn,d) = 1$. But we know from the previous corollary that $\frac{pn}{d} = \binom{p}{k} \in \mathbb{Z}$. So, we find that $d = 1$. Hence $\binom{p}{k} = pn$, so $\binom{p}{k}$ is divisible by p. $\square$

For example, 5 is prime, $\binom{5}{2} = \frac{5!}{2!3!} = \frac{120}{2 \cdot 6} = 10$ and 10 is indeed divisible by 5.

Now, this insight into the binomial coefficients is used in the following theorem.

Theorem A.33: Power of polynomials over finite fields
Suppose p is a prime number and $f \in \mathbb{F}_p[x]$. Then $f(x)^p \equiv f(x^p) \mod p$.

Proof: This theorem is proven using mathematical induction to the degree of f. Suppose that f is constant. Then $f(x) = a \in \{0, 1, \ldots, p-1\}$. According to Fermat's little theorem, we have:

$$f(x)^p = a^p \equiv a = f(x^p) \mod p$$

So, for $\deg(f) \in \{0, -1\}$, the assertion holds.

Now suppose that the assertion holds for all polynomials up to degree $n-1$. A polynomial $f \in \mathbb{F}_p[x]$ with degree n can be written as $f(x) = ax^n + g(x)$, with $g \in \mathbb{F}_p[x]$ and $g(x)^p \equiv g(x^p) \mod p$. Raising f to the power of p yields a sum with binomial coefficients, according to the binomium of Newton. Of this sum, only two terms remain, as the others vanish because the binomial coefficients are divisible by p, according to the previous theorem. Thus:

$$\begin{aligned} f(x)^p &= (ax^n + g(x))^p = \sum_{k=0}^{p} \binom{p}{k} a^k x^k g(x)^{p-k} \equiv a^p x^p + g(x)^p \\ &\equiv ax^p + g(x^p) = f(x^p) \mod p \end{aligned}$$

Thus, by mathematical induction, we have proven that for all functions $f \in \mathbb{F}_p[x]$, the relation $f(x)^p \equiv f(x^p) \mod p$ holds. $\square$

For example, 3 is prime and $(x+1)^3 = x^3 + 3x^2 + 3x + 1 \equiv x^3 + 1 \mod 3$.

As a final ingredient for proving that all cyclotomic polynomials are irreducible over $\mathbb{Q}$, we prove the following lemma.

Lemma A.34:
Let $n \in \mathbb{N}$ and let $\zeta \in \mathbb{C}$ be one of the roots of the nth cyclotomic polynomial. Let $f \in \mathbb{Q}[x]$ be the monic minimal polynomial of ζ over $\mathbb{Q}$. Let p be a prime that does not divide n. Then ζ^p is a root of f.

Proof: Note that there is only one function that is a monic minimal polynomial of ζ, by consequence of theorem A.14.

First of all, notice that ζ is a root of Φ_n. This polynomial is a monic element of $\mathbb{Q}[x]$ and $\mathbb{Q}$ is a field, so by the unique factorization theorem, there exists a unique set of monic irreducible polynomials over $\mathbb{Q}$ that when multiplied yield Φ_n. At least one of these polynomials, say $q \in \mathbb{Q}[x]$ has ζ as a root. As irreducibility and minimality are equivalent, we see that q must be a minimal polynomial of ζ over $\mathbb{Q}$. But q is also monic. As monic minimal polynomials are unique, we have $f = q$. Hence we find that f is one of the monic irreducible polynomials of the unique factorization of Φ_n. Hence, f is a divisor of Φ_n.

So, we have found that there exists a $g \in \mathbb{Q}[x]$ such that $\Phi_n = fg$. But we know that $\Phi_n \in \mathbb{Z}[x]$ and f is monic, so $g \in \mathbb{Z}[x]$ and g is monic as well.

Recall that p is a prime number. Since ζ is a root of Φ_n, we can write $\zeta = e^{\frac{2\pi ik}{n}}$ with $\gcd(k, n) = 1$. As p is prime and does not divide n, we also have $\gcd(p, n) = 1$. According to Euclid's lemma for integers, we now find that $\gcd(pk, n) = 1$. So, for any $l \in \mathbb{Z}$, we have $\gcd(pk + ln, n) = 1$. Now, we have for any $l \in \mathbb{Z}$:

$$e^{\frac{2\pi i(pk+ln)}{n}} = e^{\frac{2\pi ipk}{n}+2\pi il} = e^{\frac{2\pi ipk}{n}} \left(e^{2\pi i}\right)^l = \left(e^{\frac{2\pi ik}{n}}\right)^p \cdot 1^l = \zeta^p$$

Now, according to the Euclidean division algorithm for integers, we can choose l such that $0 < pk + ln \leq n$. Then we find that ζ^p is also a root of Φ_n.

Suppose that ζ^p is not a root of f. We are aiming to find a contradiction. As $\Phi_n = fg$, it must be a root of g. So, we find that ζ^p is a root of g. Hence, ζ is a root of $G(x) = g(x^p)$. In a similar manner as above, we now find that f is a divisor of G. Hence, there exists an $h \in \mathbb{Q}[x]$ such that $g(x^p) = f(x)h(x)$. As $f, g \in \mathbb{Z}[x]$ are monic, we find that $h \in \mathbb{Z}[x]$ is monic too (Gauss's lemma's corollary).

Furthermore, from the product formula for cyclotomic polynomials, we observe that $\Phi_n(x)$ is a divisor of $x^n - 1$. Hence, there exists a polynomial $d \in \mathbb{Q}[x]$ such that $x^n - 1 = \Phi_n(x)d(x)$. As $x^n - 1$ and Φ_n are both monic members of $\mathbb{Z}[x]$, we find that d is a monic member of $\mathbb{Z}[x]$ as well.

So far, thus, we have the following equations with all functions monic polynomials over $\mathbb{Z}$:

$$x^n - 1 = \Phi_n(x)d(x) = f(x)g(x)d(x) \qquad \text{and} \qquad g(x^p) = f(x)h(x)$$

Now, we apply the modulo p operation on both equations. The resulting functions over $\mathbb{F}_p[x]$ are denoted by the same letters. According to the previous lemma, we now find $g(x)^p \equiv g(x^p) = f(x)h(x) \mod p$.

As $\mathbb{F}_p$ is a field and g is monic, g has a unique factorization of monic irreducible polynomials over $\mathbb{F}_p[x]$. Suppose that these monic irreducible polynomials are $k_1, k_2, \ldots, k_m \in \mathbb{F}_p[x]$. Then $g = \prod_{i=1}^m k_i$, so $g^p = \prod_{i=1}^m k_i^p$. But g^p is also a polynomial in $\mathbb{F}_p[x]$, hence it also has a unique factorization of monic irreducible polynomials. We observe that this factorization is given by the same irreducible factors $k_1, \ldots, k_m$, but with higher multiplicity.

f is irreducible over $\mathbb{Q}$, but it is not necessarily irreducible over $\mathbb{F}_p$. But as $\mathbb{F}_p$ is a field, f has a unique factorization. Let $k \in \mathbb{F}_p[x]$ be a monic irreducible factor of f. Then k is also a monic irreducible factor of g^p. Hence by the previous paragraph, it must also be an irreducible factor of g.

Recall that we have $x^n - 1 = f(x)g(x)d(x)$. As k is an irreducible factor of f and g, we have that k is an irreducible factor of $x^n - 1$ with multiplicity of at least 2. But k is non-constant, so, according to the fundamental theorem of algebra, it has at least one root, $\alpha \in \mathbb{C}$. So, this is also a root of $x^n - 1$. As the multiplicity is greater than 1, the derivative of $x^n - 1$ will have this root as well. But the derivative is nx^{n-1} which has only the trivial root $x = 0$, which is clearly not a root of $x^n - 1$. Hence, we have found a contradiction. So, ζ^p is a root of f. $\square$

This lemma enables us to reach the main goal of this subsection.

Theorem A.35: Irreducibility of cyclotomic polynomials over $\mathbb{Q}$
Let $n \in \mathbb{N}$. Then Φ_n is irreducible over $\mathbb{Q}$.

Proof: Define $\zeta = e^{\frac{2\pi i}{n}}$. Then ζ is a root of Φ_n. We define f as the monic minimal polynomial of ζ over $\mathbb{Q}$. Now take η another root of Φ_n. Then we can write $\eta = e^{\frac{2\pi ik}{n}}$ with $1 \leq k \leq n$ and $\gcd(k, n) = 1$. Thus we have the following relation:

$$\eta = \left(e^{\frac{2\pi i}{n}}\right)^k = \zeta^k$$

Now consider the prime factors of k: $p_1, \ldots, p_m$ prime numbers such that $k = \prod_{i=1}^m p_i$. Then we can write:

$$\eta = \zeta^k = \zeta^{\prod_{i=1}^m p_i} = (((\zeta^{p_1})^{p_2})^{\cdots})^{p_m}$$

As $\gcd(k, n) = 1$, we have $\gcd\left(\prod_{i=1}^m p_i, n\right) = 1$. Applying Euclid's lemma for integers now yields that all for all i, $\gcd(p_i, n) = 1$. Hence, we can apply the previous lemma repeatedly, to obtain that η is also a root of f.

So, now we have found that every root of Φ_n is also a root of f. As Φ_n does not contain any roots of multiplicity greater than 1, we see that the factorization of Φ_n into irreducible polynomials over $\mathbb{Q}$ can only have one factor: f. As both are monic, we now find that $\Phi_n = f$. So, Φ_n is irreducible over $\mathbb{Q}$. □

Finally, the following corollary will prove helpful in proving that λ is irrational.

Corollary A.36:
Let $q \in \mathbb{Q}$. Then the monic minimal polynomial of $\alpha = e^{2\pi iq}$ over $\mathbb{Q}$ is cyclotomic.

Proof: Take $q \in \mathbb{Q}$ random. Then we write $q = \frac{m}{n}$ with $\gcd(m, n) = 1$. Substitution in α yields $\alpha = e^{\frac{2\pi im}{n}}$. Now α is a root of Φ_n. Furthermore, Φ_n is irreducible over $\mathbb{Q}$. This implies that Φ_n is a minimal polynomial of α. As Φ_n is monic and monic minimal polynomials are unique, we have that the monic minimal polynomial of α is Φ_n. Hence it is cyclotomic. □

A.3 The λ-polynomial

In this subsection, finally, it will be proven that λ defined by the λ-equation is irrational. To do so, the results from the previous sections will be used. This section will be less abstract than the previous two, as the focus will be on one specific polynomial, instead of a general collection of polynomials obeying certain characteristics.

The polynomial that will be considered in this section is referred to as the *λ-polynomial*, and is defined as follows:

$$p(x) = x^4 + x^3 + \frac{1}{4}x^2 + x + 1$$

First of all, it will be shown that one of the roots of this equation is in fact $e^{2\pi i\lambda}$. Then it will be shown that this polynomial is irreducible. From this, it will eventually follow that $\lambda \notin \mathbb{Q}$.

A.3.1 $e^{2\pi i\lambda}$ is a root

In this subsection, it will be proven that $\alpha = e^{2\pi i\lambda}$ is a root of the λ-polynomial.

First of all, an expression for $\cos^2\left(\frac{\pi}{8}\right)$ is found. To do so, a double-angle formula is used: $\cos(2x) = 2\cos^2(x) - 1$. Rewriting and substituting $\frac{\pi}{8}$ for x yields:

$$\cos^2\left(\frac{\pi}{8}\right) = \frac{1}{2}\left(1 + \cos\left(\frac{\pi}{4}\right)\right) = \frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right)$$

Evaluating p at $x = \alpha$ yields:

$$p(\alpha) = \frac{\alpha^2}{4}\left(4\alpha^2 + 4\alpha + 1 + 4\alpha^{-1} + 4\alpha^{-2}\right)$$

It is sufficient to show that the expression in the parentheses is 0, hence it is sufficient to show that $4p(\alpha)/\alpha^2 = 0$. Plugging in $\alpha = e^{2\pi i\lambda}$ in the expression in the parentheses, and using the definitions of the cosine and sine, the following equation is obtained:

$$\begin{aligned}\frac{4p(\alpha)}{\alpha^2} &= 4\alpha^2 + 4\alpha + 1 + 4\alpha^{-1} + 4\alpha^{-2} = 4e^{4\pi i\lambda} + 4e^{2\pi i\lambda} + 1 + 4e^{-2\pi i\lambda} + 4e^{-4\pi i\lambda} \\ &= 8\cos(4\pi\lambda) + 8\cos(2\pi\lambda) + 1\end{aligned}$$

This can be further rewritten using the same double angle formula again: $\cos(2x) = 2\cos^2(x) - 1$. This yields:

$$\begin{aligned}\frac{4p(\alpha)}{\alpha^2} &= 8(2\cos^2(2\pi\lambda) - 1) + 8(2\cos^2(\pi\lambda) - 1) + 1 \\ &= 16\cos^2(2\pi\lambda) + 16\cos^2(\pi\lambda) - 15 \\ &= 16(2\cos^2(\pi\lambda) - 1)^2 + 16\cos^2(\pi\lambda) - 15 \\ &= 16(4\cos^4(\pi\lambda) - 4\cos^2(\pi\lambda) + 1) + 16\cos^2(\pi\lambda) - 15 \\ &= 64\cos^4(\pi\lambda) - 48\cos^2(\pi\lambda) + 1 \\ &= 64\left(\cos^2\left(\frac{\pi}{8}\right)\right)^4 - 48\left(\cos^2\left(\frac{\pi}{8}\right)\right)^2 + 1\end{aligned}$$

This last step was obtained by invoking the λ-equation. Using the expression found for $\cos^2(\pi/8)$ that was found before, the right hand side of the equation above can be further rewritten. Substituting $\cos^2(\pi/8)$ for $\frac{1}{2}(1 + 2^{-\frac{1}{2}})$ in the expression above leads to:

$$\begin{aligned}\frac{4p(\alpha)}{\alpha^2} &= 64\left(\frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right)\right)^4 - 48\left(\frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right)\right)^2 + 1 \\ &= 4\left(1 + \frac{1}{\sqrt{2}}\right)^4 - 12\left(1 + \frac{1}{\sqrt{2}}\right)^2 + 1 \\ &= 4\left(\frac{3}{2} + \sqrt{2}\right)^2 - 12\left(\frac{3}{2} + \sqrt{2}\right) + 1 \\ &= 4\left(\frac{17}{4} + 3\sqrt{2}\right) - 12\left(\frac{3}{2} + \sqrt{2}\right) + 1 \\ &= 17 + 12\sqrt{2} - 18 - 12\sqrt{2} + 1 = 0\end{aligned}$$

Hence, indeed $4p(\alpha)/\alpha^2 = 0$, so $p(\alpha) = 0$. So, it was shown that $e^{2\pi i\lambda}$ is indeed a root of $x^4 + x^3 + \frac{1}{4}x^2 + x + 1$.

A.3.2 Irreducibility over $\mathbb{Q}$

This section will be devoted to proving that the λ-polynomial is irreducbile over $\mathbb{Q}$. The proof will use some results from earlier sections.

Assume that the λ-polynomial is reducible over $\mathbb{Q}$. We are going to find a contradiction. So, $x^4 + x^3 + \frac{1}{4}x^2 + x + 1$ is reducible over $\mathbb{Q}$, hence so is $4x^4 + 4x^3 + x^2 + 4x + 4$. This polynomial is primitive, and hence this polynomial is also reducible over $\mathbb{Z}$ (Gauss's lemma). So, there exist non-constant $f, g \in \mathbb{Z}[x]$ such that $f(x)g(x) = 4x^4 + 4x^3 + x^2 + 4x + 4$.

If we now apply a modulo 3 operation on both sides, and denote the resulting functions on the left hand side with the same letters, f and g, we obtain:

$$f(x)g(x) \equiv x^4 + x^3 + x^2 + x + 1 \mod 3$$

We are now going to show that such a choice for non-constant $f, g \in \mathbb{F}_3[x]$ does not exist. As $\mathbb{F}_3$ is a field, all non-zero elements from $\mathbb{F}_3$ have a multiplicative inverse, so we only have to look for monic divisors of $x^4 + x^3 + x^2 + x + 1$. We notice that at least one of the factors must have degree 1 or 2. Now, the following expressions all equal $x^4 + x^3 + x^2 + x + 1$ modulo 3:

$$\begin{array}{cc} x(x^3 + x^2 + x + 1) + 1 & x^2(x^2 + x + 1) + x + 1 \\ (x + 1)(x^3 + x) + 1 & (x^2 + 1)(x^2 + x) + 1 \\ (x + 2)(x^3 + 2x^2 + 1) + 2 & (x^2 + 2)(x^2 + x + 2) + 2x \\ & (x^2 + 2x)(x^2 + 2x) + x + 1 \\ & (x^2 + 2x + 1)(x^2 + 2x + 2) + x + 2 \end{array}$$

In the left column, we see that division by a polynomial of first degree always yields a non-zero remainder. In the right column, a similar observation regarding polynomials of 2nd degree can be made. According to Euler's division algorithm, these remainders are unique, hence there are no polynomials $f, g \in \mathbb{F}_3[x]$ such that $f(x)g(x) \equiv x^4 + x^3 + x^2 + x + 1 \mod 3$. So, we have found a contradiction, and therefore the λ-polynomial must be irreducible over $\mathbb{Q}$.

A.3.3 Completion of the proof

At this point, all ingredients for the proof of $\lambda \notin \mathbb{Q}$ are available. This last section will cover collecting all the pieces that are needed, and finally proving the statement this appendix is all about.

The λ-polynomial is an irreducible polynomial over $\mathbb{Q}$. Also, $\alpha = e^{2\pi i\lambda}$ is a root of the λ-polynomial. So, the λ-polynomial is a minimal polynomial of α over $\mathbb{Q}$. Moreover, the λ-polynomial is monic, hence it is the monic minimal polynomial of α over $\mathbb{Q}$. Furthermore, the λ-polynomial is not cyclotomic, as not all of its coefficients are elements from $\mathbb{Z}$.

Suppose now that $\lambda \in \mathbb{Q}$. Then according to the last theorem of section A.2, the monic minimal polynomial of $e^{2\pi i\lambda}$ over $\mathbb{Q}$ is cyclotomic. But the monic minimal polynomial of $e^{2\pi i\lambda}$ is the λ-polynomial, as was shown in this section, and this polynomial is not cyclotomic. Hence, there is a contradiction, so $\lambda \notin \mathbb{Q}$.

B Simulation of Shor's algorithm on a classical computer

Even though Shor's algorithm is an algorithm that is supposed to be run on a quantum computer, it can be simulated on a classical computer. The source code below does exactly that. On linux based systems, it is run by the following command:

```
./Shor <Number to factor> <Accuracy parameter>
```

In the summary of Shor's algorithm at the end of chapter 5, the number to factor refers to N, and the accuracy parameter refers to ε.

The code can also be found on GitHub, via the following link:

```
https://github.com/arriopolis/Shor.git
```

The code is compiled using a command line instruction similar to:

```
gcc -o Shor Shor.c -lm
```

If the compilation process fails, or runtime errors occur, then one can always contact me for more information.

```
1  //Standard includes
2  #include "stdio.h"
3  #include "stdlib.h"
4  #include "math.h"
5  #include "sys/time.h"
6
7  //Enable the use of booleans
8  #define true 1
9  #define false 0
10 typedef char bool;
11
12 /* Greatest Common Divisor */
13 unsigned int gcd(unsigned int a, unsigned int b)
14 {
15     //Euclidean algorithm
16     unsigned int c;
17     while (a != 0)
18     {
19         c = a;
20         a = b%a;
21         b = c;
22     }
23     return b;
24 }
25
26 /* Calculcate the remainder of a power with positive offset */
27 unsigned int remmod(unsigned int base, unsigned int exp, unsigned int offset,
       unsigned int mod)
28 {
29     //Modular exponentiotion
30     unsigned int result = 1;
31     base %= mod;
32     while (exp != 0)
33     {
```

```
34          if (exp & 1)
35          {
36              result = (result * base) % mod;
37          }
38          exp >>= 1;
39          base = (base * base) % mod;
40      }
41      result = (result + offset) % mod;
42      return result;
43  }
44
45  /* Complex variables */
46  typedef struct complex {
47      double Re;
48      double Im;
49  } COMPLEX;
50
51  /* Inverse Fast Fourier Transform */
52  void ifftstep(COMPLEX * v, unsigned int N, unsigned int i, COMPLEX * tmp,
        COMPLEX * rootsofunity)
53  {
54      COMPLEX * ve, * vo;
55      COMPLEX z,w;
56      if (N > 1)
57      {
58          ve = tmp;
59          vo = &tmp[N/2];
60          for (unsigned int k = 0; k < N/2; k++)
61          {
62              ve[k] = v[2*k];
63              vo[k] = v[2*k+1];
64          }
65          ifftstep(ve,N/2,i<<1,v,rootsofunity);
66          ifftstep(vo,N/2,i<<1,v,rootsofunity);
67          for (unsigned int k = 0; k < N/2; k++)
68          {
69              w.Re = rootsofunity[k*i].Re;
70              w.Im = rootsofunity[k*i].Im;
71              z.Re = w.Re * vo[k].Re - w.Im * vo[k].Im;
72              z.Im = w.Re * vo[k].Im + w.Im * vo[k].Re;
73              v[k].Re = ve[k].Re + z.Re;
74              v[k].Im = ve[k].Im + z.Im;
75              v[k+N/2].Re = ve[k].Re - z.Re;
76              v[k+N/2].Im = ve[k].Im - z.Im;
77          }
78      }
79      return;
80  }
81
82  void ifft(COMPLEX * v, unsigned int n)
83  {
```

```
84      unsigned int N = (1 << n);
85
86      //Set up a LUT for the cosines and sines
87      COMPLEX * rootsofunity = malloc(N * sizeof(COMPLEX));
88      for (unsigned int i = 0; i < N; i++)
89      {
90          rootsofunity[i].Re = cos(2.0*M_PI*i/(double)N);
91          rootsofunity[i].Im = sin(2.0*M_PI*i/(double)N);
92      }
93
94      COMPLEX * tmp = malloc(N * sizeof(COMPLEX));
95      ifftstep(v, N, 1, tmp, rootsofunity);
96
97      free(rootsofunity);
98      free(tmp);
99      return;
100 }
101
102 /* The order finding subroutine */
103 unsigned int findorder(unsigned int N, double epsilon, unsigned int x)
104 {
105     struct timeval time;
106     gettimeofday(&time, NULL);
107     srand((unsigned) time.tv_usec * x);
108
109     //Calculate the number of qubits needed
110     unsigned int L = ceil(log2(N)-1e-8);
111     unsigned int t = 2*L + 1 + ceil(log2(2.0+1.0/(2.0*epsilon))-1e-8);
112
113     //Allocate space for the calculation and fill with ones
114     unsigned int * psi = malloc((1 << t) * sizeof(unsigned int));
115     for (unsigned int i = 0; i < (1 << t); i++)
116     {
117         psi[i] = 1;
118     }
119
120     //Allocate space for the multiplication matrix and fill it
121     unsigned int * multorder = malloc((1 << L) * t * sizeof(unsigned int));
122     for (unsigned int i = 0; i < (1 << L); i++)
123     {
124         if (i >= N) {multorder[i] = i;}
125         else {multorder[i] = (i * x) % N;}
126     }
127
128     for (unsigned int i = 1; i < t; i++)
129     {
130         for (unsigned int j = 0; j < (1<<L); j++)
131         {
132             multorder[i*(1<<L)+j] =
                    multorder[(i-1)*(1<<L)+multorder[(i-1)*(1<<L)+j]];
133         }
```

```
134     }
135
136     //Apply the multiplications
137     unsigned int exp,ctr;
138     for (unsigned int i = 0; i < (1 << t); i++)
139     {
140         exp = i;
141         ctr = 0;
142         while (exp)
143         {
144             if (exp & 1)
145             {
146                 psi[i] = multorder[ctr*(1<<L)+psi[i]];
147             }
148             exp >>= 1;
149             ctr++;
150         }
151     }
152
153     //Measure the second register and store the resulting first register in a
        complex vector
154     unsigned int b = psi[rand() % (1 << t)];
155
156     unsigned int sum = 0;
157     for (unsigned int i = 0; i < (1 << t); i++) {if (psi[i] == b) {sum++;}}
158     double normalization = 1.0/sqrt(sum) / sqrt((double)(1 << t));
159
160     COMPLEX * phi = malloc((1 << t) * sizeof(COMPLEX));
161     for (unsigned int i = 0; i < (1 << t); i++)
162     {
163         if (psi[i] == b) {phi[i].Re = normalization;}
164         else {phi[i].Re = 0;}
165         phi[i].Im = 0.0;
166     }
167
168     //Apply the inverse Fourier transform
169     ifft(phi,t);
170
171     //Get the probabilities for measurement
172     double p = (double)rand() / (double)RAND_MAX;
173     p *= p;
174     double cumsum = 0;
175     unsigned int k;
176     for (k = 0; k < (1 << t); k++)
177     {
178         cumsum += phi[k].Re * phi[k].Re + phi[k].Im * phi[k].Im;
179         if (cumsum >= p) {break;}
180     }
181
182     //Apply the continuous fraction algorithm
183     unsigned int r = 0;
```

```
184     if (k != 0)
185     {
186         unsigned int n, d, napprox, dapprox, tmp, ctr;
187         int i;
188         unsigned int * contfracs = malloc(t * sizeof(unsigned int));
189         n = k;
190         d = (1 << t);
191         ctr = 0;
192         while (n != 0)
193         {
194             contfracs[ctr++] = d/n;
195             tmp = n;
196             n = d%tmp;
197             d = tmp;
198 
199             napprox = 0;
200             dapprox = 1;
201             for (i = ctr - 1; i >= 0; i--)
202             {
203                 tmp = contfracs[i] * dapprox + napprox;
204                 napprox = dapprox;
205                 dapprox = tmp;
206             }
207             if (dapprox >= N) {break;}
208             r = dapprox;
209         }
210 
211         free(contfracs);
212     }
213 
214     //Deallocate all the memory that was used
215     free(psi);
216     free(multorder);
217     free(phi);
218 
219     return r;
220 }
221 
222 /* Shor's algorithm */
223 //This structure contains the results of Shor's algorithm
224 typedef struct Shorstats
225 {
226     unsigned int factor;
227     unsigned int x;
228     unsigned int r;
229     unsigned int errorcode;
230     bool quantum;
231 } SHORSTATS;
232 
233 void Shor(SHORSTATS * results, unsigned int N, double epsilon, bool msg)
234 {
```

```
235     //Set up the random number generator seed
236     struct timeval t;
237     gettimeofday(&t, NULL);
238     srand((unsigned) t.tv_usec);
239
240     //Set up the default return values
241     results->factor = 0;
242     results->x = 0;
243     results->r = 0;
244     results->errorcode = 0;
245     results->quantum = false;
246
247     //Check if the number to be factored and epsilon are in the proper range
248     if (N <= 2) {results->errorcode = 1; if (msg) {printf("The number is too
            small to be factored.\n");} return;}
249     if (epsilon <= 0 || epsilon >= 1) {results->errorcode = 1; if (msg)
            {printf("The faulttolerance is invalid.\n");} return;}
250
251     //Calculate the effective fault-tolerance
252     double faulttolerance = 1.0/(2.0*(pow(2.0,ceil(log2(2.0 +
            1.0/(2.0*epsilon)) - 1e-8)) - 2.0));
253
254     //Display the welcome message
255     if (msg)
256     {
257         printf("Shor's algorithm initiated...\n");
258         printf(" - The number to be factored is %d.\n", N);
259         printf(" - The fault-tolerance towards the order finding subroutine
                is %f.\n",faulttolerance);
260     }
261
262     //Check if the number to be factored is even
263     if (N % 2 == 0) {results->factor = 2; if (msg) {printf("The number is
            even, so this case is trivial.\nA dividing factor of %d is 2.\n",
            N);} return;}
264
265     //Check if N can be written as a^b with a >= 1 and b >= 2
266     double log2N = log2(N);
267     for (unsigned int b = 2; b <= log2N; b++)
268     {
269         double a = pow(2,log2N/b);
270         if ((0.5 - fabs(a - floor(a + 1e-8) - 0.5)) <= 1e-8)
271         {
272             results->factor = floor(a + 1e-8);
273             if (msg) {printf("The number can be written as a power.\nA
                    dividing factor of %d is %d.\n", N, results->factor);}
274             return;
275         }
276     }
277
278     //Randomly choose an x in the range between 2 and N-2.
```

```
279     results->x = rand() % (N - 3) + 2;
280     if (msg) {printf("The program guesses a number to find the order of. This
            number is %d.\n", results->x);}
281
282     //Return gcd(x,N) if it is a non-trivial divisor of N
283     unsigned int temp = gcd(N,results->x);
284     if (temp > 1)
285     {
286         results->factor = temp;
287         if (msg) {printf("The program got lucky and guessed a number that is
                not coprime to N.\nA dividing factor of %d is %d.\n", N,
                results->factor);}
288         return;
289     }
290
291     //Calculate the order of x modulo N
292     results->quantum = true;
293
294     unsigned int r1 = findorder(N,epsilon,results->x);
295     unsigned int r2 = findorder(N,epsilon,results->x);
296     if (r1 == 0 && r2 == 0)
297     {
298         results->errorcode = 2;
299         if (msg) {printf("The order finding subroutine failed.\n");}
300         return;
301     }
302
303     if (r1 == 0) {results->r = r2;}
304     else if (r2 == 0) {results->r = r1;}
305     else {results->r = r1*r2/gcd(r1,r2);}
306
307     //Check if the order is correct
308     unsigned int y = 1;
309     for (unsigned int a = 0; a < results->r; a++)
310     {
311         y = (y * results->x) % N;
312     }
313     if (y != 1) {results->errorcode = 3; if (msg) {printf("The order finding
            subroutine failed.\n");} return;}
314
315     if (msg) {printf("The order is found to be %d.\n",results->r);}
316
317     //Check if the order is even
318     if (results->r % 2 != 0) {results->errorcode = 4; if (msg) {printf("The
            order is odd, so the algorithm fails.\n");} return;}
319
320     //Check if there is any hope in finding a divisor
321     if (remmod(results->x,results->r/2,0,N) == N - 1) {results->errorcode =
            5; if (msg) {printf("x^(r/2) = -1 mod N, so the algorithm
            fails.\n");} return;}
322
```

```
323     //Return one of the factors
324     unsigned int candidate = gcd(remmod(results->x,results->r/2,N-1,N),N);
325     if (candidate != 1) {results->factor = candidate; if (msg) {printf("A
            dividing factor of %d is %d.\n", N, results->factor);} return;}
326     candidate = gcd(remmod(results->x,results->r/2,1,N),N);
327     if (candidate != 1) {results->factor = candidate; if (msg) {printf("A
            dividing factor of %d is %d.\n", N, results->factor);} return;}
328
329     results->errorcode = 6;
330     if (msg) {printf("An unexpected error occured. Exiting.\n");}
331     return;
332 }
333
334 /* Main program starts here */
335 int main(int argc, char * argv[])
336 {
337     //Check the input parameters
338     unsigned int N;
339     double epsilon;
340     if (argc == 1)
341     {
342         N = 15; printf("No number to factor is supplied, so the default value
                of 15 is used.\n");
343         epsilon = 0.2; printf("No accuracy parameter is supplied, so the
                default value of 0.2 is used.\n");
344     }
345     else if (argc == 2) {N = atoi(argv[1]); epsilon = 0.2; printf("No
            accuracy parameter is supplied, so the default value of 0.2 is
            used.\n");}
346     else if (argc == 3) {N = atoi(argv[1]); epsilon = atof(argv[2]);}
347     else {printf("Too many arguments supplied. Expected: ./Shor [<Number to
            factor>] [<Accuracy parameter>].\n"); return 0;}
348
349     //Set up the statistics
350     struct timeval start, stop, starttry, stoptry;
351     gettimeofday(&start,NULL);
352     SHORSTATS * results = malloc(sizeof(SHORSTATS));
353     unsigned int numtries = 0;
354     unsigned int numquantumtries = 0;
355     unsigned int preliminaryresult = 0;
356
357     //Run the algorithm a maximum of 100 times
358     for (unsigned int i = 0; i < 100; i++)
359     {
360         numtries++;
361         gettimeofday(&starttry,NULL);
362         Shor(results,N,epsilon,false);
363         gettimeofday(&stoptry,NULL);
364         if (results->errorcode == 0) {preliminaryresult = results->factor;}
365         if (results->quantum) {numquantumtries++;}
366         if (results->quantum && results->errorcode == 0) {break;}
```

```
367         }
368         gettimeofday(&stop,NULL);
369 
370         //Display the results
371         if (results->quantum && results->errorcode == 0)
372         {
373             //The program succeeded
374             printf("The algorithm succesfully found a factor of %d using the
                    order finding subroutine.\n",N);
375             printf("The factor that was found was %d.\n",results->factor);
376             printf("The number that was guessed was %d.\n",results->x);
377             printf("The order found by the quantum routine was %d.\n",results->r);
378             printf("The number of tries was %d.\n",numtries);
379             printf("The number of times the order finding routine failed was
                    %d.\n",numquantumtries-1);
380             printf("The time elapsed during the succesful try was %f
                    seconds.\n",(stoptry.tv_sec - starttry.tv_sec) +
                    (double)(stoptry.tv_usec - starttry.tv_usec) / 1e6);
381             printf("The total elapsed time was %f seconds.\n",(stop.tv_sec -
                    start.tv_sec) + (double)(stop.tv_usec - start.tv_usec) / 1e6);
382         }
383         else
384         {
385             //The program failed
386             printf("The program did not find a dividing factor of %d using the
                    order finding subroutine after 100 tries.\n",N);
387             if (preliminaryresult == 0)
388             {
389                 printf("Neither did it find a dividing factor using classical
                        methods. Perhaps the number is prime.\n");
390             }
391             else
392             {
393                 printf("However, it did find a dividing factor using classical
                        methods.\n");
394                 printf("The factor that was found was %d.\n",preliminaryresult);
395             }
396         }
397 
398         free(results);
399         return 0;
400 }
```

C Experimental realization of Deutsch's algorithm

In section 3.4, we have seen an instructive example quantum circuit, known as Deutsch's algorithm. The idea was to find if a given function $f : \{0, 1\} \to \{0, 1\}$ is constant.

Suppose we take $f(x) = 1 - x$. Then we find $f(0) = 1$ and $f(1) = 0$, hence f is not constant. Moreover, we find that the effect of the U_f gate, as shown in figure 3.10, is now equal to the effect of the CNOT gate. Hence, the implementation of the quantum circuit that executes Deutsch's algorithm, as shown in figure 3.11, reduces to the circuit shown in figure C.1.

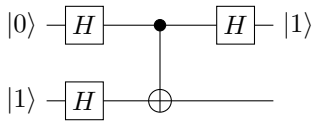


Figure C.1: The implementation of Deutsch's algorithm when we take $f(x) = 1 - x$.

This circuit, shown in figure C.1 was actually executed on the quantum computer built by IBM in New York [1]. In total, the algorithm was run 1024 times, and the results are shown in table C.1.

Outcome	Frequency
0	62
1	962

Table C.1: Results of the experimental realization of Deutsch's algorithm. In the left column, the outcome of the measurement of the first qubit is shown, and in the right column, the frequency of these measurements is displayed.

So, we observe qualitatively that the algorithm works. In 94% of the cases, the outcome was as expected. As it is very hard to keep qubits stable at this point, it seems likely that the 6% of faulty experiments can be attributed to the instability of the process.