Quantum Computing and Quantum Information – a first Overview

Prof. Dr. Ulrich Margull

October 18, 2023

Content

List of Figures

List of Tables

1. Introduction

Welcome to the course "Quantum Information and Computing"!

Quantum mechanics is a very precise concept, which has been proven correct by millions of experiments; most of our technology is build upon this theory, may it be solar cells for energy production, classical CPUs in our laptops and smartphones, LED lights, and many many more.

Nevertheless, the inner workings of quantum mechanics (or better, the inner workings of nature itself on small scale) is quite strange to us humans. We live in a world of 1m scale, and the smallest objects we handle as humans are of 1mm scale. But nature behaves different on different scales: if we throw a ball against a wall, it bounces back, and before and after it moves in a continuous movement. On the scale of quantum mechanics (i.e. single particles like atoms, electrons or photons) things are completely different, and counter intuitive to us: a particle may bounce back or just "tunnel" through the wall, it does not move continuously, it does not even have a defined position in space!

This lecture is not about quantum mechanics, but about quantum information and quantum computing. We will need some understanding of the quantum nature of things, since this is what is used in a quantum computer. But we will keep things simple: as computer scientists, we are mainly interested in 0 and 1, which simplifies things a lot.

So our key questions will be: how can we do computing using quantum behavior? What can be achieved with this technology? How does the hardware looks like, what are current problems, what is expected in future?

At first, we will have to prepare the ground for quantum computing, which means we will repeat complex numbers, vector spaces and matrices. Then, we will learn how about the qubit, which is the fundamental information unit of quantum computing, i.e. the replacement of the classical bit (Zero/One).

In a computer, many bits are combined into larger memory or processing units. This is also done with qubits, but quantum registers are much more than just a set of qubits: here, we will learn about the magic of *quantum entanglement*.

Computing means the processing of algorithms. What algorithms do we have in quantum computing? How do they work? How are they different from classical algorihtms? These questions will be covered in different chapters.

An important part of today computers is communication. How can we communicate with quantum computers? How does a quantum internet looks like? (chapter 7).

And how can we secure the communication against evil eyedroppers? In chapter 8 we will see that quantum encryption provides real benefits over classical encryption.

Real quantum computers are not perfect, they produce *noise* to the quantum states when processing them. How can we handle this noise and still achieve valid, useful results? In chapter 11 we learn how to handle noise in our quantum circuits.

In last chapter 12, we will have a look at todays quantum computers. What hardware exists today that is used for quantum computing or communication? What are its limitations? What can be expected in the futures?

So, lets go! Let us enter the fancy world of quantum computing...

2. Some basic math

2.1. Numbers

What is a number?

In mathematics, we know different types of numbers, like positive integer numbers $\mathbb{N}_0$, positive and negative integer numbers $\mathbb{Z}$, rational numbers $\mathbb{Q}$, or real numbers $\mathbb{R}$. For numbers, we can define an addition and a multiplication; from this, we can construct fields, which are sets of numbers (with addition and multiplication) that have some useful properties.

2.1.1. Field

A *field* is a set F together with two binary operations, typically called addition and multiplication. A binary operation is a mapping $F \otimes F :\rightarrow F$, i.e. a mapping that takes two elements of F and produces another element of F. For a field, the following field axioms must be true:

- Associativity of addition and multiplication: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- Commutativity: $a + b = b + a$ and $a \cdot b = b \cdot a$
- Additive and multiplicative identity. There exist a neutral element for addition (namely 0) and a neutral element for multiplication (namely 1): $a + 0 = a$ and $a \cdot 1 = a$
- Additive inverse: for each $a \in F$, there is a inverse b such that $a + b = 0$ (b typically is written as $-a$)
- Multiplicative inverse: for each $a \in F/\{0\}$ there is a inverse b such that $a \cdot b = 1$ (b typically is written as $\frac{1}{a}$).
- Distributivity of multiplication over addition: $a \cdot (b + c) = a \cdot b + a \cdot c$.

2.1.2. Complex numbers

Using numbers, we can solve algebraic equations, like $x^2-2x+1=0$ (can you spot the solution?). Here, the solution is an integer number, which means it is an element of $\mathbb{Z}$. A more difficult situation would be $x^2-2=0$, since neither an integer number nor a rational number fulfulls this equation. In this case, the real numbers come to help, and the irrational number $\sqrt{2}$ solves above equation. However, some equations cannot be solved with "normal" numbers. Consider, for example, the equation$x^2+1=0$. We all know hat no $x \in \mathbb{R}$ fulfills this equation! In the 16th century, Mathematicians realized that they could define yet a new number, which solves the above equation. A complex number consists of two parts, the "real" part and the "imaginary" part: $c=(a,b)$. The symbol i is introduced, which is defined as the solution to the above equation: $i^2=-1$. Using i, one can write a complex number like $c=a+ib$. Applying the normal calculation rules one can calculate complex numbers quite easily:

- Adding two complex numbers c_1 and c_2 is done by adding the real and imaginary parts separetely: $c_1+c_2=(a_1,b_1)+(a_2,b_2)=(a_1+a_2,b_1+b_2)$. Using the i symbol, we get $c_1+c_2=a_1+ib_1+a_2ib_2=a_1+a_2+i(b_1+b_2)$
- Multiplication of two complex numbers c_1 and c_2 is done by multiplying the numbers $c_1 \cdot c_2=(a_1+ib_1)\cdot(a_2+ib_2)=a_1a_2+i(a_1b_2)+i(b_1a_2)+i^2(b_1b_2)=a_1a_2-b_1b_2+i(b_1a_2+b_2a_1)$
- Since a complex number has two parts, the numbers cannot be ordered in one line (as all the above numbers), but fill a two-dimensional plane. So, there is no ordering between complex number!
- In the plane, a number can also be described by its polar form, where we give the angle (with relation to the positive x axis) and the distance from zero: $c=r\cdot e^{i\phi}$, with the complex exponentional funtion $e^{i\phi}=\cos\phi+i\sin\phi$.
- A complex conjugate is a complex number where the sign of the imaginary part is flipped: $\bar{c}=a-ib$ for $c=a+ib$. The product of a complex number with its own conjugate is always real: $c\cdot\bar{c}\in\mathbb{R}$ for all $c\in\mathbb{C}$. (Prove it!)

2.1.3. Exercise

Here are some exercises for complex numbers.

Let $x=3+5i$ and $y=2-4i$.

1. Calculate the complex conjugates $\bar{x}$ and $\bar{y}$, $a=x+y$, $b=x\cdot y$.
2. Write x and y in polar form $re^{i\phi}$.

Answers at the end of the script.

2.2. Vector spaces

2.2.1. What is a vector space?

A vector space can be constructed from another set, e.g. a field like the real or complex numbers. Here, we will focus mainly on finite complex vector spaces, since that is what we need in this couse.

- An element of a vector space is a *tuple* of elements, i.e. an ordered set of numbers from the underlying field: $\mathbf{v}^T = (v_0, v_1, ...v_{n-1})$ with $v_0, v_1, ... \in \mathbb{C}$. n is the size of the tupel, also called dimension of the vector space.
- The ^{T} stands for *transpose* and indicates that we have written the vector *horizontally*. Normally, we write vectors vertically:

$$\mathbf{v} = \begin{pmatrix} v_0 \\ v_1 \\ ... \\ v_{n-1} \end{pmatrix}$$

- Vectors can be added by adding them element by element:

$$\mathbf{v} + \mathbf{w} = \begin{pmatrix} v_0 \\ v_1 \\ ... \\ v_{n-1} \end{pmatrix} + \begin{pmatrix} w_0 \\ w_1 \\ ... \\ w_{n-1} \end{pmatrix} = \begin{pmatrix} v_0 + w_0 \\ v_1 + w_1 \\ ... \\ v_{n-1} + w_{n-1} \end{pmatrix}$$

- A vector can be multiplied with a number:

$$c \cdot \mathbf{v} = c \cdot \begin{pmatrix} v_0 \\ v_1 \\ ... \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} c \cdot v_0 \\ c \cdot v_1 \\ ... \\ c \cdot v_{n-1} \end{pmatrix}$$

- In the following, an element of $\mathbb{C}^n$ is called a vector, and $c \in \mathbb{C}$ a *scalar* (meaning that it is just a simple complex number).

2.2.2. Inner product in complex vector spaces

A scalar or inner product on a complex vector space V is a positive definite symmetric Hermitian sesquilinearform

$$\langle \cdot, \cdot \rangle : VxV \to \mathbb{C}$$

which means that for $x, y, z \in V$ and $\lambda \in \mathbb{C}$ the following conditions apply:

1. the form is *sesquilinear*
   - $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$

- $\langle \lambda x, y \rangle = \bar{\lambda} \langle x, y \rangle$ *sesquilinear* in the first element (i.e. when pulling out a factor, we get the conjugate of the complex nubmer)
- $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$
- $\langle x, \lambda y \rangle = \lambda \langle x \rangle$ linear in the second element

2. the form is *hermitian*: $\langle x, y \rangle = \overline{\langle y, x \rangle}$
3. the form is positive definite
   - $\langle x, x \rangle \geq 0$ (be aware that $\langle x, x \rangle \in \mathbb{R}$ since it is hermitian)
   - $\langle x, x \rangle = 0$ if and only if $x = 0$

The vector space $\mathbb{C}^n$ the following inner product can be defined::

$$[v, w] = \overline{v} \cdot w = \overline{v_0} \cdot w_0 + \overline{v_1} \cdot w_1 + ... + \overline{v_{n-1}} \cdot w_{n-1}$$

In contrast to the scalar product of real vector space, the first argument of an inner product of a complex vector space is conjugated. For $v = w$ we have: $[v, v] \in \mathbb{R}$, which can easily be shown.

2.2.3. Base

A base of a vector space of dimension n is a set of n vectors that span the complete vector space. Let $b_0, b_1, b_2, ..., b_{n-1} \in V$ be a set of vectors from V which form a *base*. Then, for each $v \in V$, there exists scalar factors $c_0, c_1, ..., c_{n-1} \in \mathbb{C}$ such that

$$v = c_0 b_0 + c_1 b_1 + ... + c_{n-1} b_{n-1}$$

Two vectors $v, w \in V$ are perpendicular to each other, if and only if their inner product is zero:

$$\langle v, w \rangle = 0$$

A base where all base vectors are perpendicular to each other is called a *orthogonal base*. Such a base can be constructed from any base using the Gram–Schmidt process.

Addionally, all base vectors can be normalized to length 1. In this case, the base is called *orthonormal*.

Example

- An example for a orthonormal base in $\mathbb{R}^3$ are the unit vectors (1,0,0), (0,1,0) and (0,0,1):

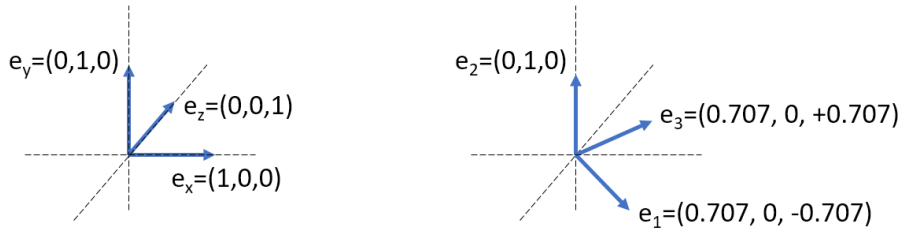


Figure 2.1.: Two different orthonormal bases in 3-dimensional space

2.2.4. Norm

Based on the inner product, we can define a norm (also called length) as follows:

$$|\quad| : \mathbb{V} \to \mathbb{R}$$

with $|v| = \sqrt{\langle v, v\rangle} = \bar{v} \cdot v$. A norm has the following properties:

- not degenerated: $|v| = 0 \Longleftrightarrow v = 0$
- triangle inequality: $|v + w| < |v| + |w|$ mit $v, w \in \mathbb{V}$
- norm respects scalar product: $|c \cdot v| = |c| \cdot |v|$ mit $c \in \mathbb{C}$

2.2.5. Orthogonality and orthonormal base

Let $v, w \in \mathbb{V}$ be two vectors. Then we can define

$$v \text{ and } w \text{ are orthogonal} \Longleftrightarrow \langle v, w\rangle = 0.$$

Then, we can define a orthogonal base as follows: a base $\mathbb{B} = v_0, v_1, \ldots v_{n-1}$ in $\mathbb{V}$ is a **Orthogonalbasis**, if

$$\langle v_i, v_j\rangle = 0 \quad \forall i, j$$

If all vectors are unit vectors (with length 1), then such a base is called **orthonormal base**. Given any base, we can always construct an orthonormal base, i.e. using the Gram-Schmidt procedure.

2.2.6. Hilbert Spaces

For completeness: an infinite-dimensional vector space with inner product, in which every cauchy sequence converges, is called a **Hilbert-Space** (named after the famous German mathematician David Hilbert). In quantum mechanics, this could be a system with an infinite amount of states, e.g. an electron somewhere in 3D space. However, for computing purposes, we typically have a limited amount of states, and therefore a finite-dimensional vector spaces. Thus, the Hilbert space is not necessary for us.

2.3. Matrices

A linear equation $x \to y$ with $x, y \in \mathbb{C}$ can be written as follows:

$$y = m \cdot x + a$$

Similarly, if we have a n-dimensional vector space V and $\boldsymbol{x} \in V$, the equation $\boldsymbol{x} \to y \in \mathbb{C}$ can be written as follows:

$$y = m_0 \cdot x_0 + m_1 \cdot x_1 + \ldots + m_{n-1} \cdot x_{n-1} + a$$

We can also have more than one equation[1]:

$$\begin{aligned} y_0 &= m_{00} \cdot x_0 + m_{01} \cdot x_1 + \ldots + m_{0,n-1} \cdot x_{n-1} + a_0 \\ y_1 &= m_{10} \cdot x_0 + m_1 \cdot x_{11} + \ldots + m_{1,n-1} \cdot x_{n-1} + a_1 \\ &\ldots \\ y_{m-1} &= m_{m-1,0} \cdot x_0 + m_{m-1,1} \cdot x_1 + \ldots + m_{m-1,n-1} \cdot x_{n-1} + a_{m-1} \end{aligned}$$

Here we can use matrices to simplify our equations. Let $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{n-1}\}$ be a vector in n-dimensional space, and $\boldsymbol{y} = \{y_0, y_1, \ldots, y_{m-1}\}$ and $\boldsymbol{a} = \{a_0, a_1, \ldots, a_{m-1}\}$ vectors in m-dimensional space, and

$$M = \begin{pmatrix} m_{00} & m_{01} & \ldots & m_{0,n-1} \\ m_{10} & m_{11} & \ldots & m_{1,n-1} \\ \ldots & \ldots & \ddots & \ldots \\ m_{m-1,0} & m_{m-1,1} & \ldots & m_{m-1,n-1} \end{pmatrix}$$

then we can write the above set of equations as

$$\boldsymbol{y} = M \cdot \boldsymbol{x} + \boldsymbol{a}$$

Note that this matrix is rectangular with (different) dimensions m and n.

[1] Since we are in Computer Science, we start counting from 0. In other domains, e.g. mathematics or physics, one typically starts counting from 1 to n, so the top left index is m_{11}, not like our m_{00}.

2.3.1. Matrix Addition

Two matrices can be added $C = A+B$ if A and B have the same dimensions. Addition of matrices is done element-wise:

$$c_{ij} = a_{ij} + b_{ij}$$

for each $i = 0, .., m-1$, $j = 0, .., n-1$. Note that C will have the same dimensions as A and B. Matrix addition follows the usual rules:

- Addition is commutative: $A + B = B + A$
- Addition is associative: $(A + B) + C = A + (B + C)$
- The zero matrix 0 (with $c_{ij} = 0$ for all $i = 0, .., m-1$ and $j = 0, .., n-1$) is the neutral element of addition: $A + 0 = 0 + A = A$

2.3.2. Matrix Multiplication

Two matrices can be multiplied, if and only if the dimensions match: let A be a matrix with dimensions m and n, and B a matrix with dimensions m' and n', then $C = A \cdot B$ exists if and only if $n = m'$. The resulting matrix C has the dimensions m and n' and can be calculated element-wise (with $i \in \{0, .., m-1\}, k \in \{0, .., n'\}$):

$$c_{ik} = \sum_{j=0}^{n-1} a_{ij} \cdot b_{jk}$$

A special form of matrices are quadratic matrices, where both dimensions are identical. We always can multiply two quadratic matrices with the same dimension.

Some properties of matrix multiplication:

- In general, matrix multiplication is NOT commutative: $A \cdot B \neq B \cdot A$.
- Associativity: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- Transpose law: $(A \cdot C)^T = C^T \cdot A^T$ (more on the transpose of a matrix in the next section)
- Homogenity: $A \cdot (\lambda C) = \lambda(A \cdot C) = (\lambda A) \cdot C$ with $\lambda \in \mathbb{C}$
- Distributivity: $(A + B) \cdot C = (A \cdot C) + (B \cdot C)$
- For quadratic matrices $A, B, E \in \mathbb{C}^{n \times n}$, there exists a unit matrix E with $A \cdot E = E \cdot A = A$. Matrix E consists of ones on the diagonal positions (i.e. $a_{ii} = 1$ for all $i \in \{0, .., n-1\}$) and zeros elsewhere. Then, if there is a B such that $A \cdot B = E$, then $B = A^{-1}$ is called the *inverse* of A. The inverse can be constructed by the Gauss-Jordan algorithm.

Exercises

1. Let $A = \begin{pmatrix} 3 & 1 \\ 2 & 5 \end{pmatrix}, B = \begin{pmatrix} 7 & 2 \\ 4 & 5 \end{pmatrix}$ and $C = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}$. Calculate $A \cdot (B + C)$ and $(A \cdot B) + (A \cdot C)$ and compare both results.

2. For a 2x2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the inverse can be calculated as follows:

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

where $ad - bc = det(A)$ is the determinant of A. Calculate the inverse A^{-1} of one of the matrices A, B or C from the previous exercise and show that the product of the matrix with its inverse is E.

2.3.3. Transpose Matrix

Let A be a matrix, then the transpose of A is $B = A^T$, where for all elements

$$b_{ij} = a_{ji}$$

Examples

- Let $A = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$, then $A^\dagger = \begin{pmatrix} 0 & 2 \\ 1 & 3 \end{pmatrix}$.
- This is also possible for non-quadratic matrices, e.g. let $A = \begin{pmatrix} 0 & 1 & 77 \\ 33 & 2 & 3 \end{pmatrix}$, then $A^\dagger = \begin{pmatrix} 0 & 33 \\ 1 & 2 \\ 77 & 3 \end{pmatrix}$.
- ... and trivial for vectors (we covered this already): let $v = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$, then $v^\dagger = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}$.

2.3.4. Unitary Matrix

Up to now, we were not concerned about the *type* of the individual elements of a matrix. For complex numbers, can define the *conjugate* matrix $D = C^*$, where each element is the conjugate:

$$d_{ij} = c^*_{ij}$$

The conjugate of a real matrix (i.e. consisting of real numbers) is the matrix itself:

$$D_{real} = D^*_{real}$$

A unitary matrix $U \in \mathbb{C}^{n \times n}$ is a complex matrix whose conjugate transpose is the inverse of the matrix itself:

$$UU^\dagger = U^\dagger U = I$$

Unitary matrices are very important for quantum computing, since all operations on qubits are described by unitary matrices. Some properties of unitary matrices:

- A unitary matrix keeps the length of a vector: let $v, w \in \mathbb{C}$, then the inner product $\langle v, w \rangle = \langle Uv, Uw \rangle$
- All eigenvalues λ of a unitary matrix U have $|\lambda| = 1$.
- Two eigenvectors $\vec{a}$ and $\vec{b}$ with different eigenvalues $\lambda_a \neq \lambda_b$ are othogonal to each other.
- Matrix U has n pairwise orthogonal eigenvectors.

Exercises

Show that the following matrices are unitary:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

2.4. Tensor Product

The tensor product is different from the cartesian product. In the tensor product, each element of the first argument (a vector or a matrix) is multiplied with the second argument. Lets do an example first. Assume we have a 2-vectors $\mathbf{a} = (a_0, a_1)$ and 3-vector $\mathbf{b} = (b_0, b_1, b_2)$. Then the result of the tensor product of both is a 6-vector

$$\mathbf{a} \otimes \mathbf{b} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_0 \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \\ a_1 \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_0 b_2 \\ a_1 b_0 \\ a_1 b_1 \\ a_1 b_2 \end{pmatrix}$$

For two given vectors v and w with dimensions n resp. m, the resulting vector $v \otimes w$ has the dimension $n \cdot m$.

2.4.1. Matrix tensor product

In the same way, a tensor product can be defined for matrices. Lets assume we have a 2x2 matrix A and a 2x3 matrix B, with

$$A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

and

$$B = \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{pmatrix}$$

The resulting tensor product is then a 4x6 matrix:

$$\begin{aligned} A \otimes B &= \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \otimes B = \begin{pmatrix} a_{00} \cdot B & a_{01} \cdot B \\ a_{10} \cdot B & a_{11} \cdot B \end{pmatrix} \\ &= \begin{pmatrix} a_{00} \cdot \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{pmatrix} & a_{01} \cdot \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{pmatrix} \\ a_{10} \cdot \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{pmatrix} & a_{11} \cdot \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{pmatrix} \end{pmatrix} \\ &= \begin{pmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{00}b_{02} & a_{01}b_{00} & a_{01}b_{01} & a_{01}b_{02} \\ a_{00}b_{10} & a_{00}b_{11} & a_{00}b_{12} & a_{01}b_{10} & a_{01}b_{11} & a_{01}b_{12} \\ a_{10}b_{00} & a_{10}b_{01} & a_{10}b_{02} & a_{11}b_{00} & a_{11}b_{01} & a_{11}b_{02} \\ a_{10}b_{10} & a_{10}b_{11} & a_{10}b_{12} & a_{11}b_{10} & a_{11}b_{11} & a_{11}b_{12} \end{pmatrix} \end{aligned}$$

Lets assume the first argument has the dimensions $m \times n$, while the second one has the dimensions $p \times q$. Since each element of the first argument is multiplied by the second argument, we end up with $(m \cdot p) \times (n \cdot q)$ dimensional result. For example, if A is a 2 by 2 matrix and B a 3 by 4 matrix, the result $A \otimes B$ has 6 rows and 8 columns.

2.4.2. Properties of the tensor product

The tensor product has the following properties(where $A \in \mathbb{C}^{m \times n}$, $B \in \mathbb{C}^{p \times q}$ $C \in \mathbb{C}^{r \times s}$, and $D \in \mathbb{C}^{s \times r}$:

- The tensor product does not communitate: $A \otimes B \neq B \otimes A$. However, communitating the factors results in a re-ordering of rows (and columns) of the resulting element.
- The tensor product respects addition in the first and second element (note that the dimensions of A, B, C and D must fit together[2]:

$$\begin{aligned} (A + B) \otimes C &= A \otimes C + B \otimes C \\ D \otimes (A + B) &= D \otimes A + D \otimes B \end{aligned}$$

[2]For the first equation, that would be that A and B have the same dimensions, and C fits to these, i.e. $m = p$, $n = q$ and $m = s$; D could be just the transpose of C

- The tensor product allows multiplication by a scalar $\lambda \in \mathbb{C}$:

$$\lambda \cdot (A \otimes B) = (\lambda \cdot A) \otimes B = A \otimes (\lambda \cdot B)$$

2.4.3. Exercise

In the following, let assume that v is a vector from $\mathbb{V} = \mathbb{C}^2$ and w a vector from $\mathbb{W} = \mathbb{C}^3$.

1. What are the dimensions of $v \otimes w$?
2. Let $v = (2, 3)$ and $w = (3, 7, 5)$. What is the result of $v \otimes w$?
3. Lets assume we have $x \in \mathbb{C}^6$, and we know that $x = v \otimes w$ is a tensor product. What are the vectors v and w, when $x^T = (8, 44, 12, 6, 33, 9)$?
4. With normal multiplication, we know that for example $12 = 3 \cdot 4 = 2 \cdot 6$. How about the tensor product? Can it happen that a vector is the result of two different tensor products, i.e. $c = a \otimes b = a' \otimes b'$ with $a, a', b, b', c \in \mathbb{V}$ and $a \neq a'$ resp. $b \neq b'$?
5. Given $x^T = (8, 0, 0, 0, 0, 18) \in \mathbb{C}^6$, what two factors $v \in \mathbb{V}$ and $w \in \mathbb{W}$ yield $v \otimes w = x$?

Answers at the end of the script.

3. Classical Bits in Matrix Formulation

Boolean algebra is part of every introductory course to computer science. It is fundamental to modern computers, since all data representation and processing is based on 0 and 1, the two fundamental states with a computer.

In order to get used to the quantum computing approach, we will first reconsider boolean algebra, using the matrix notation.

So, we define our boolean number on the $\mathbb{C}^2$ vector space. A state is then $s = \begin{pmatrix} a \\ b \end{pmatrix}$.
Since we only need two succinct states, we define

$$\mathbf{0} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

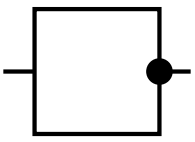
and

$$\mathbf{1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

A boolean operation is then an operation on $\mathbb{C}^2$, i.e. a matrix that is applied to the input vector.

3.1. Boolean NOT

Lets start with a simple boolean NOT: it has one input and one output, where the output is just the opposite of the input:



with

Input	Output
0	1
1	0

What does this mean for the matrix operation? The input is one bit, i.e. a 2-vector, and the output is also a 2-vector, which means that the NOT matrix is a 2x2 matrix with the following properties:

$$\begin{pmatrix}0\\1\end{pmatrix} = NOT \begin{pmatrix}1\\0\end{pmatrix}$$

and

$$\begin{pmatrix}1\\0\end{pmatrix} = NOT \begin{pmatrix}0\\1\end{pmatrix}$$

Since $NOT = \begin{pmatrix}n_{00} & n_{01}\\ n_{10} & n_1 1\end{pmatrix}$ is a 2x2 matrix, we can write:

$$\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}n_{00} & n_{01}\\ n_{10} & n_{11}\end{pmatrix} \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}n_{00}\\ n_{10}\end{pmatrix}$$

and

$$\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}n_{00} & n_{01}\\ n_{10} & n_{11}\end{pmatrix} \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}n_{10}\\ n_{11}\end{pmatrix}$$

This yields $n_{00} = 0$, $n_{10} = 1$, $n_{01} = 1$, and $n_{11} = 0$, and thus $NOT = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$.

This can be checked easily:

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

and

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

3.2. Boolean AND

How about the boolean AND? The first thing we can observe is that the Boolean AND has two inputs and one output. The output is a 2-vector like the NOT, but what about the input? We have for possible input configurations, namely 00, 01, 10 and 11, which can be expressed by a 4-vector with one 1 and three 0s. Or, more mathematically, the input is the tensor product of the two single inputs:

$$\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1 \cdot \begin{pmatrix}1\\0\end{pmatrix}\\ 0 \cdot \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1 \cdot \begin{pmatrix}0\\1\end{pmatrix}\\ 0 \cdot \begin{pmatrix}1\\0\end{pmatrix}\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$\begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

$$\begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Then, the transformation matrix for the boolean AND has the dimensions 4x2, i.e.

$$\begin{pmatrix}o_0\\o_1\end{pmatrix} = AND \begin{pmatrix}i_0\\i_1\\i_2\\i_3\end{pmatrix} = \begin{pmatrix}a_00 & a_01 & a_02 & a_03\\ a_10 & a_11 & a_12 & a_13\end{pmatrix} \begin{pmatrix}i_0\\i_1\\i_2\\i_3\end{pmatrix}$$

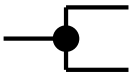
Since the output of a AND gate is $\begin{pmatrix}0\\1\end{pmatrix}$ only if the input is $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$ and $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ otherwise, it follows that

$$AND = \begin{pmatrix}1 & 1 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Show that this is the correct answer!

3.3. Exercise

1. Write down the matrices for OR, NAND, andd XOR.
2. What is the transformation matrix for a *line split*?



4. Single Qubits

In this chapter, we look into single qubits and gates for single qubits.

4.1. Qubit

A qubit $|\psi\rangle$ is described by a vector from $\mathbb{C}^2$ with length 1, which means

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

with the amplitudes $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

4.1.1. What does this mean?

What is the meaning of $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$? From a purely mathematical view, this is just a mathematical representation of the qubit, that we can manipulate according to some rules (which will follow later). The state is defined by two complex numbers $\alpha, \beta \in \mathbb{C}$.

Since the qubit is embedded in the real world, we have also a physical interpretation. Whenever we *measure* the qubit, we will only get one of two possible values: 0 or 1. However, for a normal state the absolute value of these factors determine the probability of measuring a 0 or 1:

$$P(0) = |\alpha|^2 \text{ and } P(1) = |\beta|^2$$

Since the probability of measuring either 0 or 1 must be 1, we have

$$P(0) + P(1) = |\alpha|^2 + |\beta|^2 = 1$$

Please note that the amplitudes are complex numbers, which sometimes lead to surprising results (when only used to real numbers).

Examples

a) $|0\rangle = 1 \cdot |0\rangle + 0 \cdot |1\rangle$, from which we get $P(0) = 1$ and $P(1) = 0$. This corresponds to a classical bit 0.

b) Similar we have $|1\rangle = 0 \cdot |0\rangle + 1 \cdot |1\rangle$ and $P(0) = 0$ and $P(1) = 1$.

c) $|+\rangle = \frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle)$, which is special since 0 and 1 are equally probable: $P(0) = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} = P(1)$.

d) Similar, we have $|-\rangle = \frac{1}{\sqrt{2}} \cdot (|0\rangle - |1\rangle)$.

e) What do you thing: besides $|+\rangle$ and $|-\rangle$, are the more states with equal probability $P(0) = P(1) = \frac{1}{2}$?

f) A more *complex* example of a qubit state would be $|\psi\rangle = \frac{1}{2}(|0\rangle + (1 - i\sqrt{(2)})\,|1\rangle))$

4.1.2. Collapse of the wave function

When doing a measurement, we cannot really find out the state of about the qubit, i.e. we cannot measure the *amplitudes* of the basic states. The only thing we get is zero and one, and we need to repeat the whole process to get a histogram that tells us something about the individual probabilities.

However, things are really strange here: if we measure a qubit in superposition, two things happen:

- we get a zero or a one (with respect to the probabilities), and
- the superposition is *destroyed*!

If we measure again, the second measurement will with 100% probability yield the same result as the first one! By doing the measurement, the state is destroyed, and we cannot continue (except some algorithms that use this fact in a clever way). This is called the "collapse of the wave function" (see Copenhagen interpretation of quantum mechanics) and not yet fully understood.

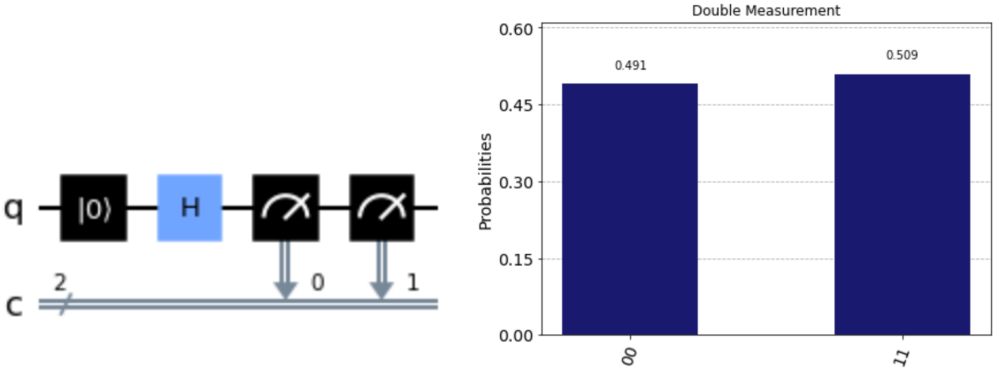


Figure 4.1.: Left: Performing a double measurement; right: resulting histogram (run in Qiskit Simualtor)

The setup in fig. 4.1, a qubit in superposition is measured twice. The first measurement produces a random qubit, while the second measurement produces exactly the same value as the first one. Thus, the values "00" and "00" have high probabilities, while "01" and "01" do not appear.

4.1.3. Using the Polar Form

Complex numbers can be written in polar form, where we state the length r (distance from zero) and the angle ϕ from the x-axis:

$$a + ib = re^{i\phi}$$

, where $e^{i\phi} = \cos(\phi) + i\sin(\phi)$ is the complex exponential function.

Using the polar form for the complex numbers in a qubit, we get

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

where the length $r_a, r_b \in \mathbb{R}$ and therefore $1 = r_a^2 + r_b^2$.

The angle of the complex numbers can be refactored using $\Delta\phi = \phi_b - \phi_a$:

$$|\psi\rangle = e^{i\phi_a}(r_a |0\rangle + r_b e^{i\phi_b - i\phi_a} |1\rangle) = e^{i\phi_a}(r_a |0\rangle + r_b e^{i\Delta\phi} |1\rangle)$$

The first factor $e^{i\phi_a}$ is called the global phase, the second one $e^{i\Delta\phi}$ local phase.

Up to now, no physical effect has been found that depends on the global phase of the qubit. All effects that we know of solemnly depend on the local phase of the qubit. Therefore, one can remove the global phase from the qubit and thus making the first amplitude real, which simplifies things a little.

4.1.4. Bloch Sphere

What does this leaves us with? Instead of two independent complex numbers (with 4 independent degrees of freedom), we now have one real number and one complex number, plus a normalization constraint $|\alpha|^2 + |\beta|^2 = 1$, which leaves us 2 independent degrees of freedom.

Therefore, we can represent the state of a qubit in the 2-dimensional plane. A very useful representation is the Block sphere, where the state of the qubit is located on the (2-dimensional) surface of a 3-D sphere with radius 1. This can be derived as follows.

When using the polar form (without the global phase), the qubit state looks as follows:

$$\begin{aligned} |\psi\rangle &= a |0\rangle + b |1\rangle \\ &= r_a \cdot e^{i\phi_a} |0\rangle + r_b \cdot e^{i\phi_b} |0\rangle \end{aligned}$$

We can define an angle $\theta = 0..\pi$ so that $r_a = cos(\frac{\theta}{2})$ and $r_b = sin(\frac{\theta}{2})$, which nicely yields

$$r_a^2 + r_b^2 = cos(\frac{\theta}{2})^2 + sin(\frac{\theta}{2})^2 = 1$$

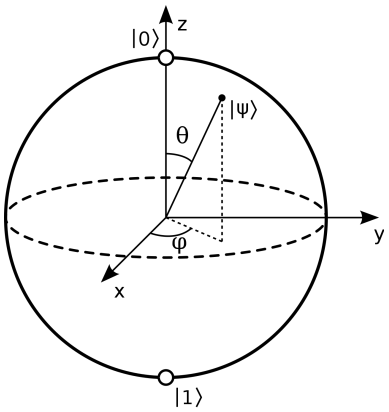


Figure 4.2.: Bloch Sphere [5]

Then, the state is fully described by the angles θ and ϕ:

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

The same way that longitude and latitude are used to locate any point on earth, we can use θ and ϕ to locate any possible state on the unit sphere, which is called Bloch sphere here. θ starts at the positive z-axis, while ϕ starts at the positive x-axis.

Examples:

- The base vector $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ is the highest point (where the positive z-axis hits the sphere).
- For $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$, $cos(\theta/2) = 0$, so $\theta = \pi$, which is on the opposite side of the sphere.
- The Pauli-X corresponds to a 180° rotation around the y-axis, and thus changes $\begin{pmatrix}1\\0\end{pmatrix}$ into $\begin{pmatrix}0\\1\end{pmatrix}$ and vice versa.

However, we will not need the Bloch Sphere in this course, so we will not look deeper into it. If you are interested, you can have a look into any quantum computing book, e.g. [7].

4.1.5. A short note on quantum computers

So, how can we utilize these strange elements for data processing?

The general workings of a quantum computer are as follows (note that this is just a typical case, individual algorithms might work differently):

a) First, we initialize all qubits to a defined state. Normally, this is the $|0\rangle$ state, resp. $|00...000\rangle$ with more than one qubit.

b) In the next step, a superposition of all possible states is created (i.e. using a Hadamard on all qubits).

c) Then, by clever manipulation of the states, one tries to *increase* the probability of the result state(s), and *decrease* the probability of all other states.

d) Finally, we *measure* the qubits. This will destroy our quantum state, but will give us a result of the computation.

e) However, since the result will have a probability less than 1 of being correct, we have to repeat the procedure. As long as the probability is larger than 1/2 (for a true/false output), we can increase the probability to any desired value by just repeating the calculation often enough. For example, we can have 1000 shots of the calculation, which typically yields a stable result.

4.2. Single Qubit Gates

Here, we look into some important quantum gates that operate on a single qubit.

Definition: Quantum Gate

A quantum gate is an operator that acts on the state of one (or more) qubits. Such an operator is represented by a unitary matrix.

4.2.1. The Paulis: X, Y, Z

The Pauli-operations have the form:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

4.2.1.1. Pauli-X

When applying a Pauli-X operation on a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we get:

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta|0\rangle + \alpha|1\rangle$$

This means that the Pauli-X exchanges the amplitudes of the qubit and therefore is called the **bitflip** operation.

When applying this to the base states, we get:

$$X|0\rangle = |1\rangle \text{ and } X|1\rangle = |0\rangle$$

oc

4.2.1.2. Pauli-Z

When applying a Pauli-Z on a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we get:

$$Z|\psi\rangle = Z(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ -\beta\end{pmatrix} = \alpha|0\rangle - \beta|1\rangle$$

So, a Pauli-Z changes the sign of the amplitude of the second base vector. The relation between first and second base is called **phase**; therefore, the Pauli-Z is also called **phase flip**.

When applying this to the base states, we get:

$$Z|0\rangle = |0\rangle \text{ and } X|1\rangle = -|1\rangle$$

4.2.1.3. Pauli-Y

When applying a Pauli-Y on a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we get :

$$Y|\psi\rangle = Y(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = i\begin{pmatrix}-\beta\\ \alpha\end{pmatrix} = -i\beta|0\rangle + i\alpha|1\rangle$$

The consequences of a Pauli-Y are slightly more complicated: the amplitudes are exchanged, the phase is flipped and the result has an imaginary factor.

What do we get when applying this to the base states?

4.2.1.4. Properties of the Paulis

The Pauli gates have the following properties:

- Each Pauli-operation is its own inverse: $X \cdot X = Y \cdot Y = Z \cdot Z = I_2$, where $I_2 = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$ is the 2x2 identity-matrix.

Exercise: confirm all of the above!

4.2.2. Hadamard

The Hadamard operation H has the form $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. Applied on a base state, this yields:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

The Hadamard operation transforms a base state in a state with superposition. The states $|+\rangle$ and $|-\rangle$ are the states with maximum superposition: both states are equally probable when doing a measurement.

We have:

- The Hadamard-Operation is its own inverse: $H \cdot H = 1$ (check this by calculation)

4.2.3. More Single Qubit Gates

There are more quantum gates that operate on a single qubit:

a) S with $S = ...$

b) T

c) $\sqrt{NOT}$

5. Quantum Registers and Entanglement

5.1. Quantum Register

5.1.1. 2-Qubit Register

A quantum register is the combination of two or more qubits, which are combined using the tensor product. Let $|\psi\rangle = \alpha_0 |0\rangle + \beta_0 |1\rangle >$ and $|\phi\rangle = \alpha_1 |0\rangle > +\beta_1 |1\rangle$ be two qubits, which are combined to a quantum register (technically, this means they are coupled). The quantum register then can be represented by

$$|\theta\rangle = |\psi\rangle \otimes |\phi\rangle = (\alpha_0 |0\rangle + \beta_0 |1\rangle) \otimes (\alpha_1 |0\rangle + \beta_1 |1\rangle)$$

$$|\theta\rangle = \begin{pmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \beta_0\beta_1 \end{pmatrix} = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle$$

with some complex amplitudes $c_{00}, c_{01}, c_{10}, c_{11} \in \mathbb{C}$[1]. Additional constraints are:

a) The sum of the squared length of the amplitudes must be one: $|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$. The reason is that when measuring, we get one of the four possible outcomes 00, 01, 10 or 11. Since the probability to one of those is one, the sum of the squares must be one, too.

b) The global phase can be removed, so that the first amplitude is real.

Note that this leaves us with *6 degrees of freedom*. This is surprising, since we had 2 degrees of freedom for a single qubit. In the classical world, we would therefore expect 4 degrees of freedom for 2 qubits. However, in quantum world we get more, namely 4 + 2 degrees of freedom. Those 2 extra degrees of freedom basically describe the added value that we get because of *entanglement*.

[1] We have $c_{00} = \alpha_0\alpha_1, c_{01} = \alpha_0\beta_1$, etc.

5.1.2. Many-Qubit Register

Lets assume we have n qubits that form a n-qubit-register. Then, the general state is

$$\Psi_n = \sum_{x \in X} c_x |x\rangle = c_{00..00} |00..00\rangle + c_{00..01} |00..01\rangle + c_{00..10} |00..10\rangle + ... + c_{11..11} |1..11\rangle$$

where $X = \{0,1\}^n$ is the set of all possible combinations of 0s and 1s with exactly n digits. Note that x consists of all binary numbers with n digits. Instead of writing binary numbers, we could also use decimal numbers. Then, we get

$$\Psi_n = \sum_{x=0}^{2^n-1} c_x |x\rangle = c_0 |0\rangle + c_1 |1\rangle + c_2 |2\rangle + ... + c_{2^n-1} |2^n - 1\rangle$$

The usual constraints apply, namely $\sum |c_x|^2 = 1$, and $c_0 \in \mathbb{R}$.

5.2. CNOT resp. controlled Pauli-X

The CNOT gate (also called CX gate) is a 2-qubit gate with a control qubit, which is unchanged, and the qubit that it operates on, and which then contains the result.:

- If the control qubit is 0, the second qubit is not changed
- If the control qubit is 1, the second qubit is bit-flipped.

The CNOT has the following form:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

5.2.1. Effects on classical qubits

Applied on the base states the qubit corresponds to a classical XOR. If the control qubit is 0, the second qubit will not change. If the control qubit is 1, the second qubit is flipped:

$$|00\rangle \to |00\rangle$$
$$|01\rangle \to |01\rangle$$
$$|10\rangle \to |11\rangle$$
$$|11\rangle \to |10\rangle$$

5.2.2. Effects on arbitrary qubits

What happens if we apply the CNOT to an arbitrary qubit?

$$CNOT \cdot |\theta\rangle = CNOT \cdot \begin{pmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \beta_0\beta_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \beta_0\beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\beta_1 \\ \alpha_1\beta_0 \end{pmatrix}$$

This means, the CNOT swaps the amplitudes of the $|10\rangle$ and $|11\rangle$ states of the qubit register.

5.3. A note on Entanglement

Looking at the previous equation, we have a qubit register $|\Phi\rangle = q_0 \otimes q_1$ which is the tensor product of two states q_0 and q_1. However, after applying CNOT we get a state that is no longer the tensor product of two separate states. Instead, both qubits in the register are now *entangled*: they cannot be viewed at separately, but must always be viewed together.

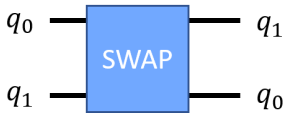
Definition: Entangled States

Two or more qubits are called entangled, if their state cannot be described as the tensor product of some individual qubit states.

Entanglement means that there are no longer individual state of the qubits!

5.4. SWAP

The idea of the SWAP gate is to exchange two qubits:



Looking at the base vectors, we see that the following mapping exists if we swap the two qubits:

$$|00\rangle \rightarrow |00\rangle$$
$$|01\rangle \rightarrow |10\rangle$$
$$|10\rangle \rightarrow |01\rangle$$
$$|11\rangle \rightarrow |11\rangle$$

This leads to the following SWAP matrix:

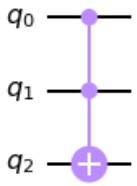
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A arbitrary quantum state $\Phi = c_{00}\,|00\rangle + c_{01}\,|01\rangle + c_{10}\,|10\rangle + c_{11}\,|11\rangle$ is transformed as follows:

$$\text{SWAP}(\Phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix} = \begin{pmatrix} c_{00} \\ c_{10} \\ c_{01} \\ c_{11} \end{pmatrix}$$

5.5. Toffoli

The Toffoli gate has 3 inputs: two control qubits and one controlled qubit. Similar to the CNOT, the controlled qubits is flipped if both inputs are one. If only one input is 1, no flip occurs.



Since we have 3 inputs, we have an 8-dimensional vector with 8 base vectors. Applying the Toffoli gate changes the base vectos as follows:

$$\begin{aligned} |000\rangle &\rightarrow |000\rangle \\ |001\rangle &\rightarrow |001\rangle \\ |010\rangle &\rightarrow |010\rangle \\ |011\rangle &\rightarrow |011\rangle \\ |100\rangle &\rightarrow |100\rangle \\ |101\rangle &\rightarrow |101\rangle \\ |\mathbf{110}\rangle &\rightarrow |\mathbf{111}\rangle \\ |\mathbf{111}\rangle &\rightarrow |\mathbf{110}\rangle \end{aligned}$$

The corresponding matrix then is:

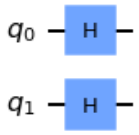
$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

5.6. Multiple Hadamard $H^{\otimes n}$

One of the single gates we discussed was the Hadamard gate. When we have a quantum register, we often apply the Hadamard to more than one qubit at a time. Note that the Hadamard operates on each qubit separately, so we do not get entanglement from it.

The notation for multiple Hadamard is $H^{\otimes n}$, which is the n-fold tensor product of the Hadamard operation.

Example



Applying a Hadamard on two qubits at a time gives us:

$$H^{\otimes 2} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Applying a Hadamard on a base vector $|u\rangle$ gives us:

$$H|u\rangle = \frac{1}{\sqrt{2^1}}(|0\rangle + (-1)^u |1\rangle)$$

where u is in $\{0, 1\}$. Using summation, we can write this as follows:

$$H|u\rangle = \frac{1}{\sqrt{2}} \sum_{\nu \in \{0,1\}} (-1)^{\nu \cdot u} |\nu\rangle$$

This notation is quite clever, as we will see later with big Hadamard matrices. So, lets check what happens with $H^{\otimes 2}$, with a two-dimensional vector $|u\rangle = |u_1 u_2\rangle$:

$$
\begin{aligned}
(H \otimes H)\,|u\rangle &= H\,|u_1\rangle \otimes H\,|u_2\rangle \\
&= \Big(\frac{1}{\sqrt{2}} \sum_{\nu_1 \in \{0,1\}} (-1)^{\nu_1 \cdot u_1}\,|\nu_1\rangle\Big) \otimes \Big(\frac{1}{\sqrt{2}} \sum_{\nu_2 \in \{0,1\}} (-1)^{\nu_2 \cdot u_2}\,|\nu_2\rangle\Big) \\
&= \frac{1}{2} \sum_{\nu \in \{0,1\}^2} (-1)^{\nu_1 \cdot u_1 + \nu_2 \cdot u_2}\,|\nu\rangle
\end{aligned}
$$

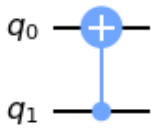
Note that the exponent on the -1 number looks like a inner product (and in fact *is* the inner product). So, we can abstract this formula to the general case for $H^{\otimes n}$:

$$
H^{\otimes n}\,|u\rangle = \frac{1}{\sqrt{2^n}} \sum_{\nu \in \{0,1\}^n} (-1)^{\nu \cdot u}\,|\nu\rangle
$$

We will use this formula later when looking at algorithms, e.g. the Deutsch-Jozsa algorithm.

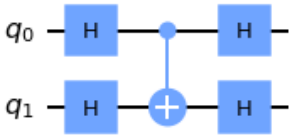
5.7. Exercises on CNOT, SWAP, and $H^{\otimes 2}$

a) We could switch the control qubit and the controlling qubit:



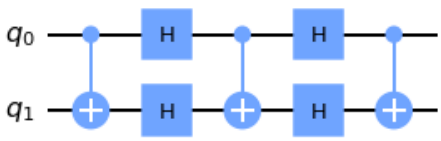
What does the corresponding matrix looks like?

b) Now lets look at a CNOT that is between two $H^{\otimes 2}$.



What do the matrices look like? Calculate the product and the resulting matrix.

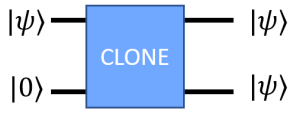
c) A SWAP can be replaced by the following circuit:



Confirm that this is correct.

5.8. No Cloning Theorem

Now lets try to copy a qubit. We need at least two qubits, one $\psi = a|0\rangle + b|1\rangle$ that can have any possible state, and another one, e.g. a $|0\rangle$. Let further U be a copy-function, as usual an unitary operator. The circuit might look like this:



Is this possible? In order to find out, lets start on the right side. Here we have ψ twice, this means

$$|\psi\psi\rangle = |\psi\rangle \otimes |\psi\rangle = (a|0\rangle + b|1\rangle) \otimes (a|0\rangle + b|1\rangle) = a^2|00\rangle + ab|01\rangle + ab|10\rangle + b^2|11\rangle$$

Ok, this should be the output of $U|\psi 0\rangle$. Since U is a linear, unitary operator, we can do the following:

$$U(|\psi 0\rangle) = U(|\psi\rangle \otimes |0\rangle) = U((a|0\rangle + b|1\rangle) \otimes |0\rangle) = U(a|00\rangle + b|10\rangle) = aU(|00\rangle) + bU()|10\rangle)$$

Remember that U a copy-operator is: this means, the left bit is copied to the right, an so we have $U(|00\rangle) = |00\rangle$ and $U(|10\rangle) = |11\rangle$. Therefore, we get

$$U(|\psi 0\rangle) = aU(|00\rangle) + bU|10\rangle) = a|00\rangle + b|10\rangle$$

Comparing this to the term above, we have

$$a^2|00\rangle + ab|01\rangle + ab|10\rangle + b^2|11\rangle = a|00\rangle + b|10\rangle$$

which is not true in the general case. However, there is one interesting exception: if and only if either a=0 or b=0, the equation is true! This means, the only bits that can be copied are the pure "classical" ones $|0\rangle$ and $|1\rangle$!

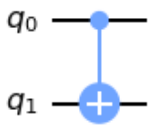
5.9. A word on the notation

In this section, we will look at the phase kickback that occurs with the controlled-not (CNOT). We will also consider the different qubit notation modes, namely the physical notation, which is used in the Homeister book, and the computer science notation, that is used by IBM Qiskit.

5.9.1. Register representation

As stated before, we can associate a qubit state like $|00\rangle$ to hardware circuit in two different ways. This can be seen with the following circuit:

On the circuit, we have the qubits stacked on each other. See for example the figure below, where we have q_0 in the upper position and q_1 in the lower position. Now we can write our combined state as follows:



a) We construct the register as $q_0 \otimes q_1$, which leads us to $|q_0 q_1\rangle$. The left qubit is the control, the right one is acted on. This is the notation used by physics (and in the Homeister book).

b) However, we could also construct the register using $q_1 \otimes q_0$, which leads us to $|q_1 q_0\rangle$. Now the right qubit is the controlling one, while the left one is acted on. This is the computer science notation, where the left-most bit typically is the least significant one (except maybe some serial network protocols where this is different).

We have to be careful here, since this has consequences on how our matrices look like and how we need to calculate the result. Either way works and is correct, but if we mix things up we will get into trouble.

Unfortunately, most of the books[2] use the physical notation, while IBM Qiskit uses the computer science notation. We will somehow use both, so sometimes we must be careful what we are doing.

5.9.2. CNOT in physical notation

In physical notation, the left qubit is the controlling one, and the CNOT looks like this (physical notation):

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Applyed base states, it is like the classical XOR gate: a 1 on the control (qu)bit flips the other bit:

$$|00\rangle \rightarrow |00\rangle$$
$$|01\rangle \rightarrow |01\rangle$$
$$|10\rangle \rightarrow |11\rangle$$
$$|11\rangle \rightarrow |10\rangle$$

[2] A notable exception is the book "Quantum Computing for Computer Scientists"[7]

We can check this, e.g. by calculating $\mathrm{CNOT}\,|10\rangle$ which yields

$$\mathrm{CNOT}\,|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

5.9.3. CNOT in Qiskit notation

In Qiskit notation, the *right-most* qubit is the controlling one: whenever it is 1, the left qubit is flippped:

$$\begin{aligned} |00\rangle &\to |00\rangle \\ |01\rangle &\to |11\rangle \\ |10\rangle &\to |10\rangle \\ |11\rangle &\to |01\rangle \end{aligned}$$

How does the CNOT matrix looks like? Well, as you can easily see, quite different from above:

$$\mathrm{CNOT}_q = \mathrm{CX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

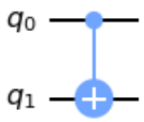
We can check this works, e.g. by calculating $\mathrm{CNOT}_q\,|01\rangle$ which yields

$$\mathrm{CX}\,|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This is also what Qiskit tells us if we ask:

In [14]:
```
qc = QuantumCircuit(2)
qc.cx(0,1)
display(qc.draw())

qc.save_unitary()
usim = Aer.get_backend('aer_simulator')
qobj = assemble(qc)
unitary = usim.run(qobj).result().get_unitary()
array_to_latex(unitary, prefix="\\text{Circuit = }\n")
```

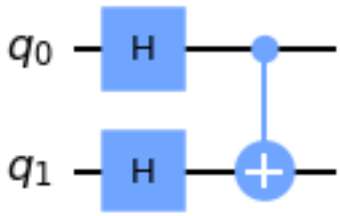


Out[14]:

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

5.10. Phase-Kickback for Controlled-Not

The CNOT gate is applied to 2 qubits. Lets see what happens if the qubits are in superposition.



We start with the $|00\rangle$ base state. Applying the two Hadamard gives us

$$(H \otimes H)\,|00\rangle = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$
$$= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = |{+}{+}\rangle$$

Applying the CNOT has the effect of exchanging the amplitudes of the states where the control bit is 1. However, in the above state this has no effect since the amplitude is the same for all states. So,

$$\mathrm{CNOT}\,|{+}{+}\rangle = \frac{1}{2}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = |{+}{+}\rangle$$

But what happens when we start with $|10\rangle$ (where the right bit is the control qubit q_0 and the left bit the controlled qubit q_1)?

$$\begin{aligned}|\phi_0\rangle = H^{\otimes 2}\,|10\rangle &= \frac{1}{2}\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\\-1\\-1\end{pmatrix}\\ &= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle) = |{-}{+}\rangle\end{aligned}$$

since

$$\frac{1}{2}\begin{pmatrix}1\\1\\-1\\-1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |{-}\rangle \otimes |{+}\rangle = |{-}{+}\rangle$$

Now we apply the CNOT, which, as a controlled Pauli X, swaps the amplitude of the controlled (i.e. second) qubit:

$$\mathrm{CNOT}\,|{-}{+}\rangle = \frac{1}{2}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\1\\-1\\-1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\1\\-1\\-1\end{pmatrix} = |{-}{+}\rangle$$

What happens with $|{+}{-}\rangle$ and $|{-}{-}\rangle$?

$$\mathrm{CNOT}\,|{+}{-}\rangle = \frac{1}{2}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\-1\\1\\-1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\-1\\-1\\1\end{pmatrix} = |{-}{-}\rangle$$

$$\mathrm{CNOT}\,|{-}{-}\rangle = \frac{1}{2}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\-1\\-1\\1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}1\\-1\\1\\-1\end{pmatrix} = |{+}{-}\rangle$$

Before, we learned the the CNOT (i.e. the controlled Pauli-X) flips the controlled qubit (i.e. the second qubit) if the controlling qubit (i.e. the first one) is 1, while the controlling qubit remains unchanged (we used it to implement a classical XOR). However, in the Hadamard base, we see that the *controlling* qubit is changed, while the *controlled* qubits remains the same!

This effect is called *phase kickback*, since in the Hadamard base, the CNOT has the effect of flipping the controlling qubit whenever the controlled qubit is $|-\rangle$.

6. Algorithms (Part 1)

6.1. Deutsch Algorithm

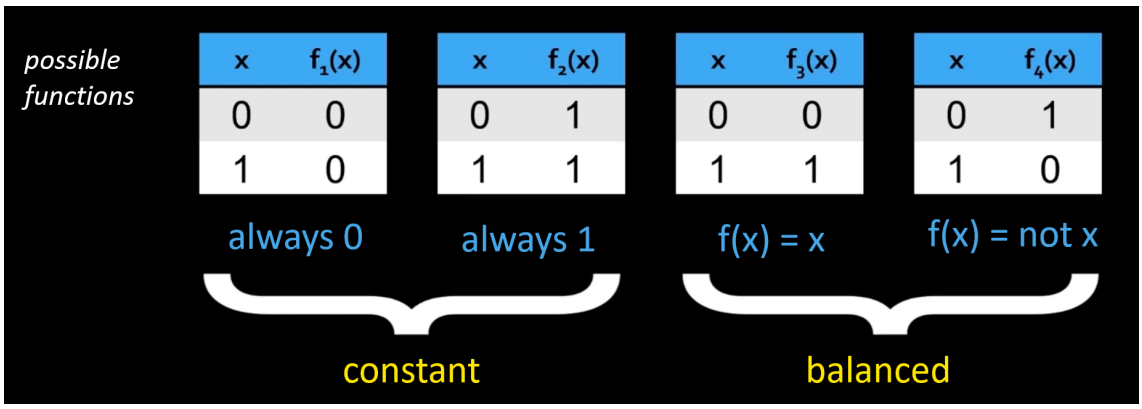
6.1.1. Problem

A first idea of a quantum algorithm is the Deutsch[1] algorithm. The algorithm is very simple: lets assume we are given a (binary) function with *one* input and *one* output (one cannot imagine a more simple system). We don't know anything about the function, and now we have to answer the following question: is the function *constant* or *balanced*? Constant means that the function always produces the same output, regardless of the input value. Balanced, on the other hand, means that function produces a different values if the input changes.

Since we have only 2 different inputs (0 and 1), there are four different functions possible:

- The function output is always 0 (the function is constant).
- The function output is always 1 (that would be constant, too).
- The identity function just produces the input value as output. This is a balanced function, since f(0)=0 while f(1)=1
- the inverse function is also balanced: f(1)=0 and f(0)=1

The following figure shows the four possible functions:



[1] David Deutsch (born on May, 18th, 1953 in Haifa) is a british physicist and professor at the university of Oxford

6.1.2. Classical solution

How can we find out whether the function is constant or balanced? Well, we need to evaluate the function and find it out! So, we might call the function $f(0)$ with input 0 and look at the output; for example, we might get a $f(0) = 0$ as the result. Is it constant or balanced? Well, we cannot decide this yet without , we need a second function call:

- if $f(1) = 0$, then the function is constant,
- but if we get $f(1) = 1$, then the function is balanced.

Can we do this with just one function call? No, this is not possible. We need *two* function calls to decide whether the function is constant or balanced.

6.1.3. Deutsch algorithm

Well, this is all quite trivial and boooring... until David Deutsch[2] found a solution that works with *one* function call! This was the first proof that with quantum computing some problems can be solved faster than with classical computing.

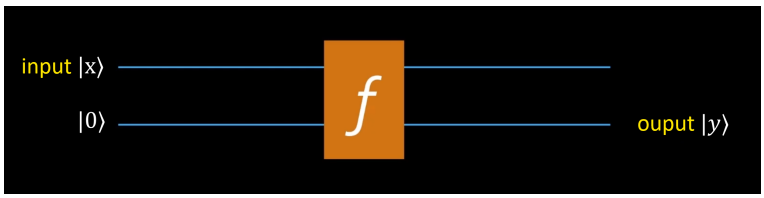
How can we implement the algorithm in a quantum computer?

As we have learned before, every quantum circuit must be reversible (since it is all about unitary functions). However, the constant functions are clearly not reversible! How can we implement them in a quantum circuit?

Well, we have a trick to solve this problem. We add another qubit to the circuit! In the following solution, we have two qubits:

- One qubit is the input to our function, but stays unchanged in the circuit. This makes our function reversible, since the output allows to reconstruct the input.
- Another qubit with the initial value of zero then contains the output of our function.

The function will look like:



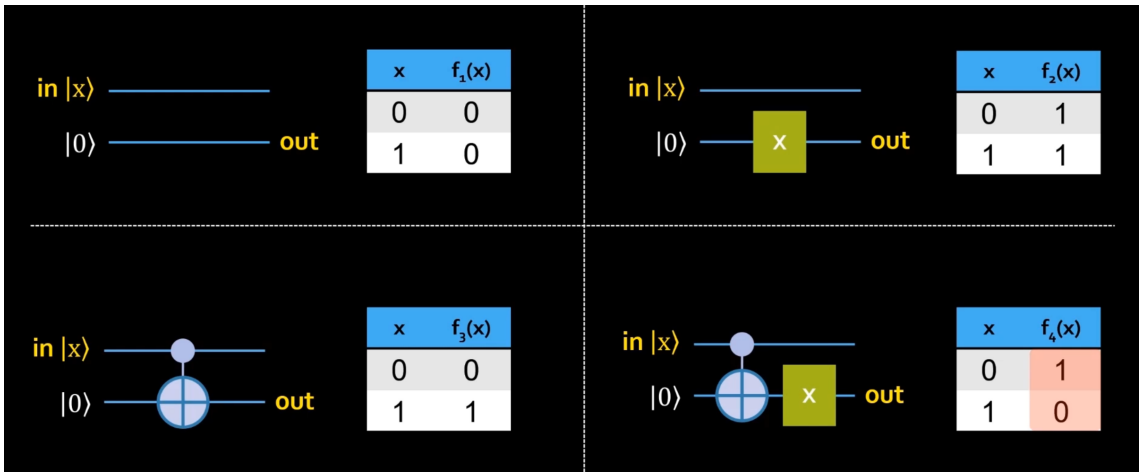
What is the function implementation?

- For the constant function $f(x) = 0$ this is trivial: we just copy the $|0\rangle$ to the output

[2]Deutsch, D. 1985 Proc. R. Soc. Lond. A 400, 97-117.

- The other constant function $f(x) = 1$ is also trivial: we invert $|0\rangle$, and we are done
- The identity function $f(x) = x$ requires to copy the input value $|x\rangle$ to the output. This can be achieved with a CNOT operation: if the input is zero, the output will not change, while for an input of one, the lower qubit will be flipped and produce a one as output
- The other balanced function $f(x) = 1 - x$ can be constructed by using a CNOT followed by a negation of the output.

The following figure shows the quantum implementation of the four functions:



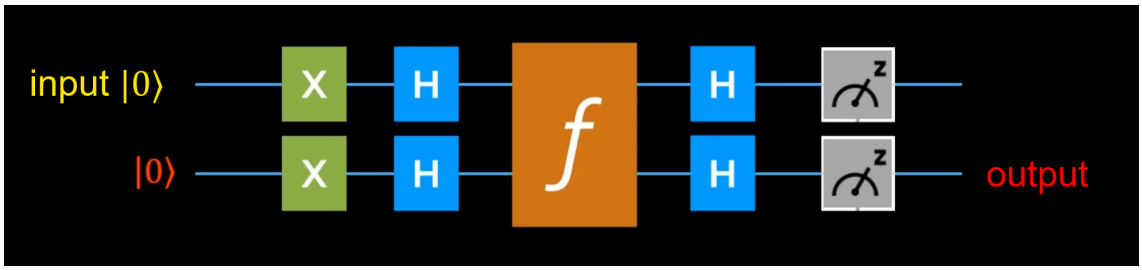
6.1.3.1. How can we detect a balanced function with just one function call?

So, here is our quantum algorithm for the examination of our function:

- First, we initialize both qubits with one (by inverting the $|0\rangle$ states).
- Then, we apply a Hadamard to both qubits, putting each on in the superposition state $\frac{1}{2}(|0\rangle + |1\rangle)$. Note that this trick is used in many quantum algorithms: superposition is the superpower of quantum computing!
- These superposition values are then feed into the function under examination.
- Another Hadamard is applied to the output, somehow reversing the first Hadamard.
- Then both qubits are measured (acutally, we would only need the upper one)

The measured value of this circuit will either be $|01\rangle$ or $|11\rangle$ (Qiskit notation[3]). In the first case, the function is balanced, in the second case, the function is constant! Well done, Mr. Deutsch!

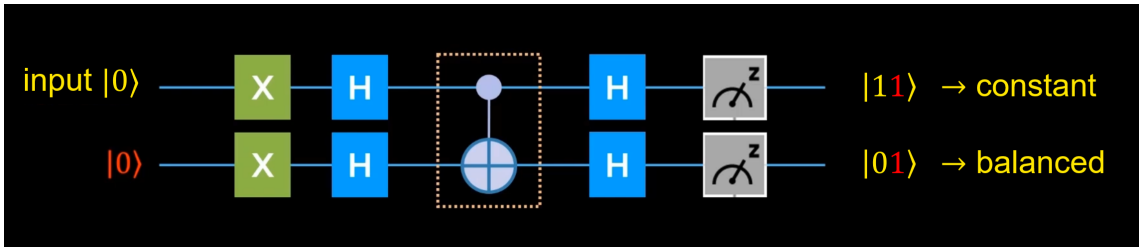
[3] Left qubit in the ket is upper qubit in the circuit



6.1.3.2. But how does it work?

For the identity function, we can immediately see that the output will be $|11\rangle$, since the Hadamard is its own inverse. But what about the other functions? How can we determine their output?

To calculate the output, we can set up the corresponding mathematical description of each circuit. Lets take one of the balanced functions:



In this circuit, we have:

- A 2-qubit register as input, which corresponds to a 4-dimensional vector, with the initial value:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- The resulting operation of applying Pauli-X on each qubit is the tensor product of two Pauli-X:

$$X \otimes X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- Applying a Hadamard on each qubit is the tensor product of two Hadamard:

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

- We know the CNOT already:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

When compining all steps of the algorithm into one mathematical representation, we have to reverse the order: while the circuit goes from left to right, the matrix equation goes from right to left. And we do process the matrix from right to left:

$$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} =$$

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} =$$

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

This function is balanced.

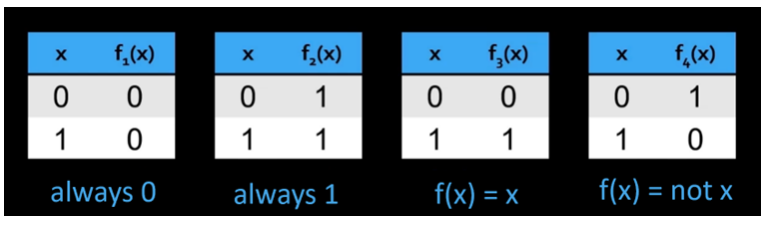
Exercise:

Repeat the same calculation for the three other functions and determine the output!

6.1.4. Is this magic?

Wow, we did one test and where able to decide which functions were balanced and which were constant. This is not possible with a classical computer! This is fantastic!

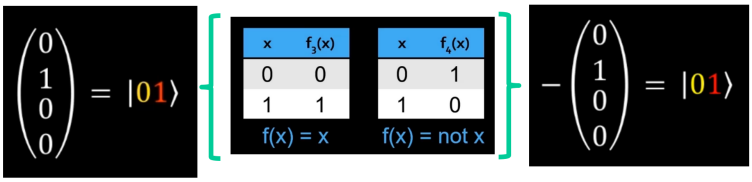
But still, with *one* test we can only discriminate between *two* options. Here are our four possible functions:



With a classical computer, we can for example evaluate an unknown function f with parameter $f(0)$. If the output is 0, then it is either function f_1 or f_3, otherwise f_2 or f_4. With the Deutsch algorithm, we can detect whether the function is constant (f_1 or f_2) or balanced (f_3 or f_4), which is not possible with a classical computer. But even with a quantum computer we need two tests to identify the function exactly!

But can't we discriminate between f_3 and f_4?

When doing the calculation, isn't there a minus sign in the result? Well, yes, there is a minus sign:



The quantum states have a different global phase, namely +1 and -1. But this does not help us: the global phase of a quantum state has no physical meaning, and we cannot discrimate two states which differ only in the global phase. Physically, they are undistinguishable!

6.2. Deutsch-Jozsa Algorithm

Please see the section 9.8 in the book *Dancing with Qubits*[6], also available in Moodle.

6.3. Grover Algorithm

t.b.d.

7. Quantum Communication

7.1. Quantum Teleportation

8. Quantum Cryptography

8.1. What is cryptography?

8.2. Classical cryptographic key exchange: Diffie-Hellmann

8.3. Quantum Key Exchange: BB84

9. Algorithms (Part 2)

9.1. Shor Algorithm

t.b.d.

10. Algorithm Complexity

t.b.d.

11. Error Correction

11.1. Classic Error Correction

11.2. Example: A 3-Bit Code

11.3. Shor 9-Bit Code

11.4. Surface Codes

12. Quantum Hardware

12.1. Further Hardware Technologies

The field of quantum hardware is evolving fast. Besides the traditional techniques, like the transmon (e.g. IBM, Google), cold atoms or photons (e.g. ID Quantique) much research is done on new technologies. Due to the promising perspectives of quantum computing (aka the hype around QC), there has been huge amounts of money invested in this field, and there are many start-ups around trying to develop - or participate - in the next big quantum computing evolutin.

Here is a list of recent technologies for might play a role in the future:

- Six qubit quantum processor, based on quantum dots in silicon[4]. Advantage: decades of experience of producing silicon chips, fabs are in place, can be integrated with normal processors. Temperature: ? Qubit Lifetime: ?

A. About the Bell Inequality and the Nobel Prize in Physics in 2022

The most important element of quantum mechanics is entanglement of quantum objects, which is also the most "strange" one in our classical world view.

In 1935, Einstein, Podolsky and Rosen published a paper on what they described as "paradoxon" of quantum mechanics, the famous EPR paradox[2]. They described the following *Gedankenexperiment*:

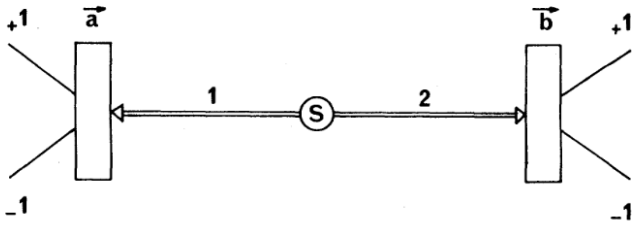


Figure A.1.: Setup of EPR experiment [2]. Physicists use +1/-1 instead of 0 and 1

A source creates an entangled pair of quantum particles, e.g. a Bell pair

$$|\Phi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

One particle is sent to Alice, the other to Bob. Both perform a measurement on their particle at exactly the same point in time (or as close as possible, since time might be relative). The outcome will be random: Alice measures a zero or a one with equal probability; the same is true for Bob. However, when their later compare their results, each measurement result of Alice is exactly the opposite of Bob: if Alice measured a 0, then Bob got a 1, and vice versa. As we already discussed, this is due to the entanglement that creates a correlation between Alice and Bobs measurement.

EPR now argued that this cannot be the full picture, and argued that quantum mechanics is incomplete. One problematic aspect is that the measurement of Alice and Bob are performed at different places, but at the same time. How could it be that the result is correlated? How does the mechanism work that produces a 1 at Alice measurement and a 0 at Bobs? A communication mechanism is not possible, since this contradicts the theory of special relativity, which states that nothing - not

even information - can travel faster than light. How do the particles synchronize? The problem here is the *non-locality* of the effect.

An other aspect is *realism*: a measurement should give us information about the world. If we measure the length of an object, we acquire knowledge about this object, namely how long it is. What do we measure in the above EPR setup? Alice and Bob get a random answer. If we measure the correlation (0 on one side, 1 on the other), then what does it tell us about the physical world?

One proposed solution to this problem were *hidden variables*. The idea is as follows: when creating the pair, some information is stored inside the quantum object (information that we cannot see). For example, one particle stores a 1, the other a 0. Then, when the measurement is performed, it merely shows the already predefined measurement.

A.1. Bell Inequality

Many people worked on this problem; one of them was John Steward Bell. In 1964, he published a paper with the title ON THE EINSTEIN PODOLSKY ROSEN PARADOX[1]. In this paper, he basically showed that in a certain setup, *any* hidden variable theory that relies on a classical world view produces a different result as quantum mechanics.

His argumentation is as follows[1]. The setup is close to the EPR setup. However, the measurement is done in two different setups, e.g. in Z-direction or in X-direction. For photons, this would for example be a horizontal or vertical polarization. Alice and Bob decide randomly, which setup they use for one measurement. Ideally, the choice is done *after* the particles have been created.

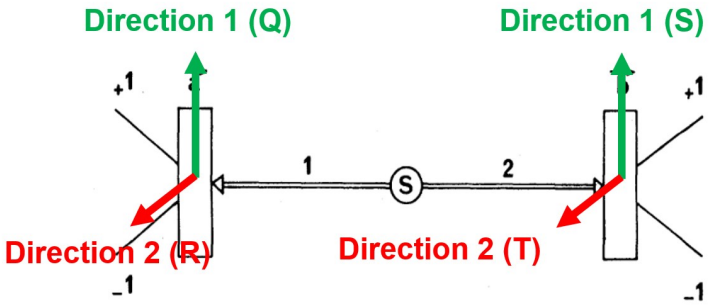


Figure A.2.: Setup of Bell experiment.

In the Bell experiment, we have 4 different setups:

- Alice uses Q setup and Bob S
- Alice uses Q setup and Bob T

[1]Here I follow the discussion in [3]

- Alice uses R setup and Bob S
- Alice uses R setup and Bob T

Each result can be either -1 or +1, see figure A.3.

Figure A.3.: Setup of Bell experiment[3].

Now, we assume that the particles are prepared for measurement at the time of their creation. This means, the values for Q, R, S, T are somehow fixed. Let $P(q, r, s, t)$ be the probability that the system is in the state $Q = q, R = r, S = s, T = t$. For example, $P(+1, +1, -1, -1)$ could be 0.125 (we don't care about the value, the argumentation is valid for all possible scenarios).

Bell now used the following term for his argumentation

$$E_{Bell} = E(QS) + E(RS) + E(RT) - E(QT)$$

and he showed that in a classical setup, this expectation value $E_{Bell} \leq 2$. However, we will see later that in a quantum setup, $E_{Bell} = 2\sqrt{2}$ is possible.

A.1.1. Classical Scenario

In the first step, the entangled pair is created. The hidden variable theory assumes that the values for each possible measurement are already fixed at this point. So lets assume that the probabilities are fixed.

Each measurement yields either $+1$ or -1. We know that

$$QS + RS + RT - QT = (Q + R) \cdot S + (R - Q) \cdot T$$

and since Q and R are either $+1$ or -1, one of $Q + R$ or $Q - R$ is zero, while the other is ± 2. So, we have

$$\begin{aligned} E(QS + RS + RT - QT) &= \sum_{q,r,s,t} P(q, r, s, t)(qs + rs + rt - qt) \\ &\leq \sum_{q,r,s,t} P(q, r, s, t) \cdot 2 \\ &= 2 \end{aligned}$$

since the sum of the probabilities for all possible cases q, r, s, t is one.

On the other hand,

$$\begin{aligned} E(QS + RS + RT - QT) &= \sum_{q,r,s,t} P(q,r,s,t)(qs + rs + rt - qt) \\ &= \sum_{q,r,s,t} P(q,r,s,t)qs + \sum_{q,r,s,t} P(q,r,s,t)rs + \\ &+ \sum_{q,r,s,t} P(q,r,s,t)rt - \sum_{q,r,s,t} P(q,r,s,t)qt \\ &= E(QS) + E(RS) + E(RT) - E(QT) \end{aligned}$$

which gives us the Bell inequality

$$E(QS) + E(RS) + E(RT) - E(QT) \leq 2$$

A.1.2. Quantum Scenario

However, for a quantum system this is different due to entanglement. Bell choose

$$|\psi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

as the entangled pair. The first qubit is sent to Alice, the second to Bob. The perform the following measurements on their qubits, using the following observables (i.e. using this transformation into the orthonormal basis):

$$\begin{aligned} Q &= Z_1 \qquad S = \frac{-Z_2 - X_2}{\sqrt{2}} \\ R &= X_1 \qquad T = \frac{Z_2 - X_2}{\sqrt{2}} \end{aligned}$$

What are the expectation values for these measurements, i.e. E(QS), E(RS), E(RT) and E(QT)? How can we calculate the expectation value? Well, we need a formula here, which we didn't cover in the basic section of our class, so here it is:

The expectation value of a quantum state ϕ with respect to a measurement operator M can be calculated as follows:

$$E(|\phi\rangle) = \langle\phi| M |\phi\rangle$$

where $\langle\phi|$ is the conjugate transposed $|\phi\rangle$.

Putting the operators in matrix form we get:

$$\begin{aligned} Q = Z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ R = X &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ S &= \frac{-Z - X}{\sqrt{2}} = \frac{1}{\sqrt{2}} \left[- \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right] = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix} \\ T &= \frac{Z - X}{\sqrt{2}} = \frac{1}{\sqrt{2}} \left[\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix} \end{aligned}$$

The tensor product of QS is then

$$\begin{aligned} Q \otimes S &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{aligned}$$

Now we can start to calculate the average:

$$\begin{aligned} E(QS) &= \langle \phi | \, Q \otimes S \, | \phi \rangle \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} \\ &= \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \\ &= \frac{1}{2\sqrt{2}} \cdot 2 = \frac{1}{\sqrt{2}} \end{aligned}$$

Similarly, we get $E(RS) = E(RT) = \frac{1}{\sqrt{2}}$. For $E(QT)$, we get

$$\begin{aligned} E(QT) &= \langle \phi | \, Q \otimes T \, | \phi \rangle \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} \\ &= \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \\ &= \frac{1}{2\sqrt{2}} \cdot -2 = -\frac{1}{\sqrt{2}} \end{aligned}$$

Putting it all together, we get

$$E(QS) + E(RS) + E(RT) - E(QT) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} - (-\frac{1}{\sqrt{2}}) = 2\sqrt{2}$$

This is in contradiction to the above Bell inequality. Thus, a quantum system cannot be modeled with classical probability like proposed by the hidden variable model.

A.2. Nobel Prize 2022

- John Clauser was able to measure the Bell inequality in experiment. However, in these first experiments the random selection of the setting (Q or R, S or T) took place every time before creating the entangled pair.
- Alain Aspect further refined the measurement by doing the random selection *after* creation of the entangled pair.
- Anton Zeilinger further refined the quantum experiments and developed new techniques, like the quantum entangler which is required for large distance quantum networks.

B. Answers

Answers to 2.1.3:

a) $\bar{x} = 3 - 5i$, $\bar{y} = 2 - 4i$, $x + y = 5 + i$, $x \cdot y = 26 - 2i$.

b) The polar coordinates are $x = (5.831, 59.04)$ and $x = (4.472, 63.43)$

Answers to 2.4.3:

a) The result is a 6-dimensional vector.

b)

$$\mathbf{v} \otimes \mathbf{w} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 7 \\ 5 \end{pmatrix} = \begin{pmatrix} 6 \\ 14 \\ 10 \\ 9 \\ 21 \\ 15 \end{pmatrix}$$

c)

$$\mathbf{v} \otimes \mathbf{w} = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 11 \\ 3 \end{pmatrix} = \begin{pmatrix} 8 \\ 44 \\ 12 \\ 6 \\ 33 \\ 9 \end{pmatrix}$$

d) Yes, this is possible. We could have for example a factor $\lambda \in \mathbb{C}/0$ with $c' = (\lambda a) \otimes b = a \otimes (\lambda b)$, where $\lambda a \neq a$ and $\lambda b \neq b$.

e) There exist no such vectors, x is not the result of a tensor product! This is quite important for later, since quantum state that *cannot* be represented as the tensor product are in fact *entangled*.

Bibliography

[1] John S Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[2] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

[3] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[4] Stephan G. J. Philips, Mateusz T. M?dzik, Sergey V. Amitonov, Sander L. de Snoo, Maximilian Russ, Nima Kalhor, Christian Volk, William I. L. Lawrie, Delphine Brousse, Larysa Tryputen, Brian Paquelet Wuetz, Amir Sammak, Menno Veldhorst, Giordano Scappucci, and Lieven M. K. Vandersypen. Universal control of a six-qubit quantum processor in silicon. *Nature*, 609(7929):919–924, September 2022.

[5] Smite-Meister. Bloch sphere, 2022. Own work, CC BY-SA 3.0.

[6] Robert S Sutor. *Dancing with Qubits: How quantum computing works and how it can change the world*. Packt Publishing Ltd, 2019.

[7] Noson S. Yanofsky and Mirco Mannucci. *Quantum Computing for Computer Scientists*. 2008.