Noise in quantum and classical computation & non-locality
Unger, F.P.

Citation for published version (APA):
Chapter 2

Preliminaries

In this chapter we will discuss some general techniques and results which will be needed later. In later chapters we will provide pointers to this chapter whenever needed.

Section 2.1 is intended to set up our linear algebra notation, and in particular introduce the famous Dirac notation. Some more facts and notation can be found in Appendix A but it is very concise and mostly intended as a reminder. For an introduction to linear algebra the reader can consult for example [55].

In Section 2.2 we will summarize all prerequisites from quantum mechanics necessary to understand this thesis.

In Sections 2.3 and 2.4 we will give formal definitions of some notions in computational complexity theory and communication complexity theory, although an intuitive understanding of these concepts will be sufficient to read this thesis. Computational complexity theory will be used to partially motivate the results in Chapter 7 and all the reader needs to know about communication complexity is that there are functions which have high communication complexity.

Section 2.5 from this chapter will summarize facts about the Bloch sphere, which will be a convenient tool in Chapters 4 and 5 for characterizing 1-qubit operations.

Section 2.6 discusses Semidefinite Programming, which will be needed in Chapter 7 to characterize XOR games.

2.1 Linear algebra notation

Hilbert space  We denote the $d$-dimensional complex vector space by $\mathbb{C}^d$. It consists of all column vectors with $d$ complex entries. For vectors elements $\phi, \psi \in \mathbb{C}^d$, we have

\[
\langle \phi | \psi \rangle = \sum_{i=1}^{d} \phi_i \overline{\psi_i}
\]

where $\overline{\psi_i}$ denotes the complex conjugate of $\psi_i$. The inner product is a complex number which measures the overlap between the two vectors. It is symmetric, conjugate linear in the first argument, and linear in the second argument.

Inner product space  An inner product space is a vector space equipped with an inner product: A function $\langle \cdot | \cdot \rangle : \mathbb{C}^d \times \mathbb{C}^d \rightarrow \mathbb{C}$ that is conjugate linear in the first argument and linear in the second argument. It must also satisfy the following properties:

1. Positive definiteness: $\langle \phi | \phi \rangle > 0$ for all non-zero vectors $\phi$.
2. Conjugate linearity in the first argument: $\langle \lambda \phi + \mu \psi | \phi \rangle = \lambda^* \langle \phi | \phi \rangle + \mu \langle \psi | \psi \rangle$ for all vectors $\phi, \psi$ and scalars $\lambda, \mu$.
3. Linearity in the second argument: $\langle \phi | \lambda \psi + \mu \psi \rangle = \lambda \langle \phi | \psi \rangle + \mu \langle \phi | \psi \rangle$ for all vectors $\phi, \psi$ and scalars $\lambda, \mu$.
4. Conjugate symmetry: $\langle \phi | \psi \rangle = \langle \psi | \phi \rangle^*$.

The inner product space is called a Hilbert space if it is complete, i.e., every Cauchy sequence converges to a limit in the space. In this case, the norm of a vector $\phi$ is defined as $\|\phi\| = \sqrt{\langle \phi | \phi \rangle}$.

Hilbert space  We denote the $d$-dimensional complex vector space by $\mathbb{C}^d$. It consists of all column vectors with $d$ complex entries. For vectors elements $\phi, \psi \in \mathbb{C}^d$, we have

\[
\langle \phi | \psi \rangle = \sum_{i=1}^{d} \phi_i \overline{\psi_i}
\]

where $\overline{\psi_i}$ denotes the complex conjugate of $\psi_i$. The inner product is a complex number which measures the overlap between the two vectors. It is symmetric, conjugate linear in the first argument, and linear in the second argument.

Inner product space  An inner product space is a vector space equipped with an inner product: A function $\langle \cdot | \cdot \rangle : \mathbb{C}^d \times \mathbb{C}^d \rightarrow \mathbb{C}$ that is conjugate linear in the first argument and linear in the second argument. It must also satisfy the following properties:

1. Positive definiteness: $\langle \phi | \phi \rangle > 0$ for all non-zero vectors $\phi$.
2. Conjugate linearity in the first argument: $\langle \lambda \phi + \mu \psi | \phi \rangle = \lambda^* \langle \phi | \phi \rangle + \mu \langle \psi | \psi \rangle$ for all vectors $\phi, \psi$ and scalars $\lambda, \mu$.
3. Linearity in the second argument: $\langle \phi | \lambda \psi + \mu \psi \rangle = \lambda \langle \phi | \psi \rangle + \mu \langle \phi | \psi \rangle$ for all vectors $\phi, \psi$ and scalars $\lambda, \mu$.
4. Conjugate symmetry: $\langle \phi | \psi \rangle = \langle \psi | \phi \rangle^*$.

The inner product space is called a Hilbert space if it is complete, i.e., every Cauchy sequence converges to a limit in the space. In this case, the norm of a vector $\phi$ is defined as $\|\phi\| = \sqrt{\langle \phi | \phi \rangle}$.
\[ \phi = \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{d-1} \end{pmatrix}, \quad \psi = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{d-1} \end{pmatrix} \]

we can define a complex inner product by

\[ \langle \phi, \psi \rangle = \sum_{i=0}^{d-1} \phi_i^* \psi_i, \]

where \( \phi_i^* \) is the complex conjugate of \( \phi_i \). This makes \( \mathbb{C}^d \) a complex Hilbert space, since \( d \) is finite. From this inner product we define the standard norm (or Euclidean norm) as

\[ ||\phi|| = \sqrt{\langle \phi, \phi \rangle}. \]

**Dirac notation** It will be convenient to write vectors \( \phi \in \mathbb{C}^d \) in Dirac notation by sandwiching \( \phi \) in between “|” and “⟩” and write \( |\phi\rangle \) instead of \( \phi \). We will denote the conjugate transpose of a vector \( |\phi\rangle \) by \( \langle \phi | \), which is the row vector with entries

\[ \langle \phi | = (\phi_0^*, \phi_1^*, \ldots, \phi_{d-1}^*) . \]

With this notation the inner product \( \langle \phi, \psi \rangle \) can be simply written as the matrix product of \( \langle \phi | \) and \( |\psi\rangle \) as

\[ \langle \phi, \psi \rangle = \langle \phi | \langle \psi \rangle. \]

**Standard basis** It is easy to see that the vectors

\[ |0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \quad |d-1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \]

are linearly independent and orthonormal (since \( \langle i | j \rangle = 0 \) if \( i \neq j \) and 1 otherwise) with respect to our inner product. We therefore call \( \{|0\rangle, |1\rangle, \ldots, |d-1\rangle\} \) the standard basis or computational basis.

**Matrices** We let \( \mathbb{C}^{d \times d} \) be the \( d^2 \)-dimensional complex vector space of all \( d \times d \) matrices with complex entries. For a matrix \( A \in \mathbb{C}^{d \times d} \) we let \( A_{ij} \) be the entry of \( A \) in the \( i \)-th row and \( j \)-th column. The identity matrix is denoted by \( \mathbb{I}_d \). For a matrix \( A \in \mathbb{C}^{d \times d} \) we let \( A^T \) be the transpose of \( A \), which means that \( (A^T)_{ij} = A_{ji} \). Similarly, \( A^\dagger \) denotes the conjugate transpose and has entries \( (A^\dagger)_{ij} = (A_{ji})^* \). We can also equip \( \mathbb{C}^{d \times d} \) with an inner product by defining for \( A, B \in \mathbb{C}^{d \times d} \)

\[ \langle A, B \rangle = \text{Tr}(A^\dagger B) = \sum_{i,j=1}^{d} A_{ij}^* B_{ij}. \]
This inner product is called the Hilbert-Schmidt inner product. 

$E_{ij}$ is the all-zero matrix, except for the entry $i, j$ which is equal to 1. Evidently, the $E_{ij}$ form an orthonormal basis for $\mathbb{C}^{d \times d}$ with respect to our inner product. Later in Chapter 4 we will present a different orthonormal basis (tensor products of Pauli matrices), which is in certain applications more natural.

**Positive semidefinite** A hermitian matrix $A \in \mathbb{C}^{d \times d}$ is called positive semidefinite if for all $\psi \in \mathbb{C}^d$ it holds

$$\langle \psi | A | \psi \rangle \geq 0. \quad (2.1)$$

We also write $A \succeq 0$ for “$A$ is positive semidefinite”. We write $A \succeq B$ if $A - B \succeq 0$. Further, we write $A \succ 0$ if the above inequality is strict. We then say that $A$ is positive definite.

### 2.2 Quantum states, operations and computation

**Quantum states** We model a closed physical system (of finite dimension $d$) abstractly by attaching to it the $d$-dimensional Hilbert space $\mathcal{H}_d = \mathbb{C}^d$. The set of all linear operators $\rho$ mapping $\mathcal{H}_d$ into itself is written $\mathbb{B}(\mathcal{H}_d)$. It is isomorphic to $\mathbb{C}^{d \times d}$, the set of all $d \times d$-matrices. We assume that linear operators are always represented in matrix form. The set of possible states the system can be in is the set of all bounded linear operators $\rho \in \mathbb{B}(\mathcal{H})$ which are normalized (i.e. $\text{Tr}(\rho) = 1$) and positive semidefinite $\rho \succeq 0$ (and therefore by definition also hermitian). Any such $\rho$ we call a density matrix. For convenience we often do not make a distinction between the physical system and the Hilbert space associated to it and speak of “the physical system $\mathcal{H}_d$”.

**Measurements** In order to gain some information about the state of a system one can perform a measurement. A measurement on system $\mathcal{H}$ is given by a set of measurement operators $\mathcal{M} = \{M_1, \ldots, M_m\} \subset \mathbb{B}(\mathcal{H})$, which have the property that $\sum_i M_i^\dagger M_i = I_d$. The outcome of the measurement $\mathcal{M}$ is one of the labels $1, \ldots, m$. If the system is in state $\rho$ before the measurement then outcome $1 \leq i \leq m$ occurs with probability

$$p_i = \text{Tr}(M_i^\dagger M_i \rho)$$

and if outcome $i$ occurred, the new state after the measurement is

$$\rho_i = \frac{M_i \rho M_i^\dagger}{\text{Tr}(M_i^\dagger M_i \rho)}.$$
Note that in general a measurement is a non-reversible operation, i.e. it is in general impossible to “undo” the measurement and obtain the premeasurement state.

**General evolution**  The most general way to change the state of a quantum system is by applying a *completely positive trace-preserving map*, short CPTP map. They can change the dimensionality of a system, though we will be mostly concerned with operations that preserve dimensionality. A CPTP map $\mathcal{E}$ mapping a $d$-dimensional system into a $d'$-dimensional system is given by Kraus operators $\{E_1, \ldots, E_m\} \subset \mathbb{C}^{d' \times d}$ with the property that

$$\sum_i E_i^\dagger E_i = \mathbb{I}.$$  

Then $\mathcal{E}$ changes the density matrix $\rho$ into

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger.$$  

A special case is when $\mathcal{E}$ is given by one Kraus operator $E_1 = U$ only, which necessarily has to be unitary. We then say that $\rho$ evolves (under unitary evolution) into the state

$$\rho \rightarrow U \rho U^\dagger.$$  

Obviously, *unitary operations* are reversible.

**Pure states**  A state $\rho$ is called *pure* if $\text{rank}(\rho) = 1$, which is equivalent to saying that there is some $\phi \in \mathbb{C}^d$ with the property that $\rho = |\phi\rangle\langle \phi|$. States which are not pure are called *mixed*. Often the short-hand $|\phi\rangle$ is used for $|\phi\rangle\langle \phi|$.  

For every density matrix $\rho$ it is possible to find normalized vectors $|\phi_i\rangle \in \mathbb{C}^d$ and non-negative real numbers $p_i$ with $\sum_i p_i = 1$ with

$$\rho = \sum_{i=1}^d p_i |\phi_i\rangle\langle \phi_i|.$$  

(2.2)  

For example, since $\rho$ is hermitian and positive semidefinite it can be unitarily diagonalized, and thus it is possible to choose $p_i$ and $|\phi_i\rangle$ to be the eigenvalues respectively eigenvectors of $\rho$. We call $\{p_i, |\phi_i\rangle\langle \phi_i|\}$ an *ensemble* for $\rho$. However, in general the choice of an ensemble for a state is not unique.

It is straightforward to see that a quantum system state which is in a probabilistic mixture of states $|\phi_i\rangle\langle \phi_i| \in \mathbb{C}^{d \times d}$, each occurring with probability $p_i$, has the same measurement statistics as the state $\rho = \sum_{i=1}^d p_i |\phi_i\rangle\langle \phi_i|$ for any measurement. This means that it is impossible to distinguish the ensemble $\{p_i, |\phi_i\rangle\langle \phi_i|\}$ from the state $\rho$ by observing the system. (Note that this remains true even after
applying arbitrary quantum operations/measurements on the system.) Hence, the description of the state of a system in terms of ensembles and the description in terms of density matrices are absolutely equivalent from a physics point of view.

Note that the (pure) states $|\phi_i\rangle$ of an ensemble for $\rho$ are elements of the $d$-dimensional Hilbert space $\mathbb{C}^d$ only. Since these $d$ dimensions are enough to describe the pure states of the physical system and as we have seen we may interpret every state as a probabilistic mixture of pure states, the system is called $d$-dimensional. The reason for describing states in the larger Hilbert space of hermitian matrices in $\mathbb{C}^{d \times d}$ is that it is more suitable when using general CPTP maps as defined above.

Subsystems and entanglement If two physical systems are represented by Hilbert spaces $\mathcal{A} = \mathbb{C}^a$ respectively $\mathcal{B} = \mathbb{C}^b$, then the joint system is represented by the Hilbert space $\mathcal{A} \otimes \mathcal{B} := \mathbb{C}^{ab}$. Its inner product is defined as above. It holds $\mathbb{B}(\mathcal{A} \otimes \mathcal{B}) = \mathbb{B}(\mathcal{A}) \otimes \mathbb{B}(\mathcal{B})$.

The system $\mathcal{A} \otimes \mathcal{B}$ can be in a state $C \in \mathbb{B}(\mathcal{A}) \otimes \mathbb{B}(\mathcal{B})$ for which there are no $A_i \in \mathbb{B}(\mathcal{A})$, $B_i \in \mathbb{B}(\mathcal{B})$ such that $C = \sum_i A_i \otimes B_i$. We then say that $C$ is entangled.

We say that a measurement $\mathcal{M} = \{M_1, \ldots, M_m\} \subset \mathbb{B}(\mathcal{A} \otimes \mathcal{B})$ acts only on subsystem $\mathcal{A}$ if there are $M'_1, \ldots, M'_m \subset \mathbb{B}(\mathcal{A})$ such that for all $1 \leq i \leq m$ : $M_i = M'_i \otimes I_B$, where $I_B$ is the identity operator on $\mathcal{B}$. Similarly, we say that a CPTP map $\mathcal{E}$, given by Kraus operators $\{E_1, \ldots, E_m\} \subset \mathbb{B}(\mathcal{A} \otimes \mathcal{B})$, acts on subsystem $\mathcal{A}$ only if there are $E'_1, \ldots, E'_m \subset \mathbb{B}(\mathcal{A})$ such that for all $0 \leq i \leq m$ : $E_i = E'_i \otimes I_B$.

Qubits We say that the Hilbert space $\mathcal{H}_2 = \mathbb{C}^2$ represents one qubit and generally $\mathcal{H}_d = \mathbb{C}^d$ represents one qudit. The space $\mathcal{H}_2^{\otimes n}$ represents $n$ qubits.

2.2.1 Quantum circuits and quantum computation

We will only define quantum circuits on qubits, as this is all we will need later. It is straightforward to generalize this to arbitrary $d$-dimensional qudits.

Quantum circuits A quantum circuit on $n$ qubits and of depth $T$ consists of a set of (quantum) gates $\mathcal{G}_i$, each of which contains 3 parameters: $W_i \subseteq \{1, \ldots, n\}$ (the qubits $\mathcal{G}_i$ acts on), an integer number $1 \leq t_i \leq T$ (the execution time) and a quantum operation $\mathcal{E}_i$ which acts on $|W_i|$ qubits. Further, we require that for each $1 \leq t \leq T$ and every $1 \leq j \leq n$ there is exactly one $i$ with $j \in W_i$ and $t_i = t$. In other words for each qubit and each time there is exactly one gate which acts upon this qubit (which may be the identity gate). The number of qubits $|W_i|$ is called the fan-in of gate $\mathcal{G}_i$. 
Evolution We fix a unique total order $\prec$ on the set of gates by defining that

$$G_i \prec G_j \iff (t_i < t_j) \text{ or } (t_i = t_j \text{ and } \min W_i < \min W_j).$$

Without loss of generality we let the gates be numbered from 1 to $S$ and we assume that this numbering is consistent with the ordering $\prec$, i.e.,

$$G_i \prec G_j \iff i < j.$$

The computation of a quantum circuit on input $\rho \in H_2^\otimes n$ is inductively defined through the following set of quantum states $\rho_i$

1. $\rho_0 = \rho$
2. If $i < S$ then $\rho_{i+1} = \mathcal{E}'_{i+1}(\rho_i)$.

If $\mathcal{E}_i$ has Kraus operators $E_j$, we let $\mathcal{E}'_i$ be the quantum operation with Kraus operators $E_j \otimes I \otimes \{1, \ldots, n\} \setminus W_i$ where the $E_j$ act on the Hilbert space of the qubits $\{1, \ldots, n\} \setminus W_i$. We define the output of the computation to be $\rho_S$.

If $x \in \{0, 1\}^n$ we denote by $\rho_x \in \mathbb{C}^{2^n \times 2^n}$ the output of the computation with input $\ket{\text{bin}(x)}$, where $\text{bin}(x)$ is the number with binary representation $x$ and $\ket{\text{bin}(x)}$ is short for $\ket{\text{bin}(x)}\bra{\text{bin}(x)}$, as defined above. We say that $\rho_x$ is the output of the computation with classical input $x$.

Computation of a function Often it is necessary to give quantum circuits additional work space, e.g. to “store” intermediate results. To take care of that we pad inputs with additional qubits in state $\ket{0}$, which are input-independent. These qubits are called ancilla qubits. The precise definition is as follows.

We say that a quantum circuit on $m$ qubits computes a boolean function $f : \{0, 1\}^n \mapsto \{0, 1\}$, $n \leq m$, with error $0 \leq \epsilon < \frac{1}{2}$ using $m-n$ ancillas, if there is some measurement $\mathcal{M} = \{M_0, M_1\}$ such that for every $x \in \{0, 1\}^n$ the outcome of $\mathcal{M}$ on $\rho_{2^n-m}$ is equal to $f(x)$ with probability at least $1-\epsilon$. Formally,

$$\forall x \in \{0, 1\}^m : \text{Tr}(M^\dagger f(x)M f(x)\rho_{2^n-m}) \geq 1 - \epsilon.$$

We say that $\mathcal{M}$ measures the first qubit if $M_i = M_i^0 \otimes \mathbb{I}_{2^m-1}$, for some measurement operators $\{M_i^0, M_i^1\}$ on the first qubit.

Efficient quantum computation A family of boolean functions $f_1, f_2, f_3, \ldots$ with $f_n : \{0, 1\}^n \mapsto \{0, 1\}$ is said to be uniformly computable in polynomial-time with bounded error by a quantum computer, if there is some polynomial $p(\cdot)$, $0 \leq \epsilon < 1/2$, $k > 0$ and a (classical) Turing machine $M$ with the property that

1. For each $n$ the Turing machine $M(n)$ runs in time at most $p(n)$. 
2.3. **Complexity classes**

2. The output \( M(n) \) is a description of a quantum circuit \( C_n \) which has \( m_n \) qubits and each gate has fan-in at most \( k \).

3. \( C_n \) computes \( f_n \) with error at most \( \epsilon \) using \( m_n - n \) ancillas and a measurement on the first qubit.

Note that \( n \leq m_n \leq p(n) \) and that the depth of \( C_n \) is at most \( p(n) \). The output \( M(i) \) (which is the description of \( C_i \)) should be a list of numbers specifying for each gate: \( W_i, t_i \) and numbers specifying the Kraus operators. Note that since the running time of \( M(n) \) is at most polynomial in \( n \), the number of gates and the number of ancilla qubits are also polynomially bounded.

At first sight it might look that changing the parameters \( \epsilon, k \) or the type of gates might change the computational power (i.e. set of functions that can be computed). However, this is not the case. Firstly, it is not hard to show [68] that every family of functions which can be computed with error \( 0 \leq \epsilon < 1/2 \), can also be computed with arbitrarily small error \( 0 < \epsilon' \), with small overhead in the size of the circuit. Secondly, it turns out that there is a finite set of unitary gates \( \mathcal{U} \) (with fan-in 2) such that every family of functions which can be efficiently computed in the above way can also be efficiently computed with circuits using only gates from \( \mathcal{U} \). However, the polynomial \( p(\cdot) \) might change. For example it is possible to choose \( \mathcal{U} \) to be the CNOT gate, the Hadamard gate and the \( \pi/8 \) gate [18], see (5.2) on page 63 and (5.19) on page 74 for definitions of this gate set.

This makes it possible to define the complexity class BQP without reference to a particular gate set and to fix \( 0 < \epsilon < 1/2 \) arbitrarily, see Section 2.3 later.

### 2.3 Complexity classes

One—or perhaps the most—important part of computer science is to determine how difficult computational problems are, in terms of the number of resources (e.g. time) needed to solve the problem. A thorough introduction can be found in [71].

Computational problems are usually phrased as membership problems (with binary yes/no answers), because every computational problem can be reduced to membership problems. A membership problem is modeled by a set \( A \subseteq \{0, 1\}^\ast \), a subset of all 0/1-strings. The task is to determine whether a given \( x \in \{0, 1\}^\ast \) is in \( A \) or not.

We say that the problem \( A \subseteq \{0, 1\}^\ast \) is **polynomial-time decidable** if there is a Turing machine \( M \) and a polynomial \( p \) such that for each input \( x \in \{0, 1\}^\ast \) the machine \( M \) stops after at most \( p(|x|) \) steps and outputs whether \( x \in A \) or not. Here \(|x|\) is the number of bits of \( x \). The class P consists of all problems \( A \) which are polynomial-time decidable. It contains interesting problems, like (the decision version of) Linear Programming, finding the shortest path between two nodes in a graph, deciding whether a number is prime or not and many others.
Another important complexity class is NP, which we define via projection. A problem \( A \subseteq \{0,1\}^* \) is in NP if there is some set \( B \in \text{P} \) and a \( k \in \mathbb{N} \) such that
\[
x \in A \iff \exists y \in \{0,1\}^{\lfloor |x|^k \rfloor} : xy \in B,
\]
where \( xy \) is the concatenation of \( x \) and \( y \). We say that the problems in NP can be “accepted by a non-deterministic polynomial-time Turing machine”.

Changing “polynomial time” into “exponential time” in the above definitions gives the classes EXP respectively NEXP, which are the classes of problems which can be decided/accepted by a deterministic/non-deterministic Turing machine in exponential time. If in the definition of \( \text{P} \) we do not require that \( M \) stops after \( p(|x|) \) steps, but only demand that \( M \) uses at most \( p(|x|) \) many cells of the tape of \( M \), then we get the definition of \( \text{PSPACE} \).

The following inclusions are obvious
\[
P \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}.
\]
All inclusion are conjectured to be strict, but currently there is no proof for any of them. We do know however, that \( P \subset \text{EXP} \) and \( \text{NP} \subset \text{NEXP} \), so at least one of the inclusions must be strict.

The class \( \text{BPP} \) is the class of all problems which can be decided by a probabilistic polynomial-time Turing machine which always outputs the correct result with probability at least \( 2/3 \). More formally, a problem \( A \subseteq \{0,1\}^* \) is in \( \text{BPP} \) if there is some set \( B \subseteq \{0,1\}^* \) which can be decided in polynomial time and a polynomial \( q(\cdot) \) such that
\[
x \in A \rightarrow \exists S \subseteq \{0,1\}^{\lfloor q(|x|) \rfloor} \text{ s.t. } |S| \geq 2^{\frac{2q(|x|)}{3}} \forall y \in S : xy \in B,
x \notin A \rightarrow \exists S \subseteq \{0,1\}^{\lfloor q(|x|) \rfloor} \text{ s.t. } |S| \geq 2^{\frac{2q(|x|)}{3}} \forall y \in S : xy \notin B.
\]
The choice of the constant \( 2/3 \) is not unique. It can be shown that any constant \( 1/2 < c < 1 \) yields the same class. Clearly,
\[
P \subseteq \text{BPP} \subseteq \text{PSPACE}.
\]
The first inclusion is conjectured to be an equality [69].

We now define the class \( \text{BQP} \) [105], the equivalent of \( \text{BPP} \) for quantum computers. A set \( A \subseteq \{0,1\}^* \) is in \( \text{BQP} \) if its characteristic function
\[
\chi_A(x) = \begin{cases} 1 & \text{ if } x \in A \\ 0 & \text{ if } x \notin A \end{cases}
\]
is uniformly computable in polynomial-time with error \( \epsilon = 1/3 \) by a quantum computer (see end of Section 2.2). The following inclusions are known
\[
\text{BPP} \subseteq \text{BQP} \subseteq \text{PSPACE}.
\]
2.4 Communication complexity

We briefly review the field of communication complexity [63, 28, 102, 19]. Assume Alice and Bob wish to compute some Boolean function $f(x, y)$ of input $x \in \{0, 1\}^n$, known to Alice only, and input $y \in \{0, 1\}^n$, known to Bob only. The aim is that Alice learns the value $f(x, y)$. To this end they exchange messages, using as little communication as necessary.

![Communication Protocol Diagram]

Alice and Bob send messages $m_1, \ldots, m_e$ back and forth and Alice outputs $f(x, y)$.

Figure 2.1: A communication protocol

**Deterministic communication complexity** The deterministic communication complexity $D(f)$ of $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ is the smallest number $c$ such that any protocol which always computes the correct result, needs at least $c$ bits of communication for at least one input pair $x, y \in \{0, 1\}^n$. It is clear that this task cannot be accomplished in general without at least some communication, unless $f(x, y)$ does not actually depend on $y$. In fact, it is well-known that there are functions $f$ where $D(f)$ is $n$. An example is the inner product function (see [63]), which is defined as $IP_n(x, y) = \bigoplus_{i=1}^n (x_i \wedge y_i)$.

**Randomized communication complexity** There is a probabilistic version of communication complexity, in which there is a source of random bits, to which Alice and Bob both have access. This is called the public coin model [104] because
Alice and Bob both see the same random bits. In this model, Alice is not required to learn the value of $f(x, y)$ with certainty. Instead, we shall be satisfied if she can obtain an answer that is correct with probability bounded away from $\frac{1}{2}$. In other words, there must exist some real number $p > \frac{1}{2}$ such that the probability that Alice outputs the correct value $f(x, y)$ is at least $p$ for all pairs $(x, y)$ of inputs. The probability is taken over the value of random variables shared between them. The error probability of a protocol is defined as $\epsilon = 1 - p$ and we define the randomized communication complexity $R_\epsilon(f)$ to be the minimum number of bits needed such that the output is correct with probability at least $1 - \epsilon$. Also in this randomized setting there are “hard” functions. For example, it is known that the inner product function has randomized communication complexity $n - O(\log(1/\epsilon))$, if the outputs have to be correct with probability at least $1 - \epsilon$ (see also [63]).

**Entanglement-assisted communication complexity** There is another version of communication complexity, introduced in [28], in which Alice and Bob may share entanglement. More precisely, Alice and Bob may hold an arbitrary (entangled) quantum state shared between them. Particularly useful is often an arbitrary amount of EPR pairs

$$|\phi^+\rangle = \frac{|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B}{\sqrt{2}},$$

where Alice has access to the qubit(s) with subscript $A$ and Bob has access to the qubit(s) with subscript $B$. We denote by $R^*_\epsilon(f)$ the number of classical bits needed to communicate in order for one party to compute the correct result.

Shared entanglement between Alice and Bob helps sometimes but not always. Some functions can be computed with exponentially less communication than with a purely classical protocol [23]. However, for other functions, for example the inner product function, the communication cannot be reduced significantly. Alice and Bob cannot succeed with probability $1 - \epsilon > \frac{1}{2}$ if they transmit less than $\max(\frac{1}{2}(1 - 2\epsilon)^2, (1 - 2\epsilon)^4)n - \frac{1}{2}$ bits, even if they share prior entanglement [29]. In Appendix C we will give a better bound of

$$R^*_\epsilon(IP_n) \geq n - 2\log_2 \frac{1}{1 - \epsilon}$$

using a reduction by [103] to a lower bound for the number of qubits needed to communicate with shared entanglement [65].

**Worst-case partition communication complexity** For functions of the form $f : \{0, 1\}^n \rightarrow \{0, 1\}$ which depend only on one input string and any $S \subseteq \{0, 1\}^n$, such a function is called a partition function. The complexity of computing such a function is called the worst-case partition communication complexity. There is another model, called private coin model, in which Alice and Bob do not have access to the same random bits. But we will not need this model subsequently.
\{1, \ldots, n\} let \(D^S(f)\) be the deterministic communication complexity of \(f\) if the bits with indices in \(S\) are given to Alice and all others to Bob. As in [63] we then define the \textit{worst-case partition communication complexity} as \(D_{\text{worst}}(f) = \max_{S \subseteq \{1, \ldots, n\}} D^S(f)\). In [91] this is called symmetric communication complexity. The \textit{randomized worst-case partition communication complexity} is defined analogously by replacing “\(D_{\text{worst}}\)” with “\(R_{\epsilon_{\text{worst}}}\)” and providing Alice and Bob with shared random bits.

\section{2.5 Bloch sphere}

In later chapters it will be convenient to use the \textit{Bloch sphere} representation of 1-qubit states and 1-qubit operations, which we review now (see e.g. Section 4.2 and Chapter 8 in [68]).

\subsection{2.5.1 Pauli matrices}

The following \textit{Pauli matrices} are the basis of the Bloch vector representation

\[
X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.
\]

(2.3)

It is immediately obvious that they span the whole space \(\mathbb{C}^{2 \times 2}\) and are therefore a basis. They are hermitian and self-inverse. They obey the following commutation relations:

\[
XY = iZ \quad YX = -iZ \\
ZX = iY \quad XZ = -iY \\
YZ = iX \quad ZY = -iX
\]

\subsection{2.5.2 Bloch-vector representation}

For \(\mathbf{r} \in \mathbb{R}^3\) define \(\mathbf{r} \cdot \sigma = r_x X + r_y Y + r_z Z\), where \(\sigma = (X, Y, Z)\) is the vector of Pauli matrices. Then, all 1-qubit density matrices \(\rho\) can be uniquely written in the form

\[
\rho = \frac{I_2 + \mathbf{r} \cdot \sigma}{2} = \frac{I_2 + r_x X + r_y Y + r_z Z}{2},
\]

where \(\mathbf{r} \in \mathbb{R}^3\) and \(||\mathbf{r}|| = \sqrt{r_x^2 + r_y^2 + r_z^2} \leq 1\). We call \(\mathbf{r}\) the \textit{Bloch vector} of \(\rho\).

For \(\mathbf{n} \in \mathbb{R}^3\) with \(||\mathbf{n}|| = 1\) and \(\theta \in \mathbb{R}\) we define

\[
U_n(\theta) = \exp \left( -i\theta \mathbf{n} \cdot \sigma \right) = I_2 \cos \frac{\theta}{2} - i \mathbf{n} \cdot \sigma \sin \frac{\theta}{2}.
\]

We first note that \(U_n(\theta)U_n(\theta)^* = I_2\), i.e., \(U_n(\theta)\) is unitary. Second, let the result of the unitary quantum operation \(U_n(\theta)\) applied to state \(\rho = I/2 + \mathbf{r} \cdot \sigma/2\) be
\[ \rho' = U_n(\theta)^* \rho U_n(\theta) = \frac{I}{2} + \mathbf{r}' \cdot \sigma/2. \] 
Then \( \mathbf{r}' \) is the image of rotating \( \mathbf{r} \) around \( \mathbf{n} \) by an angle \( \theta \). Third, all 1-qubit unitaries \( U \) can be written as

\[ U = U_n(\theta) \]

with \( \mathbf{n} \in \mathbb{R}^3, \theta \in \mathbb{R} \) and \( ||\mathbf{n}|| = 1 \) (ignoring an unimportant phase factor \( \alpha \in \mathbb{C} \) with \( |\alpha| = 1 \)).

Thus, one-qubit states and unitaries are isomorphic to vectors, respectively, rotations in \( \mathbb{R}^3 \). The set of all rotations in \( \mathbb{R}^3 \) is the group \( SO(3) \).

**Arbitrary quantum operations** For non-unitary one-qubit quantum operations the picture is a little bit more complicated. We present a characterization of trace-preserving completely-positive maps on one-qubit operations due to Ruskai, Szarek, and Werner [84, Sections 1.2 and 1.3]. They show that any one-qubit CPTP map \( G \) can be written as a convex combination of gates of the form \( U \circ J \circ V \), where \( U \) and \( V \) are one-qubit unitary operations (acting on the density matrix by conjugation with some unitary \( U, V \in \mathbb{C}^{2 \times 2} \)), and \( J \) is one-qubit map that in the Pauli basis has the form

\[ J = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \lambda_1 & 0 & 0 \\
0 & 0 & \lambda_2 & 0 \\
t & 0 & 0 & \lambda_1 \lambda_2
\end{pmatrix} \] (2.4)

for some \( \lambda_1, \lambda_2 \in [-1, 1] \) and \( t = \pm \sqrt{(1 - \lambda_1^2)(1 - \lambda_2^2)} \). In other words, if a one-qubit state \( \rho \) has Bloch vector \( \mathbf{r} \), then the Bloch vector \( \mathbf{r}' \) of \( \rho' = J(\rho) \) is given by

\[ \begin{pmatrix} 1 \\ \mathbf{r}' \end{pmatrix} = J \begin{pmatrix} 1 \\ \mathbf{r} \end{pmatrix}. \]

### 2.6 Semidefinite programming

In Chapter 7 we will use semidefinite programming methods as a tool to characterize quantum XOR games. We briefly review the most important facts here, closely following [17]. A *semidefinite program* (for short SDP) is an optimization problem of the form

\[
p^* = \sup \quad \text{Tr}(CX) \\
\text{subject to} \quad \text{Tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\
X \succeq 0
\] (2.5)

where \( A_1, \ldots, A_m, C \in \mathbb{C}^{d \times d} \) are given hermitian matrices and the constraint \( X \succeq 0 \) means that \( X \in \mathbb{C}^{d \times d} \) is a positive semidefinite hermitian matrix, i.e. \( X = X^\dagger \) and for all \( |v\rangle \in \mathbb{C}^d : \langle v | X | v \rangle \geq 0 \). Note that with these constraints
∀ \in : \text{Tr}(A_iX) \in \mathbb{R} \text{ and } \text{Tr}(CX) \in \mathbb{R}. \text{ We call (2.5) an SDP in standard primal form.}

In Chapter 7 we will express the optimal winning probability of a quantum XOR game as the optimal solution \( p^* \) of an SDP. Lower bounds on the optimal value of (2.5) can be shown by finding an \( \hat{X} \) satisfying the constraints. Then it is clear that the optimal value of (2.5) is at least \( \text{Tr}(CX) \).

However, we also want to be able to give upper bounds on \( p^* \). One way to find upper bounds is by duality. To this end, we first write down the Lagrangian of (2.5) in terms of the objective function and the constraints as

\[
L(X, y, \Lambda) = \text{Tr}(CX) + \sum_i y_i (\text{Tr}(A_iX) - b_i) + \text{Tr}(\Lambda X),
\]

where \( y \in \mathbb{R}^m \) and \( \Lambda \in \mathbb{C}^{d \times d} \) are called Lagrange multipliers. Now, for any \( y \in \mathbb{R}^m \) and any hermitian \( \Lambda \in \mathbb{C}^{d \times d} \) with \( \Lambda \succeq 0 \) the value of \( p^* \) can be upper bounded by

\[
p^* \leq \sup_X L(X, y, \Lambda),
\]

because every solution \( \hat{X} \) for (2.5) satisfies \( \text{Tr}(A_i \hat{X}) - b_i = 0 \) and \( \text{Tr}(\Lambda \hat{X}) \geq 0 \). Note that there are no constraints on \( X \), other than \( X \in \mathbb{C}^{d \times d} \) and \( X = X^\dagger \). Hence, if we can find \( y, \Lambda \) such that the right hand side of (2.7) is small we can also give good upper bounds on \( p^* \). Let us write this out more explicitly

\[
\sup_X L(X, y, \Lambda) = \sup_X \text{Tr}(CX) + \sum_i y_i (\text{Tr}(A_iX) - b_i) + \text{Tr}(\Lambda X)
\]

\[
= \sup_X \sum_i y_i b_i + \text{Tr} \left( \left( \sum_i y_i A_i + C + \Lambda \right) X \right)
\]

\[
= \begin{cases} 
\sum_i y_i b_i & \text{if } \sum_i y_i A_i + C + \Lambda = 0 \\
\infty & \text{if } \sum_i y_i A_i + C + \Lambda \neq 0.
\end{cases}
\]

Obviously, we are not interested in the trivial upper bound \( p^* \leq \infty \). So, to get good upper bounds we have to find \( y, \Lambda \geq 0 \) with \( \sum_i y_i A_i + C + \Lambda = 0 \). Since \( \Lambda \geq 0 \), we need to require \( \sum_i y_i A_i + C \preceq 0 \) and we get the following dual problem

\[
d^* = \inf_{y_1, \ldots, y_m \in \mathbb{R}} \sum_i y_i b_i \quad \text{subject to } \sum_i y_i A_i + C \preceq 0
\]

From our derivations it immediately follows that

\[
p^* \leq d^*
\]

a property known as weak duality. If this inequality is tight, i.e. \( p^* = d^* \), we speak of strong duality. There are many results about when strong duality holds [17]. The most important one is Slater’s condition. It states that if (2.5) is strictly feasible, which in our case means that some \( X > 0 \) satisfies the constraints of (2.5), then strong duality always holds. This is the criterion which we will also use later.
Real SDPs versus complex SDPs  Most literature deals only with real SDPs, which means that all (entries of all) variables are real numbers. We have given a direct derivation for complex SDPs, because it is more natural for our application in Chapter 7. The complex SDPs we are using can be transformed into real SDPs, using that for any hermitian $X \in \mathbb{C}^{d \times d}$ it holds

$$X \succeq 0 \iff \forall a, b \in \mathbb{R}^d : \begin{pmatrix} a^T, b^T \end{pmatrix} \begin{pmatrix} Re(X) & -Im(X) \\ Im(X) & Re(X) \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \succeq 0,$$

where $Re(X)$ is the real part and $Im(X)$ the imaginary part of $X$. Note that $X \in \mathbb{C}^{d \times d}$ is hermitian if and only if $Re(X) = Re(X^T)$ and $Im(X) = -Im(X^T)$, which is the case if and only if $\tilde{X}$ is symmetric. See also Exercise 4.42 in [17] for a more thorough discussion.