Masaryk University
Faculty of Informatics

Unambiguous discrimination using QISKIT

Bachelor's Thesis

Martin Horáček

Brno, Spring 2020

Masaryk University
Faculty of Informatics

Unambiguous discrimination using QISKIT

Bachelor's Thesis

Martin Horáček

Brno, Spring 2020

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Martin Horáček

Advisor: doc. RNDr. Jan Bouda, Ph.D.

Acknowledgements

I would like to thank my supervisor, doc. RNDr. Jan Bouda, Ph.D., for his guidance. I am also grateful to RNDr. Daniel Reitzner, PhD., for our discussions.

Abstract

We review methods for unambiguous discrimination of pure quantum states. We describe how these methods can be implemented in current quantum computers. We implement unambiguous state discrimination methods in Python3 using the Qiskit library. We implement optimal discrimination of two pure states with the same prior probabilities and optimal discrimination of arbitrary ensembles of pure states with arbitrary prior probabilities. As part of the implementation, we include the decomposition of unitary operators into basic gates and the realization of positive operator valued measure (POVM) measurements. All provided discrimination methods can be run on current IBM quantum computers.

Keywords

unambiguous state discrimination, USD, POVM realization, SDP, optimal, Qiskit, unitary decomposition

Contents

1 Introduction

Quantum information processing promises improvements over classical information processing in simulation, cryptography, and computation fields. Quantum mechanics introduces new properties that enable these improvements, but it also poses several limitations. One of these limitations is the impossibility of perfectly distinguishing non-orthogonal quantum states. Techniques resolving this impossibility are essential in many applications of quantum information processing.

Two important tasks arise from this fundamental problem. The first task is called quantum state tomography or quantum state estimation. It tries to estimate the state of a system by repeated measurements. The second task is called quantum state discrimination, where the set of possible states is previously known, and the goal is to determine the state of the system.

There are several strategies for the quantum state discrimination. The minimum-error discrimination minimizes the probability of an incorrect answer, and an incorrect answer can be returned. The strategy examined in this thesis is called unambiguous state discrimination (USD). It allows us to return an inconclusive answer, and whenever a conclusive answer is returned, then it must be the correct one. The probability of the inconclusive answer should be minimized. Unambiguous state discrimination has its applications in cryptographic protocols, where both the user and the attacker can use it.

Methods for unambiguous state discrimination of pure states have been proposed, and we think it would be beneficial from the educational and practical point of view to see how these methods can be implemented in current quantum computers. We decided to use Python3 and the Qiskit library. Qiskit is a framework for quantum computation. It can simulate quantum circuits, and run them on IBM quantum computers.

We created the QUSD package. It implements several USD methods. They can be either simulated by Qiskit simulators or run on IBM quantum computers. Two general techniques (unitary decomposition, POVM measurement realization) that are used in USD methods were implemented as part of the QUSD package.

No prior knowledge of quantum information processing is required for the comprehension of this thesis. We suppose the reader has good knowledge of linear algebra and its theorems.

In Chapter 2, we introduce the framework for quantum information processing. In Chapter 3, we review unambiguous state discrimination techniques. In Chapter 4, we describe the Qiskit library. In Chapter 5, we describe how techniques from Chapter 3 can be realized on current quantum computers. In Chapter 6, we present the QUSD package, describe its API, and evaluate its performance. In Appendix A, we review notions and theorems from linear algebra that are used extensively in this thesis.

2 Quantum information processing

In this chapter, we lay down foundations of quantum information processing. We present the most common notation used in quantum physics. We define what a qubit is, how to compose larger systems from qubits, how the state of a system can be changed, and how we can learn something about the state using a measurement. For more detailed explanations see [1, 2].

We advise the reader to familiarize themselves with the most heavily used terms from linear algebra in Appendix A.

2.1 Dirac notation

Quantum mechanics uses the Dirac notation for vectors. It is also called the bra-ket notation. The vector v is denoted by $|v\rangle$. $\langle v|$ is the vector dual to $|v\rangle$. The inner product between $|u\rangle$ and $|v\rangle$ is denoted by $\langle u|v\rangle$. In some cases, we denote the inner product by $(|u\rangle, |v\rangle)$, because it can make expressions easier to read. The outer product between $|u\rangle$ and $|v\rangle$ is denoted by $|u\rangle\langle v|$. The application of a linear map A to $|v\rangle$ can be written without brackets as $A|v\rangle$. The tensor product between $|u\rangle$ and $|v\rangle$ is written as $|u\rangle \otimes |v\rangle$.

2.2 Quantum system

The state of a quantum system is described by a normalized vector from a Hilbert space. The state can be used for computations in the following way. The input is encoded as the initial state of the system. The state is then evolved and output is obtained by a measurement.

The quantum system used most often in quantum computing is called a qubit.

2.2.1 Qubit

A qubit is a quantum system with associated two-dimensional Hilbert space $\mathcal{H}$. We work only with this mathematical abstraction, the physical qubit can then be realized in various ways. We define $\{|0\rangle, |1\rangle\}$ to

be an orthonormal basis of $\mathcal{H}$. This basis is called the computational basis. The state of a qubit is a vector $|v\rangle \in \mathcal{H}$

$$|v\rangle = a|0\rangle + b|1\rangle, \tag{2.1}$$

where $|a|^2 + |b|^2 = 1$.

Vectors $e^{i\lambda}|v\rangle$, where $\lambda \in \mathbb{R}$, represent the same state as vector $|v\rangle$. They differ only in their global phase, which is physically insignificant. It means that vectors yield the same measurement statistics, and they cannot be distinguished.

Two often used states with their own special label are states

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ |-\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle. \end{aligned} \tag{2.2}$$

States $|+\rangle$, $|-\rangle$ form an orthonormal basis.

2.2.2 Bloch sphere

It is often useful to visualize the state of a qubit using the Bloch sphere. We can always rewrite the vector $|v\rangle$ up to the global phase as

$$|v\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle. \tag{2.3}$$

Angles θ, φ are shown in the figure 2.1. The term $e^{i\varphi}$ is called relative phase. In contrary to the global phase, the relative phase can be determined by measurements.

The Bloch sphere representation does not have any extension to multi-qubit systems.

2.2.3 Multi-qubit systems

The tensor product is used to combine vector spaces into a larger vector space. Suppose we have n qubits with their two-dimensional Hilbert spaces $\mathcal{H}$. The vector space of the composite system is then $\mathcal{H}^{\otimes n} = \mathcal{H} \otimes ... \otimes \mathcal{H}$.

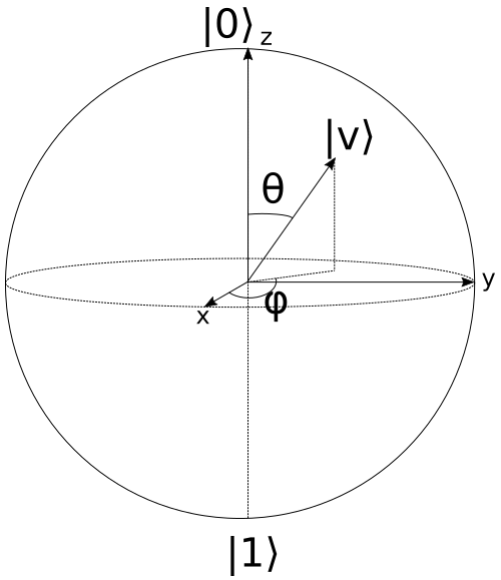


Figure 2.1: Visualization of a vector in the Bloch sphere

The state of a 2-qubit system with both qubits in states $|0\rangle$ is $|0\rangle \otimes |0\rangle$. To make the notation more compact, we write tensor products also as

$$|0\rangle \otimes |1\rangle \equiv |0\rangle|1\rangle \equiv |01\rangle. \tag{2.4}$$

To make this notation even more succinct, we often write labels in decimal base, e.g. $|101\rangle \equiv |5\rangle$.

The computational basis of an n-qubit system can be written as

$$\{|i\rangle | i \in \{0, ..., 2^n - 1\}\}. \tag{2.5}$$

States that can be written as tensor product of 1-qubit states are called product states. States that cannot be written this way are called entangled states. The state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{2.6}$$

is an example of an entangled state. The state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle = |0\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = |0\rangle \otimes |+\rangle \tag{2.7}$$

is an example of a product state.

2.3 Evolution of a quantum system

A closed quantum system is a system that does not interact with its surroundings. If we assume that qubits form a closed quantum system, then the evolution of their state is described by a linear transformation. Since the resulting state must be normalized, this transformation must be unitary. If we start in a state $|v\rangle$, then after the state evolves according to unitary U, it will end up in the state $U|v\rangle$.

Often we want to apply a unitary U only to one qubit, and leave other qubits unchanged. This can be achieved by applying the unitary $I^{\otimes i_1} \otimes U \otimes I^{\otimes i_2}$ to the state of the system, where $i_1 + i_2 + 1$ is the number of qubits, and U is applied to $(i_1 + 1)$-th qubit.

Unitary operators can be composed from simpler unitary operators. In quantum computing, we are interested in sets of unitary operators that can decompose arbitrary unitary operator. In chapter 5, we will show how any unitary operator can be decomposed into unitary operators that are supported by IBM quantum computers.

2.4 Measurement

Measurements are used to obtain information about the system. The process of measurement is not linear, and therefore the evolution can be no longer described by a unitary operator. The measurement is inherently a random process. Results of measurements cannot be predicted, and only their probabilities are known.

Measurements in quantum mechanics are destructive. They collapse the state of a quantum system into a discrete set of states. Repeating the same measurement on the collapsed state will always have the same outcome.

2.4.1 Measurement in the computational basis

The measurement in the computational basis is the most basic type of measurement. It is the only type of measurement implemented in current IBM quantum computers. Suppose we perform the measurement

in the computational basis on the state

$$|v\rangle = \sum_{i=0}^{n-1} a_i |i\rangle. \tag{2.8}$$

The measurement will return outcome i with probability

$$|\langle i|v\rangle|^2 = |a_i|^2, \tag{2.9}$$

and the system will end up in the state $|i\rangle$. States are normalized vectors, therefore

$$\sum_{i=0}^{n-1} |a_i|^2 = 1, \tag{2.10}$$

where $0 \leq |a_i|^2 \leq 1$.

The global phase does not change measurement statistics, because the probability of outcome i for the state $e^{i\varphi}|v\rangle$ is

$$|e^{i\varphi} a_i|^2 = e^{i\varphi} a_i e^{-i\varphi} a_i^* = a_i a_i^* = |a_i|^2. \tag{2.11}$$

2.4.2 Measurement in an orthonormal basis

There is nothing significantly different about the computational basis. The measurement can be defined using any orthonormal basis. Suppose $\{|b_1\rangle, ..., |b_n\rangle\}$ is an orthonormal basis, and we measure the state $|v\rangle$ (2.8). Performing the measurement will return outcome i and collapse the state to $|b_i\rangle$ with the probability $|\langle b_i|v\rangle|^2$.

In later chapters, we will see how to perform this kind of measurement on a quantum computer that can measure only in the computational basis.

2.4.3 Projective measurement

We can define the measurement using an observable. The observable is a Hermitian operator M. Because M is Hermitian, it is diagonalizable and all its eigenvalues are real numbers. We can write M as

$$M = \sum_i m_i P_i, \tag{2.12}$$

where P_i are orthogonal projectors. The probality of outcome i is

$$\langle v|P_i|v\rangle, \tag{2.13}$$

and the state after the measurement is

$$\frac{P_i|v\rangle}{\sqrt{\langle v|P_i|v\rangle}}. \tag{2.14}$$

If P_i are projectors with rank 1, then $P_i = |b_i\rangle\langle b_i|$ and the measurement corresponds to the measurement in an orthonormal basis, because

$$\langle v|P_i|v\rangle = \langle v|b_i\rangle\langle b_i|v\rangle = |\langle b_i|v\rangle|^2. \tag{2.15}$$

The expected value of M for $|v\rangle$ can be computed as

$$\langle M\rangle = \langle v|M|v\rangle. \tag{2.16}$$

The expected value can be used to compute coordinates of a state in the Bloch sphere. Observables, which expected value are Bloch sphere coordinates, are called Pauli operators

$$\begin{aligned} X &= |1\rangle\langle 0| + |0\rangle\langle 1| \\ Y &= i|1\rangle\langle 0| - i|0\rangle\langle 1| \\ Z &= |0\rangle\langle 0| - |1\rangle\langle 1|. \end{aligned} \tag{2.17}$$

The Bloch sphere xyz-coordinates are then $(\langle X\rangle, \langle Y\rangle, \langle Z\rangle)$.

2.4.4 POVM

A more general way of defining a measurement is the positive operator valued measure (POVM) measurement. It does not require orthogonality of its operators. The POVM is a set of positive operators $\{E_1, ..., E_n\}$, where it holds that

$$\sum_{i=1}^{n} E_i = I. \tag{2.18}$$

The probability of outcome i is $\langle v|E_i|v\rangle$. Restrictions on POVM say that the probability must be non-negative, and that the probability must sum up to 1, because

$$\sum_{i=1}^{n} \langle v|E_i|v\rangle = \langle v| \sum_{i=1}^{n} E_i|v\rangle = \langle v|I|v\rangle = \langle v|v\rangle = 1. \tag{2.19}$$

POVMs can be physically realized, and we will see in later chapters how it can be done on current quantum computers.

2.5 Mixed states

Suppose that a measurement in the computational basis is performed on the $|+\rangle$ state, but the outcome of the measurement is unknown. Although the system is either in state $|0\rangle$ or $|1\rangle$, we know only that the state is $|0\rangle$ with probability $\frac{1}{2}$ or $|1\rangle$ with probability $\frac{1}{2}$. This is one way how to obtain a mixed state. States, such as $|+\rangle$ or $|0\rangle$, are called pure states. Pure states can be seen as special cases of mixed states.

Generally, a mixed state is a set $\{(p_i, |v_i\rangle)\}_i$, where p_i is the probability corresponding to the state $|v_i\rangle$. The set representation is not computationally convenient, and a density operator

$$\rho = \sum_i p_i |v_i\rangle\langle v_i| \tag{2.20}$$

is used for computations with mixed states.

For purposes of this thesis, we will define only the POVM measurement of ρ. Suppose that $\{E_1, ..., E_n\}$ is a POVM measurement. The probability of outcome i, when performing the measurement on the state ρ, is

$$\mathrm{Tr}(\rho E_i). \tag{2.21}$$

ρ is always a positive operator with trace 1. Therefore $\mathrm{Tr}(\rho E_i) \geq 0$ for each $1 \leq i \leq n$, and

$$\sum_{i=1}^{n} Tr(\rho E_i) = Tr(\rho \sum_{i=1}^{n} E_i) = Tr(\rho I) = Tr(\rho) = 1. \tag{2.22}$$

The definition indeed gives a valid probability distribution.

3 Unambiguous state discrimination

In this chapter, we review methods for the unambiguous discrimination of pure states. We will first show why these methods are necessary, then we will present methods for unambiguous discrimination of two states with equal prior probability. Finally, we will present a general method for arbitrary pure states with arbitrary prior probabilities.

Let us consider the following scenario. Alice and Bob agree upon a pure ensemble. The pure ensemble is a set of pure states. Each state from the ensemble has a probability of being prepared. Alice prepares a state from the ensemble according to its probability. Alice gives the prepared state to Bob. Bob now has to say which state from the ensemble he was given. He is not allowed to make an error, but he can admit that he does not know. His task is to minimize the probability of the inconclusive answer. This is the problem of unambiguous state discrimination (USD).

3.1 Perfect discrimination

We say that states can be discriminated perfectly iff the optimal probability of an inconclusive answer is 0. If states are orthogonal to each other, then they can be perfectly discriminated. They form an orthonormal set, and they can be extended into an orthonormal basis. We can perform a measurement in this basis to achieve the perfect discrimination.

However, non-orthogonal states cannot be perfectly discriminated. We will show this for states $|+\rangle, |0\rangle$. A more general proof can be found in the QCQI book [1, page 87].

Proof. Suppose there exists a POVM $\{E_1, E_2\}$ such that

$$\begin{aligned} \langle +|E_1|+\rangle &= 1 \\ \langle 0|E_2|0\rangle &= 1. \end{aligned} \tag{3.1}$$

Since probabilities sum up to 1, it holds that

$$\langle 0|E_1|0\rangle = 0. \tag{3.2}$$

Because E_1 is a positive operator, it has a unique square root $\sqrt{E_1}$. $\sqrt{E_1}$ is also a positive operator. Notably, it is Hermitian. We can write the inner product as

$$\begin{aligned} 0 = \langle 0|E_1|0\rangle = \\ = (|0\rangle, E_1|0\rangle) = \\ = (|0\rangle, \sqrt{E_1}\sqrt{E_1}|0\rangle) = \\ = (\sqrt{E_1}|0\rangle, \sqrt{E_1}|0\rangle). \end{aligned} \tag{3.3}$$

Therefore $\sqrt{E_1}|0\rangle = 0$. We can write the probability of outcome 1 given that the state is $|+\rangle$ as

$$\begin{aligned} \langle +|E_1|+\rangle = \frac{1}{2}(\langle 0|E_1|0\rangle + \langle 0|E_1|1\rangle + \langle 1|E_1|0\rangle + \langle 1|E_1|1\rangle) = \\ = \frac{1}{2}\langle 1|E_1|1\rangle \leq \frac{1}{2} < 1, \end{aligned} \tag{3.4}$$

where we used the fact that $\sqrt{E_1}|0\rangle = 0$, and that $\langle 1|E_1|1\rangle$ is the probability of outcome 1 given that the state is $|1\rangle$.

This contradicts our assumption that required $\langle +|E_1|+\rangle = 1$. □

3.2 Orthogonal projector discrimination

First methods proposed for USD worked with a pure ensemble of two states with equal prior probability. We denote these states as $|p\rangle, |q\rangle$.

Ivanovic [3] proposed a straightforward method, that is not optimal. We define two orthogonal projector measurements P and Q

$$\begin{aligned} P_1 = |p\rangle\langle p|, P_2 = I - P_1, \\ Q_1 = |q\rangle\langle q|, Q_2 = I - Q_1. \end{aligned} \tag{3.5}$$

The strategy is to randomly perform the P or Q measurement with some probability. Suppose we perform the measurement defined by P, and get outcome 2. The input state has to be $|q\rangle$, because

$$\langle p|P_2|p\rangle = 0. \tag{3.6}$$

The probability of the conclusive answer given that $|q\rangle$ was on the input is $\langle q|P_2|q\rangle = 1 - |\langle p|q\rangle|^2$. Because both states have same prior probabilities, the probability of the conclusive answer is

$$P_C = \frac{1}{2}(1 - |\langle p|q\rangle|^2). \tag{3.7}$$

If we perform the measurement defined by Q, then the probability of the conclusive answer stays the same.

Although not considered by Ivanovic, this method can be also used for the discrimination of two states with different prior probabilities. Let us denote the prior probabilities of $|p\rangle, |q\rangle$ as r, s respectively. The prepared state is a mixed state, and it can be represented as the density operator

$$\rho = r|p\rangle\langle p| + s|q\rangle\langle q|. \tag{3.8}$$

Probabilities of conclusive outcomes are

$$\begin{aligned} \mathrm{Tr}(\rho P_2) &= r\langle p|P_2|p\rangle + s\langle q|P_2|q\rangle = s(1 - |\langle p|q\rangle|^2) \\ \mathrm{Tr}(\rho Q_2) &= r\langle p|Q_2|p\rangle + s\langle q|Q_2|q\rangle = r(1 - |\langle p|q\rangle|^2). \end{aligned} \tag{3.9}$$

We can see that $\mathrm{Tr}(\rho Q_2) \geq \mathrm{Tr}(\rho P_2)$ iff $r \geq s$. Let α be the probability of performing the measurement P, and $1 - \alpha$ the probability of performing the measurement Q. Suppose $r \geq s$, then it holds that

$$\begin{aligned} P_C(\alpha) &= \alpha\,\mathrm{Tr}(\rho P_2) + (1-\alpha)\,\mathrm{Tr}(\rho Q_2) \leq \\ &\leq \alpha\,\mathrm{Tr}(\rho Q_2) + (1-\alpha)\,\mathrm{Tr}(\rho Q_2) = \mathrm{Tr}(\rho Q_2). \end{aligned} \tag{3.10}$$

Therefore to maximize the conclusive probability, we have to set

$$\alpha = \begin{cases} 0, & \text{if} \quad r > s \\ [0,1] & \text{if} \quad r = s \\ 1, & \text{otherwise.} \end{cases} \tag{3.11}$$

The case, where $r = s$, is the case considered by Ivanovic, and changing α does not change the probability of the conclusive answer. For other cases, this method completely ignores the less probable state, and always returns the inconclusive answer for it.

3.3 Optimal discrimination using a sequence of measurements

This method was proposed by Ivanovic [3]. It optimally unambiguously discriminates between states $|p\rangle$, $|q\rangle$ with equal prior probabilities.

The idea is to add a second quantum system that starts in the state $|s\rangle$. We then apply a unitary U to the composed system. The action of U is

$$\begin{aligned} U|p\rangle|s\rangle &= \alpha|p_1\rangle|s_1\rangle + \beta|p_2\rangle|s_2\rangle \\ U|q\rangle|s\rangle &= \gamma|q_1\rangle|s_1\rangle + \delta|q_2\rangle|s_2\rangle, \end{aligned} \tag{3.12}$$

where $\langle p_1|q_1\rangle = 0$ and $\langle s_1|s_2\rangle = 0$. A measurement is performed on the second system. If the outcome is s_1, then the measurement of the first system is performed. Because p_1 and q_1 are orthogonal, they can be perfectly discriminated. Otherwise the answer is inconclusive.

Ivanovic considered the case of 1-qubit states. He provided a definition of the unitary U and states from equation 3.12. States $|p\rangle$, $|q\rangle$ are written in a basis $\{|b_1\rangle, |b_2\rangle\}$

$$\begin{aligned} |p\rangle &= \sqrt{a}|b_1\rangle + \sqrt{1-a}|b_2\rangle \\ |q\rangle &= \sqrt{a}|b_1\rangle - \sqrt{1-a}|b_2\rangle, \end{aligned} \tag{3.13}$$

where $a > \frac{1}{2}$. The second qubit starts in the state $|b_1\rangle$. Although the original paper does not specify, how to find this basis, we propose the following approach. First we define a unitary R

$$\begin{aligned} |p'\rangle &= R|p\rangle = |0\rangle \\ |q'\rangle &= R|q\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle. \end{aligned} \tag{3.14}$$

The unitary R is used, because it makes further computations with states easier. In chapter 5, we will show how R can be implemented. The basis $\{|b_1\rangle, |b_2\rangle\}$ is then defined as

$$\begin{aligned} |b_1\rangle &= \cos\frac{\theta}{4}|0\rangle + \sin\frac{\theta}{4}|1\rangle \\ |b_2\rangle &= \sin\frac{\theta}{4}|0\rangle - \cos\frac{\theta}{4}|1\rangle. \end{aligned} \tag{3.15}$$

States $|p'\rangle$, $|q'\rangle$ written in this basis have the required form with $a = \cos^2 \frac{\theta}{4}$. The unitary U is defined by its action on $|b_1 b_1\rangle$ and $|b_2 b_1\rangle$ as

$$\begin{aligned} U|b_1 b_1\rangle &= \sqrt{1-a}|b_1 b_1\rangle - \sqrt{a}|b_2 b_2\rangle \\ U|b_2 b_1\rangle &= \sqrt{1-a}|b_1 b_2\rangle + \sqrt{a}|b_2 b_1\rangle. \end{aligned} \tag{3.16}$$

The action of U on states $|b_1 b_2\rangle$ and $|b_2 b_2\rangle$ follows from the unitarity of U. States $|p_1\rangle$, $|q_1\rangle$ are

$$\begin{aligned} |p_1\rangle &= \frac{1}{\sqrt{2}}|b_1\rangle + \frac{1}{\sqrt{2}}|b_2\rangle \\ |q_1\rangle &= \frac{1}{\sqrt{2}}|b_1\rangle - \frac{1}{\sqrt{2}}|b_2\rangle. \end{aligned} \tag{3.17}$$

States $|p_1\rangle$, $|q_1\rangle$ are orthogonal as was required in equation 3.12. The probability of conclusive answer is $\frac{1}{2}(1 - |\langle p'|q'\rangle|^2) = \frac{1}{2}(1 - |\langle p|q\rangle|^2)$. It is obvious that unitary R did not change the probability of conclusive answer. States $|p_2\rangle$ and $|q_2\rangle$ are generally not the same, and the described procedure can be repeated with them on the input. It leads to a potentially infinite sequence of measurements with total conclusive probability $1 - |\langle p|q\rangle|$. This probability is optimal, as was shown by Dieks [4]. In practice, the probability gets close to the optimal probability after 3–4 iterations.

3.4 Optimal discrimination optimized

Peres [5] showed how the previous sequence of measurements can be replaced with single measurement. He achieved that just by changing the unitary U. The definition of states $|b_1\rangle$, $|b_2\rangle$ and unitary R is the same. The unitary U is defined by its action on $|b_1 b_1\rangle$ and $|b_2 b_2\rangle$ as

$$\begin{aligned} U|b_1 b_1\rangle &= \sqrt{\frac{1-a}{a}}|b_1 b_1\rangle + \sqrt{\frac{2a-1}{a}}|b_2 b_2\rangle \\ U|b_2 b_2\rangle &= -\sqrt{\frac{2a-1}{a}}|b_1 b_1\rangle + \sqrt{\frac{1-a}{a}}|b_2 b_2\rangle. \end{aligned} \tag{3.18}$$

The final states of the system after applying U are

$$\begin{aligned} U|p'\rangle|b_1\rangle &= \sqrt{1-a}(|b_1\rangle + |b_2\rangle)|b_1\rangle + \sqrt{2a-1}|b_2 b_2\rangle \\ U|q'\rangle|b_1\rangle &= \sqrt{1-a}(|b_1\rangle - |b_2\rangle)|b_1\rangle + \sqrt{2a-1}|b_2 b_2\rangle. \end{aligned} \tag{3.19}$$

If the measurement on the second qubit yields result b_1, then states of the first qubit are orthogonal to each other as required. The probability of conclusive answer is

$$1 - |\sqrt{2a-1}|^2 = 2(1-a) = 1 - |\langle p|q\rangle|, \tag{3.20}$$

which is optimal.

3.5 2 states with different prior probabilities

Jaeger and Shimony [6] examined the case of two states with different prior probabilities. Let us denote the probability of state $|p\rangle$ as r, and suppose WLOG $r \geq \frac{1}{2}$. The idea behind unambiguous discrimination is the same as in Ivanovic method. The input state is evolved according to the equation 3.12, only the states and unitary U have to be chosen differently.

The probability of the conclusive answer P_C is maximal iff

$$|\beta|^2 = \max(|\langle p|q\rangle|\sqrt{\frac{1-r}{r}}, |\langle p|q\rangle|^2). \tag{3.21}$$

The value of P_C is then

$$P_C = \begin{cases} 1 - 2\sqrt{r(1-r)}|\langle p|q\rangle| & \text{if} \quad \sqrt{\frac{1-r}{r}} \geq |\langle p|q\rangle| \\ r(1 - |\langle p|q\rangle|^2) & \text{otherwise.} \end{cases} \tag{3.22}$$

Figure 3.1 shows conclusive probabilities for an ensemble of states $|0\rangle$ and $|+\rangle$, where r is the probability of state $|0\rangle$. The conclusive probability of state $|0\rangle$ is the probability of recognizing state $|0\rangle$, while performing the optimal USD for the whole ensemble. We can see that the state $|0\rangle$ is ignored for small r. The state $|+\rangle$ is ignored for high values of r. For such values of r, the orthogonal projector discrimination from section 3.2 is optimal.

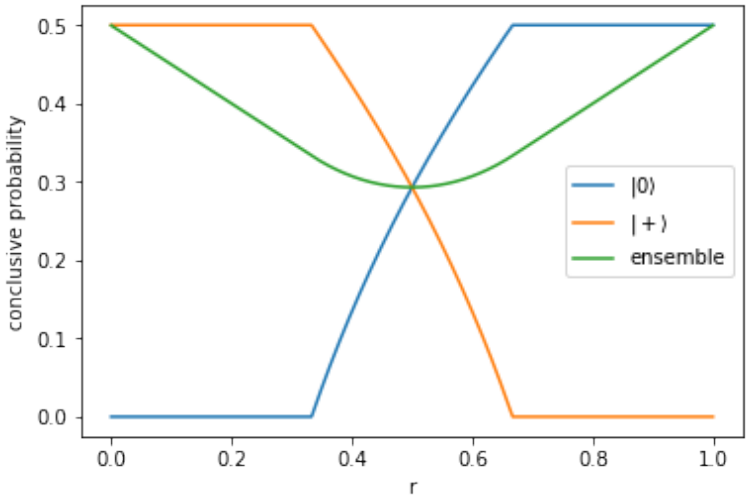


Figure 3.1: Optimal conclusive probabilities for an ensemble of $|0\rangle$ and $|+\rangle$, where r is the probability of $|0\rangle$ on the input.

Figure 3.2 shows conclusive probabilities for a fixed value of r and varying inner product between states.

Although Jaeger and Shimony did not specify the unitary U, we can still use their results to check the correctness of our implementation of the general USD method.

3.6 Optimal discrimination of arbitrary ensembles

A geometric method for three state ensembles was proposed by Peres and Terno [7]. It was shown by Chefles [8] that USD of a set of states is possible iff these states are linearly independent.

Eldar [9] introduced an optimal method for arbitrary ensembles with linearly independent states. Let n be the number of states and $\{|\varphi_i\rangle | 1 \leq i \leq n\}$ be a linearly independent set of states. The probability of state $|\varphi_i\rangle$ is denoted by η_i. The measurement will be defined as a POVM measurement with $n+1$ elements. Each state $|\varphi_i\rangle$ has an associated positive operator Π_i. The operator Π_0 represents the inconclusive measurement. For these operators to define an USD mea-

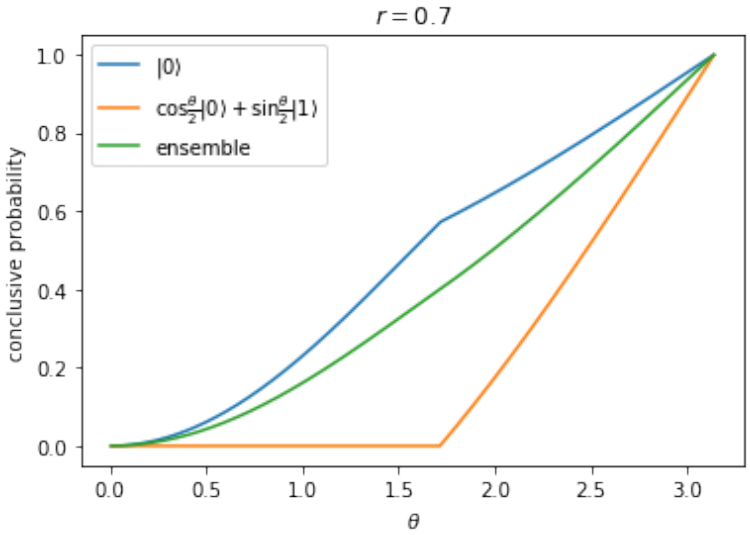


Figure 3.2: Optimal conclusive probabilities for an ensemble of $|0\rangle$ and $\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$, where the probability of $|0\rangle$ is 0.7.

surement, it must hold for each $1 \leq i, j \leq n$ that

$$\langle\varphi_i|\Pi_j|\varphi_i\rangle = \begin{cases} p_i & \text{if} \quad i = j \\ 0 & \text{otherwise,} \end{cases} \tag{3.23}$$

where p_i is the probability of the conclusive result for the state $|\varphi_i\rangle$ on the input.

For each state $|\varphi_i\rangle$ there exists a vector $|\overline{\varphi}_i\rangle$ that is orthogonal to other states from the ensemble, and $\langle\varphi_i|\overline{\varphi}_i\rangle = 1$. Notice that $|\overline{\varphi}_i\rangle$ does not have to be normalized. If we write states $|\varphi_i\rangle$ as columns into a matrix Φ then vectors $|\overline{\varphi}_i\rangle$ are columns of matrix $\overline{\Phi}$ defined by

$$\overline{\Phi} = \Phi(\Phi^\dagger\Phi)^{-1}. \tag{3.24}$$

Positive operators Π_i can then be defined as

$$\begin{aligned} \Pi_i &= p_i|\overline{\varphi}_i\rangle\langle\overline{\varphi}_i| \quad \text{for} \quad 1 \leq i \leq n \\ \Pi_0 &= I - \sum_{i=1}^{n} \Pi_i. \end{aligned} \tag{3.25}$$

It remains to choose probabilities p_i in such a way that maximizes P_C. The problem of finding optimal p_i can be defined as a semi-definite program (SDP). It is a convex optimization problem. It always converges to the global optimum. We want to maximize

$$P_C = \sum_{i=1}^{n} \eta_i p_i \tag{3.26}$$

subject to

$$\begin{aligned} \Pi_0 &\geq 0 \\ \sum_{i=1}^{n} p_i = 1 \quad p_i &\geq 0. \end{aligned} \tag{3.27}$$

The condition $\Pi_0 \geq 0$ is required, because POVM elements must be positive operators.

4 Qiskit library

Qiskit [10] is an open-source framework for quantum computation. It covers every stage of the quantum programming workflow. Its API provides methods for designing quantum circuits, optimizing them, and then simulating them. Qiskit simulators are capable of realistically simulating noisy quantum computers. It is also easy to switch from simulators to real IBM quantum computers by adding an IBMQ token.

4.1 Elements of Qiskit

The Qiskit library consists of four parts (elements). The first element is Terra. It provides tools for the composition of quantum circuits. It implements transpilers that transform gates and optimize circuits.

The second element is Aer. It provides simulators that run circuits compiled by Terra. These simulators can simulate noisy quantum computers if the noise model is specified. Simulators are a very useful debugging tool because computational time on IBMQ devices is limited.

The last two elements of Qiskit are Ignis and Aqua. Ignis is a framework for suppressing errors in quantum circuits, and Aqua is a library with the most common quantum algorithms.

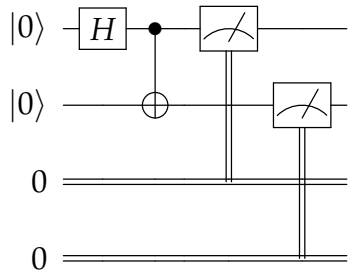
4.2 IBM Quantum

IBM provides public access to a number of their quantum computers. If the IBMQ token is set up correctly, then using them is as simple as changing a few lines in the code.

Current IBM quantum computers have several limitations. The only gates that can be performed on them are CNOT, U1, U2, U3. The only possible measurement is the measurement in the computational basis. CNOT can be performed only on some pairs of qubits. These pairs are specified by a coupling map of the device.

4.3 Circuit model of computation

Quantum programs in Qiskit are defined as circuits (4.1). Each single wire in a circuit represents a qubit. In Qiskit, states of all qubits start always in the state $|0\rangle$. Quantum gates represent unitary operators. There are several commonly used gates that have their special symbol. Time in the circuit flows from left to right. The circuit also contains classical bits. They are represented by double wires. Classical bits start in 0, and they are used to store results of measurements. The measurement is represented by a special symbol. Measurements in Qiskit are always performed in the computational basis. These essential parts of quantum circuits can be seen together in an example circuit (4.1).



(4.1)

4.3.1 Common quantum gates

Some gates are used so often that they have their unique symbols. They are shown in the table 4.1 together with the unitary operator they represent. CNOT gate applies X to the target qubit whenever the control qubit is in the state $|1\rangle$. Toffoli gate is a variant of the CNOT gate with two control qubits.

The U3 gate is a universal 1-qubit gate. It can represent any 1-qubit gate. Gates U2 and U1 can be seen as U3 gate with some parameters fixed. Their unitary representation is

$$\begin{aligned} \mathrm{U3}(\theta,\phi,\lambda) &= \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{pmatrix} \\ \mathrm{U2}(\phi,\lambda) &\equiv \mathrm{U3}(\pi,\phi,\lambda) \\ \mathrm{U1}(\lambda) &\equiv \mathrm{U3}(0,0,\lambda). \end{aligned} \tag{4.2}$$

Table 4.1: Most common gates and their symbols.

name	symbol	unitary
X / NOT gate		$\|1\rangle\langle 0\| + \|0\rangle\langle 1\|$
Y gate		$i\|1\rangle\langle 0\| - i\|0\rangle\langle 1\|$
Z gate		$\|0\rangle\langle 0\| - \|1\rangle\langle 1\|$
Hadamard gate		$\|+\rangle\langle 0\| + \|-\rangle\langle 1\|$
CNOT / CX gate		$\|0\rangle\langle 0\| \otimes I + \|1\rangle\langle 1\| \otimes X$
Toffoli / CXX gate		$(I - \|11\rangle\langle 11\|) \otimes I + \|11\rangle\langle 11\| \otimes X$

Because U3 gate is universal for 1-qubit gates, it can implement its own adjoint. The relation is

$$\mathrm{U3}^{\dagger}(\theta, \phi, \lambda) \equiv \mathrm{U3}(-\theta, -\lambda, -\phi). \tag{4.3}$$

4.4 Transpilers

A transpiler is a source-to-source compiler. The functionality of a Qiskit transpiler can be divided into small tasks called passes. These passes can

- remove redundant gates, and optimize the circuit,
- unroll gates to gates implemented by the device,
- swap qubits to match the coupling map of the device.

It is important to optimize circuits, because quantum computers are noisy. Therefore longer circuits have worse results. The Qiskit transpilers can decompose 2-qubit unitaries into elementary gates. Unitaries on more qubits, that will be required for the implementation of USD techniques, cannot be decomposed by Qiskit transpilers. We

are required to write our own decomposition. The Toffoli gate is an exception, because Qiskit transpilers can decompose it into U3 and CNOT gates.

4.5 Providers and backends

The Backend class [11] is an interface to a simulator or a quantum device. The Backend object contains parameters of the device such as coupling map and available gates. The Provider class manages backends from one provider. We use Aer provider for simulations and IBMQ provider for experiments on real quantum devices.

Aer provides three backends:

QasmSimulator simulates a real quantum device. It returns a dictionary of measurement results.

StatevectorSimulator returns the final state of the circuit. This is typically used without measurements.

UnitarySimulator computes the unitary operator represented by the circuit. The circuit must not contain measurements.

StatevectorSimulator and UnitarySimulator can be used to check the correctness of circuits.

4.6 Special simulator functions

Running the circuit with a simulator backend has another advantage. We can use *initialize* and *unitary* methods of the QuantumCircuit class.

The *initialize* method prepares the starting state of the circuit. Normally all qubits start in state $|0\rangle$. Using this method, we can prepare any input state. This was useful in the process of implementing and testing USD methods, because we could use arbitrary input states, and did not have to come up with preparation circuits.

The *unitary* method can simulate arbitrary unitary gate, which is useful if we do not know how to decompose some unitary.

The limitation of these two methods is that they can be used only with simulator backends.

4.7 Register endianity

Qiskit uses little endian ordering for qubits and classical bits. Our theoretical computations use big endian ordering. It is necessary to change endianity of measurement results to get the expected results. This applies also to results from StatevectorSimulator and UnitarySimulator.

5 Implementation of unambiguous state discrimination

In chapter 3, we theoretically described methods for USD. In this chapter, we will present techniques used for the implementation of USD methods on real quantum computers. Described techniques can be used in various areas of quantum information processing.

5.1 Measurement in an orthonormal basis

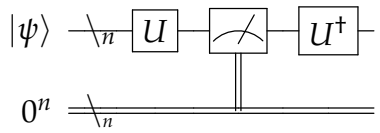
In Qiskit, measurements can be performed only in the computational basis. A technique for performing measurements in any orthonormal basis is required, because this type of measurement is directly used in methods from sections 3.2, 3.3, 3.4. It is also used by the POVM realization method that will be described in the section 5.4.

Suppose we want to perform a measurement in an orthonormal basis $\{|b_1\rangle, ..., |b_n\rangle\}$, where n is a power of 2. This measurement can be realized by transforming $|b_{i+1}\rangle$ to $|i\rangle$ using a unitary U

$$U = \sum_{i=0}^{n-1} |i\rangle\langle b_{i+1}|. \tag{5.1}$$

The measurement in the computational basis is then performed. The outcome i corresponds to the state $|b_{i+1}\rangle$. The post-measurement state is then transformed by U^{-1} from $|i\rangle$ back to $|b_{i+1}\rangle$. The transformation back to $|b_{i+1}\rangle$ is not necessary if we are interested only in the outcome of the measurement.

The operator U is unitary, therefore $U^{-1} = U^\dagger$. The whole procedure is shown in the following circuit diagram,



(5.2)

where one wire represents n-qubits for better readability.

5.1.1 1-qubit orthonormal basis

Methods from sections 3.2, 3.3, 3.4 all use measurements in 1-qubit orthonormal basis $\{|b_1\rangle, |b_2\rangle\}$. The unitary U for basis transformation from equation 5.1 can be implemented with one U3 gate.

Let $|b_1\rangle = a|0\rangle + b|1\rangle$. Because $\langle b_1|b_2\rangle = 0$, the state $|b_2\rangle$ must be of the form

$$|b_2\rangle = e^{i\varphi}(-b^*|0\rangle + a^*|1\rangle). \tag{5.3}$$

Because the global phase is insignificant, we can set $\varphi = 0$. The adjoint of unitary U from equation 5.1 is then

$$U^\dagger = \begin{pmatrix} a & -b^* \\ b & a^* \end{pmatrix}. \tag{5.4}$$

Entries of U3 (4.2) can be directly compared with values of $U^\dagger$. Parameters of the U3 gate are

$$\begin{aligned} \theta &= 2\cos^{-1}|a| \\ \phi &= \arg(b) - \arg(a) \\ \lambda &= -\arg(b) - \arg(a). \end{aligned} \tag{5.5}$$

Parameters for U3 gate corresponding to unitary U can be obtained from equation 4.3.

5.2 Rotation to xz-plane

In methods from sections 3.3 and 3.4, we transformed states $|p\rangle$ and $|q\rangle$ into states $|0\rangle$ and $\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$ to make further computations easier. The unitary R from equation 3.14 can be composed from one U3 and one U1 gate.

The first unitary U_1 will rotate the state $|p\rangle$ to the state $|0\rangle$. U_1 is the inverse of the unitary that transforms $|0\rangle$ to $|p\rangle$. If (x, y, z) are Bloch sphere coordinates of state $|p\rangle$, then

$$\begin{aligned} \theta_1 &= cos^{-1}(z) \\ \phi_1 &= tan^{-1}(\frac{y}{x}) \\ U_1^\dagger &= \mathrm{U3}(\theta_1, \phi_1, 0) \\ U_1 &= \mathrm{U3}(-\theta_1, 0, -\phi_1), \end{aligned} \tag{5.6}$$

where the last line follows from 4.3. State $|q\rangle$ is transformed up to a global phase into

$$|\overline{q}\rangle = U_1|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \tag{5.7}$$

The unitary U_2 that transforms $|\overline{q}\rangle$ into $\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$ can be realized with U1 gate as

$$U_2 = \mathrm{U1}(-\varphi). \tag{5.8}$$

The state $|0\rangle$ is not changed by U_2, thus $|\overline{p}\rangle = |0\rangle$ stays the same. The unitary R is then $R = U_2U_1$. We call this procedure a rotation to xz-plane, because

$$\langle\overline{q}|Y|\overline{q}\rangle = \langle\overline{p}|Y|\overline{p}\rangle = 0. \tag{5.9}$$

States $|\overline{p}\rangle$, $|\overline{q}\rangle$ are in the xz-plane of the Bloch sphere.

5.3 Conditional termination circuits

In section 3.3, we reviewed an iterative USD method proposed by Ivanovic. In each iteration the computation either continues or ends depending on the result of a measurement on one qubit. It is not possible in Qiskit to terminate the execution of a circuit based on the measurement result, but we can store the result of the measurement as a bit, and control gates in further iterations by values of bits from all previous iterations.

Qiskit allows gates controlled by a classical register in a simulator, but IBM quantum computers do not support this feature. To overcome this limitation, we can use the principle of deferred measurement from the QCQI book [1, page 186]. It says that if a qubit is measured to a

classical register, and that register is used as a control bit, then we can postpone the measurement until the end of the circuit, and use a gate controlled by the qubit that was to be measured.

The theoretical description of Ivanovic's method allowed potentially infinite sequences of measurements. This is not possible now, because each iteration requires a new additional qubit. Let n be the number of iterations chosen beforehand. The circuit will contain $n+1$ qubits. Initially, the qubit with index 0 is set to one of the input states. In iteration $i \geq 1$, we always prepare the qubit with index i for measurement. If the qubit is in the state $|0\rangle$, then the measurement of the qubit with index 0 will yield the conclusive result, and the computation should be stopped. Because we cannot do this directly, all gates in iteration i will be controlled by qubits $\{1, ..., i-1\}$. Iterations with inconclusive result set their corresponding qubit to $|1\rangle$. This allows further iterations to continue.

5.4 Realization of POVM

In section 3.6 we reviewed a general method for unambiguous discrimination of pure states with arbitrary prior probabilities. It used a POVM measurement, but it did not specify how to physically realize this measurement. The POVM had a rank 1 element for each state, and one element with unspecified rank for the inconclusive answer.

The POVM realization is a two step process. Firstly, the POVM will be decomposed to a POVM with rank 1 elements. Secondly, the POVM with rank 1 elements will be implemented using a measurement in specific orthonormal basis.

5.4.1 POVM to POVM with rank 1 elements

Let us consider a POVM with elements $E_1, ..., E_n$. Each E_i is a positive operator, and it has a diagonalization

$$E_i = \sum_{j=1}^{m_i} \lambda_j |\lambda_j\rangle\langle\lambda_j|, \tag{5.10}$$

where m_i is the number of non-zero eigenvalues, $\lambda_j \neq 0$. Because $\lambda_j \geq 0$, $\lambda_j|\lambda_j\rangle\langle\lambda_j|$ is a positive operator. Let us denote $\lambda_j|\lambda_j\rangle\langle\lambda_j|$ by E_{ij}. Then the set

$$\{E_{ij}|1 \leq i \leq n, 1 \leq j \leq m_i\} \tag{5.11}$$

is a valid POVM measurement with rank 1 elements. If we associate measuring E_{ij} with outcome i, we get the same measurement statistics as for the original POVM.

5.4.2 Neumark theorem construction

The POVM realization method is based on the constructive proof of Neumark theorem [12]. The Neumark theorem claims that POVM measurements can be realized physically. It is done by extending the quantum system with additional qubits, and then performing a measurement in an orthonormal basis on the bigger system.

Suppose we want to perform a POVM measurement specified by rank 1 elements $\{E_1, ..., E_n\}$. Each element can be written as

$$E_i = |v_i\rangle\langle v_i|, \tag{5.12}$$

where vectors $|v_i\rangle$ do not have to be normalized. If we write vectors $|v_i\rangle$ into rows of a matrix, then this matrix has mutually orthonormal columns. We can add columns to the matrix in such a way, that they will form an orthonormal basis. This extended matrix is unitary. Let us denote its rows as $|u_i\rangle$. Vectors $|u_i\rangle$ form an orthonormal basis, and the outcome i of the measurement in this basis corresponds to the measurement of E_i.

5.5 Unitary decomposition

All unitary operators from USD methods have to be decomposed into circuits formed by CNOT and U3 gates because these are the only gates supported by current IBM quantum computers. As the unitary decomposition problem has many applications in quantum information processing, many decomposition methods were developed. We chose to implement the method described in the QCQI book [1]. This method uses $\mathcal{O}(n^2 4^n)$ 1-qubit and CNOT gates for the decomposition

of a unitary on n qubits. Because we use the Qiskit library, we can use its transpilers, that can already decompose Toffoli and controlled U3 gates.

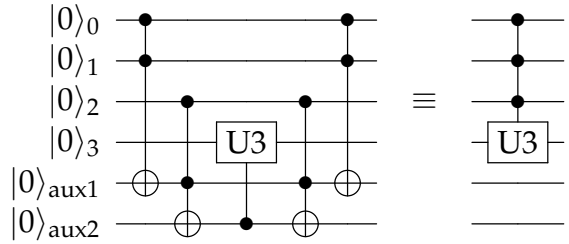
It is enough for quantum computers to support only CNOT and U3 gates because they can implement arbitrary unitary operators. We say that CNOT and U3 are universal for quantum computation.

5.5.1 Multicontrolled U3

The concept of control can be generalized to more qubits, and the controlled gate can be arbitrary 1-qubit gate. As Qiskit is capable of decomposing controlled U3 and Toffoli gate, it is enough to decompose the multicontrolled U3 into these gates. This construction uses auxiliary qubits that start in state $|0\rangle$, and are returned to state $|0\rangle$ afterwards. If n is the number of control qubits, then the circuit will use $n-1$ auxiliary qubits for the computation. The decomposition for $n = 3$ and gate

$$U = |111\rangle\langle 111| \otimes \mathrm{U3} + (I - |111\rangle\langle 111|) \otimes I \tag{5.13}$$

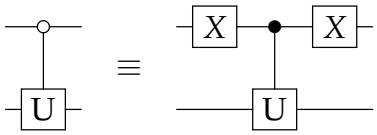
is



(5.14)

Because auxiliary qubits are returned to the state $|0\rangle$, they can be used for the decomposition of more gates in a sequence.

So far the unitary was applied whenever the state of the control qubit was $|1\rangle$. If we want to apply the unitary whenever the state of the control qubit is $|0\rangle$, then we apply NOT gate to the control qubit before and after the controlled gate. This is usually visualized as



(5.15)

5.5.2 Two-level unitary

A unitary that has non-trivial action on at most two states is called a two-level unitary. Such unitary was directly used in the USD method by Peres (3.4). A unitary composed from two two-level unitaries was used in the iterative USD method by Ivanovic (3.3).

Suppose that the two-level unitary acts on n-qubit states $|k\rangle$ and $|m\rangle$, where $0 \leq k, m < 2^n$. The action of this unitary on $|k\rangle$ and $|m\rangle$ is specified by a 1-qubit unitary U. We can find the shortest sequence of binary numbers $(a_1, ..., a_l)$, where $a_1 = k$, $a_l = m$, and each successive number differs in exactly one bit from the previous one. Let us denote by b_i the position, where this bit is for numbers a_i and a_{i+1}. For each $1 \leq i \leq l-2$ we add a multicontrolled NOT to the circuit. It has $n-1$ control qubits, and it applies NOT to the b_i-th qubit. Bits (except b_i-th bit) of a_i set to 1 specify positions of qubits that control target iff they are in state $|1\rangle$. Bits set to 0 specify positions of qubits controlling iff they are in state $|0\rangle$. For $i = l-1$ we replace NOT gate with U. Changes to other states than $|k\rangle$ and $|m\rangle$ are reverted by applying multicontrolled NOT gates in the reverse order.

5.5.3 Decomposing unitary into two-level unitaries

Two-level unitaries can be used to decompose an arbitrary unitary matrix. The decomposition proceeds by columns. Concrete steps are described in the QCQI book [1, page 189]. A n-qubit unitary is decomposed into $O(4^n)$ two-level unitaries.

This decomposition is directly used in the implementation of the general USD method (3.6) for the measurement in an orthonormal basis from the Neumark theorem construction.

5.6 Mitigation of problems with floating-point arithmetic

Computers represent real numbers as approximations using a finite number of bits. Therefore operations with real numbers are imprecise, mathematical identities do not have to hold, and algorithms do not have to work as expected. It is necessary to employ techniques that

are robust to these imprecisions. We present mitigation techniques that were required for the robust implementation of USD methods.

5.6.1 Floating-point number comparison

The most common operation used in the implementation is the comparison of floating-point numbers. It is used in unit and randomized tests to check correctness. It is used in functions that test properties of matrices and vectors.

Direct comparison of two floating-point numbers, as can be done with integers, does not produce required results. Instead, the difference between numbers has to be checked. An appropriate allowed difference (tolerance) must be chosen. It is not enough to compare only the absolute difference between numbers, because if numbers were small, then their absolute difference could be lower than the tolerance, and still they could be quite different. The solution is to define also relative tolerance. The comparison equation used by the NumPy library [1] is

$$|a - b| \leq (\epsilon + \delta * |b|), \tag{5.16}$$

where a, b are numbers being compared, ϵ is the absolute tolerance, δ is the relative tolerance.

Our implementation and Qiskit library use NumPy *isclose* method for comparison of floating-point numbers.

5.6.2 Matrix properties

Checks for matrix properties are used in many places. Functions for unitary decomposition require a unitary matrix on the input. The POVM measurement consists of positive operators. Properties of input matrices are checked to ensure correctness. These checks should be robust to imprecisions of floating-point arithmetic. Otherwise, there will be false positives, and valid inputs will be considered invalid.

The check we found most susceptible to errors was the check for positivity. Qiskit and also our implementation test positivity by checking whether the matrix is Hermitian, and then checking that eigen-

1. `https://numpy.org/doc/stable/reference/generated/numpy.isclose`

values are nonnegative. However, the eigenvalue decomposition can return small negative eigenvalues even for positive matrices. Tolerance $\epsilon \geq 0$ is therefore chosen, and it is checked that for each eigenvalue λ_i it holds that

$$\lambda_i + \epsilon \geq 0. \tag{5.17}$$

Another problem was encountered while using the Qiskit *unitary* method (4.6), and when testing unitarity of inputs to internal classes of our implementation. The unitarity check is done by checking that

$$UU^\dagger \approx I. \tag{5.18}$$

Algorithms doing POVM realization and unitary decomposition in our implementation, although correct in infinite precision setting, can return matrices that do not pass this check. Also it can happen that a matrix U passes this unitarity test, but $U^\dagger$ does not pass it.

One solution is to set different tolerance or to avoid checking matrix properties, when passing them between internal functions. The problem with this solution is that we cannot change the tolerance used by Qiskit in its QuantumCircuit.unitary method. We need a way to change an almost unitary matrix A into a unitary matrix $\overline{A}$ such that it is as close as possible to A. This can be done[2] by computing the singular value decomposition $A = U\Sigma V^\dagger$, and setting $\overline{A} = UV^\dagger$.

5.6.3 Extending orthogonal states to orthonormal basis

In the Neumark theorem construction (5.4.2), we extended a list of mutually orthogonal states to an orthonormal basis. This can be done with the following approach.

Let $\{|v_1\rangle, ..., |v_k\rangle\}$ be a set of mutually orthogonal states from a n-dimensional Hilbert space $\mathcal{H}$. We iteratively add $|i\rangle$ for $0 \leq i \leq n-1$ to the set iff the set extended with states from previous iterations combined with $|i\rangle$ forms a linearly independent set of states. We are left with a set $\{|v_1\rangle, ..., |v_n\rangle\}$, where $|v_i\rangle$ for $1 \leq i \leq k$ are the original states. This set forms a basis of $\mathcal{H}$, but it is not orthonormal. Now it should be enough to do the Gram–Schmidt process, but the traditional Gram–

2. `https://people.eecs.berkeley.edu/~wkahan/Math128/NearestQ.pdf`

Schmidt process is not numerically stable. It is therefore necessary to do a modified Gram–Schmidt[3] process that is numerically stable.

In our experience, for some mutually orthogonal states even this modified process did not yield an orthonormal basis. The modified Gram–Schmidt process is therefore repeated for 100 iterations, where output of one iteration is passed as input to next iteration. It was confirmed by randomized testing, that this can produce an orthonormal basis with high precision.

3. https://www.math.uci.edu/~ttrogdon/105A/html/Lecture23.html

6 QUSD package

We implemented unambiguous discrimination methods that were theoretically described in chapter 3. We used techniques from chapter 5 for the implementation. Implemented methods can be accessed through the API of QUSD (Qiskit unambiguous state discrimination) package. In this chapter, we describe the API of the package, we point to example Jupyter notebooks, we describe tests for the package. Finally we also evaluate the correctness of our implementation.

6.1 Used libraries

The QUSD package was implemented in Python3, and it uses three libraries:

NumPy is a package for scientific computing. Its linear algebra functions are used for most computations in QUSD package.

Qiskit is the package described in chapter 4. It is used for creating circuits and simulating them.

CVXPY is a package for convex optimization. It is used for the optimization of SDP.

6.2 API

QUSD package provides a united interface to all USD methods. Each USD class corresponds to one USD method: OrthogonalProjectorUSD (3.2), PeresUSD (3.4), IvanovicUSD (3.3), POVM_USD (3.6). Constructors of these classes take a PureEnsemble object. The PureEnsemble class represents the ensemble of pure states. IvanovicUSD class is the only exception, as the number of iterations must be also specified. The interface to these classes is specified by the USDBase class. The specifications of public methods are described in the documentation, here we only briefly comment on their usage.

The *get_required_registers* method returns the number of quantum and classical bits that the circuit implementing given method will

use. The *get_theoretical_probabilities* method returns a dictionary that contains overall probabilities of answers for the ensemble, and also probabilities of answers given the prepared state is one of the states from the ensemble. The *discriminate* method runs the discrimination method for a specified number of samples, and returns results. The user can provide arbitrary backend, such as IBMQ backend. The *discriminate_statevector* method uses the StatevectorSimulator from the Qiskit library. Instead of sampling and returning counts as *discriminate* method does, it returns precise probabilities of results computed by the simulator. This is very useful for testing and debugging.

Apart from USD methods, unitary decomposition is available from the UnitaryDecomposer class, and POVM realization is available from the POVMRealization class.

6.3 Examples

We provide 4 example Jupyter notebooks:

01_2state_usd.ipynb notebook shows how to use classes OrthogonalProjectorUSD, PeresUSD, and IvanovicUSD. It compares their conclusive answer probabilities for the state $|0\rangle$ and states

$$\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle, \tag{6.1}$$

where $0 \leq \theta \leq \pi$.

02_Nstate_usd.ipynb notebook shows how to use the POVM_USD class, and it shows that it does the optimal USD for the ensemble from Peres and Terno [7] paper.

03_unitary_decomposition.ipynb notebook shows how the methods for unitary decomposition can be used out of the USD context.

04_quantum_device_experiment.ipynb runs POVM_USD and PeresUSD on a real IBM quantum computer. Because current quantum computers are noisy, it is possible to get an incorrect answer when using them as a backend.

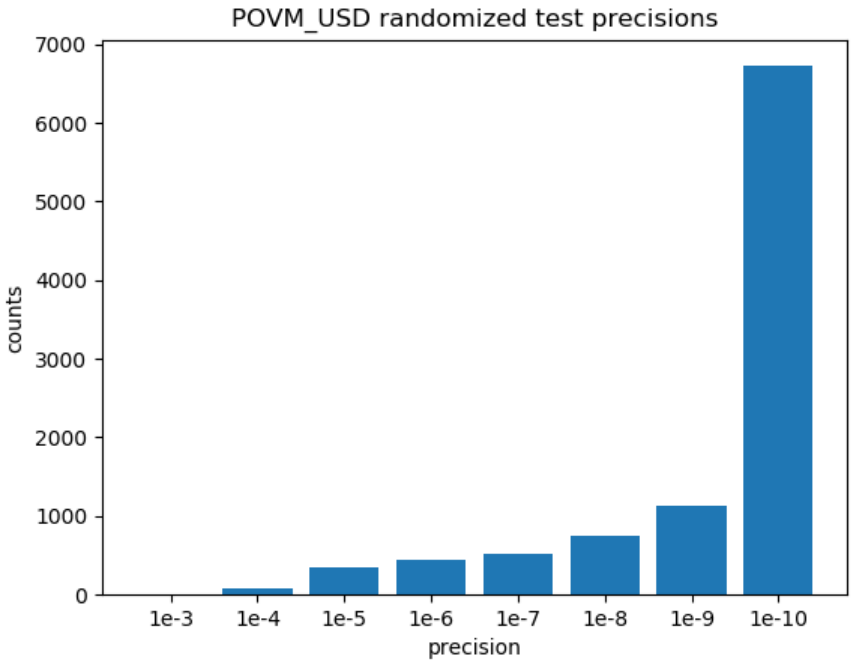


Figure 6.1: This histogram shows precisions for randomized tests of POVM_USD class. Tests ran with 10000 random ensembles.

6.4 Tests and evaluation

The package is tested extensively by unit tests. Unit tests have 100% code coverage. Tests for USD classes use the Qiskit StatevectorSimulator to get precise measurement statistics. They compare actual results to theoretically computed results to 10^{-8} precision. Tests for POVM_USD use the ensemble from Peres and Terno [7] paper, and theoretically computed results for 2 state ensembles with different prior probabilities using computations from section 3.5.

Along with unit tests, we implemented randomized tests for all USD classes and unitary decomposition. Randomized tests for unitary decomposition decompose a random unitary, and compare results from Qiskit UnitarySimulator with the original unitary up to 10^{-8} precision elementwise. Tests ran for over 100000 random unitaries on different numbers of qubits.

Randomized tests for USD classes generate random ensembles. They use the *discriminate_statevector* method of USD classes, because it returns precise measurement statistics. They compare simulation results with probabilities computed by the *get_theoretical_probabilities* method. They check whether the absolute difference is below some threshold. *get_theoretical_probabilities* returns analytically computed probabilities for classes OrthogonalProjectorUSD, PeresUSD and IvanovicUSD. For POVM_USD class it returns probabilities computed using the semidefinite programming optimization.

For classes OrthogonalProjectorUSD, PeresUSD and IvanovicUSD, we ran randomized tests with over 10000 random ensembles. Absolute differences between simulated and theoretical probabilities were lower than 10^{-8} for all random ensembles.

Randomized tests for the POVM_USD class tested random ensembles on 1, 2, 3 and 4 qubits with all possible numbers of states. Worst absolute differences between simulated and theoretical probabilities were lower than 10^{-4}. This precision seems worse than for other classes. In figure 6.1 is the histogram of precisions for 10000 random ensembles. Although the precision seems worse, it is actually worse for only a smaller number of cases. We believe that the reason for lower precision in some cases is the more complicated computation in POVM_USD class, and the fact that absolute difference behaves badly when comparing numbers close to 0. Optimal USD often ignores less probable states, and states measured with zero probability are encountered often.

7 Conclusion

We introduced the quantum information processing framework. We defined operations that can be done on quantum systems. Then we reviewed unambiguous state discrimination methods. The first few described methods discriminated ensembles of two pure states with equal prior probabilities. Lastly, we described a general method for optimal unambiguous discrimination of arbitrary pure ensembles.

Theoretical descriptions of USD methods did not specify how they should be implemented on real quantum devices. Therefore, we described the techniques required for the implementation of USD methods. We showed how limitations posed by the Qiskit library and IBM quantum computers could be overcome. We showed how POVM measurements can be realized, and how unitaries used by USD methods can be decomposed into circuits.

Then we briefly described the Qiskit library. We focused mainly on features that helped us with writing implementations of USD methods, testing them, and debugging them.

Then we described the API of the QUSD package. Example Jupyter notebooks show how the package can be used. In one notebook, we ran the USD methods on IBM quantum computers to show that it is really possible as it was claimed throughout this thesis. As part of the implementation, we wrote extensive unit tests. To ensure the robustness of the implemented methods, we also wrote randomized tests. These tests showed to be incredibly useful for finding problems with floating-point arithmetic. We believe that our implementation is robust and correct.

A Linear algebra for quantum information processing

In this appendix, we present notions and theorems from linear algebra that are extensively used in the field of quantum information processing.

All considered vector spaces in this thesis are finite dimensional vector spaces over complex numbers with inner product. They are often referred to as Hilbert spaces.

We denote the set of all linear operators on a vector space $\mathcal{H}$ as $\mathcal{L}(\mathcal{H})$.

Definition A.1. $U \in \mathcal{L}(\mathcal{H})$ is called unitary iff $UU^\dagger = I$.

Theorem A.2. Unitary operators preserve inner product.

Quantum states are normalized vectors, and the previous property of unitary operators is necessary to keep states normalized after the unitary evolution.

Definition A.3. $B \in \mathcal{L}(\mathcal{H})$ is called adjoint of $A \in \mathcal{L}(\mathcal{H})$ iff for all $|u\rangle, |v\rangle \in \mathcal{H}$ it holds that $(B|u\rangle, |v\rangle) = (|u\rangle, A|v\rangle)$.

This operator always exists and is unique. We denote the adjoint of A as $A^\dagger$.

Theorem A.4. If $U \in \mathcal{L}(\mathcal{H})$ is a unitary operator, then $U^{-1} = U^\dagger$.

This theorem is used a lot, because we often need to find the inverse of a unitary operator, and taking adjoint is simpler than computing the inverse.

Definition A.5. $H \in \mathcal{L}(\mathcal{H})$ is called Hermitian or self-adjoint iff $H = H^\dagger$.

Theorem A.6. Hermitian operators are normal, and they have real eigenvalues.

Definition A.7. $A \in \mathcal{L}(\mathcal{H})$ is called positive iff for all $|u\rangle \in \mathcal{H}$ it holds that $\langle u|A|u\rangle \geq 0$.

Usually, in linear algebra, the operator A is called positive semidefinite, but in quantum computing literature, it is usually called a positive operator. Therefore, we call it positive in this thesis. The positivity of A is denoted by $A \geq 0$.

Theorem A.8. Positive operators are Hermitian, and they have real nonnegative eigenvalues.

Bibliography

1. NIELSEN, Michael A.; CHUANG, Isaac L. *Quantum computation and quantum information*. 10th anniversary ed. New York: Cambridge University Press, 2010. ISBN 978-1-107-00217-3.
2. KAYE, Phillip; LAFLAMME, Raymond; MOSCA, Michele. *An Introduction to Quantum Computing*. New York: Oxford University Press Inc., 2007. ISBN 978-0-19-857049-3.
3. IVANOVIC, Igor. How to differentiate between non-orthogonal states. *Physics Letters A*. 1987, vol. 123, no. 6, pp. 257–259. ISSN 0375-9601. Available from DOI: 10.1016/0375-9601(87)90222-2.
4. DIEKS, Dennis. Overlap and distinguishability of quantum states. *Physics Letters A*. 1988, vol. 126, no. 5, pp. 303–306. ISSN 0375-9601. Available from DOI: 10.1016/0375-9601(88)90840-7.
5. PERES, Asher. How to differentiate between non-orthogonal states. *Physics Letters A*. 1988, vol. 128, no. 1, pp. 19. ISSN 0375-9601. Available from DOI: 10.1016/0375-9601(88)91034-1.
6. JAEGER, Gregg; SHIMONY, Abner. Optimal distinction between two non-orthogonal quantum states. *Physics Letters A*. 1995, vol. 197, no. 2, pp. 83–87. ISSN 0375-9601. Available from DOI: 10.1016/0375-9601(94)00919-G.
7. PERES, Asher; TERNO, Daniel R. Optimal distinction between non-orthogonal quantum states. *Journal of Physics A: Mathematical and General*. 1998, vol. 31, no. 34, pp. 7105–7111. Available from DOI: 10.1088/0305-4470/31/34/013.
8. CHEFLES, Anthony. Unambiguous discrimination between linearly independent quantum states. *Physics Letters A*. 1998, vol. 239, no. 6, pp. 339–347. ISSN 0375-9601. Available from DOI: 10.1016/S0375-9601(98)00064-4.
9. ELDAR, Yonina. A semidefinite programming approach to optimal unambiguous discrimination of quantum states. *IEEE Transactions on Information Theory*. 2003, vol. 49, pp. 446–456. Available from DOI: 10.1109/TIT.2002.807291.

10. ABRAHAM, Héctor et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. Available from DOI: 10.5281/zenodo.2562110.

11. MCKAY, David C. et al. *Qiskit Backend Specifications for OpenQASM and OpenPulse Experiments*. 2018. Available from arXiv: 1809.03452 [quant-ph].

12. BOUDA, Jan. *Density Matrices And Superoperators* [*online*]. 2001 [cit. 2020-05-01]. Available also from: https://is.muni.cz/th/h2lv6. Master's thesis.