POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Quantum Matching Pursuit Algorithms

Tesi di Laurea Magistrale in
Computer science and Engineering - Ingegneria Informatica

Author: **Stefano Vanerio**

Student ID: 963974
Advisor: Prof. Stefano Zanero
Co-advisors: Armando Bellante
Academic Year: 2021-2022

# Acknowledgements

Per la realizzazione di questa tesi:

Ringrazio il mio relatore, Professor Stefano Zanero per avermi permesso di approfondire questo argomento, tanto interessante quanto impegnativo.

Un ringraziamento speciale va ad Armando Bellante, per la sua comprensione, la pazienza e la disponibilità nonostante il fuso orario avverso. I suoi consigli sono stati fondamentali per questo lavoro.

Per essermi stati vicini in questi anni:

Voglio ringraziare la mia famiglia, per avermi sempre supportato e dato affetto anche nei momenti più difficili. Un grazie a mio fratello Daniele e a mia sorella Noemi, per avermi dato il giusto esempio da seguire, per avermi sempre aiutato e per gli importanti consigli. Un ringraziamento particolare a mia mamma, per l'amore incondizionato e per essere stata presente per me. Grazie per avermi ascoltato sempre e comunque, per i preziosi insegnamenti e per avermi insegnato come superare le difficoltà a testa alta.

Un ringraziamento speciale alla mia compagna Ilaria, per avermi trasmesso la sua immensa forza e il suo coraggio. Grazie per il tuo amore e per tutto il tempo che mi hai dedicato. Grazie perché ci sei sempre stata e hai sempre saputo come farmi stare meglio.

Senza di voi non sarei mai riuscito ad arrivare fino a qui.

Inoltre, voglio ringraziare Agnese che durante questi anni di università è sempre stata un punto di riferimento, un prezioso aiuto e un'amica sincera. Voglio ringraziare gli amici che mi hanno aiutato a sopportare le difficoltà, facendomi distrarre e trascorrere dei bei momenti. Un grazie ai ragazzi del TPA e del QG per le belle serate passate insieme. Voglio ringraziare anche i miei amici della Svezia, che mi hanno insegnato che nella vita c'è ben altro oltre l'università.

Infine, voglio ringraziare me stesso. Per la perseveranza, i sacrifici e l'impegno che ho messo in questo percorso. Qualsiasi cosa mi riservi il futuro mi sono impegnato al massimo, ed è questo quello che conta.

# Abstract

Quantum machine learning is an emerging discipline that combines the benefits of quantum computing with machine learning techniques. Indeed, quantum computing speeds up the computation of several tasks with respect to what the current classical computing techniques achieve. In this work, we provide an analysis of three novel quantum algorithms. The first two algorithms are related to the sparse representation theory and are based respectively on Matching Pursuit and Orthogonal Matching Pursuit. These algorithms solve a linear system minimizing greedily the $l_0$-norm of the solution. Instead, the third one is a nearest neighbor classification algorithm that exploits the reduction of dimensionality to achieve better performance. The use of quantum procedures introduces some approximation error in the computation of several intermediate steps. For each of these algorithms, we provide a detailed description of their procedure, their running time, their probability of failure, and what requirements must be met to ensure a specific error in the final results. The runtimes of these algorithms are lower than their classical equivalents. In addition to that, we study, through the use of numerical experiments, how the outputs of the algorithms are affected by the errors in the intermediate results. In this analysis, we use both artificially generated datasets and real cybersecurity-related datasets.

**Keywords:** quantum computing, quantum machine learning, sparse representation, matching pursuit, orthogonal matching pursuit, eigenvectors-based classification

# Abstract in lingua italiana

Il machine learning quantistico è una disciplina emergente che combina i benefici del calcolo quantistico con tecniche di machine learning. Infatti, il calcolo quantistico accellera la computazione di diverse operazioni rispetto alla complessità delle attuali tecniche classiche. In questo lavoro, presentiamo tre nuovi algoritmi quantistici. I primi due algoritmi sono relativi alla teoria delle rappresentazioni sparse e sono basati su Matching Pursuit e Orthogonal Matching Pursuit. Questi algoritmi risolvono un sistema lineare minimizzando in maniera avida la norma $l_0$ della soluzione. Il terzo algoritmo, invece, è un algoritmo di classificazione nearest neighbour che sfrutta la riduzione della dimensionalità per ottenere prestazioni migliori. L'uso di procedure quantistiche introduce errore di approssimazione nel calcolo di alcuni step intermedi. Per ognuno di questi algoritmi, forniamo una spiegazione dettagliata del loro funzionamento, del loro tempo di esecuzione, della loro probabilità di successo e quali requisiti devono essere soddisfatti per ottenere uno specifico errore di approssimazione nei risultati finali. Il tempo di esecuzione di questi algoritmi è inferiore rispetto ai loro equivalenti classici. In aggiunta a ciò, analizziamo, tramite l'uso di esperimenti numerici, come i risultati degli algoritmi sono influenzati dall'errore nei risultati intermedi. Per questa analisi usiamo sia dati generati artificialmente che dati reali relativi alla sicurezza informatica.

**Parole chiave:** calcolo quantistico, machine learning quantistico, rappresentazioni sparse, matching pursuit, orthogonal matching pursuit, classificazione basata su autovettori

# Contents

# Introduction

In recent years, the field of Quantum Machine Learning has gained increasing interest. This field looks into how machine learning algorithms might benefit from the computational advantages of quantum computing. Through the use of quantum computing, it is possible to exploit phenomena such as superposition and entanglement to create more efficient routines than the classical ones.

The beginning of Quantum Machine Learning can be traced back to 2009 with the publication of the paper from Harrow, Hassidim, and Lloyd [36] where the authors present a quantum routine to solve linear systems of equations. This routine uses a number of operations that is only poly-logarithmic in the size of the system. This algorithm, referred to as HHL after its authors, has given the basis to discover a different number of routines related to linear algebra that are fundamental for machine learning. This work and its outcomes have sparked interest in the field, leading to the development of algorithms that perform clustering [43], support vector machines [65], and principal component analysis [49], with polynomial to exponential speed-ups over the state-of-the-art classical solutions. More recently, Chakraborty et al. [15, 16], Gilyén et al. [33] have generalized how to use block-encodings in quantum settings, creating a useful framework to design quantum machine learning algorithms, including our work.

Currently, most of the work related to Quantum Machine Learning is still theoretical, due to the early stage of development of quantum computers. Indeed, we can locate ourselves in the *noisy intermediate-scale quantum (NISQ)* era. While some tools can be used to simulate the functioning of a quantum computer, such as the SQLearn library [30] or the IBM's open source software development kit (SDK) Qiskit [5], the use of a functioning quantum computer is still far in time. To use the algorithms cited above and the ones presented in this work, we need fault-tolerant quantum computers with a huge number of qubits. There are several techniques designed to run on NISQ shallow circuits but, unfortunately, they lack theoretical guarantees on the runtime [9].

In this work, we focus on a specific topic called Sparse Representation [92] (also known as sparse approximation). Sparse representation is a field of study in which the objective is

to find the sparsest solutions of linear systems, between the possible ones. More formally, the algorithms try to find a good-enough approximation of a given vector, while minimizing the $l_0$-*norm* of the solution. The origin of this field is closely linked to compressed sensing [13], a methodology used to acquire and reconstruct signals for which not enough samples are provided. The exploitation of sparse representation techniques has found a large use in different topics such as machine learning, signal processing, data mining, medical imaging, and image restoration [51, 68]. It is also possible to find applications in anomaly detection [2]. One important use is in classification tasks. For instance, the Sparse Representation Classification method [38, 86], used to classify different images in categories, had been demonstrated to be robust to noise.

Finding the sparsest representation of a vector is an NP-Hard problem. For this reason, the problem is usually solved through *greedy algorithms*. These algorithms, through the use of an iterative process, provide a locally optimal solution that can be considered an approximation of the globally optimal one. Some examples of greedy algorithms for this task are the Matching Pursuit [52], the Orthogonal Matching Pursuit [61], the Subspace Pursuit [20], and CoSaMP [58]. Another way to obtain a sparse enough solution is minimizing the $l_1$-*norm* of the solution instead of the $l_0$-*norm*, as shown in [14, 24]. In this case, techniques such as Lasso [75] can achieve good results and generate sparse-enough solutions.

In this work, we propose a quantum version of the Orthogonal Matching Pursuit algorithm and we extend the quantum version of Matching Pursuit of Bellante and Zanero [7], considering quantum states as inputs.

In addition to them, we also present a quantum algorithm based on Eigenvectors Classification [77]. Using this algorithm, it is possible to perform classification of a sample after performing dimensionality reduction through Principal Component Analysis [54]. This method is the basis of techniques used for face recognition with eigenfaces [77, 89]. Furthermore, we show how it can be used also for anomaly detection in a cybersecurity scenario.

For each of the algorithms, we prove upper bounds on their running time and show how the error caused by the use of quantum computing routines can impact the quality of the solutions.

## Main Contributions

We can summarize our contributions as:

- **Quantum Matching Pursuit**: we extend the algorithm provided by Bellante and Zanero [7] to handle quantum states as inputs. We provide the estimation of the running time and the analysis of the error. The procedure is summarized in Algorithm 4.1. We report the final running time of the algorithm in Theorem 4.1.
- **Quantum Orthogonal Matching Pursuit**: we describe the algorithm, prove the running time, and perform an error analysis. We analyze how some theoretical guarantees, studied for the classical version of the algorithm to obtain better results, are present also in the quantum version. We summarize the procedure in Algorithm 4.2. The running time of this algorithm is reported in Theorem 4.5.
- **Quantum Eigenvectors Classification/Recognition**: we describe a quantum algorithm for classification, analyzing in detail its running time complexity and how to bound its final error. Algorithm 4.3 summarizes the procedure. The running time of this new algorithm is reported in Theorem 4.8.

All the algorithms obtain a polynomial speed-up compared to their classical versions and work for both classical and quantum inputs. For each of them, we provide some experiments to test their stability, by introducing artificially some simulated quantum error. The experiments in Section 5.3.2 are related to malware detection using Quantum Orthogonal Matching Pursuit, and the ones in Section 5.4.2 are related to network anomaly detection using Quantum Eigenvectors Classification/Recognition.

## Thesis Outline

This work is divided into 7 chapters:

1. **Classical Background**: we describe the mathematical notation we use in this work. Then, we proceed with an overview of the classical version of the algorithms that we will analyze later on.
2. **Quantum Background**: we introduce some basic concepts related to Quantum Computing, such as quantum data representation, quantum RAM, quantum data access, the block-encodings framework, and any relevant subroutine that we will use in this work.
3. **Intermediate Technical Results**: we present some technical results in the form

of lemmas and claims. These results will be used for proving the runtimes of the algorithms.

4. **New Quantum Algorithms**: we analyze and prove quantum versions of the algorithms known as Matching Pursuit, Orthogonal Matching Pursuit, and Eigenvectors Classification/Recognition.

5. **Experiments**: we show the results of the experiments conducted on the simulations of each of the algorithms presented in Chapter 4. We show how the algorithms' performance changes with different levels of error. We also test the Quantum Orthogonal Matching Pursuit and the Quantum Eigenvector Classification/Recognition algorithms with real data, obtaining performance comparable to other state-of-the-art methods for reasonable runtime parameters.

6. **Limitations and Future Work**: we discuss some limitations of this work and describe some possible future research directions.

7. **Conclusions**: we summarize the conclusions of this work.

# 1 | Classical Background

In this chapter, we describe the mathematical notation that we use inside the work. After that, we briefly introduce the field of sparse representation and we proceed with an overview of the classical version of the algorithms analyzed in this work.

## 1.1. Mathematical Notation

We denote vectors with lowercase letters as $x$. With the notation $x \in \mathbb{R}^n$, we point out, for example, that all the $n$ components of the vector are real numbers. We denote the $j^{th}$ element of a vector $x$ as $x_j$. We use lowercase letters also for scalar numbers such as $a$. The difference between scalar numbers and vectors will be clear by the context or clarified with a note when necessary.

We use $\|\cdot\|$ to denote the Euclidean norm, also known as *l2-norm*, of a vector. The notation $\|\cdot\|_0$, instead, denotes the sparsity (the number of non-zero components) of a vector. The generic *$l_p$-norm*, for a certain value $p$, corresponds to $\|x\|_p = (\sum_{i=1}^{d} |x_i|^p)^{1/p}$. The notation $\langle x, y\rangle$, given two vectors $x$, $y$ with the same dimension, denotes the inner product between them. We use $\langle x|y\rangle$ if the two vectors are normalized. We define as $d(x, y) = \|x - y\| = \sqrt{\|x\|^2 + \|y\|^2 - 2\langle x, y\rangle}$ the Euclidean Distance between the vectors $x$ and $y$.

We use capital letters as $A$ to refer to matrices. Let $A \in \mathbb{R}^{n\times m}$ be a matrix, we use the notation $a_{i,:}$ to refer to its $i^{th}$ row while we use the notation $a_i$ to refer to its $i^{th}$ column. We use $a_{i,j}$ to denote the entry of the matrix present in the $i^{th}$ row and in the $j^{th}$ column. We denote with $A^T$ the transpose of the matrix, with $A^*$ its conjugate transpose and with $A^+$ its pseudoinverse, which corresponds to $A^+ = (A^*A)^{-1}A^*$. Given $A \in \mathbb{R}^{n\times n}$ a square invertible matrix, we denote with $A^{-1}$ the matrix for which $AA^{-1} = A^{-1}A = \mathbb{I}$ where $\mathbb{I}$ is the identity matrix. We denote a matrix $A$ with as columns the set of vectors $\{a_1, ..., a_n\}$ as $A = \begin{bmatrix} a_1 & ... & a_n \end{bmatrix}$.

We can perform the Singular Value Decomposition of a matrix $A \in \mathbb{R}^{n\times m}$ and obtain three matrices such that $A = U\Sigma V$, where $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{m\times m}$ and $\Sigma$ is a rectangular

diagonal matrix. The elements of the matrix $\Sigma$ are called singular values. We denote the $i^{th}$ singular value of a matrix as $\sigma_i$. With $\sigma_{max}$ we identify the maximum singular value, while with $\sigma_{min}$ we identify the minimum one.

We denote with $\|\cdot\|_F$ the Frobenius norm of a matrix. Given a matrix $A \in R^{n\times m}$ we can define the Frobenius norm as $\|A\|_F = \sqrt{\sum_i^n \sum_j^m a_{ij}^2}$. Finally, we denote with $\|A\|$ the maximum singular value of $A$.

The condition number of a matrix $A$ corresponds to $\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}}$.

We use the Big-O notation $O(\cdot)$ to describe the running time upper bound of an algorithm. We also use the notation $\widetilde{O}(\cdot)$ to omit polylogarithmic terms. For example, an algorithm with complexity $O(n \log n)$ has complexity $\widetilde{O}(n)$.

## 1.2. Classical Algorithms

### 1.2.1. Sparse Representation

The main focus of our work is on finding sparse solutions for linear systems. For this reason, before diving into the description of the single algorithms analyzed, we state the main problem that the algorithms aim to resolve.

The problem corresponds to representing a dense signal as a sparse combination of a few unit vectors, known as *atoms* and grouped in a structure called *dictionary*. We define as *over-complete* the dictionaries in which the number of atoms is larger than the number of signal components. These dictionaries are frequently used. The sparse representation of a signal is the set of coefficients of the linear combination of atoms used to describe it. More formally, we define the following problem.

**Definition 1.1** (Problem $\mathcal{P}_0^\epsilon$ [7]). *Let $s \in \mathbb{R}^n$ be a vector that represents the signal to analyze and $D \in \mathbb{R}^{n\times m}$ a matrix called dictionary with columns such that $\|d_j\| = 1$ for each $j \in [m]$. Let $\epsilon \in \mathbb{R}^+$ be a threshold on the residual, we define problem $\mathcal{P}_0^\epsilon$ as:*

$$\arg\min_x \|x\|_0 \ \ such \ \ that \ \ \|s - Dx\| \leq \epsilon. \tag{1.1}$$

The following algorithms solve this problem in a greedy way, minimizing at each step the difference between the signal $s$ and the approximated value.

### 1.2.2. Matching Pursuit

Matching Pursuit (MP) [52] is a greedy algorithm that tries to solve the sparse representation problem by maximizing at each step the amount of signal that the approximation is describing in an iterative way. It was one of the first algorithms used and, thanks also to its simplicity, one of the most popular.

This work is based on an already developed quantum version of the algorithm, developed by Bellante and Zanero [7], and extends the previous result thanks to the inclusion of newly discovered quantum routines and a different approach related to the check of the exit condition.

#### Algorithm

This description of the algorithm is a rework of the one in Section C by Bellante and Zanero [7].

Let us denote the signal with $s \in \mathbb{R}^n$, the dictionary matrix with $D \in \mathbb{R}^{n \times m}$, the approximation threshold with $\epsilon \in \mathbb{R}^+$ and the sparsity threshold with $L \in \mathbb{N}$. To obtain the solution $x \in \mathbb{R}^m$, such that we have a sparse-enough representation, we proceed in an iterative way, trying to optimize the solution at each step.

We define the *residual* as the difference between the approximated solution and the real signal

$$r = s - Dx. \tag{1.2}$$

We start from an empty vector $x = 0^{\otimes m}$ then the algorithm looks for the atom that reduces the residual. We initialize the residual equal to the signal $r = s$.

At each step, we select one atom to reduce the residual norm.

At each step, the algorithm searches the index $j^*$ of the atom of the dictionary that is closer to the residual vector and computes $z_j \in \mathbb{R}$ that corresponds to the best scaling factor for the atom $d_j$

$$j^* = \arg_{\mathrm{j}} \min \|r - z_j d_j\|, \tag{1.3}$$

$$z_j = \arg_{\mathrm{z}} \min \|r - z d_j\|. \tag{1.4}$$

Mallat and Zhang [52] have shown that $z_j = \langle r, d_j \rangle$, from which we obtain

$$\|r - z_j d_j\|^2 = \|r\|^2 - |\langle r, d_j \rangle|^2. \tag{1.5}$$

Considering that the value of the residual does not change in the previous equation, we can just look for the greatest inner product to select our next atom

$$j* = \arg_{\mathrm{j}} \max |\langle r, d_j \rangle|. \tag{1.6}$$

For this reason, we just compute the inner products between all the atoms of the dictionary and the residual vector. This step is called *sweep stage*. We update the solution and the residual using the atom with the greatest inner product

$$x_{j*} = x_{j*} + z_{j*}, \tag{1.7}$$

$$r = r - z_{j*} d_{j*} = s - Dx. \tag{1.8}$$

After each iteration, the residual will be less since the sparse combination of the atoms of the dictionary will describe more of the initial signal. The algorithm can select the same atom more than once.

The stop condition of the algorithm is the following

$$\|x\|_0 > L \text{ or } \|r\| \leq \epsilon. \tag{1.9}$$

Algorithm 1.1 summarizes the procedure.

Algorithm 1.1 Matching Pursuit [7]

**Input** $s \in \mathbb{R}^n$ signal, $D \in \mathbb{R}^{n \times m}$ dictionary, $L \in \mathbb{N}$ sparsity threshold, $\epsilon \in \mathbb{R}^+$ error reconstruction tolerance

**Output** $x \in \mathbb{R}^m$, an approximated solution of $\mathcal{P}_0^\epsilon$ (Def. 1.1)

Initialize $r = s$, $x = 0^{\otimes m}$.
**while** not ($\|x\|_0 > L$ or $\|r\| \leq \epsilon$) **do**
  **for all** $j \in [m]$ **do**
    Compute $\langle d_j, r \rangle$
  **end for**
  Select $j^* = \arg\max(|\langle d_j, r \rangle|)$
  Assign $z_{j*} = \langle d_{j*}, r \rangle$
  Update the solution $x_{j*} = x_{j*} + z_{j*}$
  Update the residual $r = r - z_{j*} d_{j*}$
**end while**
Output $x$

## Alternative exit condition and norm bounds

We present here an alternative exit condition for the algorithm. This condition had been highlighted in Equation 17 presented in [52] by Mallat and Zhang. We denote with $r_{(i)}$ the residual vector and with $d_{(i)}$ the selected atom at the $i^{th}$ iteration. First of all, it is necessary to recall that the signal can be described as

$$s = \sum_{i=0}^{k} \langle r_{(i)}, d_{(i)} \rangle \, d_{(i)} + r_{(k+1)}. \tag{1.10}$$

From Equation 1.10, it is easy to deduce a description of the residual as subtraction between the full signal and the linear combination of the selected atoms multiplied for their coefficient

$$r_{(k+1)} = s - \sum_{i=0}^{k} \langle r_{(i)}, d_{(i)} \rangle \, d_{(i)}. \tag{1.11}$$

At this point, if we define the energy described until iteration $k$ as $\xi = \sum_{i=0}^{k} |\langle r_{(i)}, d_{(i)} \rangle|^2$, we obtain that

$$\|r_{(k+1)}\|^2 = \|s\|^2 - \xi = \|s\|^2 - \sum_{i=0}^{k} |\langle r_{(i)}, d_{(i)} \rangle|^2. \tag{1.12}$$

The previous equation holds since, at each step, the energy described by the residual is the combination of the energy described by the previous residual and the inner product between it and the selected atom. For the interested reader, a more formal explanation is provided by Mallat and Zhang [52]. This equation can be used as an update rule for the residual norm at each iteration.

In addition to the previous result, it is also possible to define a bound on the norm of the solution vector $x$, thanks to the same property.

**Claim 1.2** (Bound on the norm of the solution vector)**.** *Given a signal $s \in \mathbb{R}$, a dictionary $D \in \mathbb{R}^{n \times m}$ and a solution vector $x \in \mathbb{R}^n$ such that $x$ is a solution of $\mathcal{P}_0^\epsilon$ (Def. 1.1) we can say that $\|x\| \leq \|s\|$.*

*Proof.* First of all, we must highlight that $\|x_k\|^2 \leq \sum_{i=0}^{k} |\langle r_{(i)}, d_{(i)} \rangle|^2$ as explained above. Considering the bound on the norm $\|x_k\|$ and Equation 1.12 we can say that

$$\|r_{k+1}\|^2 \leq \|s\|^2 - \|x_k\|^2. \tag{1.13}$$

From the previous equation, it is easy to deduce that

$$\|x_k\|^2 \leq \|s\|^2 - \|r_{k+1}\|^2 \leq \|s\|^2 \tag{1.14}$$

and for this reason we know that $\|x_k\| \leq \|s\|$. □

These two conditions will be particularly useful in our quantum version of this algorithm.

## Computational complexity

For the analysis of the running time, we proceed considering the complexity of each single step. The initialization has complexity $O(n)$ because we initialize the vector $r = s$, and it is linear in the dimension of $s$. After that, the *sweep stage* has complexity $O(nm)$ because it is necessary to compute the inner products of all the $m$ atoms and the residual, considering that all the vectors have dimension $n$. The selection of the best atom can be done during the sweep stage, saving at each computation the greatest value and its index. For this reason, it has no additional complexity. The solution update has complexity $O(1)$ because one single component of the solution vector is updated. The update of the residual has complexity $O(n)$ because all the components of the residual vector are updated.

Considering that the complexity of the sweep stage is the bottleneck of the algorithm, we can say that the overall complexity for one iteration corresponds to $O(nm)$. Considering also that we denote $k$ as the number of iterations of the cycle necessary to satisfy the exit condition, we obtain as final complexity

$$O(knm). \tag{1.15}$$

## Related work

The first published version of Matching Pursuit was the one created by Mallat and Zhang [52] (1993) to describe noisy signals of dimension $n$ as a combination of a subset of $m$ waveforms grouped in a matrix called dictionary. The complexity of this algorithm is $O(knm)$, where $k$ is the number of iterations of the algorithm. It is possible to apply this algorithm to subjects different from simple signals. One example is the work of Bergeaud and Mallat [8] (1995), in which the Matching Pursuit is applied to images using the multiscale time-frequency Gabor dictionary.

After that, different versions of the algorithm had been developed while trying to reduce

its computational complexity, for example, the Thresholding algorithm and the Weak Matching Pursuit [91]. In the first of these two approaches, the selection of the atoms is different. In the Thresholding algorithm, the inner products between the signal and the atoms are performed once. At this point, the solution vector is modified iteratively, adding at each iteration the inner product with the greatest absolute value to the components and recomputing the residual vector until a threshold value is reached. Then, the residual is computed with the selected atoms, and the complexity corresponds to $O(kn + nm)$. In Weak Matching Pursuit, the inner products are computed as in the classical matching pursuit but, instead of selecting the maximum value, the algorithm selects the first inner product bigger than a value chosen by the user. This algorithm has worst-case complexity $O(knm)$, but the complexity in the single iteration can be better on average because it is possible to stop the computation of the inner products in advance and not compute all the $m$ inner products.

Another interesting algorithm is the one of Krstulovic and Gribonval [44] (2006), where the authors had obtained a run-time of $O(k \log(n))$ using a specific dictionary (the multiscale time-frequency Gabor), but the algorithm does not achieve this speed-up with non-analytical dictionaries. In addition, different extensions to the algorithm had been made, for example, the popular Orthogonal Matching Pursuit [61] (1993) that we will describe in the next section, together with many other versions and algorithms.

Regarding previous work related to Quantum Computing and Matching Pursuit, some works suggest the use of matching pursuit to simulate the dynamics of quantum mechanical processes [87, 88] (2003, 2004).

In addition to that, Bellante and Zanero [7] (2022) analyzed a quantum version of this algorithm, obtaining a polynomial speed-up with a final complexity of $O(k\, n \log n + k\frac{\sqrt{m}}{\xi}\log(\frac{3km}{\delta}))$, considering $\delta$ probability of success and $\xi$ a precision parameter on the output. Our work extends this result by considering different input models and exploiting the use of the block-encodings framework.

### 1.2.3. Orthogonal Matching Pursuit

As anticipated in the previous section, Orthogonal Matching Pursuit algorithm (OMP) [61] is a popular extension of the well-known Matching Pursuit algorithm [52]. As its predecessor, OMP is a greedy algorithm that, through the use of local minimization, finds an approximation of a signal obtained as a sparse combination of some atoms of a dictionary.

#### Algorithm

The main difference between Orthogonal Matching Pursuit and Matching Pursuit consists in the update of the solution obtained by performing an orthogonal projection of the signal on the subspace spanned by the atoms selected so far. In this way, each atom is never selected more than once and the coefficients of all the selected atoms are updated at each iteration.

Let us denote the signal with $s \in \mathbb{R}^n$, the solution vector with $x \in \mathbb{R}^m$, the dictionary matrix with $D \in \mathbb{R}^{n \times m}$, the approximation threshold with $\epsilon \in \mathbb{R}^+$ and the sparsity threshold with $L \in \mathbb{N}$. We define also an iteration index $t \in \mathbb{N}$, the set $\Lambda_t \subset \mathbb{N}$ of the indices of the selected atoms at iteration $t$ and a vector of residuals $r \in \mathbb{R}^n$.

During the initialization, we set the residual vector equal to the signal and the solution to the zero vector

$$r = s, \tag{1.16}$$

$$x = 0^{\otimes m}. \tag{1.17}$$

We also initialize the counter and the empty set, used to store the indices of the selected atoms

$$t = 0, \tag{1.18}$$

$$\Lambda_0 = \emptyset. \tag{1.19}$$

Once we have initialized the values, the algorithm looks for the atom in the dictionary with the greatest correlation that has not already been selected

$$j* = \arg_{\mathrm{j} \in [\mathrm{m}] \setminus \Lambda_\mathrm{t}} \max |\langle r, d_j \rangle|. \tag{1.20}$$

After selecting the best atom, the iteration counter and the indices list are updated

$$t = t + 1, \tag{1.21}$$

$$\Lambda_t = \Lambda_{t-1} \cup j^*. \tag{1.22}$$

We build the matrix $A_t$ using as column vectors the atoms with the indices in $\Lambda_t$ in the following way. Given $\Lambda_t = \{j_1, ..., j_t\}$, we create

$$A_t = \begin{bmatrix} d_{j_1} & ... & d_{j_t} \end{bmatrix}. \tag{1.23}$$

At this point, the algorithm must resolve the following OLS problem

$$x_t = \arg_x \min \|s - A_t x\|. \tag{1.24}$$

The solution $x_t$ would have $t$ non-zero components. The residual is updated as

$$r = s - A_t x. \tag{1.25}$$

The stop condition of the algorithm is the following

$$\|x\|_0 > L \text{ or } \|r\| \leq \epsilon. \tag{1.26}$$

As mentioned at the beginning of this section, an interesting property of the algorithm is that the projection of Equation 1.24 avoids selecting the same atom more than once. For this reason, we can consider the $l_0$ norm of the solution equivalent to the number of iterations $t$.

Another interesting observation is that this least square problem admits a closed solution of the form

$$P_t = (A_t^T A_t)^{-1} A_t^T = A_t A_t^+, \tag{1.27}$$

such that $r = s(\mathbb{I} - P_t) = s - A_t x$.

The procedure is summarized by Algorithm 1.2.

Algorithm 1.2 Orthogonal Matching Pursuit

**Input** $s \in \mathbb{R}^n$ signal, $D \in \mathbb{R}^{n \times m}$ dictionary, $L \in \mathbb{N}$ sparsity threshold, $\epsilon \in \mathbb{R}^+$ error reconstruction tolerance

**Output** $x \in \mathbb{R}^m$, an approximated solution of $\mathcal{P}_0^\epsilon$ (Def. 1.1)

Initialize $r = s$, $x = 0^{\otimes m}$, $t = 0$, $\Lambda_0 = \emptyset$
**while** not $(t > L$ or $\|r\| \leq \epsilon)$ **do**
  **for all** $j \in ([m] \setminus \Lambda_t)$ **do**
    Compute $\langle d_j, r \rangle$
  **end for**
  Select $j^* = \arg_j \max(|\langle d_j, r \rangle|)$
  Update $t = t + 1$
  Update $\Lambda_t = \Lambda_{t-1} \cup j^*$
  Construct $A_t$ concatenating the atoms of $D$ with the indices in $\Lambda_t$
  Compute $x = \arg_{x'} \min \|s - A_t x'\|$
  Compute $r = s - A_t x$
**end while**
Output $x$

## Computational complexity

The initialization has complexity $O(n)$ because we initialize the vector $r = s$. The complexity of the *sweep stage*, with a naive approach, is $O(nm)$ as for the Matching Pursuit algorithm. The various updates of the lists have complexity $O(1)$. The update of the solution, assuming that the cost of inverting a $n \times k$ matrix is in the worst case $O(k^3)$ [18], has complexity bounded by the solution of the least square, $O(nk^2 + k^3)$. The update of the residual has complexity $O(nk)$ because of the multiplication between the matrix $A_t \in \mathbb{R}^{n \times k}$ and the solution vector $x \in \mathbb{R}^k$. The complexity can be dominated by the sweep stage or by the update of the solution, depending on the dimensions of the dictionary, the signal, and the number of iterations needed.

Therefore, assuming that the orthogonal matching pursuit converges after $k$ iteration, its asymptotic computational complexity without any optimization scales as

$$O(k(nm + nk^2 + k^3)) \tag{1.28}$$

with a memory use of $O(nm)$.

Considering the use of techniques such as the Cholesky decomposition [94] we can obtain

a better complexity that corresponds to

$$O(k(mk + k^2)) \tag{1.29}$$

with required memory of $O(m^2 + nm + k^2)$. Another possibility is the use of the Matrix Inversion Lemma [72] with a final complexity of

$$O(k(nk + mk)) \tag{1.30}$$

and a required memory of $O(m^2 + nm + nk)$.

## Related work

The first version of Orthogonal Matching Pursuit was created by Pati et al. [61] (1993) and is based on the Matching Pursuit algorithm [52]. The main difference between the two is that OMP guarantees to avoid the selection of the same atom more than once. This happens because the solution vector is not updated but completely recomputed solving an Ordinary Least Square problem after the selection of the best atom at each iteration. As explained above, the running time of this algorithm corresponds to $O(k(nm + nk^2 + k^3))$.

Different and faster versions of the OMP algorithm had been proposed. We report here the most relevant ones related to our work.

Needell and Vershynin (2009) had created a regularized version of OMP called ROMP [59]. In this algorithm, thanks to the validity of the Restricted Isometry property for the dictionary, which we will explain better later in these sections, it is possible to recover the signal with guaranteed uniformity and stability [59]. Combining this result with other techniques that help to maintain the sparsity of the solution, Needell and Tropp have also developed another variant of the OMP algorithm known as CoSaMP[58] (2009). This version has a final complexity of $O(n \log^2 n)$ for cases in which the RIP property is satisfied and the signal is sparse. As before, the worst-case complexity corresponds to the same one of OMP.

Donoho et al. (2012) had developed another version called stagewise OMP, or StOMP [26]. This algorithm selects multiple atoms at each step based on a threshold value. The algorithm provides the same solution of OMP with a lower running time of $O(svnm)$, where $s$ and $v$ are small parameters that depend on the dataset.

Instead, if we focus on speed-up on the general OMP algorithm different techniques had been developed. One of these is the version of OMP obtained using Cholesky decompo-

sition [94] (2012). This version exploits the Gram matrix generated by the dictionary to avoid the direct matrix inversion, obtaining a final complexity of $O(k(mk+k^2))$. Another interesting implementation of OMP is the one using QR decomposition [72] (2012). This version is based on the factorization of the dictionary into two matrices, $Q$ and $R$, obtaining a running time of $O(k(mk+nk+k^2))$. Finally, another implementation of OMP based on the Matrix Inversion Lemma (MIL) [72], that again exploits the use of the Gram matrix, with complexity $O(k(nk+mk))$.

Another way to obtain a sparse representation is to use an $l_1$-minimization approach such as Lasso [75]. Some works [14, 24, 85], have demonstrated that the minimization of the $l_1$-*norm* can also minimize the $l_0$-*norm* of the solution if some conditions are satisfied.

In addition, from how much we know, there are no works providing a Quantum version of the Orthogonal Matching Pursuit algorithm. If we also consider the results related to the $l_1$-*norm* relaxation, the recent work from [17] provides a quantum algorithm for LASSO with complexity $\widetilde{O}(\sqrt{m}/\epsilon^2)$, with classical inputs and outputs.

## Theoretical guarantees

### Mutual Incoherence

An interesting result about the connection between the complete recovery of a signal and the mutual incoherence of the dictionary had been studied in past literature [25, 64, 82].

**Definition 1.3** (Mutual Incoherence [25]). *For $n$ vectors $x_i \in \mathbb{R}^m$, the mutual incoherence $\gamma$ is the largest absolute value of normalized correlation between these vectors.*

$$\gamma = \max_{i \neq j} \frac{|\langle x_i, x_j \rangle|}{\|x_i\|_2 \|x_j\|_2} \text{ with } i, j \in \{0, ..., n-1\} \text{ and } i \neq j. \tag{1.31}$$

We highlight that the mutual incoherence value is low if the atoms are not similar to each other. At this point, we can define different properties related to this value.

**Theorem 1.4** (Condition for exact recovery [25]). *Given $D \in \mathbb{R}^{n \times m}$ a matrix with $l_2$-normalized columns and $b \in \mathbb{R}^n$. Let $x \in \mathbb{R}^m$ such that $Dx = b$, if $D$ and $x$ satisfy the following inequality*

$$\gamma < \frac{1}{2s-1} \tag{1.32}$$

*where $\gamma$ is the mutual incoherence of the column vectors of $D$ and $s$ is the sparsity of $x$, then we can recover the solution $x$ by OMP without reconstruction error after $s$ iterations.*

We can also define a property on the uniqueness of the solution.

**Theorem 1.5** (Condition on uniqueness of the solution[4]). *Given $D \in \mathbb{R}^{n\times m}$ a matrix with $l_2$-normalized columns and $b \in \mathbb{R}^n$. If $Dx = b$, $x \in \mathbb{R}^m$ and we know that the sparsity $s$ of the solution is*

$$s < \frac{1}{2\gamma} \tag{1.33}$$

*where $\gamma$ is the mutual incoherence of the column vectors of $D$, $x$ is the unique solution of Problem $\mathcal{P}_0^0$ (Def. 1.1).*

### Restricted Isometry Property

Another important property exploited to obtain a speed-up in some variations of the OMP algorithm is the use of the Restricted Isometry Property (RIP).

**Definition 1.6** (Restricted Isometry Property [12]). *Let $D$ be an $n \times m$ matrix and let $1 \leq t \leq m$ be an integer. We can say that the matrix satisfies the Restricted Isometry Property if exists a constant $\delta_t \in (0, 1)$ such that for every $n \times t$ submatrix $D_t$ of $D$ and for every $n$-dimensional vector $x$ we have that*

$$(1 - \delta_t)\|x\|^2 \leq \|D_t x\|^2 \leq (1 + \delta_t)\|x\|^2. \tag{1.34}$$

*We say that the RIP of order $K$ is satisfied if for all $x$ holds that $\|x\|_0 \leq K$.*

It is possible, using the already mentioned algorithms ROMP and CoSaMP for a RIP of order $CK$ where $C \geq 2$ depends on the algorithm and $\delta$ is low enough, to recover exactly any $K$-sparse signal [58, 59]. The RIP property of order $K + 1$ with isometry constant $\delta < \frac{1}{3\sqrt{K}}$ can be used also for a straightforward implementation of OMP for signals without noise and is enough for the exact recovery of a $K$-sparse signal in $K$ iterations as shown in [23]. Different works have then improved the bounds on the isometry constant to $\delta < \frac{1}{\sqrt{K+1}}$ [55, 56].

### 1.2.4. Eigenvectors Classification/Recognition

In this work, we describe a simple implementation of a quantum algorithm that performs Eigenvector Classification and, in the experiment Section 5.4.1, as an application of it, we also show how Eigenfaces Recognition [77] is performed. In the following sections, we focus primarily on the use in the general classification task, and we generalize the necessary concepts to draw a parallel between it and face recognition.

#### Algorithm

Given a signal $s \in \mathbb{R}^n$ that we want to classify, it is possible to do it more efficiently if we have previous knowledge about similar data. The first step, which in this algorithm we consider a pre-computation step, is to extract a limited amount of information from a training dataset but, at the same time, try to make the description capture the greatest amount of variance. The selection of this information, also known as components, can be performed using Principal Component Analysis (PCA).

In this algorithm, PCA can be used to extract the eigenvectors of the covariance matrix that is obtained using the samples of the training dataset. The elements of this dataset belong to a specific *category*, for example face images or network anomalies. Let $A \in \mathbb{R}^{n \times m}$ be a matrix composed by $m$ of the eigenvectors of the covariance matrix of dimension $n$ and let $\mu \in \mathbb{R}^n$ be the mean vector of the training dataset. We consider $A$ and $\mu$ as inputs for the algorithm.

Firstly, we subtract the mean vector $\mu$ from the test sample $s$. After that, we multiply the result by the transposed eigenvectors matrix

$$w = A^T(s - \mu). \tag{1.35}$$

In this way, we obtain the vector $w \in \mathbb{R}^m$ such that each of its components corresponds to an inner product between the sample and an eigenvector and describes how much the corresponding column of $A$ is similar to the sample. For this reason, we call the vectors obtained in this way *weights vectors*.

To classify our test sample, we need the weights vectors of other samples already divided into classes, computed, and stored. For this reason, we consider $V = \{v_1, ..., v_p\}$ the set of the already computed weights vectors with $v_j \in \mathbb{R}^m \ \forall j \in [p]$. We also consider that these $p$ samples are divided into different classes and are obtained through the same process used to obtain $w$.

We can test if the sample $s$ is classified in the same class of a certain element whose weights are in $V$ by testing the similarity of their weights vectors. To perform this task, we compute the Euclidean distance between the weights vector of the sample and the stored ones

$$d^* = \min \|w - v_j\| \; \forall j \in p. \tag{1.36}$$

At this point, we extract the minimum value $d^*$ and its index $j^*$.

It is then possible to identify, using two parameters $\delta_1, \delta_2 \in \mathbb{R}^+$, if the sample is classified in the same class as one sample already in the system, can be added as a new sample with a different class but of the same category or it is not a sample related the problem that we are analyzing. The output of the procedure will be

- if $d^* \leq \delta_1 \rightarrow$ recognized in the same class of the sample of index $j^*$,
- if $\delta_1 < d^* < \delta_2 \rightarrow$ recognized as an element of the same category but not in a class,
- if $d^* \geq \delta_2 \rightarrow$ not an element of the category.

We summarize the procedure in Algorithm 1.3.

Algorithm 1.3 Eigenvectors based Classification/Recognition

**Input**: $A \in \mathbb{R}^{n \times m}$ matrix of eigenvectors, $\mu \in \mathbb{R}^n$ mean vector of the starting dataset. $s \in \mathbb{R}^n$ sample to classify, $V = \{v_1, ..., v_p\}$ set of stored weights vectors with $v_j \in \mathbb{R}^m \; \forall j \in [p]$. $\delta_1, \delta_2 \in \mathbb{R}^+$ used as threshold for the outputs.

**Output**: if $d^* \leq \delta_1$: *recognized in the same class of* $v_{j^*}$,
if $\delta_1 < d^* < \delta_2$: *recognized as similar element,*
if $d^* \geq \delta_2$: *not a similar element.*

1: Compute the weights vector of the test sample:
$w = A^T(s - \mu)$
2: Select the minimum distance between the weights vector of the test sample and the weights already stored:
$d^* = \min \|w - v_j\| \; \forall j \in [p]$
3: Save the index of the closer weights vector:
$j^* = \arg\min_j \|w - v_j\| \; \forall j \in [p]$
4: Output: if $d^* \leq \delta_1$: *recognized in the same class of* $v_{j^*}$,
if $\delta_1 < d^* < \delta_2$: *recognized as similar element,*
if $d^* \geq \delta_2$: *not a similar element.*

## Computational complexity

The subtraction between the sample and the mean vector has complexity $O(n)$ because it is linear in the dimension of the vectors. The product between the matrix and the vector of the test sample has complexity $O(nm)$ because it depends on the dimension of $A$. The computation of the distances between the weights vector of the test sample and the stored ones has complexity $O(pm)$ because the computation of $p$ distances of vectors of length $m$ is performed. The complexity for just the computation of the weights is $O(nm)$ and for the overall procedure is $O(nm + pm)$.

## Eigenfaces Recognition

The previous algorithm can also be used as a Face Recognition method. It is possible to follow the same procedure with just some little modifications.

We must consider as samples the images of faces with dimensions $h \times v$, where $h$ is the height in pixels and $w$ is the weight, transform them into grey-scale representations and, after that, into vectors of dimension $n = h \times v$. In addition, each class will contain only images of one subject and, if the sample is classified in a class, is recognized as the subject described by the class.

## Related work

First of all, we want to point out that the algorithm we study can be seen as an ensemble of two famous methods, PCA [54] and 1-Nearest-Neighbour [10].

The Principal Components Analysis had been first studied before the worldwide availability of electronic computers by Pearson [62](1901) and Hotelling [37](1933). After that, this method had been studied and used in a lot of different disciplines, with the general objective of reducing the dimensionality of data.

As anticipated, this algorithm is also based on the 1-Nearest-Neighbour algorithm. This method is one of the oldest algorithms studied to perform classification. It is based on computing the similarity of two samples and had been vastly studied and applied in different fields. One of the most important extensions used in Machine Learning is the method known as K-nearest-neighbour, in which the $K$ different closest samples are considered for the classification instead of only the closest. This algorithm was first developed in 1951 by Fix and Hodges [31], and extended in 1967 by Cover and Hart [19].

The application of these methods to the classification of data had been used for many years. In this section, we focus only on some specific applications on which we have based

our studies and experiments, Face Recognition and Anomaly Detection in network traffic.

The first intuition for the use of Principal Components for Face Recognition had been made in 1987 by Sirovich and Kirby [71]. Turk and Pentland had shown in 1991 the first Eigenfaces Recognition method [76] connecting the use of PCA to the task of classification of face images. From this first work, different other improvements had been made, for example, the use of specific different facial features to improve the performances [89] (2002). Another use that we have considered is the one related to anomaly detection in network traffic [93]. Also in this case, there are plenty of results for classification with different metrics and nearest neighbors methods as K-NN [48, 73].

Focusing on quantum algorithms, the closer algorithm to ours is the one developed by Wiebe et al. [83] (2014). In this work, the authors define a nearest-neighbor classifier based on the Euclidean distance between the sample and some centroids (defined as the mean samples for each class) of complexity $O(\sqrt{p}\log(p)\,d\,r^2/\epsilon)$, where $p$ is the number of centroids, $r$ the biggest entry of the vectors considered, $d$ the sparsity of the test sample and $\epsilon$ an error parameter. This routine is defined without the use of the novel routines related to the block-encoding frameworks, without considering performing dimensionality reduction before the computation, and with a dependency in the running time on the sparsity of the input and the maximum entry of the vectors that we avoid with our algorithm. In the QSFA [40] algorithm the authors also reduce the dimensionality of the data and perform clustering with complexity $\widetilde{O}(K/\eta^2)$ where $K$ is the number of clusters and $\eta$ a precision parameter. This procedure exploits a classifier to check the distance with the centroids based on the Frobenius distance. Finally, another important algorithm to mention is the one by Dang et al. [21]. This quantum algorithm is a K-NN method for image classification based on dimensionality reduction. The complexity of this algorithm is described using the running time of oracles that perform different operations, such as amplitude estimation, as the basic unit. In this way, the algorithm performs the task in $O(\sqrt{kMT})$ where $k$ is the number of neighbours considered, $M$ the dimension of the sample, and $O(T)$ the complexity of the oracle. With our algorithm, we consider only the nearest neighbour for the classification and we express in a more precise way the complexity of each step.

In addition to them, there are a big number of other quantum algorithms used to perform classification and clustering such as Q-SVM [65] and Q-Means [43]. This work can be considered as a possible way to use the output provided by the already studied Q-PCA [49] method to perform classification or recognition tasks.

# 2 | Quantum Background

This chapter introduces the main quantum computing concepts we use in this work. We want to avoid an extensive explanation of all quantum computing concepts that would require an entire book itself, so we just summarize the most important concepts. We advise the reader to consult Nielsen and Chuang [60] or Kaye et al. [39] for a more complete explanation of the subject. We have extracted and summarized this explanation from these works.

After a description of the notation and the main concepts, we proceed by explaining the most important quantum routines used in Quantum Machine Learning.

## 2.1. Quantum Information and Computation

Quantum information is the result of reformulating the rules of classical information using the new tools provided by quantum mechanics. First, we introduce some basic concepts about the most used notation to refer to a quantum state, and then we describe how it is possible to combine these states and how to apply operations.

### 2.1.1. Bra-Ket Notation

The most commonly used notation is called Bra-Ket, or Dirac's, notation. This notation makes the algebraic operations simpler to understand by hiding the complexity of the denoted objects.

We now introduce the notation used to represent a vector $x \in \mathbb{C}^n$. For convention, we denote this vector with the notation *ket* $|x\rangle$ if it correspond to a column vector

$$|x\rangle = \begin{bmatrix} x_1 \\ ... \\ x_n \end{bmatrix}.$$

While the notation *bra* $\langle x|$ is used to its complex conjugate

$$\langle x| = \begin{bmatrix} x_1^* & \dots & x_n^* \end{bmatrix}.$$

We also want to recall that the conjugate transpose of a real vector is just the transposition of the vector, as the imaginary part is not present.

This notation helps to be more concise with operations such as the inner product. Considering two vectors $x, y \in \mathbb{C}^n$, represented by $|x\rangle$ and $|y\rangle$, their inner product can be denoted by $\langle x|y\rangle$.

$$\langle x|y\rangle = \begin{bmatrix} x_1^* & \dots & x_m^* \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix} = \sum_{i=0}^{n} x_i y_i^*.$$

As it is possible to denote in a compact way the inner product, the same is possible for the outer product. The notation, using two vectors that can have different dimensions $x \in \mathbb{C}^n$, $y \in \mathbb{C}^m$, is $|x\rangle \langle y|$.

$$|x\rangle \langle y| = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} \cdot \begin{bmatrix} y_1^* & \dots & y_m^* \end{bmatrix} = \begin{bmatrix} x_1 y_1^* & \dots & x_1 y_m^* \\ \dots & & \dots \\ x_n y_1^* & \dots & x_n y_m^* \end{bmatrix}.$$

Another frequently used notation is the one that refers to the tensor product. Considering the same states as before, the tensor product between the two vectors corresponds to

$$|x\rangle \otimes |y\rangle = |xy\rangle = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} \otimes \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix} = \begin{bmatrix} x_1 & \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix} \\ \dots & \dots \\ x_n & \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix} \end{bmatrix} = \begin{bmatrix} x_1 y_1 \\ \dots \\ x_{n-1} y_{n-1} \\ \dots \\ x_n y_m \end{bmatrix}.$$

### 2.1.2. Quantum Bits

We consider the *bit* as the unit of information for the classical computation. Looking at quantum computation, we can consider a different unit of information called *qubit*. While classical bits can have only value 0 or 1, a qubit represents a quantum state that can be described by a unit vector in complex space, $|\phi\rangle \in \mathbb{C}^2$. This state is considered a linear combination of two basis states, usually $|0\rangle$ and $|1\rangle$. The qubit can be in state $|0\rangle$, $|1\rangle$ or

in a superposition of the two

$$\alpha\,|0\rangle + \beta\,|1\rangle$$

where

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}.$$

We call *amplitudes* the values $\alpha$ and $\beta$, and their squared values represent the probability of measuring $|0\rangle$ or $|1\rangle$ as the final result. It is important to highlight that since this notation represents a distribution of probability, we have that

$$|\alpha|^2 + |\beta|^2 = 1.$$

During the computation, all the superpositions are possible, and in this way, a qubit represents more information than a classical bit.

As introduced before, the couple of states $|0\rangle$ and $|1\rangle$ is a basis and, to be more specific, it is called the *computational basis* or *z-basis*. A computational basis of dimension 2 is just a couple of states, $|x\rangle$, $|y\rangle$ with the following property

$$\langle x|x\rangle = 1, \langle y|y\rangle = 1,$$
$$\langle x|y\rangle = 0, \langle y|x\rangle = 0.$$

All these properties have their origin in the first postulate of quantum mechanics

**Postulate 2.1** (First Postulate of Quantum Mechanics [60])**.** *Associated with any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.*

There are also other important bases as the *x-basis* that corresponds to

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}, |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}.$$

It is possible to move from one basis to another using some algebraic operations, for example,

$$|0\rangle = \frac{|+\rangle + |-\rangle}{\sqrt{2}}, |1\rangle = \frac{|+\rangle - |-\rangle}{\sqrt{2}}.$$

Here, as an example, we show the transformation of the representation of a generic state

in *z-basis* to its corresponding representation in *x-basis*

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha\frac{|+\rangle + |-\rangle}{\sqrt{2}} + \beta\frac{|+\rangle - |-\rangle}{\sqrt{2}} = \frac{\alpha+\beta}{\sqrt{2}}\,|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}\,|-\rangle\,.$$

If we consider a generic quantum state $|\varphi\rangle \in \mathbb{C}^2$, we can describe it as a unitary vector with the following corresponding description

$$|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,.$$

where we consider $e^{i\phi}$ the relative phase of the quantum state. We ignore the global phase of the state in this definition because it has no observable consequences on a single qubit. With this notation, it is easy to see how it is possible to describe a state as a point in spherical coordinates. The sphere on which we can visualize the quantum states is called Bloch Sphere, and we show it in Figure 2.1. A quantum state represented by a linear combination of basis states is called a *pure state*. For this kind of state sum of squared amplitudes correspond to 1. The coordinates of a pure quantum state on the Bloch sphere correspond to

$$(x, y, z) = (\sin\theta\cos\phi,\ \sin\theta\cos\phi,\ \cos\theta). \tag{2.1}$$

States that can not be represented as a combination of basis states because generated by phenomena of interference between different pure states are called *mixed state*. For these states, the sum of the amplitudes is not one. Mixed states do not correspond to points on the surface of the sphere but to points inside the Bloch Sphere. In this work, we do not consider phenomena of interference and, for this reason, we do not enter into major details about mixed states.

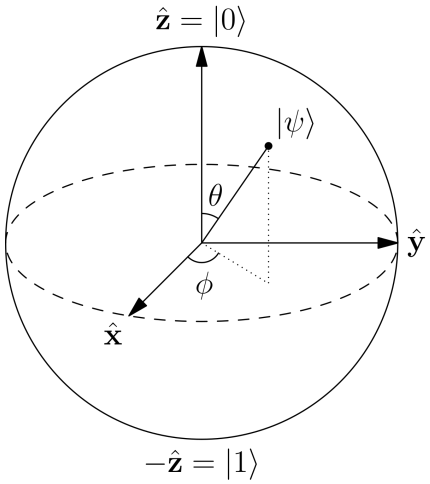



Figure 2.1: Visualization of a Qubit on the Bloch sphere [84]

### 2.1.3. Quantum Register

If we want to describe a higher amount of information, a possible way is to use more qubits together, creating a *quantum register*. A register formed by $n$-qubits can describe a vector in a complex space of dimension $2^n$, with complex amplitudes $\alpha_i \in \mathbb{C}$

$$|\varphi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = |\varphi\rangle = \alpha_0 |0\rangle + ... + \alpha_{2^n-1} |2^{n-1}\rangle . \tag{2.2}$$

such that the condition $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ is satisfied. Each value $i$, for $i \in [0, 2^n - 1]$ corresponds to a different basis state. We use binary encoding to describe the state $|i\rangle$ and, in this way, it is enough to use $n$ qubits. We show here a simple example of the *computational basis* with two qubits

$$|0\rangle = |00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |1\rangle = |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$|2\rangle = |10\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |3\rangle = |11\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

all the vectors must satisfy the unitary $l_2$-norm condition to represent correct quantum states. We can describe any quantum state of a system of 2 qubits as a linear combination of the vectors of the computational basis as

$$|\varphi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle + \alpha_3 |3\rangle . \tag{2.3}$$

### 2.1.4. Quantum Logical Gates

It is common to use a notation that hides the low-level complexity of the circuits used to describe the evolution of a quantum system. This notation exploits the use of unitary matrices. We describe here the postulate that refers to the evolution of a quantum system.

**Postulate 2.2** (Second Postulate of Quantum Mechanics [60]). *The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\phi\rangle$ of the system at time $t_1$ is related to the state $|\phi'\rangle$ of the system at time $t_2$ by a unitary $U$ which*

*depends only on the times $t_1$ and $t_2$ from the relation $|\phi'\rangle = U|\phi\rangle$.*

A unitary $U \in \mathbb{C}^{n\times n}$ is a matrix that has the property $U^\dagger U = UU^\dagger = \mathbb{I}$, where $U^\dagger$ denotes the conjugate transpose of the matrix $U$. The unitary matrices have the fundamental property of preserving the norm of the vectors to which are applied. A unitary matrix describes the rotation of a pure quantum state, represented by a vector on the Bloch sphere. Any unitary matrix that can be defined corresponds to a valid reversible quantum gate or a combination of them in a quantum circuit. It is important to point out that all the quantum gates must satisfy the condition to be reversible, which means that the quantum state does not collapse on itself and it is possible to apply the inverse of the operation performed. This happens due to unitary constraints. All the quantum algorithms that can be defined are a combination of some unitary operations and some measurements applied to quantum states.

Now we analyze some of the simplest quantum logical gates for a single qubit with their corresponding unitary matrix.

The first to mention has the same functionality as a classical NOT gate and is called $X$-Gate. Its unitary matrix corresponds to

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

This gate corresponds to a rotation of 180° for the $\theta$ angle of the Bloch Sphere. Given a quantum state described as $\alpha|0\rangle + \beta|1\rangle$ if we apply the gate to it, we swap the amplitudes of the two states of the computational basis

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix},$$

$$\alpha|0\rangle + \beta|1\rangle \rightarrow \beta|0\rangle + \alpha|1\rangle.$$

In this case, it is possible to apply the same gate again to the state, returning to the initial situation. It is trivial to verify that $X^\dagger X = XX^\dagger = \mathbb{I}$. Other important gates that act in a very similar way are the $Y$-gate and the $Z$-gate, which perform a rotation of 180° on the other two axes. These three gates are called Pauli Gates.

The Hadamard gate, also known as $H$-gate, is another fundamental gate for quantum computation. This gate transforms the basis of a certain state from the *computational*

*basis* to the *x-basis*. Its unitary matrix corresponds to

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

If we apply this gate to the basis states of the computational basis, we obtain the ones of the x-basis in the following way

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle,$$
$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle.$$

We can easily extend this gate for multiple inputs. If, for example, we consider a $n$-qubit gate, we can create a uniform superposition of all the states of the computational basis

$$H_n|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle.$$

This property is useful to generate a superposition that can be used as input for a quantum algorithm.

In addition to these simple gates that act independently by the value of the qubits, there also exist other gates that change their behavior depending on one or more specific qubits in the input. These gates are called *controlled gates*. A controlled gate acts at least on two qubits because one is necessary to control the choice of the operation and the other one is the state on which we want to apply the operation. In general, it is quite common to define these gates as *controlled rotations* because they apply a unitary, which performs a rotation only if the condition on the control qubits is satisfied.

It is easy to define the generic controlled gate $CU$ for two qubits. We start from the unitary $U$ that represents the operation that we want to perform on the qubit that we want to modify as

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}.$$

The complete controlled-$U$ is defined in the following way

$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}.$$

Applying this unitary on the basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ we obtain the following results

$$CU\,|00\rangle = |00\rangle\,,$$
$$CU\,|01\rangle = |01\rangle\,,$$
$$CU\,|10\rangle = \mathbb{I}\,|1\rangle \otimes U\,|0\rangle = |1\rangle \otimes (u_{00}\,|0\rangle + u_{10}\,|1\rangle),$$
$$CU\,|11\rangle = \mathbb{I}\,|1\rangle \otimes U\,|1\rangle = |1\rangle \otimes (u_{01}\,|0\rangle + u_{11}\,|1\rangle).$$

One of the most used gates is the $CNOT$ gate in which the unitary $U$ corresponds to the $X$-gate.

### 2.1.5. Measurement of Quantum States

It is not easy to retrieve the classical values represented by a quantum state. First of all, we recall the third postulate of quantum mechanics that guarantees the possibility of measuring a state but also explains how this measure modifies the state itself.

**Postulate 2.3** (Third Postulate of Quantum Mechanics [60]). *Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. The index $m$ refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that results in $m$ occurs is given by $p(m) = \langle\psi|\, M_m^\dagger M_m\,|\psi\rangle$, and the state of the system after the measurement is*

$$\frac{M_m\,|\psi\rangle}{\sqrt{\langle\psi|\, M_m^\dagger M_m\,|\psi\rangle}}.$$

*The measurement operators satisfy the completeness equation $\sum_m M_m^\dagger M_m = \mathbb{I}$. This equation expresses the fact that probabilities sum to one $\sum_m p(m) = \sum_m \langle\psi|\, M_m^\dagger M_m\,|\psi\rangle = 1$.*

We provide here an example to be more clear in the explanation. Given a quantum state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ if we measure the state with respect to the *computational basis* $\{|0\rangle\,, |1\rangle\}$ we obtain the following situation.

Our set of measurement operators corresponds to

$$\{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$$

for which it is trivial to verify the completeness equation, and we have

$$M_0^\dagger M_0 + M_1^\dagger M_1 = |0\rangle\langle 0||0\rangle\langle 0| + |1\rangle\langle 1||1\rangle\langle 1| = |0\rangle\langle 0| + |1\rangle\langle 1| = \mathbb{I}.$$

If we want to compute the probability of measuring the state $|0\rangle$, we obtain

$$p(0) = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = (\alpha^* \langle 0| + \beta^* |1\rangle)|0\rangle\langle 0|(\alpha|0\rangle + \beta|1\rangle) =$$
$$= \alpha^*\alpha\langle 0||0\rangle\langle 0||0\rangle + \alpha^*\langle 0||0\rangle\langle 0||1\rangle + \beta^*\alpha\langle 1||0\rangle\langle 0||0\rangle + \beta\beta^*\langle 1||0\rangle\langle 0||1\rangle = |\alpha|^2.$$

Similarly, we obtain the probability $|\beta|^2$ of measuring the state $|1\rangle$. As we have already anticipated, the sum of these two probabilities gives 1. If we measure the state, the system changes and the superposition collapses on one single quantum state of the computational basis that we have chosen to perform the measurement.

If a procedure provides only one quantum state as output, since the measure modifies the state itself, we can think of copying it before measuring. However, this is not possible due to the following theorem.

**Theorem 2.4** (No cloning [60]). *There is no quantum algorithm that can create a copy of a quantum state.*

This happens because if we measure the state, the algorithm is not reversible anymore, and we cannot describe it using a unitary transformation. For this reason, to retrieve the complete description of a quantum state, the value of all its amplitudes, we need to construct from scratch multiple times the same state and measure all of them. This process is called *Quantum Tomography*.

## 2.2. Quantum Machine Learning

In this section, we explain the representation of the data in quantum machine learning, the QRAM model, and the block-encoded framework. After that, we proceed with a summary of all the routines that we have used in this work, and, for brevity, we do not explain the functioning of each routine. If the reader is interested in examining them more in detail, we provide references.

### 2.2.1. Quantum Data Representation

Considering quantum computing, data can be represented in different ways. We have already introduced the concepts of *quantum states* and *quantum registers*. It is possible to use the qubits in the same way as classical bits to encode in *binary encoding* a scalar number. For example, the number 5 could be encoded using 3 qubits in the form $|101\rangle$. Each qubit contains just the equivalent of a classical bit. To be more general the encoding of a certain scalar $x \in \mathbb{N}$ corresponds to

$$|x\rangle = |b_1\rangle \otimes ... \otimes |b_n\rangle \tag{2.4}$$

where $b_i \in \{0, 1\}$. For what concerns the dimension, we need $n$-qubits for natural numbers in the range $[0, 2^n - 1]$ as a normal binary encoding. It is also possible to use one extra qubit to store the sign.

The important benefit compared to the classical systems is the one related to storing vectors or matrices. In this case, another way to store the information is provided, called *amplitude encoding*. For example, we can represent a real vector $x \in \mathbb{R}^n$ as

$$|x\rangle = \frac{1}{\|x\|} \sum_{i=0}^{n-1} x_i |i\rangle = \frac{1}{\|x\|} \begin{bmatrix} x_0 \\ ... \\ x_{n-1} \end{bmatrix}$$

where we store each element of the vector, the values $x_i$, in the amplitudes of the different states $|i\rangle$. In this case, it is important to notice how we need a reduced space to store the vector. It is enough to use $\lceil \log(n) \rceil$ qubits to store a vector of size $n$. Once we have a classical register containing the norm of the vector $x$ and a quantum register of dimension $\lceil \log(n) \rceil$, we can fully describe the state. We show here an example.

Given

$$a = \begin{bmatrix} 5 \\ 4 \\ 3 \\ 0 \end{bmatrix}$$

The corresponding amplitude encoding is

$$\|a\| = \sqrt{50}, \quad |a\rangle = \frac{5}{\sqrt{50}} |00\rangle + \frac{4}{\sqrt{50}} |01\rangle + \frac{3}{\sqrt{50}} |10\rangle$$

the coefficient for the term $|11\rangle$ corresponds to 0. Here $\lceil \log(4) \rceil = 2$ qubits are used for the amplitude encoding.

Also, matrices can be represented easily with this type of encoding. We can represent a matrix $A \in \mathbb{R}^{n \times m}$ as the quantum state

$$|A\rangle = \frac{1}{\|A\|_F} \sum_{i}^{n-1} \sum_{j}^{m-1} a_{i,j} |i\rangle |j\rangle = \frac{1}{\|A\|_F} \begin{bmatrix} a_{0,0} \\ \dots \\ a_{0,m-1} \\ \dots \\ a_{n-1,0} \\ \dots \\ a_{n-1,m-1} \end{bmatrix}.$$

In this case, analogously with the storing of vectors, we need just a classical register to store $\|A\|_F$ and $\lceil \log(nm) \rceil$ qubits.

## QRAM access model

A fundamental element that we use in our work is the *Quantum Random Access Memory*, or *QRAM* [34]. This device has the capability of storing a classical description of quantum states. This memory acts similarly to the classical RAM, guaranteeing efficient quantum access to stored classical data. First of all, we must define what we mean by efficient quantum access.

**Definition 2.5** (Efficient Quantum Access to a Matrix [7])**.** *We say to have efficient quantum access to a matrix $A \in \mathbb{R}^{n \times m}$ if we can perform the following mappings in time $O(\text{polylog}(nm))$:*

- $U : |i\rangle |0\rangle \to |i\rangle |a_{i,\cdot}\rangle = |i\rangle \frac{1}{\|a_{i,\cdot}\|} \sum_{j}^{m} a_{i,j} |j\rangle$ , $for\ i \in \mathbb{R}^n$,

- $V : |0\rangle \to \frac{1}{\|A\|_F} \sum_i^n \|a_{i,\cdot}\| \, |i\rangle$.

If we combine the unitaries defined above, we obtain

$$|A\rangle = U(V \otimes \mathbb{I}) \, |0\rangle \, |0\rangle = \frac{1}{\|A\|_F} \sum_i^n \sum_j^m a_{i,j} \, |i\rangle \, |j\rangle = \frac{1}{\|A\|_F} \sum_i^n \|a_{i,\cdot}\| \, |i\rangle \, |a_{i,\cdot}\rangle$$

using just two registers of dimensions $\lceil \log(n) \rceil + \lceil \log(m) \rceil$. A useful data structure defined to provide efficient quantum access to stored states is the *KP-Tree*. Kerenidis and Prakash explain how this structure works in [41, 42], we report here its formal definition and an example of its functioning.

**Theorem 2.6** (QRAM Data structure/KP-Tree [41, 42])**.** *Let $A \in \mathbb{R}^{n \times m}$ be a matrix. Entries $(i, j, a_{i,j})$ arrive in the system in some arbitrary order, and $w$ denotes the number of entries that have already arrived in the system. There exists a data structure to store the matrix $A$ with the following properties:*

- *The size of the data structure is $O(w \log^2(nm))$.*
- *The time to store a new entry $(i, j, a_{i,j})$ is $O(\log^2(nm))$.*
- *A quantum algorithm that has quantum access to the data structure can perform the mapping $U : |i\rangle \, |0\rangle \to |i\rangle \, |a_{i,\cdot}\rangle = |i\rangle \frac{1}{\|a_{i,\cdot}\|} \sum_j^m a_{i,j} \, |j\rangle$ for $i \in [n]$, corresponding to the rows of the matrix currently stored in memory and the mapping $V : |0\rangle \, |j\rangle \to |\widetilde{A}\rangle \, |j\rangle = \left( \frac{1}{\|\widetilde{A}\|_F} \sum_i^n \|a_{i,\cdot}\| \, |i\rangle \right) |j\rangle$ for $j \in [m]$, where $\widetilde{A} \in \mathbb{R}^n$ has entries $\widetilde{A}_i = \|a_{i,\cdot}\|$ in time* $\text{polylog}(mn)$.

This structure consent to create quantum states efficiently and a dataset stored in it occupies space with a linear dependence on the number of its entries. The fact that the time to *insert*, *update*, or *remove* an already stored element is logarithmic in the dimension of the dataset is another crucial feature of this structure. In this work, we will frequently suppose to have matrices stored in this kind of structure.

Here we report an example from [41] where the authors show how it is possible to generate a quantum state from a normalized vector stored in this data structure.

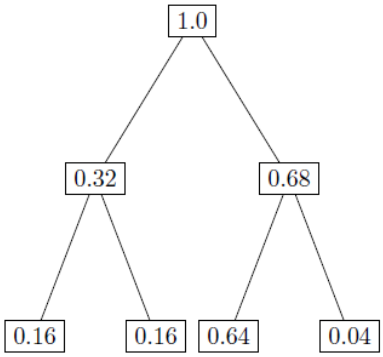



Figure 2.2: KP-Tree used to store a 4-dimensional state $|\psi\rangle$ [41]

In Figure 2.2, it is possible to see a simple schema of a KP-tree that stores the quantum state

$$|\psi\rangle = 0.4\,|00\rangle + 0.4\,|01\rangle + 0.8\,|10\rangle + 0.2\,|11\rangle\,.$$

As it is possible to see, the leaves store the squared amplitudes of the basis states. We obtain this state by applying some rotation on specific qubits. First of all, we start from the state $|0\rangle\,|0\rangle$ and we apply a rotation on the first qubit obtaining

$$|0\rangle\,|0\rangle \to (\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)\,|0\rangle\,.$$

After that, we apply another rotation on the second qubit, conditioned on the amplitudes of the first one, obtaining

$$\begin{aligned}
&(\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)\,|0\rangle \to \\
&\to \sqrt{0.32}\,|0\rangle\,\frac{1}{\sqrt{0.32}}(0.4\,|0\rangle + 0.4\,|1\rangle) + \sqrt{0.68}\,|1\rangle\,\frac{1}{\sqrt{0.68}}(0.8\,|0\rangle + 0.2\,|1\rangle) \to \\
&\to 0.4\,|00\rangle + 0.4\,|01\rangle + 0.8\,|10\rangle + 0.2\,|11\rangle\,.
\end{aligned}$$

that corresponds to our required quantum state.

Chen and de Wolf [17] have shown how it is possible to modify the KP-Tree structure to obtain a structure that can efficiently update the description of a certain vector $\theta \in \mathbb{R}^d$ to the one of $a\theta + be_j$ where $a, b \in \mathbb{R}$ and $j \in [d]$. In addition to that, to be more efficient, this version of the algorithm for the creation of quantum states can recognize when a node is equal to zero and avoid the application of the rotation in that case. Finally, it is possible to store also not normalized vectors by placing in the root node the value of the norm.

In Figure 2.3a, it is possible to see a tree that stores the value of a vector $\theta$, and in Figure

2.3b, it is possible to see the tree of the vector $\frac{4}{5}\theta + 6e_4$, after the update.


(a) KP-Tree of vector $\theta$

(b) KP-Tree of vector $\frac{4}{5}\theta + 6e_4$

Figure 2.3: Update of a vector stored in a KP-Tree [17]

As it is clear, the nodes with value 0 are not represented because, for the algorithm, it is enough to recognize that the root of the branch is zero to understand how to modify the construction routine. For a more precise explanation of the technical part of this procedure, we advise the reader to consult Definition 2.9 and Theorem 2.12 of [17].

## Block-encodings framework

Now we introduce one of the most important frameworks for our work. This framework had been first formalized by Chakraborty et al. [15], Gilyén et al. [33] starting from the result, related to hamiltonian simulations, of Low and Chuang in [50]. After this introduction, lots of different works used this framework to solve machine learning-related tasks [16, 40, 57]. We first introduce what a block-encoding is.

**Definition 2.7** (Block-encoding [15]). *Suppose that $A$ is an $s$-qubit operator, $\alpha, \epsilon \in \mathbb{R}^+$ and $q \in N$, then we say that the $(s+q)$-qubit unitary $U_A$ is an $(\alpha, q, \epsilon)$ block-encoding of $A$, if*

$$\|A - \alpha(\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})\| \leq \epsilon,$$

*where $\|\cdot\|$ is the operator norm.*

In matrix form, we can consider the subnormalised matrix $A$ as the top-left block of $U_A$

$$U_A = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

It is possible to define a high number of different routines to sum two block-encodings together, multiply them or apply a block-encoding to quantum states. These routines and their running time is analyzed in the following sections. To proceed, we must first define a value that appears in the following definition.

**Definition 2.8** (Parameter $\mu_p$). *Let $A \in \mathbb{C}^{n\times m}$. Fix $p \in [0,1]$. We define the parameter $\mu_p(A)$ as*

$$\mu_p(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}$$

*where $s_q(A) = \max_j \|a_{j,\cdot}\|_q^q$ is the q-th power of the maximum q-norm of any row of $A$.*

There are different ways, as explained in [15], to obtain a block-encoding of a matrix. We focus here on how it is possible to create a block-encoding starting from data stored in QRAM.

**Theorem 2.9** (Block-encoding from QRAM data structure [16, 41, 42]). *Given $A \in \mathbb{C}^{n\times m}$. Let $A^{(p)}$ denote the matrix of the same dimension of $A$ with $a_{i,j}^{(p)} = (a_{i,j})^{(p)}$.*

- *Fix $p \in [0,1]$. If $A \in \mathbb{C}^{n\times m}$, and $A^{(p)}$ and $(A^{(1-p)})^\dagger$ are both stored in quantum-accessible data structures, then there exist unitaries $U_R$ and $U_L$ that can be implemented in time $O(\text{polylog}(nm/\epsilon))$ such that $U_R^\dagger U_L$ is a $(\mu_p(A), \lceil\log(N+M+1)\rceil, \epsilon)$-block-encoding of $A$.*
- *On the other hand, if $A$ is stored in a quantum-accessible data structure, then there exist unitaries $U_R$ and $U_L$ that can be implemented in time $O(\text{polylog}(nm/\epsilon))$ such that $U_R^\dagger U_L$ is a $(\|A\|_F, \lceil\log(n+m)\rceil, \epsilon)$-block-encoding of $A$.*

The block-encodings are unitary matrices that contain any matrix in them with a small approximation error and, using the QRAM, it is possible to create them efficiently. This allows to perform fast arithmetic operations between matrices and fast singular value transformation as explained in [33]. These results serve as the foundation for the main linear algebra routines and, for this reason, this framework is ideal for our applications as well as many others.

### 2.2.2. Quantum Amplitude Amplification and Estimation

Quantum amplitude amplification is a quantum routine that, given a certain quantum state, can amplify the amplitude of a certain substate reducing the amplitude of the others in which we are not interested. Quantum amplitude estimation, instead, is an algorithm used to estimate the amplitude of a certain substate.

In this work, we use a simplified version of these routines that hide the more technical details. Our objective is to achieve an overall optimal complexity, and we show that minimizing the run-time of these specific routines is not important for the final complexity of our algorithms. If the reader wants to enter more in detail on the topic, it is possible to consult various materials as [3, 11].

**Lemma 2.10** (Amplitude amplification and estimation [11, 42]). *If there is unitary operator $U$ such that $U|0\rangle^l = |\phi\rangle = \sin(\theta)|x,0\rangle + \cos(\theta)|G,0^\perp\rangle$ where $|G\rangle$ is an arbitrary garbage state and $|0^\perp\rangle$ is any state orthogonal to $|0\rangle$ then $\sin^2(\theta)$ can be estimated within error $(1 \pm \eta)$ in time $O\left(\frac{T(U)}{\eta \sin(\theta)}\right)$ and $|x\rangle$ can be generated in expected time $O\left(\frac{T(U)}{\sin(\theta)}\right)$ and has success with probability $1-\delta$ with an additional running time of $O(\log(1/\delta))$.*

Note that following this statement, if the amplitude amplification succeeds and we measure $|0\rangle$ in the ancillary register, we obtain the state $|x\rangle$ without any approximation error involved.

### 2.2.3. Quantum Block-Encodings Routines

In this section, we report and describe the routines related to the block-encoding framework used in this work.

#### Application of Block-Encodings to states

First of all, we report some theorems useful to transform a generic block-encoding to another with a different encoding.

**Corollary 2.10.1** (Uniform Block Amplification [16]). *Let Let $A \in \mathbb{R}^{n\times m}$ and $\delta \in (0,1]$. Suppose $U$ is a $(\alpha, a, \epsilon)$-block-encoding of $A$, such that $\epsilon \leq \delta/2$, that can be implemented at a cost of $T_U$. Then a $(\sqrt{2}\|A\|, a+1, \delta)$- block-encoding of $A$ can be implemented at a cost of*

$$O\left(\frac{\alpha T_U}{\|A\|}\log\left(\frac{\|A\|}{\delta}\right)\right). \tag{2.5}$$

One of the most important operations in linear algebra is the matrix-vector product. This operation has classical complexity that is linear in the dimension of the matrix. This complexity can be lowered in the quantum context, as shown in the following procedures.

We report here the complexity of applying a generic block-encoded matrix to a quantum state, which we can consider representing a vector. The result is a quantum state that corresponds to the result of the matrix-vector product. We have extended the previous

result proven by Chakraborty et al. [15, 16], adding the complexity of retrieving the norm of the solution using amplitude estimation (Lemma 2.10).

**Lemma 2.11** (Applying a Block-encoded Matrix on a Quantum State [15, 16]). *Let $A$ be an $s$-qubit operator such that its singular values lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Also let $\delta \in (0,1)$, and $U_A$ be an $(\alpha, a, \epsilon)$-block-encoding of $A$, such that*

$$\epsilon \leq \frac{\delta \|A\|}{2\kappa} \tag{2.6}$$

*that can be implemented in time $T_A$. Furthermore, suppose $|b\rangle$ be an $s$-qubit quantum state that can be prepared in $T_b$. Then we can prepare a state $|\varphi\rangle$ such that $\left\| |\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|} \right\| \leq \delta$ with success probability $\Omega(1)$ at a cost*

$$O\left(\frac{\alpha\kappa}{\|A\|}(T_A + T_b)\right). \tag{2.7}$$

*If we want to estimate the norm of the result up to a relative error $\eta$ with probability $1-\delta'$ we have a complexity of $O\left(\frac{\alpha\kappa}{\eta\|A\|}(T_A + T_b)\log(1/\delta')\right)$.*

Another possibility is to apply a pre-amplified block-encoding obtained following Corollary 2.10.1 to a quantum state to obtain, in some cases, a better complexity.

**Corollary 2.11.1** (Applying a pre-amplified Block-encoded Matrix on a Quantum State [16]). *Let $A$ be an $s$-qubit operator such that its singular values lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Also let $\delta \in (0,1)$, and $U_A$ be an $(\alpha, a, \epsilon)$-block-encoding of $A$, such that*

$$\epsilon \leq \frac{\delta \|A\|}{4\kappa}, \tag{2.8}$$

*that can be implemented in time $T_A$. Furthermore, suppose $|b\rangle$ be an $s$-qubit quantum state that can be prepared in $T_b$. Then we can prepare a state $|\varphi\rangle$ such that $\left\| |\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|} \right\| \leq \delta$ with success probability $\Omega(1)$ at a cost*

$$O\left(\frac{\alpha\kappa}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + \kappa T_b\right). \tag{2.9}$$

*If we want to estimate the norm of the result up to a relative error $\eta$ with probability $1-\delta'$ we have a complexity of $O\left(\left(\frac{\alpha\kappa}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + \kappa T_b\right)\log(1/\delta')\right)$.*

Another important lemma guarantees the robustness of the state that we prepare by applying the block-encoding to a quantum state.

**Lemma 2.12** (Robustness of state preparation [16]). *Let $A$ be an $s$-qubit operator such that its singular values lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Suppose that $|b'\rangle$ is a quantum state such that $\||b\rangle - |b'\rangle\| \leq \frac{\epsilon}{2\kappa}$ and $|\varphi\rangle$ is a quantum state such that $\left\||\varphi\rangle - \frac{A|b'\rangle}{\|A|b'\rangle\|}\right\| \leq \frac{\epsilon}{2}$. Then we have that $\left\||\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|}\right\| \leq \epsilon$.*

## Arithmetic operations with Block-Encodings

For many applications, it can be useful also to create a linear combination of block-encodings, and, for this reason, we report the following lemmas.

**Corollary 2.12.1** (Linear Combination of Two Block Encoded Matrices [16, 33]). *For $j \in \{0,1\}$, let $A_j$ be an $s$-qubit operator and $y_j \in \mathbb{R}^+$. Let $U_j$ be a $(\alpha_j, a_j, \epsilon_j)$-block-encoding of $A_j$, implemented in time $T_j$. Then we can implement a $(y_0\alpha_0 + y_1\alpha_1, 1 + max(a_0, a_1), y_0\epsilon_0 + y_1\epsilon_1)$ encoding of $y_0A_0 + y_1A_1$ in time $O(T_0 + T_1)$.*

**Lemma 2.13** (Product of block-encoded matrices [33]). *If $U$ is an $(\alpha, a, \delta)$-block-encoding of an $s$-qubit operator $A$, and $V$ is an $(\beta, b, \varepsilon)$-block-encoding of an $s$-qubit operator $B$ then $(I_b \otimes U)(I_a \otimes V)$ is an $(\alpha\beta, a + b, \alpha\varepsilon + \beta\delta)$-block-encoding of $AB$.*

As before, if we exploit the possibility of using pre-amplified block-encodings, we can also use the following lemma that, in some cases, can guarantee a smaller error in the block-encoding of the result.

**Lemma 2.14** (Product of Amplified Block-Encodings [16]). *Let $\delta \in (0, 1]$. If $U_A$ is an $(\alpha_A, a_A, \epsilon_A)$-block-encoding of an $s$-qubit operator $A$ implemented in time $T_A$, and $U_B$ is a $(\alpha_B, a_B, \epsilon_B)$-block-encoding of an $s$-qubit operator $B$ implemented in time $T_B$, such that $\epsilon_A \leq \frac{\delta}{4\sqrt{2}\|B\|}$ and $\epsilon_B \leq \frac{\delta}{4\sqrt{2}\|A\|}$. Then we can implement a $(2\|A\|\|B\|, a_A + a_B + 2, \delta)$-block-encoding of $AB$ implemented at a cost of*

$$O\left(\left(\frac{\alpha_A}{\|A\|}T_A + \frac{\alpha_B}{\|B\|}T_B\right)\log\left(\frac{\|A\|\|B\|}{\delta}\right)\right). \tag{2.10}$$

### 2.2.4. Quantum OLS and Matrix Inversion

One important routine that we will use in this work is matrix inversion. Gilyén et al. [33] have defined an efficient routine to perform this task.

**Corollary 2.14.1** (Matrix Inversion using QSVT [16, 53]). *Let $A \in \mathbb{R}^{n\times m}$ be a matrix with singular values in the range $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$ for some $\kappa \geq 1$. Let $\delta \in (0, 1]$. Let $U_A$ be an*

*$(\alpha, a, \epsilon)$-block-encoding of $A$ implemented in time $T_A$, such that*

$$\epsilon \leq \frac{\delta \|A\|^2}{2\kappa^2} \tag{2.11}$$

*Then we can implement a $\left(\frac{2\kappa}{\|A\|}, a+1, \delta\right)$-block-encoding of $A^+$ at a cost of*

$$O\left(\frac{\kappa\alpha}{\|A\|}\log\left(\frac{\kappa}{\delta\|A\|}\right)T_A\right). \tag{2.12}$$

We want to highlight also as the previous result can be used to solve the Ordinary Least Square problem. We report here a routine obtained by Chakraborty et al. [16] with the modification of avoiding the use of a regularization term.

**Lemma 2.15** (Quantum Ordinary Least Squares revised from Theorem 35 of [16]). *Let $A \in \mathbb{R}^{n\times m}$ be the data matrix with condition number $\kappa$. Let $U_A$ be a $(\alpha, a, \epsilon)$-block-encoding of $A$ implemented in time $T_A$ Furthermore, suppose $U_b$ be a unitary that prepares $|b\rangle$ in time $T_b$. Then for any $\delta \in (0,1)$ such that*

$$\epsilon \leq \frac{\delta\|A^T A\|}{32\alpha\,\kappa^3 log^3(2\kappa/\delta)} \tag{2.13}$$

*we can prepare a state $|\varphi\rangle$ such that*

$$\left\| |\varphi\rangle - \frac{A^+|b\rangle}{\|A^+|b\rangle\|} \right\| \leq \delta \tag{2.14}$$

*with probability $\Theta(1)$, at a cost*

$$O\left(\kappa\log\kappa\left(\frac{\alpha}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + T_b\right)\right) \tag{2.15}$$

*using only $O(\log\kappa)$ additional qubits.*

*If we want to estimate the norm of the result up to a relative error $\eta$ with probability $1-\delta'$ we have a complexity of $O\left(\frac{1}{\eta}\left(\kappa\log\kappa\left(\frac{\alpha}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + T_b\right)\right)\log(1/\delta')\right)$.*

### 2.2.5. Quantum Inner Product Estimation

One of the most significant routines used in this work is the estimation of the inner product of two vectors represented by quantum states.

**Lemma 2.16** (Inner product estimation [7, 43]). *Let there be quantum access to the matrices $V \in \mathbb{R}^{n\times m}$ and $\mathbb{C} \in \mathbb{R}^{k\times m}$, through the unitaries $U_v : |i\rangle|0\rangle \to |i\rangle|v_{i,\cdot}\rangle$ and $U_c : |j\rangle|0\rangle \to |j\rangle|c_{j,\cdot}\rangle$ that run in time $T$, and the norms of the vectors are known. Then, for any $\delta \in (0,1]$ and $\epsilon \in R^+$, there exists a quantum algorithm that computes $|i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|\overline{\langle v_{i,\cdot}, c_{j,\cdot}\rangle}\rangle$, such that $|\overline{\langle v_{i,\cdot}, c_{j,\cdot}\rangle} - \langle v_{i,\cdot}, c_{j,\cdot}\rangle| \leq \epsilon$, with probability greater than $1-2\delta$ in time $\widetilde{O}\left(T\frac{\|v_i\|\|c_j\|}{\epsilon}\log(1/\delta))\right)$.*

If we want to compute the inner product of a vector and all the column vectors of a certain matrix, it is possible to apply the following corollary.

**Corollary 2.16.1** (Matrix-vector inner product estimation [7]). *Let there be quantum access to a matrix $A \in \mathbb{R}^{n\times m}$ with unitary columns and to a vector $x \in \mathbb{R}^n$, through the unitaries $U_A : |0\rangle|0\rangle \to \frac{1}{\|A\|_F}\sum_i^n |i\rangle|a_{i,\cdot}\rangle$ and $U_x : |0\rangle \to \frac{1}{\|x\|_2}\sum_i^n x_i|i\rangle$, that run in time less than $T$, and the norm of the vector is known. Then, for any $\delta \in (0,1]$ and $\epsilon \in R^+$, there exists a quantum algorithm that computes $|0\rangle|0\rangle \to \frac{1}{\|A\|_F}\sum_i^n |i\rangle|\overline{\langle a_{i,\cdot}, x\rangle}\rangle$, such that $|\overline{\langle a_{i,\cdot}, x\rangle} - \langle a_{i,\cdot}, x\rangle| \leq \epsilon$ consistently across multiple runs, with probability greater than $1-2\delta$ in time $\widetilde{O}\left(\frac{\|x\|T}{\epsilon}\log(m/\delta)\right)$.*

From the procedure that generates the value of the inner product between two vectors, it is also possible to obtain a routine that estimates the Euclidean distance between two vectors.

**Lemma 2.17** (Distance estimation [43]). *Assume for a data matrix $V \in \mathbb{R}^{n\times m}$ and a matrix $C \in \mathbb{R}^{k\times d}$ that the following unitaries $U_v : |i\rangle|0\rangle \to |i\rangle|v_{i,\cdot}\rangle$ and $U_c : |i\rangle|0\rangle \to |i\rangle|c_{i,\cdot}\rangle$ can be performed in time $T$ and the norms of the vectors are known. For any $\delta \in (0,1]$ and $\epsilon \in R^+$, there exists a quantum algorithm that computes $|i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|\overline{d^2(v_i, c_j)}\rangle$, where $|\overline{d^2(v_i, c_j)} - d^2(v_i, c_j)| \leq \epsilon$ with probability at least $2\delta$, in time $\widetilde{O}\left(T\frac{\|v_i\|\|c_j\|\log(1/\delta)}{\epsilon}\right)$.*

It is possible to draw a connection between Lemma 2.16 and Lemma 2.17. To obtain the distance, the first step is to estimate the inner product between the two vectors. This value, together with the norms of the vectors that are already known, is used to construct the distance

$$d^2(v_j, c_i) = \|v_j\|^2 + \|c_i\|^2 - 2\langle v_j, c_i\rangle. \tag{2.16}$$

If we consider using normalized vectors, we can define the distance as

$$d^2(|v_j\rangle, |c_i\rangle) = 1 + 1 - 2\langle v_j|c_i\rangle. \tag{2.17}$$

With this result in mind, we define the following corollary.

**Corollary 2.17.1** (Matrix-vector distance estimation). *Let there be quantum access to a matrix $A \in \mathbb{R}^{n\times m}$ with unitary columns and to a unitary vector $x \in \mathbb{R}^n$, through the unitaries $U_A : |0\rangle|0\rangle \to \frac{1}{\|A\|_F}\sum_i^n |i\rangle|a_{i,\cdot}\rangle$ and $U_x : |0\rangle \to \frac{1}{\|x\|_2}\sum_i^n x_i|i\rangle$, that run in time less than $T$. Then, for any $\delta \in (0,1]$ and $\epsilon \in R^+$, there exists a quantum algorithm that computes $|0\rangle|0\rangle \to \frac{1}{\|A\|_F}\sum_i^n |i\rangle|\overline{d^2(a_{i,\cdot},x)}\rangle$, such that $|\overline{d^2(a_{i,\cdot},x)} - d^2(a_{i,\cdot},x)| \leq \epsilon$ consistently across multiple runs, with probability greater than $1-2\delta$ in time $\widetilde{O}\left(\frac{T}{\epsilon}\log(m/\delta)\right)$.*

### 2.2.6. Quantum Search

It is possible to use an algorithm based on the famous Grover search algorithm [35] to find the component with the minimum absolute value inside a vector.

**Lemma 2.18** (Finding the minimum [7, 27]). *Let there be quantum access to a vector $u \in [0,1]^N$ via the operation $|j\rangle|0\rangle \to |j\rangle|u_j\rangle$ in time $T$. Then, we can find the minimum $u_{min} = \min_{j\in[N]} u_j$ and its index $j_{min} = \arg\min_{j\in[N]} u_j$ with success probability $1-\delta$ in time $O\left(T\sqrt{N}\log(\frac{1}{\delta})\right)$.*

As shown by Bellante and Zanero [7], it is easy to create an algorithm that finds the maximum absolute value from this result.

**Corollary 2.18.1** (Finding the maximum absolute value [7]). *Let there be quantum access to a vector $u \in [0,1]^N$ via the operation $|j\rangle|0\rangle \to |j\rangle|u_j\rangle$ in time $T$. Then, we can find the maximum absolute value $u_{max} = \max_{j\in[N]} u_j$ and its index $j_{max} = \arg\max_{j\in[N]} u_j$ with success probability $1-\delta$ in time $O\left(T\sqrt{N}\log(\frac{1}{\delta})\right)$.*

### 2.2.7. Vector State Tomography

As already explained in Section 2.1.5, the tomography of a quantum state is not easy to perform. We report a lemma that states the complexity necessary to perform it.

**Lemma 2.19** (Vector State Tomography [43, 78]). *Given access to unitary $U$ such that $U|0\rangle = |x\rangle$ and its controlled version in time $T(U)$, there is a tomography algorithm with time complexity $O(T(U)\frac{d\log d}{\epsilon^2}\log\left(\frac{1}{\delta}\right))$ that produces unit vector $\widetilde{x} \in \mathbb{R}^d$ such that $\|\widetilde{x} - x\|_2 \leq \epsilon$ where $\epsilon \in \mathbb{R}^+$ with probability at least $\delta$.*

Recent studies have shown how, in some specific settings, it is possible to reduce this complexity up to a linear dependence on the error parameter [78].

# 3 | Intermediate Technical Results

In this chapter, we explain and prove some technical results used in the following chapters. These statements are not the main results of our work but are useful to define the different algorithms.

## 3.1. Application of block-encoding to a quantum state

The first result is a lemma that modifies the run-time of applying a block-encoding to a quantum state (Lemma 2.11), considering knowing beforehand the norm of the final state.

**Lemma 3.1** (Applying a pre-amplified Block-encoded Matrix on a Quantum State knowing the final norm)**.** *Let $A$ be an $s$-qubit operator such that its singular values lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Also let $\delta \in (0,1)$, and $U_A$ be an $(\alpha, a, \epsilon)$-block-encoding of $A$, such that*

$$\epsilon \leq \frac{\delta \|A\|}{4\kappa}, \tag{3.1}$$

*that can be implemented in time $T_A$. Furthermore, suppose $|b\rangle$ be an $s$-qubit quantum state that can be prepared in time $T_b$ and suppose that we already know the value of $\|A|b\rangle\|$. Then we can prepare a state $|\varphi\rangle$ such that $\left\||\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|}\right\| \leq \delta$ with success probability $\Omega(1)$ at a cost*

$$O\left(\frac{\|A\|}{\|A|b\rangle\|}\left(\frac{\alpha}{\|A\|}\log\left(\frac{\kappa}{\delta}\right)T_A + T_b\right)\right). \tag{3.2}$$

*Proof.* We base our proof on the proof of Lemma 24 in [15] and on the proof of Corollary 10 in [16]. The first step is the pre-amplification of the block-encoding $U_A$, following Lemma 2.10.1, at the cost of $O\left(\frac{\alpha}{\|A\|}T_A\log\left(\frac{\|A\|}{\gamma}\right)\right)$ and obtaining a $(\sqrt{2}\|A\|, a+1, \gamma)$-block-encoding of $A$, with $\gamma = 2\epsilon$. Given the pre-amplified version of $U_A$ and the quantum

state $|b\rangle$, we can say that

$$\|A - \sqrt{2}\|A\|(\langle 0|^{\otimes a} \otimes \mathbb{I})U(|0\rangle^{\otimes a} \otimes \mathbb{I})\| \leq \gamma, \tag{3.3}$$

$$\left\| \frac{1}{\sqrt{2}\|A\|} A\,|b\rangle - (\langle 0|^{\otimes a} \otimes \mathbb{I})U(|0\rangle^{\otimes a} \otimes \mathbb{I})\,|b\rangle \right\| \leq \gamma/(\sqrt{2}\|A\|). \tag{3.4}$$

We obtain a final state that corresponds to $|0\rangle^{\otimes a}\left(\frac{1}{\sqrt{2}\|A\|}A\,|b\rangle\right) + |0^{\perp}\rangle$. If we already know the norm $\|A\,|b\rangle\,\|$, for example, because it has been estimated using amplitude estimation, we can perform a number of amplitude amplification rounds, conditioned on having $|0\rangle$ in the first register, that is exactly $\frac{\sqrt{2}\|A\|}{\|A|b\rangle\|}$ to obtain a state $|\varphi\rangle$ such that $\left\||\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|}\right\| \leq \frac{\gamma}{\|A|b\rangle\|}$. If we select a value for $\delta$ such that $\delta \geq \frac{\gamma}{\|A|b\rangle\|}$, we obtain $\left\||\varphi\rangle - \frac{A|b\rangle}{\|A|b\rangle\|}\right\| \leq \delta$. Considering also that $\|A\,|b\rangle\,\| \geq \frac{\|A\|}{\kappa}$, we obtain that $\epsilon \leq \frac{\delta\|A\|}{4\kappa}$. □

If we use an estimation of the norm instead of the correct norm, we can consider applying the fixed-point amplitude amplification routine by Yoder et al. [90]. We can amplify the state with a value that corresponds to the estimation of the norm summed to the maximum error that can be in the estimation. In this way, we avoid the possibility of having states that are not amplified enough while keeping the runtime as low as possible.

## 3.2. Results on the error of vector approximations

In this section, we explain some results useful to bound the errors of different quantum routines.

We define an upper bound for the error on the value of an unnormalized vector approximation. Our inputs are a bound on the error of the normalized vector approximation and a bound on the error of its norm.

**Claim 3.2** (Error on Unnormalized Projected vector)**.** *Given access to a normalized vector $|\phi'\rangle$ such that $\|\,|\phi'\rangle - |\phi\rangle\,\| \leq \delta$ and given access to a value $\|\phi\|'$ such that $|\|\phi\|' - \|\phi\|| \leq \eta\|\phi\|$, we can say that the unnormalized vector $\|\phi\|'\,|\phi'\rangle$ has the following error guarantees*

$$\|\|\phi\|\,|\phi\rangle - \|\phi\|'\,|\phi'\rangle\,\| \leq \delta\|\phi\| + \eta\|\phi\|, \tag{3.5}$$

$$\|\|\phi\|\,|\phi\rangle - \|\phi\|'\,|\phi'\rangle\,\| \leq \delta\|\phi\|' + \eta\|\phi\|. \tag{3.6}$$

*Proof.* Our starting point is the following inequality

$$\| \, |\phi\rangle - |\phi'\rangle \, \| \leq \delta. \tag{3.7}$$

The first step is to add the norm of an arbitrary vector $\|a\|$ to both sides of the equation

$$\| \, |\phi\rangle - |\phi'\rangle \, \| + \|a\| \leq \delta + \|a\|. \tag{3.8}$$

After that, we use the Triangular inequality ($\|x\| + \|y\| \geq \|x + y\|$) to obtain

$$\| |\phi\rangle - |\phi'\rangle + a\| \leq \| \, |\phi\rangle - |\phi'\rangle \, \| + \|a\| \leq \delta + \|a\|. \tag{3.9}$$

We can multiply everything by the norm of $\|\phi\|$ obtaining

$$\| \|\phi\| \, |\phi\rangle - \|\phi\| \, (|\phi'\rangle - a)\| \leq \|\phi\|(\delta + \|a\|). \tag{3.10}$$

At this point, we can choose the value of the variable $a$ such that it solves the following equation

$$\|\phi\| \, (|\phi'\rangle - a) = \|\phi\|' \, |\phi'\rangle \, . \tag{3.11}$$

The value corresponds to

$$a = \frac{\|\phi\| - \|\phi\|'}{\|\phi\|} \, |\phi'\rangle \, . \tag{3.12}$$

We substitute the value of $a$ inside Inequality 3.10 and we obtain

$$\left\| \|\phi\| \, |\phi\rangle - \|\phi\| \left( |\phi'\rangle + \frac{\|\phi\|' - \|\phi\|}{\|\phi\|} \, |\phi'\rangle \right) \right\| \leq \|\phi\| \left( \delta + \left\| \frac{\|\phi\| - \|\phi\|'}{\|\phi\|} \, |\phi'\rangle \right\| \right), \tag{3.13}$$

$$\| \|\phi\| \, |\phi\rangle - \|\phi\|' \, |\phi'\rangle \| \leq \delta \|\phi\| + |\|\phi\| - \|\phi\|'| = \delta \|\phi\| + \eta \|\phi\|. \tag{3.14}$$

The procedure to obtain the other bound is similar. It is enough, starting from Inequality 3.9 multiply by $\|\phi\|'$ instead of $\|\phi\|$ and select $a = \frac{\|\phi\| - \|\phi\|'}{\|\phi\|'} \, |\phi\rangle$. □

Here we show a graphical intuition useful to understand better the proof.

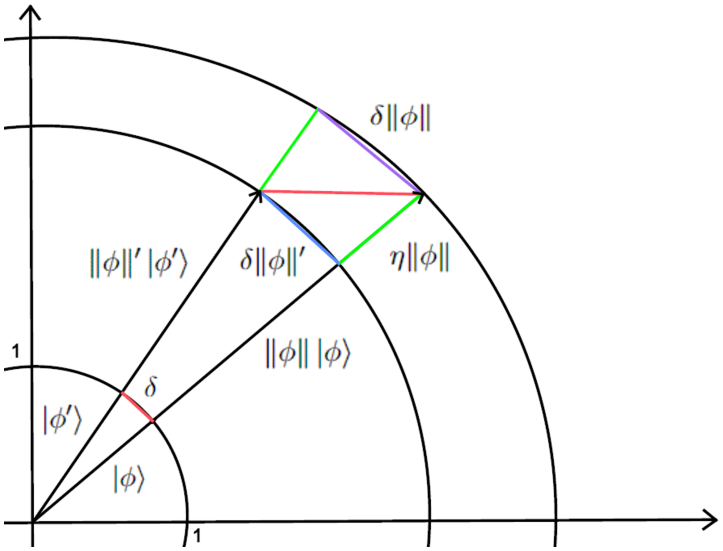



Figure 3.1: Graphical solution of unnormalized projection error

It is possible to observe in Figure 3.1 that the distance between the vectors' heads, highlighted in red, corresponds to the true reconstruction error. It is easy to see how applying the triangular inequality this value can be bounded.

The second result is used to find an upper bound on the error of an unnormalized inner product. We obtain this value knowing the error of an approximation of the normalized inner product between a unitary vector and the approximation of another vector.

**Claim 3.3.** *Let $\phi \in \mathbb{R}^n$ and $\epsilon_\phi, \eta, \epsilon, \epsilon_{in} \in \mathbb{R}^+$. Let $d \in \mathbb{R}^n$ such that $\|d\| = 1$. Given a normalized vector $|\phi'\rangle$ such that $\| \, |\phi'\rangle - |\phi\rangle \, \| \leq \epsilon_\phi$, $\|\phi\|'$ such that $|\|\phi\|' - \|\phi\|| \leq \eta\|\phi\|$, a value $\widetilde{z}$ such that $|\widetilde{z} - \langle d|\phi'\rangle| \leq \epsilon_{in}$, we can guarantee $|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle| \leq \epsilon$ if*

$$\|\phi\|'\epsilon_{in} + \|\phi\|\epsilon_\phi + \|\phi\|\eta \leq \epsilon. \tag{3.15}$$

*Proof.* We start from the known $|\widetilde{z} - \langle d|\phi'\rangle| \leq \epsilon_{in}$. We multiply both sides for the estimated norm $\|\phi\|'$ obtaining $|\|\phi\|'\widetilde{z} - \|\phi\|'\langle d|\phi'\rangle| \leq \|\phi\|'\epsilon_{in}$.

We can select a value $\vec{\epsilon_v}$ such that $\|\phi\|'\,|\phi'\rangle = \phi + \vec{\epsilon_v}$ and we obtain

$$|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle - \langle d, \vec{\epsilon_v}\rangle| \leq \|\phi\|'\epsilon_{in}. \tag{3.16}$$

We add to both sides of the inequality the term $|\langle d, \vec{\epsilon_v}\rangle|$ and we use the triangular in-

equality obtaining

$$|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle - \langle d, \vec{\epsilon_v}\rangle| + |\langle d, \vec{\epsilon_v}\rangle| \leq \|\phi\|'\epsilon_{in} + |\langle d, \vec{\epsilon_v}\rangle|, \tag{3.17}$$

$$|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle| \leq \|\phi\|'\epsilon_{in} + |\langle d, \vec{\epsilon_v}\rangle|. \tag{3.18}$$

Thanks to Cauchy-Schwartz, we can say that $|\langle d, \vec{\epsilon_v}\rangle| \leq \|d\|\|\vec{\epsilon_v}\| = \|\vec{\epsilon_v}\|$ because the vector $d$ is unitary. Then we substitute the upper bound in Inequality 3.18 to obtain

$$|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle| \leq \|\phi\|'\epsilon_{in} + \|\vec{\epsilon_v}\|. \tag{3.19}$$

Finally, we apply the bound $\|\vec{\epsilon_v}\| \leq \epsilon_\phi\|\phi\| + \eta\|\phi\|$, obtained using Claim 3.2, and we bound the final error with the value $\epsilon$

$$|\|\phi\|'\widetilde{z} - \langle d, \phi\rangle| \leq \|\phi\|'\epsilon_{in} + \|\phi\|\epsilon_\phi + \|\phi\|\eta \leq \epsilon. \tag{3.20}$$

□

We now show another claim used to bound the error related to the computation of the distance between a correct vector and an approximated one, knowing an approximation of the inner product between the two.

**Claim 3.4.** *Let $\phi \in \mathbb{R}^n$ and $\epsilon_\phi, \eta, \epsilon, \epsilon_{in} \in \mathbb{R}^+$. Let $s \in \mathbb{R}^n$ with norm $\|s\|$. Given a normalized vector $|\phi'\rangle$ such that $\| |\phi'\rangle - |\phi\rangle \| \leq \epsilon_\phi$, $\|\phi\|'$ such that $|\|\phi\|' - \|\phi\|| \leq \eta\|\phi\|$, a value $\widetilde{z}$ such that $|\widetilde{z} - \langle s|\phi'\rangle| \leq \epsilon_{in}$ where $|s\rangle$ is the normalized vector $s$, we can guarantee $\left|\widetilde{d^2} - d^2(s,\phi)\right| \leq \epsilon$, where $d^2(s,\phi)$ is the squared Euclidean distance and $\widetilde{d^2} = \|s\|^2 + \|\phi\|'^2 - 2\|s\|\|\phi\|'\widetilde{z}$, if*

$$3\eta\|\phi\|^2 + 2\|s\|\|\phi\|\eta + 2\|s\|\|\phi\|\epsilon_\phi + 2\|s\|\|\phi\|'\epsilon_{in} \leq \epsilon \tag{3.21}$$

*Proof.* We start with our final objective

$$\left|\widetilde{d^2} - d^2(s,\phi)\right| \leq \epsilon. \tag{3.22}$$

We substitute inside it the analytical description of the values $d^2(s,\phi)$ and $\widetilde{d^2}$ obtaining

$$\left|\|s\|^2 + \|\phi\|'^2 - 2\|s\|\|\phi\|'\widetilde{z} - (\|s\|^2 + \|\phi\|^2 - 2\|\phi\|\langle s|\phi\rangle)\right| \leq \epsilon, \tag{3.23}$$

$$\left|\|\phi\|'^2 - \|\phi\|^2 - 2\|s\|(\|\phi\|'\widetilde{z} - \|\phi\|\langle s|\phi\rangle)\right| \leq \epsilon. \tag{3.24}$$

We can substitute the approximated norm of $\phi$ with its value in the worst-case $\|\phi\|' \leq \|\phi\|(1+\eta)$. Substituting it inside the first occurrence in the previous equation, and considering that $\eta^2 \leq \eta$ because $\eta \in [0,1]$, we obtain

$$\left|(\|\phi\| + \eta\|\phi\|)^2 - \|\phi\|^2 - 2\|s\|(\|\phi\|'\widetilde{z} - \|\phi\|\langle s|\phi\rangle)\right| \leq \epsilon \tag{3.25}$$

$$\left|2\eta\|\phi\|^2 + \eta^2\|\phi\| - 2\|s\|(\|\phi\|'\widetilde{z} - \|\phi\|\langle s|\phi\rangle)\right| \leq \epsilon \tag{3.26}$$

$$\left|3\eta\|\phi\|^2 - 2\|s\|(\|\phi\|'\widetilde{z} - \|\phi\|\langle s|\phi\rangle)\right| \leq \epsilon \tag{3.27}$$

At this point, we want to understand how much error can be introduced by the use of the value $\widetilde{z}$. To achieve this objective we can apply Claim 3.3 considering $|s\rangle$ as the vector of unitary norm, the vector $|\phi'\rangle$ such that $\| \, |\phi'\rangle - |\phi\rangle \, \| \leq \epsilon_\phi$, the bound on the norm $|\|\phi\|' - \|\phi\|| \leq \eta\|\phi\|$ and the condition on the normalized inner product $|\widetilde{z} - \langle s|\phi'\rangle| \leq \epsilon_{in}$. We obtain

$$|\|\phi\|'\widetilde{z} - \langle |s\rangle, \phi\rangle| \leq \|\phi\|'\epsilon_{in} + \epsilon_\phi\|\phi\| + \eta\|\phi\|. \tag{3.28}$$

Substituting back this bound in Inequality 3.27 and considering the greatest possible final error, we obtain

$$3\eta\|\phi\|^2 + 2\|s\|\|\phi\|\eta + 2\|s\|\|\phi\|\epsilon_\phi + 2\|s\|\|\phi\|'\epsilon_{in} \leq \epsilon \tag{3.29}$$

□

Another trivial claim shows how the sum of two scalars with additive error provides a sum with the sum of the two errors as the final absolute error.

**Claim 3.5.** *Let $z_1, z_2 \in \mathbb{R}$ and $z = z_1 + z_2$. Let $\overline{z_1} \in \mathbb{R}$ be an approximation of $z_1$ such that $|\overline{z_1} - z_1| \leq \epsilon_1$ with $\epsilon_1 \in \mathbb{R}^+$. Let $\overline{z_2} \in \mathbb{R}$ be an approximation of $z_2$ such that $|\overline{z_2} - z_2| \leq \epsilon_2$ with $\epsilon_2 \in \mathbb{R}^+$. We define $\overline{z} = \overline{z_1} + \overline{z_2}$. We can say that $|\overline{z} - z| \leq \epsilon_1 + \epsilon_2$.*

*Proof.* $|\overline{z} - z| = |\overline{z_1} + \overline{z_2} - (z_1 + z_2)| \leq |\overline{z_1} - z_1| + |\overline{z_2} - z_2| \leq \epsilon_1 + \epsilon_2$. □

It is easy to prove that this is also valid for the subtraction of scalars.

## 3.3. Quantum Projection

We also need an operator that projects a quantum state on the subspace spanned by the columns of a matrix. For this reason, we define the following lemma.

**Lemma 3.6** (Quantum Projection). *Let $A \in \mathbb{R}^{n\times m}$ be the data matrix with condition number $\kappa$ and with singular values that lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Let $U_A$ be a $(\alpha, a, \epsilon)$-block-encoding of $A$ implemented in time $T_A$. Furthermore, suppose $U_s$ be a unitary that prepares $|s\rangle$ in time $T_s$. Suppose also that we know the norms $\|s\|$, $\|A^+s\|$ and $\|AA^+s\|$. Then for any $\delta \in (0,1)$ such that*

$$\epsilon \leq \frac{\delta\|A^TA\|}{64\alpha\,\kappa^4 log^3(2\kappa/\delta)} \tag{3.30}$$

*we can prepare a state $|\varphi\rangle$ such that*

$$\left\| |\varphi\rangle - \frac{AA^+|s\rangle}{\|AA^+|s\rangle\|} \right\| \leq \delta \tag{3.31}$$

*with probability $\Theta(1)$, at a cost*

$$\widetilde{O}\left(\frac{\kappa\alpha\|A^+s\|}{\|AA^+s\|}(T_A + T_s)\right). \tag{3.32}$$

*Proof.* First of all, we solve the OLS problem following Lemma 2.15 using $U_A$, the block-encoding of matrix $A$, and $U_s$ the unitary that prepares the state $|s\rangle$ obtaining a quantum state $|\widetilde{A^+s}\rangle$ such that $\left\||\widetilde{A^+s}\rangle - |A^+s\rangle\right\| \leq \delta'$. This step is possible only if the block-encoding of $A$ is good enough. For this reason, we impose

$$\epsilon \leq \frac{\delta'\|A^TA\|}{32\alpha\,\kappa^3 log^3(2\kappa/\delta')}, \tag{3.33}$$

and the run-time is $\widetilde{O}(T_{OLS}) = \widetilde{O}\left(\frac{\kappa\alpha}{\|A\|}(T_A + T_s)\right)$.

Now, to have a good-enough input for the next step of the procedure, we need to place a constraint on the value of the $\delta'$ parameter following Lemma 2.12. If we want to guarantee that the error of our approximation of the value $|AA^+s\rangle$ will not be more than $\delta$, we must bound the value of $\delta'$ in the following way

$$\left\||A^+s\rangle - |\widetilde{A^+s}\rangle\right\| \leq \delta' \leq \frac{\delta}{2\kappa}. \tag{3.34}$$

At this point, we apply to $|\widetilde{A^+s}\rangle$ the pre-amplified block-encoding of the matrix $A$ following Lemma 3.1 and obtain a state $|\varphi\rangle$ such that $\left\||\varphi\rangle - \frac{AA^+|s\rangle}{\|AA^+|s\rangle\|}\right\| \leq \delta$. To obtain this result, we must also guarantee that

$$\epsilon \leq \frac{\delta\|A\|}{4\kappa}, \tag{3.35}$$

and the run-time for this step is

$$\widetilde{O}\left(\frac{\|A^+s\|\|A\|}{\|AA^+s\|}\left(\frac{\alpha}{\|A\|}T_A + T_{OLS}\right)\right). \tag{3.36}$$

Putting together the run-time of the two routines and the constraints on $\epsilon$, we obtain

$$\widetilde{O}\left(\frac{\|A^+s\|\|A\|}{\|AA^+s\|}\left(\frac{\alpha}{\|A\|}T_A + \frac{\kappa\alpha}{\|A\|}(T_A+T_s)\right)\right) \sim \widetilde{O}\left(\frac{\kappa\alpha\|A^+s\|}{\|AA^+s\|}(T_A+T_s)\right), \tag{3.37}$$

with a value of $\epsilon$ such that

$$\epsilon \leq \frac{\delta\|A^TA\|}{64\alpha\,\kappa^4 log^3(4\kappa^2/\delta)}. \tag{3.38}$$

□

We can also prove the complexity when the norms of the intermediate results are not known and the complexity necessary to estimate them.

**Lemma 3.7** (Quantum Projection with Norm Estimation). *Let $A \in \mathbb{R}^{n\times m}$ be the data matrix with condition number $\kappa$ and with singular values that lie in $\left[\frac{\|A\|}{\kappa}, \|A\|\right]$. Let $U_A$ be a $(\alpha, a, \epsilon)$-block-encoding of $A$ implemented in time $T_A$. Furthermore, suppose $U_s$ be a unitary that prepares $|s\rangle$ in time $T_s$ and we know the norm $\|s\|$. Then for any $\delta \in (0,1)$ and $\eta \in (0,1)$ such that*

$$\epsilon \leq \frac{\delta\|A^TA\|}{64\alpha\,\kappa^4 log^3(2\kappa/\delta)} \tag{3.39}$$

*we can prepare a state $|\varphi\rangle$ such that*

$$\left\||\varphi\rangle - \frac{AA^+|s\rangle}{\|AA^+|s\rangle\|}\right\| \leq \delta \tag{3.40}$$

*with probability $\Theta(1)$, at a cost*

$$\widetilde{O}\left(\frac{\kappa^2\alpha}{\|A\|}(T_A+T_s)\right). \tag{3.41}$$

*If we want to estimate the norms $\|A^+s\|$ and $\|AA^+s\|$ up to a relative error $\eta$ with probability $1-\delta'$ we have a complexity of $O\left(\frac{1}{\eta}\frac{\kappa^2\alpha}{\|A\|}(T_A+T_s)\log(1/\delta')\right)$.*

*Proof.* The proof follows similarly from the one of the previous lemma. We solve the OLS problem using $U_A$, the block-encoding of $A$, and $U_s$ the unitary that prepares the state $|s\rangle$ exploiting Lemma 2.15 and obtaining the state $|\widetilde{A^+|s\rangle}\rangle$ in $\widetilde{O}\left(\frac{\kappa\alpha}{\|A\|}(T_A+T_s)\right)$. After that, it is necessary to estimate the norm of the solution. Applying the second result of the lemma, we obtain the value $\|\widetilde{A^+|s\rangle}\|$ such that it is an estimation of the correct norm with $\eta$-multiplicative precision in time $\widetilde{O}\left(\frac{\kappa\alpha}{\eta\|A\|}(T_A+T_s)\right)$.

At this point following Lemma 2.11.1 we apply the block-encoding $U_A$ to $|\widetilde{A^+|s\rangle}\rangle$ and we estimate the norm $\|A|\widetilde{A^+|s\rangle}\|$ of the result in time $\widetilde{O}\left(\frac{\kappa^2\alpha}{\eta\|A\|}(T_A+T_s)\right)$ because the time necessary to obtain the state $|\widetilde{A^+|s\rangle}\rangle$ is $\widetilde{O}\left(\frac{\kappa\alpha}{\|A\|}(T_A+T_s)\right)$.

To find an estimation of the norm $\|AA^+s\|$, we can multiply the two norms that we have estimated and the one already known of the vector $s$ in the following way

$$\|\widetilde{AA^+s}\| = \|\widetilde{A|\widetilde{A^+|s\rangle}}\|\,\|\widetilde{A^+|s\rangle}\|\,\|s\|. \tag{3.42}$$

We can bound the error of this final value to be an $\eta$-multiplicative error, we here skip all the calculus for brevity, but it is possible to check them in Section A.1 of Appendix A.

The final running time corresponds to

$$\widetilde{O}\left(\left(\frac{\kappa^2\alpha}{\eta\|A\|}+\frac{\kappa\alpha}{\eta\|A\|}\right)(T_A+T_s)\right)\sim\widetilde{O}\left(\frac{1}{\eta}\left(\frac{\kappa^2\alpha}{\|A\|}\right)(T_A+T_s)\right) \tag{3.43}$$

□

It is important to highlight how, if for an algorithm it is necessary to create a quantum state $|AA^+s\rangle$ multiple times, it is possible to apply Lemma 3.7 one time and Lemma 3.6 all the other times exploiting the norms already estimated to speed-up the computation. As previously stated, if we use an estimation of the norm rather than the true norm, we might consider using the fixed-point amplitude amplification procedure provided by Yoder et al. [90]. We recall that we can amplify the state with a value equal to the norm estimation added to the maximum error that can appear in the estimation. In this way, we avoid the risk of having states that are not amplified sufficiently while keeping the runtime as low as possible.

# 4 | New Quantum Algorithms

## 4.1. Q-MP

In this section, we present the quantum version of the algorithm known as Matching Pursuit. The algorithm tries to build an every-time-better approximation of a signal $s \in \mathbb{R}^n$ describing it as a combination of a group of atoms of a dictionary $D \in \mathbb{R}^{n \times m}$ as its classical counterpart.

The main difference between our work and the one of Bellante and Zanero [7] is that our algorithm can compute the results even for signals that are not stored in a QRAM structure, but are the result of a unitary quantum process. In addition, we provide this version of the algorithm in the block-encoding framework. This framework permits the use of different routines, such as the product between a matrix and a vector, not used in the previous work. We avoid the classical re-computation of the residual vector at each step to speed up the computation with relation to the dependency on the size of the signal in input. Finally, we exploit a different exit condition based on the energy described by the approximation of the signal as explained in Section 1.2.2.

In the following sections, we explain the algorithm, prove a bound on the running time, and show the propagation of the error.

### 4.1.1. Algorithm

Algorithm 4.1 summarizes the Quantum Matching Pursuit procedure.

**Algorithm 4.1** Quantum Matching Pursuit

**Input** Unitary $U_s$ that creates $|s\rangle$ where $s \in \mathbb{R}^n$ is the signal to analyse, $\|s\|$ stored classically. Unitary $U_D$ that is a block-encoding of the dictionary $D \in \mathbb{R}^{n\times m}$. Unitary $U_d$ that creates the superposition of the columns of $D$.
$L \in \mathbb{N}$ sparsity required, $\epsilon_z \in \mathbb{R}^+$ precision parameter on the inner products, $\epsilon \in \mathbb{R}^+$ error reconstruction tolerance.

**Output** $x \in \mathbb{R}^m$, an approximated solution of $\mathcal{P}_0^\epsilon$ (Def. 1.1).

Initialize in QRAM: $x_0 = 0^{\otimes m}$.
Initialize: $t = 0$, $\xi = 0$
**while** not ($\|x_t\|_0 > L$ or $\|s\|^2 - \xi \leq \epsilon^2$) **do**
  Use inner product estimation on $U_d$, $U_s$ to create:
    $|\varphi_1\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle |\overline{z_1}_j\rangle$ where $z_{1j} = \langle d_j, s\rangle$ and $|\overline{z_1}_j - z_{1j}| \leq \frac{1}{2}\epsilon_z$
  **if** t=0 **then**
    $|\varphi\rangle = |\varphi_1\rangle$
  **else**
    Use inner product estimation on $U_d$, $U_{\phi_t}$ to create:
    $|\varphi_2\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle |\overline{z_2}_j\rangle$ where $z_{2j} = \langle d_j, \phi_t\rangle$ and $|\overline{z_2}_j - z_{2j}| \leq \frac{1}{2}\epsilon_z$
    Subtract the states $|\varphi_1\rangle$ and $|\varphi_2\rangle$, conditioned on the first register to be equal:
    $|\varphi\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle |\overline{z}_j\rangle$ where $z_j = \langle d_j, r\rangle$ and $|\overline{z}_j - z_j| \leq \epsilon_z$
  **end if**
  Apply finding the maximum absolute value on the unitary that implements $|\varphi\rangle$, to extract $j^*$ and $z_{j^*}$
  Update $t = t + 1$
  Update the component of index $j^*$ of $x_t$ in QRAM as: $x_{t,j^*} = x_{t-1,j^*} + z_{j^*}$
  Update the stored norm of the solution as: $\|x_t\|^2 = \|x_{t-1}\|^2 - |x_{t-1,j^*}|^2 + |x_{t,j^*}|^2$
  Use quantum application of block-encoding to obtain a unitary $U_{\phi_t}$ that creates:
    $|\widetilde{\phi_t}\rangle$ where $|\phi_t\rangle = |Dx_t\rangle$ and $\left\||\widetilde{\phi_t}\rangle - |\phi_t\rangle\right\| \leq \frac{1}{8\|s\|}\epsilon_z$,
    $\widetilde{\|\phi_t\|}$ where $\|\phi_t\| = \|Dx_t\|$ and $\left|\widetilde{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8}\epsilon_z$
  Update $\xi = \xi + z_{j^*}^2$
**end while**
Output $x$

We first focus on the Q-MP procedure and defer the error analysis to Section 4.1.2. For the moment, we consider that all the errors depend on the parameter $\epsilon_z \in \mathbb{R}^+$.

We start by initializing the empty solution vector $x = 0^{\otimes m}$ in QRAM. We also initialize the value $\xi = 0$ and the counter $t = 0$.

For each iteration of the algorithm, we proceed in the following way.

First of all, we use the matrix-vector product routine of Corollary 2.16.1 with as input the unitary $U_d$ that generates the superposition of the atoms of $D$, and the unitary $U_s$ that generates $|s\rangle$, together with the norm $\|s\|$, to produce the state

$$|\varphi_1\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle \, |\overline{z_1}_j\rangle \, . \tag{4.1}$$

We recall that $z_{1j} = \langle d_j, s\rangle$ and we select an error for the routine such that $|\overline{z_1}_j - z_{1j}| \leq \frac{1}{2}\epsilon_z$.

During the first iteration, we skip the computation of the second inner product and assign $|\varphi\rangle = |\varphi_1\rangle$.

Except for the first iteration, we use again the matrix-vector product routine of Corollary 2.16.1 with as inputs the unitary $U_d$ and the unitary $U_{\phi_t}$ that creates the state $|\widetilde{\phi_t}\rangle$ that corresponds to the approximated signal, together with the approximated norm $\widetilde{\|\phi_t\|}$, to obtain the state

$$|\varphi_2\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle \, |\overline{z_2}_j\rangle \, . \tag{4.2}$$

We recall that $z_{2j} = \langle d_j, \phi_t\rangle$. The procedure computes $\overline{z_2}_j$ such that $|\overline{z_2}_j - z_{2j}| \leq \frac{1}{2}\epsilon_z$.

Later in this section, we will explain how to derive the unitary $U_{\phi_t}$, which produces the approximation of the signal, and the norm $\widetilde{\|\phi_t\|}$, which are needed to obtain $|\varphi_2\rangle$ for all the iterations except the first one.

Once we have the unitaries that create the states $|\varphi_1\rangle$ and $|\varphi_2\rangle$, we perform a basis-encoded state subtraction between the second register of them conditioned on the first register having the same index. In this way, we obtain the following quantum state

$$|\varphi\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle \, |\overline{z}_j\rangle = \frac{1}{\sqrt{m}} \sum_j^m |j\rangle \, |\overline{z_1}_j - \overline{z_2}_j\rangle \, . \tag{4.3}$$

We recall that $z_j = \langle d_j, s\rangle - \langle d_j, \phi_t\rangle = \langle d_j, s - \phi_t\rangle = \langle d_j, s - Dx_t\rangle = \langle d_j, r_t\rangle$ and the procedure computes $\overline{z}_j$ such that $|\overline{z}_j - z_j| \leq \epsilon_z$.

Once we have defined the unitary that creates the quantum register $|\varphi\rangle$, which contains a superposition of all the inner products, we can perform the finding the maximum absolute value routine from Corollary 2.18.1. In this way, we find $j^*$, the index of the atom that has the highest absolute inner product, and the value $z_{j^*}$.

With these two values, we can proceed in updating the solution $x_t \in \mathbb{R}^m$ and the squared norm of the solution $\|x_t\|^2$ in the following way

$$x_{t,j^*} = x_{t-1,j^*} + z_{j^*}, \tag{4.4}$$

$$\|x_t\|^2 = \|x_{t-1}\|^2 - |x_{t-1,\,j^*}|^2 + |x_{t,\,j^*}|^2. \tag{4.5}$$

We still have to define the unitary $U_{\phi_t}$ that computes an approximation of the value $\phi_t = Dx_t$ for the next iteration of the algorithm. We can define the procedure as follows. Firstly, we highlight that we have access to the solution $x_t$, stored in QRAM, and we can create the quantum state

$$|x_t\rangle = \frac{1}{\sqrt{m}} \sum_{j}^{m} x_{t,j} \,|j\rangle\,. \tag{4.6}$$

The state that approximate $|\phi_t\rangle$ is required multiple times and, to be more efficient, we first estimate its norm to have the possibility of applying the routines we have defined in Chapter 3 that also need the norm of the vector. We use Corollary 2.11.1 to obtain an estimation of the norm $\|\phi_t\|$, with as input the unitary $U_D$, that is the block-encoding of matrix $D$, and the access to state $|x_t\rangle$. In this way, we obtain a value $\widetilde{\|\phi_t\|}$ such that $\left|\widetilde{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8}\epsilon_z$.

Then, for each request of the state, we define the unitary $U_{\phi_t}$ as the application of $U_D$ to $|x_t\rangle$ following Lemma 3.1, that requires as input also to the already estimated norm $\widetilde{\|\phi_t\|}$. The routine generates the quantum state $|\widetilde{\phi_t}\rangle$ such that $\left\||\widetilde{\phi_t}\rangle - |Dx_t\rangle\right\| \leq \frac{1}{8\|s\|}\epsilon_z$.

To choose if the algorithm should continue or not with other iterations, we check how much energy of the signal it remains to describe. We provide a complete description of this property in Section 1.2.2. At each step, we sum the square of the estimated inner product that we have found to the previously stored value of the energy $\xi$

$$\xi = \xi + z_{j^*}^2. \tag{4.7}$$

Thanks to this value, we can consider the exit condition as

$$\|r\|^2 = \|s\|^2 - \xi \leq \epsilon^2 \tag{4.8}$$

where $\epsilon \in \mathbb{R}^+$ is the error reconstruction tolerance.

We repeat the procedure until either the sparsity of the solution is such that $\|x_t\|_0 > L$ for a threshold $L \in \mathbb{N}$, or until the squared norm of the residual is smaller than $\epsilon^2$. We can check the sparsity of the solution just by looking at how many components of $x_t$ had been updated. These conditions are checked classically.

It is crucial to note that the error in the inner products might cause convergence problems, as already mentioned by Bellante and Zanero [7]. As the authors did, we also analyse two alternative versions of the quantum matching pursuit algorithm:

1. **Single Error:** we chose to recompute classically the inner product $z_{j^*}$. In this way, the error can only guide us to the selection of the wrong atom $j^*$.
2. **Double Error:** we chose to not recompute classically the inner product $z_{j^*}$ and this value is used to update the solution. In this case, the error affects both the selection of the best atom and the value of $z_{j^*}$.

### 4.1.2. Error Propagation

As already anticipated, if we focus on the error propagation, we can define two different algorithms, the **Single Error** version and the **Double Error** version. The difference between the two consists in a different error present in the final result. We first analyze the Single Error case.

#### Single Error

We can analyze the error, in this case, considering that after we have found the maximum inner product, we recompute the inner product classically. This additional step has complexity $\widetilde{O}(kn)$, as explained in Section A.2, and guarantees, if the procedure that selects the atoms is good enough, to have a solution that is the same as the one we can find classically.

We define the error on the final value of the inner product with $\epsilon_z \in \mathbb{R}^+$. This error can guide us to the selection of the wrong atom compared to the one selected from the classical procedure. If we bound this value with the minimum value necessary to not misselect an atom, we can guarantee a result equivalent to the classical one. Unfortunately, this value

depends on the dictionary and on the signal analyzed, and it is not easy to provide a bound for it. In the Experiments Section 5.2, we show some results with different choices for the parameter $\epsilon_z$.

We now proceed with a complete analysis of all the errors involved. We consider the errors of the generic $t^{th}$ iteration, and we follow the steps of Algorithm 4.1. The first three steps do not add any errors.

Considering step number 4, we want to fix a bound on the error of the approximated inner products $\overline{z_{1j}}$, for $j \in [m]$, that, for the moment, we define as a variable $\epsilon_1 \in \mathbb{R}^+$. For this reason, we must describe a bound on the error admitted in the result of the normalized inner product procedure of Corollary 2.16.1, that we define as $\epsilon_{in_1} \in \mathbb{R}^+$, because it adds error in the value $\overline{z_{1j}}$. To obtain this result, we can use Claim 3.3 with as input the dictionary atoms $d_j$, that are unitary, the signal state $|s\rangle$ and its norm $\|s\|$, both with no approximation error. In this case, if we consider the variable $\epsilon_1$ as bound for the error of the final unnormalized inner product, starting from the error $\epsilon_{in_1}$ of its normalized version, we can say that $\|s\|\epsilon_{in_1} \leq \epsilon_1$.

For step number 8, we want to fix a bound on the error of the approximated inner products $\overline{z_{2j}}$, for $j \in [m]$, with a variable $\epsilon_2 \in \mathbb{R}^+$. For this reason, we must describe a bound on the error that can be present due to the use of the normalized inner product estimation procedure of Corollary 2.16.1 that, this time, we define as $\epsilon_{in_2} \in \mathbb{R}^+$. We must also bound the error on the value of $|\widetilde{\phi_t}\rangle$, that is the approximation of $|\phi_t\rangle$, called $\epsilon_\phi \in \mathbb{R}^+$ and required by Lemma 3.1. In addition, we also need a bound on the error of the value $\widetilde{\|\phi_t\|}$, that is an estimation of $\|\phi_t\|$, called $\eta \in \mathbb{R}^+$ and required by Corollary 2.11.1. We use these values for the estimation of the second inner product. We can use Claim 3.3 with as input the dictionary atoms $d_j$, that are unitary, and the approximated values $|\widetilde{\phi_t}\rangle$ and $\widetilde{\|\phi_t\|}$ to obtain the bounds. In this case, if we consider a variable $\epsilon_2 \in \mathbb{R}^+$ as bound for the error of the final unnormalized inner product, we can say that $\widetilde{\|\phi_t\|}\epsilon_{in_2} + \|\phi_t\|\epsilon_\phi + \|\phi_t\|\eta \leq \epsilon_2$.

During step number 9, we obtain an error on the subtraction of the inner products $\overline{z_j} = \overline{z_{1j}} - \overline{z_{2j}}$. This error coincides with the sum of the errors of the two inner products estimated $\epsilon_1 + \epsilon_2$, as shown in Claim 3.5.

At this point, it is necessary to bound the above error with $\epsilon_z$, the value that we have described at the beginning of the section, as $\epsilon_1 + \epsilon_2 \leq \epsilon_z$. We can choose to select the two errors as

$$\epsilon_1 \leq \frac{1}{2}\epsilon_z,\ \epsilon_2 \leq \frac{1}{2}\epsilon_z. \tag{4.9}$$

If we consider $\epsilon_2 \leq \frac{1}{2}\epsilon_z$, as explained above, we can choose $\epsilon_{in_2} \leq \frac{1}{4\widetilde{\|\phi_t\|}}\epsilon_z$, $\epsilon_\phi \leq \frac{1}{8\|\phi_t\|}\epsilon_z$ and

$\eta \leq \frac{1}{8\|\phi_t\|}\epsilon_z$. The same can be done for $\epsilon_1 \leq \frac{1}{2}\epsilon_z$ and we obtain $\epsilon_{in_1} \leq \frac{1}{2\|s\|}\epsilon_z$.

Steps from number 10 to number 14 do not add any errors.

Considering step number 15 instead, to guarantee that the approximation of the $|\phi_t\rangle$ vector has at maximum an error of $\epsilon_\phi$ we must have an error in the block-encoding of the matrix $D$ that satisfies the conditions of Lemma 3.1. We define this error as $\zeta$ such that

$$\zeta \leq \frac{\epsilon_\phi \|D\|}{4\kappa(D)} \leq \frac{\epsilon_z \|D\|}{32\kappa(D)\|\phi_t\|}. \tag{4.10}$$

All the other steps do not add any relevant error. Here we show a summary of all the different errors considered.

**Summary of the errors**

The only error that we must provide in advance is $\epsilon_z$. All the other errors that we have analyzed can be bounded using it. The running time of the algorithm can be estimated using these values.

- $\epsilon_{in_1} \leq \epsilon_z/(2\|s\|)$
- $\epsilon_{in_2} \leq \epsilon_z/(4\widetilde{\|\phi_t\|})$
- $\epsilon_\phi \leq \epsilon_z/8\|s\| \leq \epsilon_z/(8\|\phi_t\|)$
- $\eta \leq \epsilon_z/8\|s\| \leq \epsilon_z/(8\|\phi_t\|)$
- $\zeta \leq \epsilon_z\|D\|/(32\kappa(D)\|s\|) \leq \epsilon_z\|D\|/(32\kappa(D)\|\phi_t\|)$

It is possible to see how the errors that are upper-bounded by values in which is present $\|\phi_t\|$, that is not known during the computation, can be easily bounded in a more strict way substituting it with the norm $\|s\|$ because $\phi$ is just an approximated version of $s$. We estimate the value $\widetilde{\|\phi_t\|}$ during the computation, and the error $\epsilon_{in_2}$ can vary following it or be bounded as the other errors using $\|s\|$.

## Double Error

In this version of the algorithm, we use the same procedures as before, and, for this reason, we suppose the same errors, but we consider to **not** recompute the inner product classically. We show how this can cause a big increase in the error of the inner product. During some steps of each iteration cycle, we add a certain amount of error since we use quantum routines. This error will add at each iteration. We now analyze in detail what happens.

We have already seen how the first inner product estimation procedure (step 4) introduces an error called $\epsilon_1$. This value remains the same also in this version.

Considering the second inner product estimation procedure (step 8), we have the same error terms we have studied before with the addition of the error propagated through the iterations. We can still consider as bound for the error of this procedure the value $\epsilon_2$, and we can still identify the error in the final inner product as $\epsilon_1 + \epsilon_2$.

We have defined the error $\epsilon_2$ in the single error version as a value such that

$$\|\phi_t\|\epsilon_\phi + \|\phi_t\|\eta + \widetilde{\|\phi_t\|}\epsilon_{in_2} \leq \epsilon_2. \tag{4.11}$$

The difference in this version is that the error $\epsilon_2$ increases over time. This happens because of how the solution $x$ is updated, with components containing some error. At each iteration, the algorithm adds an extra error to the previous one.

As before, we want to find a bound for the errors $\epsilon_{in_1}, \epsilon_{in_2}, \epsilon_\phi$ and $\eta$ based on the value $\epsilon_z$ used to not miss-select the inner products. After some steps, we obtain as a final worst-case bound for the errors the value

$$2\epsilon_{in_1} + \epsilon_{in_2} + \epsilon_\phi + \eta \leq \frac{\epsilon_z}{2^{(L-1)}\|s\|}. \tag{4.12}$$

We provide the complete proof in Section A.3 of the Appendix. This value depends on the error and the number of iterations of the algorithm.

To conclude this explanation, we will show in the Experiment Section 5.2 how the final error of this version continuously grows considering more iterations but how, if we use small enough errors, the solution is not too distant from the **Single Error** one. In addition to that, we will also show how this error affects the exit condition of the algorithm.

For a general solution that is needed to be robust for big data or the introduction of high amounts of error, we recommend the use of the **Single Error** version. We continue the analysis of the running time of this algorithm considering this version.

### 4.1.3. Running Time

**Theorem 4.1** (Quantum Matching Pursuit). *Let $U_D$ be a $(\alpha, a, \zeta)$-block encoding of dictionary $D \in \mathbb{R}^{n\times m}$ such that*

$$\zeta \leq \frac{\epsilon_z \|D\|}{32\kappa(D)\|s\|} \tag{4.13}$$

*that can be generated in time $O(T_D)$ and with $\epsilon_z \in \mathbb{R}^+$ that is a precision parameter on the estimation of the inner products. Let $U_d$ be a unitary that generates the superposition of the columns of $D$ in time $O(T_d)$ and let $U_s$ a unitary that generates the state $|s\rangle$ where $s \in \mathbb{R}^n$ is the signal to analyze in time $O(T_s)$. Let $\|s\|$ be the known norm of the signal stored classically and let $L \in \mathbb{N}$ be a sparsity threshold such that $L \leq n$. Let $\epsilon \in \mathbb{R}^+$ be the error reconstruction tolerance. Then, there exists a procedure that simulates the matching pursuit algorithm in running time*

$$\widetilde{O}\left(k\left(\frac{\|s\|}{\epsilon_z}\left(\sqrt{m}\,(T_s + T_d + \alpha T_D) + \frac{\alpha\,\kappa(D)}{\|D\|}T_D\right) + n\right)\right) \tag{4.14}$$

*where $k \in \mathbb{N}$ is the total number of iterations.*

If we also consider that $\sqrt{m} \geq \frac{\kappa(D)}{\|D\|}$, condition that can be satisfied by well-formed dictionaries with not too low minimum singular value, the running time corresponds to

$$\widetilde{O}\left(k\left(\|s\|(T_s + T_d + \alpha\, T_D)\,\frac{\sqrt{m}}{\epsilon_z} + n\right)\right). \tag{4.15}$$

*Proof.* The proof consists in proving the running time of the algorithm.

From Corollary 2.16.1, we know that the cost of step 4 is $\widetilde{O}\left(\frac{T_s+T_d}{\epsilon_{in_1}}\right)$ given $O(T_s)$ the time necessary to prepare $|s\rangle$ and $O(T_d)$ the time necessary to create a superposition of the columns of the matrix. Therefore, we can prepare the state $|\varphi_1\rangle$ in time $\widetilde{O}\left(\frac{T_s+T_d}{\epsilon_{in_1}}\right) = \widetilde{O}\left(2\frac{\|s\|(T_s+T_d)}{\epsilon_z}\right) \sim \widetilde{O}\left(\frac{\|s\|(T_s+T_d)}{\epsilon_z}\right)$.

The following step needs the dictionary and the state $|\widetilde{\phi_t}\rangle$. For the moment, we define the overall required time to obtain $|\widetilde{\phi_t}\rangle$ as $\widetilde{O}(T)$, and we will explain its value later in this section.

Also the norm $\widetilde{\|\phi_t\|}$ is needed, but it is enough to estimate it just once for each cycle of the algorithm. We can obtain this result in time defined as $O(T_{norm})$ considering that we already know the norm $\|x_t\|$, which is stored classically. We will explain the value of $O(T_{norm})$ later in this section.

Given that the cost of preparing the necessary inputs of the procedure can be bounded by

$\widetilde{O}(T+T_d)$, we know that the cost of step 5, from Corollary 2.16.1 is $\widetilde{O}\left(\frac{T+T_d}{\epsilon_{in_2}}\right)$. Therefore, we can prepare the state $|\varphi_2\rangle$ in time $\widetilde{O}\left(\frac{T+T_d}{\epsilon_{in_2}}\right) \sim \widetilde{O}\left(\frac{\widetilde{\|\phi_t\|}(T+T_d)}{\epsilon_z}\right)$.

We can then subtract the two quantum states obtaining the state $|\varphi\rangle$ in time $\widetilde{O}(1)$.

We consider the total complexity of these first three steps as $\widetilde{O}\left(\frac{\widetilde{\|\phi_t\|}(T+T_d)}{\epsilon_z} + \frac{\|s\|(T_s+T_d)}{\epsilon_z}\right) \sim \widetilde{O}\left(\frac{\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}(T+T_d)}{\epsilon_z}\right) \sim \widetilde{O}\left(\frac{\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}T}{\epsilon_z}\right)$ considering that $\|s\| \geq \widetilde{\|\phi_t\|}$.

The state $|\varphi\rangle$ is the superposition of the $m$ differences between inner products, and we must determine the one with the greatest value. Using Corollary 2.18.1, we can find the value $z_{j^*}$ and its index $j^*$ in time $O\left(\sqrt{m}\frac{\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}T}{\epsilon_z}\right)$.

With the best $j^*$ and $z_{j^*}$, we can proceed to update the solution $x \in \mathbb{R}^m$ and the norm of the solution $\|x_t\|$ in time $\widetilde{O}(1)$. The last step consists of checking if it is necessary to continue the iterations or not. This is done classically in $\widetilde{O}(1)$.

The cost of the $t^{th}$ iteration of the loop, results in

$$\widetilde{O}\left(\left(\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}T\right)\frac{\sqrt{m}}{\epsilon_z}+T_{norm}\right). \tag{4.16}$$

If we assume that the algorithm performs $k$ iterations of the loop, the total complexity is

$$\widetilde{O}\left(k\left(\left(\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}T\right)\frac{\sqrt{m}}{\epsilon_z}+T_{norm}\right)\right). \tag{4.17}$$

We can consider the complexity of finding the norm, $O(T_{norm})$, using Lemma 2.11 as $\widetilde{O}(T_{norm}) = \widetilde{O}\left(\frac{\|s\|}{\epsilon_z}\frac{\alpha\,\kappa(D)}{\|D\|}T_D\right)$.

The last term we need is the complexity $O(T)$. From Lemma 3.1 considering that we obtain the unitary $U_D$ in time $O(T_D)$ and $x_t$ is accessible using the Q-RAM we can define $\widetilde{O}(T) = \widetilde{O}\left(\frac{\alpha\|x_t\|}{\widetilde{\|\phi_t\|}}T_D\right)$.

Substituting in Equation 4.17 the two terms we obtain

$$\widetilde{O}\left(k\left(\left(\|s\|(T_s+T_d)+\widetilde{\|\phi_t\|}\frac{\alpha\|x_t\|}{\widetilde{\|\phi_t\|}}T_D\right)\frac{\sqrt{m}}{\epsilon_z}+\frac{\|s\|}{\epsilon_z}\frac{\alpha\,\kappa(D)}{\|D\|}T_D\right)\right). \tag{4.18}$$

From what is explained in Section 1.2.2, we can consider that the norm of the solution $x_t$ is bounded by the norm of the signal $s$ and, for this reason, the complexity is

$$\widetilde{O}\left(k\left(\left(\|s\|(T_s+T_d)+\alpha\|s\|T_D\right)\frac{\sqrt{m}}{\epsilon_z}+\frac{\|s\|}{\epsilon_z}\frac{\alpha\,\kappa(D)}{\|D\|}T_D\right)\right). \tag{4.19}$$

If we consider the **Single Error** version, the final complexity will be

$$\widetilde{O}\left(k\left(\frac{\|s\|}{\epsilon_z}\left(\sqrt{m}\,(T_s+T_d+\alpha T_D)+\frac{\alpha\,\kappa(D)}{\|D\|}T_D\right)+n\right)\right) \tag{4.20}$$

Considering that $\sqrt{m} \geq \frac{\kappa(D)}{\|D\|}$, we can ignore the right side of the equation, and the complexity of the algorithm is

$$\widetilde{O}\left(k\left(\|s\|(T_s+T_d+\alpha\,T_D)\,\frac{\sqrt{m}}{\epsilon_z}+n\right)\right). \tag{4.21}$$

□

At this point, we can analyze how the final running time would change with additional information on the time necessary to run the required unitaries. The terms that can assume different values in the running time are $T_d$, $T_s$, and $T_D$.

Assuming that we store the matrix $D$ and the signal $s$ in QRAM or we have efficient quantum access, these terms assume value $\widetilde{O}(1)$ as explained in Section 2.2.1. In that case, the final running time assumes the value

$$\widetilde{O}\left(k\left(\|s\|\,\mu(D)\,\frac{\sqrt{m}}{\epsilon_z}+n\right)\right). \tag{4.22}$$

If we instead consider having efficient quantum access to the block-encoding $U_D$ and to $U_s$, but not to $U_d$, we can consider obtaining $U_d$ as the application of the block-encoding $U_D$ to a quantum state that corresponds to the superposition of the indices of the columns. This state is trivial to obtain in polylogarithmic time using a line of Hadamard gates. In this case, we must consider the complexity of applying a block-encoding to a state and, following Lemma 2.11, we obtain a running time of $\widetilde{O}(T_d) = \widetilde{O}\left(\frac{\mu(D)\kappa(D)}{\|D\|}\right)$.

### 4.1.4. Success probability

For this analysis, we keep the same style as the one of Bellante and Zanero [7] due to the similarity of the algorithms.

We must consider that, like the majority of quantum algorithms, our algorithm has a chance of failing. This failure probability results from the fact that the majority of the employed routines do not always output the desired results. We now examine the run-time overhead required to ensure a significant output with sufficient probability. The inner products at steps 4 and 8, the search at step 11, and the computation of the approximate

solution and its norm at step 15 are the procedures that increase the probability of failure.

We now define the runtime overhead related to these probabilities more precisely considering the same parameter $\delta$, which describes the probability of failure for all of them. Corollary 2.16.1, necessary for computing the inner products, fails with a probability smaller than $2\delta$ if we consider a running time overhead of $O(\log(\frac{m}{\delta}))$, since the procedure performs $m$ inner products. Corollary 2.18.1, used to find the maximum absolute value, fails with a probability smaller than $\delta$ considering a running time overhead of $O(\log(\frac{1}{\delta}))$. Lemma 2.11, used for the computation of the approximated solution and its norm, fails with a probability of $\delta$ with an overhead of $O(\log(\frac{1}{\delta}))$.

Our analysis proceeds first by computing the overall probability of success and, after that, computing the necessary extra runtime to guarantee a good output.

Each loop iteration has a success probability of

$$1 - p_f \geq (1-\delta)(1-2\delta)(1-2\delta)(1-\delta) \geq 1 - 6\delta + 13\delta^2 - 12\delta^3 + 4\delta^4, \tag{4.23}$$

where $p_f$ is the probability of failure.

Then, if we suppose the terms with a degree greater than one small enough, we can simplify it as

$$1 - p_f \geq 1 - 6\delta, \tag{4.24}$$

with a total run-time overhead of $O\left(\log(\frac{m}{\delta})^2 \log(\frac{1}{\delta})^2\right)$.

For this reason, we can consider the probability of getting the wrong solution in each iteration of the algorithm as bounded by $6\delta$. To find the final probability of failure of the whole algorithm, we can exploit the union bound considering that it performs $k$ iterations

$$p(\cup_i^k p_f) \leq \sum_i^k p_f \leq 6k\delta. \tag{4.25}$$

Finally, we can define the overall probability of success as $1 - \delta'$ adjusting the running time overhead to $O(\text{polylog}(\frac{km}{\delta'})) \sim \widetilde{O}(1)$.

## 4.2. Q-OMP

In this section, we present the quantum version of the algorithm known as Orthogonal Matching Pursuit. The algorithm builds an every-time-better approximation of a signal $s \in \mathbb{R}^n$, describing it as a combination of a subset of atoms of $D \in \mathbb{R}^{n \times m}$, as explained in Section 1.2.3.

In this version of the algorithm, we have exploited different quantum routines to obtain a speed-up in various steps of the classical algorithm. The first and most important choice we have made is the avoidance of the direct computation of the residual value $r \in \mathbb{R}^n$ to avoid tomography. This was possible thanks to an exit condition based on the distance between the approximated signal $\phi_t \in \mathbb{R}^n$ and the signal $s$, computed in a fully quantum way. We obtain the approximated signal $\phi_t$ by an orthogonal projection of the signal $s$ on the subspace generated by the selected atoms. This routine has an important speed-up in the quantum settings compared to classical computation.

It is also important to highlight how, thanks to the fact that the approximated signal $\phi_t$ is recomputed at each iteration starting from classical data, the error between the various iterations is not propagated.

The following sections provide a full explanation of how the algorithm works, with an estimation of the running time and an analysis of the errors obtained using quantum routines.

### 4.2.1. Algorithm

Algorithm 4.2 summarizes the Quantum Orthogonal Matching Pursuit procedure.

Algorithm 4.2 Quantum Orthogonal Matching Pursuit (Q-OMP)

**Input** Unitary $U_s$ that creates $|s\rangle$, where $s \in \mathbb{R}^n$ is the signal to analyse, $\|s\|$ stored classically. Quantum access to dictionary $D \in \mathbb{R}^{n\times m}$ via the column tree data structure. $L \in \mathbb{N}$ sparsity required, $\epsilon \in \mathbb{R}^+$ error reconstruction tolerance.
$\epsilon_f \in \mathbb{R}^+$ precision parameter on the squared norm of the residual, $\epsilon_z \in \mathbb{R}^+$ error on the inner product estimation, $\epsilon_x, \eta_x \in \mathbb{R}^+$ precision parameters on the solution.

**Output** Quantum state $|\widetilde{x}\rangle$ and $\widetilde{\|x\|}$ such that $x \in \mathbb{R}^m$ is an approximated solution of $\mathcal{P}_0^\epsilon$ (Def. 1.1) or set of selected atoms $\Lambda_k$ where $k = \min(L, t)$

Initialize: $t = 0$, $\Lambda_0 = \emptyset$, $D_0 = D$
**while** not ($t > L$ or $\widetilde{d^2} \leq \epsilon^2$) **do**
  Use inner product estimation with $U_s$ to create:
    $|\varphi_1\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle\,|\overline{z_1}_j\rangle$ where $z_{1j} = \langle d_j, s\rangle$ and $|\overline{z_1}_j - z_{1j}| \leq \frac{1}{2}\epsilon_z$
  **if** t=0 **then**
    $|\varphi\rangle = |\varphi_1\rangle$
  **else**
    Use inner product estimation with $U_{\phi_t}$ to create:
    $|\varphi_2\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle\,|\overline{z_2}_j\rangle$ where $z_{2j} = \langle d_j, \phi_t\rangle$ and $|\overline{z_2}_j - z_{2j}| \leq \frac{1}{2}\epsilon_z$
    Subtract the states $|\varphi_1\rangle$ and $|\varphi_2\rangle$, conditioned on the first register to be equal:
    $|\varphi\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle\,|\overline{z}_j\rangle$ where $z_j = \langle d_j, r\rangle$ and $|\overline{z}_j - z_j| \leq \epsilon_z$
  **end if**
  Apply finding the maximum on the unitary that creates $|\varphi\rangle$, to extract index $j^*$
  Update $t = t + 1$ and $\Lambda_t = \Lambda_{t-1} \cup j^*$
  Update the norm tree of $D_t$ in QRAM, removing the path to $d_{j^*}$
  Update the norm tree of $A_t$ in QRAM, adding the path to $d_{j^*}$
  Use quantum projection and norm estimation to obtain a unitary $\overline{U_{\phi_t}}$ that creates:
    $|\overline{\phi_t}\rangle$ where $|\phi_t\rangle = |A_t A_t^+ s\rangle$ with $\left\||\overline{\phi_t}\rangle - |\phi_t\rangle\right\| \leq \frac{1}{8\|s\|^2}\epsilon_f$,
    $\overline{\|\phi_t\|}$ where $\|\phi_t\| = \|A_t A_t^+ s\|$ with $\left|\overline{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8\|s\|}\epsilon_f$
  Use distance estimation with $U_s$ and $\overline{U_{\phi_t}}$ to obtain:
    $\widetilde{d^2}$ where $d^2(s, \phi_t) = \|r_t\|^2$ and $\left|\widetilde{d^2} - d^2(s, \phi_t)\right| \leq \epsilon_f$
  Use quantum projection and norm estimation to obtain a unitary $U_{\phi_t}$ that creates:
    $|\widetilde{\phi_t}\rangle$ where $|\phi_t\rangle = |A_t A_t^+ s\rangle$ with $\left\||\widetilde{\phi_t}\rangle - |\phi_t\rangle\right\| \leq \frac{1}{8\|s\|}\epsilon_z$,
    $\widetilde{\|\phi_t\|}$ where $\|\phi_t\| = \|A_t A_t^+ s\|$ with $\left|\widetilde{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8}\epsilon_z$
**end while**
Optionally, use quantum OLS and amplitude estimation to obtain:
    $|\widetilde{x}\rangle$ where $|x\rangle = |A_t^+ s\rangle$ and $\|\,|x\rangle - |\widetilde{x}\rangle\,\| \leq \epsilon_x$,
    $\widetilde{\|x\|}$ where $\|x\| = \|A_t^+ s\|$ and $\left|\widetilde{\|x\|} - \|x\|\right| \leq \eta_x\|x\|$
Output $|\widetilde{x}\rangle$ and $\widetilde{\|x\|}$ or $\Lambda_k$ with $k = \min(L, t)$

In this first section, we focus only on the various steps of the algorithm and not on the error analysis. The bounds on the errors added in the different routines will be analyzed extensively in Section 4.2.3. For the moment, it is enough for the reader to consider the user-provided errors $\epsilon_z \in \mathbb{R}^+$,$\epsilon_f \in \mathbb{R}^+$, $\epsilon_x \in \mathbb{R}^+$ and $\eta_x \in \mathbb{R}^+$ as all the other errors depend on them.

We start by initializing $t = 0$ and $\Lambda_0 = \emptyset$. For each iteration of the algorithm, we repeat the following steps.

We use the matrix-vector product estimation routine of Corollary 2.16.1 providing quantum access to the matrix $D_t$, the unitary $U_s$, and the known norm $\|s\|$ to obtain the unitary that produces the state

$$|\varphi_1\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle \, |\overline{z_{1j}}\rangle \, . \tag{4.26}$$

We select a precision for the procedure which guarantees to obtain values $\overline{z_{1j}}$ such that $|\overline{z_{1j}} - z_{1j}| \leq \frac{1}{2}\epsilon_z$, with $z_{1j} = \langle d_j, s\rangle$.

In the first iteration of the algorithm, we skip the computation of the second inner product, and we consider $|\varphi\rangle = |\varphi_1\rangle$.

If it is not the first iteration, we use again Corollary 2.16.1 providing quantum access to the matrix $D_t$, the unitary $U_{\phi_t}$ that generates the approximated signal $|\widetilde{\phi_t}\rangle$ and the approximated norm $\widetilde{\|\phi_t\|}$ to produce the unitary that creates the state

$$|\varphi_2\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle \, |\overline{z_{2j}}\rangle \, . \tag{4.27}$$

We use a precision for the procedure which guarantees to obtain values $\overline{z_{2j}}$ such that $|\overline{z_{2j}} - z_{2j}| \leq \frac{1}{2}\epsilon_z$, with $z_{2j} = \langle d_j, \phi_t\rangle$.

We will explain later in this section how to obtain the unitary $U_{\phi_t}$, that generates $|\widetilde{\phi_t}\rangle$, an approximation of the signal that we have described until now, and the norm $\widetilde{\|\phi_t\|}$.

Once we have the two quantum registers, $|\varphi_1\rangle$ and $|\varphi_2\rangle$, we perform a basis-encoded state subtraction between the second register of them conditioned on the first register having the same index. In this way, we obtain the unitary that generates the following quantum state

$$|\varphi\rangle = \frac{1}{\sqrt{m-t}} \sum_{j\in[m]\setminus\Lambda_t} |j\rangle \, |\overline{z}_j\rangle \, , \tag{4.28}$$

and we obtain $\overline{z}_j$ such that $|\overline{z}_j - z_j| \leq \epsilon_z$ with $z_j = \langle d_j, s - \phi_t\rangle = \langle d_j, s - A_t x\rangle = \langle d_j, r_t\rangle$.

Considering that we have defined an approximated unitary that creates $|\varphi\rangle$, which corresponds to a superposition of all the approximated inner products, we can now perform the finding the maximum absolute value routine from Corollary 2.18.1. We identify the index $j^*$ such that $j^* = \arg\max_{j\in[m]\setminus\Lambda_t} |\langle d_j, r_t\rangle|$.

With this value, we can update the iteration counter $t = t + 1$, the set of chosen indices $\Lambda_t = \Lambda_{t_1} \cup j^*$, and the matrices $A_t$ and $D_t$. We will better explain the update of the matrices in Section 4.2.2.

We now must define the unitary $\overline{U_{\phi_t}}$ that creates a state that approximates $|\phi_t\rangle = |A_t A_t^+ s\rangle$. This state will be required multiple times in a single iteration of the algorithm. To implement a faster unitary that gives the state, we estimate the norm $\|\phi_t\|$ applying Lemma 3.7 with as input the block-encoding of the matrix $A_t$, created following Theorem 2.9, the unitary $U_s$ that creates the state $|s\rangle$, and the norm $\|s\|$. We obtain an approximation called $\overline{\|\phi_t\|}$ with precision such that $\left|\overline{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8\|s\|}\epsilon_f$.

At this point, we define the unitary $\overline{U_{\phi_t}}$ that creates the approximated signal state. We use the procedure that solves the projection problem of Lemma 3.6 with as input the block-encoding of $A_t$, the unitary $U_s$, the norm $\|s\|$ and the norm $\overline{\|\phi_t\|}$. The procedures generates the state $|\overline{\phi_t}\rangle$ such that $\left\||\overline{\phi_t}\rangle - |\phi\rangle\right\| \leq \frac{1}{8\|s\|^2}\epsilon_f$.

Finally, we can estimate the Euclidean Distance between the signal $s$ and the vector $\phi_t$. We use Lemma 2.17 with as input the unitary $U_s$ that creates the state $|s\rangle$, the unitary $\overline{U_{\phi_t}}$ that creates the state $|\overline{\phi_t}\rangle$, the norms $\|s\|$ and $\overline{\|\phi_t\|}$. We obtain the approximated squared distance $\widetilde{d^2}$ such that

$$\left|\widetilde{d^2} - d^2(s, \phi_t)\right| \leq \epsilon_f. \tag{4.29}$$

We highlight that the squared value of the distance corresponds to the squared norm of the residual $\|r_t\|^2 = d^2(s, \phi_t)$.

At this point, for the following iteration, we need a unitary $U_{\phi_t}$ that produces the approximation of the state $|\phi_t\rangle$, but with a different level of accuracy. The reason behind the use of two different unitaries for the same state generation is related to the final running time of the procedure, as shown in Section 4.2.5. We can obtain the unitary $U_{\phi_t}$ using the same procedures that we have used to define $\overline{U_{\phi_t}}$ with the same inputs, with the exception of the required precision. At this point, we define the unitary that creates the state $|\widetilde{\phi_t}\rangle$ such that $\left\||\widetilde{\phi_t}\rangle - |\phi\rangle\right\| \leq \frac{1}{8\|s\|}\epsilon_z$ and we estimate the norm $\widetilde{\|\phi_t\|}$ such that $\left|\widetilde{\|\phi_t\|} - \|\phi_t\|\right| \leq \frac{1}{8}\epsilon_z$.

We repeat the procedure until the squared estimated norm of the residual is lower than

the squared value of the error reconstruction tolerance $\epsilon \in \mathbb{R}^+$ or the sparsity of the solution is such that $t = \|x\|_0 > L$ for a sparsity threshold $L \in \mathbb{N}$.

As an output of the procedure, we can provide a state $|\widetilde{x}\rangle$ such that $\| \, |\widetilde{x}\rangle - |x\rangle \, \| \leq \epsilon_x$ where $\epsilon_x \in \mathbb{R}^+$ is a precision parameter chosen by the user, and $|x\rangle = |A_t^+ s\rangle$. This is possible just by solving the OLS problem using Lemma 2.15 with as input the block-encoding of the matrix $A_t$ and the unitary $U_s$. After that, we can also obtain the value of the estimated norm $\widetilde{\|x\|}$ such that $|\widetilde{\|x\|} - \|x\|| \leq \eta_x \|x\|$ with $\eta_x \in \mathbb{R}^+$ just applying the second result of the same lemma and then multiplying the result for the known $\|s\|$.

Another option is to provide the solution classically and output the set of the indices of the selected atoms $\Lambda_k$ with $k = \min(L, t)$ that are already described in classical form and solve just one ordinary least square procedure classically. If the atoms were all correctly selected during the procedure and the algorithm stopped at the correct iteration, this solution is equal to the one obtained using the classical Orthogonal Matching Pursuit.

### 4.2.2. Creation of $A_t$ and $D_t$

For this algorithm we need to access $A_t \in \mathbb{R}^{n \times t}$ and $D_t \in \mathbb{R}^{n \times (m-t)}$ in polylogarithmic time. We recall that, at each iteration, one atom is selected and extracted from the matrix $D_t$ and added to the matrix $A_t$. In this section, we present a possible way of doing it, using two additional data structures to manage the selection of only one part of the atoms of the dictionary $D \in \mathbb{R}^{n \times m}$.

For the dictionary $D$, we can consider a structure similar to the one in Figure 4.1. In this structure, the main tree highlighted is the one that refers to the norm of the various columns of the matrix. Each column has a unitary norm and, for this reason, the root element that contains the sum of the squared value of the leaves contains the value $m$. We define this structure based on the one explained by Chen and de Wolf [17] in Section 2.4.

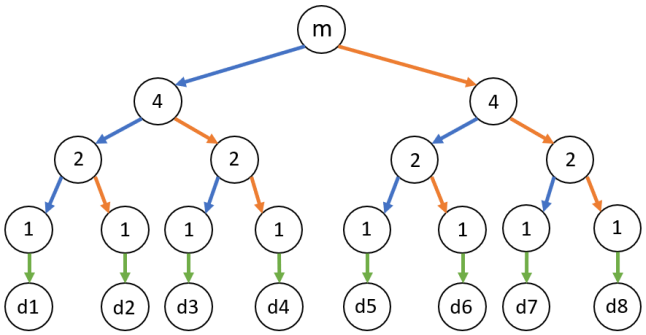



Figure 4.1: KP-Tree used for $D$

For each atom, instead, since these elements are already normalized, and no modifications are applied directly during the computation, we can consider a KP-Tree as explained in Appendix A of [41] by Kerenidis and Prakash. Figure 4.2 shows an example of an atom with 4 components.

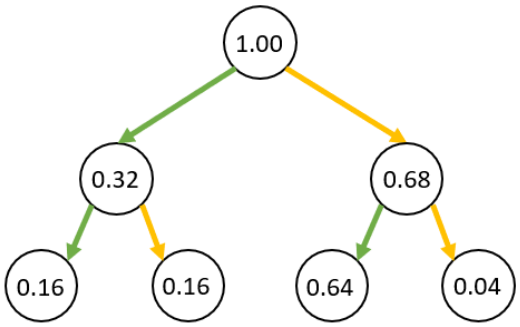
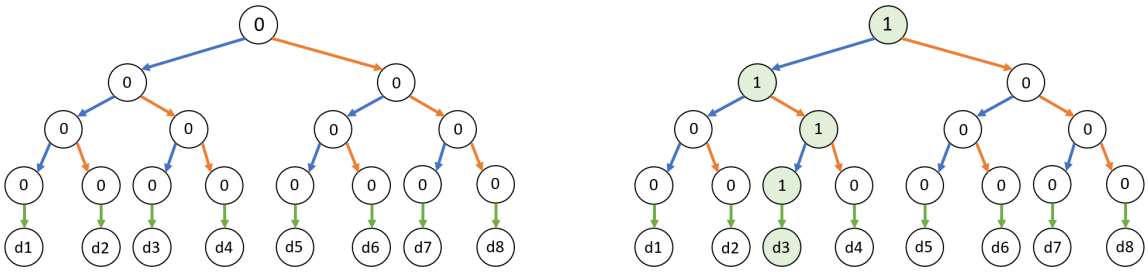



Figure 4.2: Example of KP-Tree used for an atom

We use these structures as the basis to construct the data structures used for matrices $A_t$ and $D_t$.

We can consider the data structure that stores the matrix $A_t$ as the same data structure of the dictionary $D$ with the difference that we initialize it with all 0 for what concerns the nodes of the column norm tree. At each step of computation, we update a single leaf and modify the value from 0 to 1. In addition, we also update all the other nodes on the path from the root node to the leaf adding 1 to the value previously stored. This update can be performed in time $O(\log(m))$.

Figure 4.3a shows the KP-Tree during the initialization, while Figure 4.3b shows the tree after the first insert.

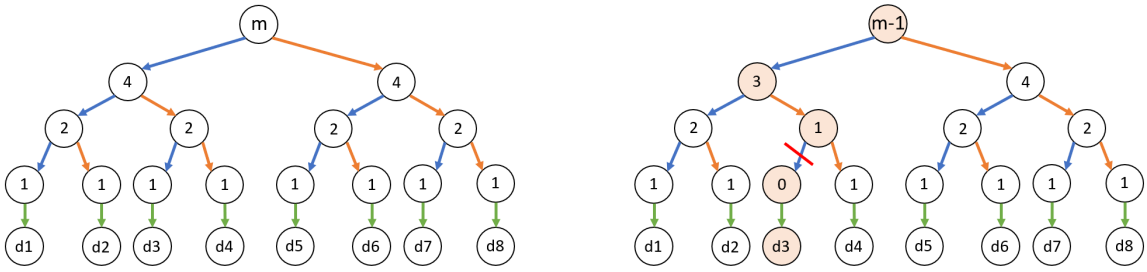



(a) Initialization of $A_t$ (b) First insert in $A_t$

Figure 4.3: KP-Tree used for $A_t$

The situation is similar for the matrix $D_t$. We start from the same data structure of the dictionary $D$ and modify its entries. We can obtain the data structure that describes $D_t$ at each step changing, from 1 to 0, the leaf value that refers to the norm of the atom that we remove. In addition, also all the nodes of the path to reach it are updated, decreasing their value by 1. This update can be performed in time $O(\log(m))$.

Figure 4.4a shows the KP-Tree during the initialization, while Figure 4.4b shows the tree after the first delete.




(a) Initialization of $D_t$ (b) First delete in $D_t$

Figure 4.4: KP-Tree used for $D_t$

Thanks to these data structures, it is possible to access the elements of matrices $A_t$ and $D_t$ in polylogarithmic time in relation to the size of the stored data. In this way, it is easy to obtain a block-encoded version of the matrices as explained in Section 2.2.1 with the $\alpha$ parameter equal to the Frobenius norm of the matrix.

A possible trivial optimization is to share the trees that describe the atoms between the two structures and differentiate just the trees that refer to the norms of the selected columns to use less memory space.

### 4.2.3. Error Propagation

In this algorithm there is no propagation of error between the different iterations. The error introduced in the solution is limited to one single iteration because, at each step, we compute a new solution starting from the atoms of the matrix $A_t$ and the signal $s$ that are stored classically.

At this point, we can have two different problems in our algorithm. The first one is the miss-selection of the greatest inner product, which can be caused by the error in the estimation of the inner products. We bound this value with $\epsilon_z \in \mathbb{R}^+$. This error is the same one considered in Section 4.1.2 of the Q-MP algorithm. In general, this value depends on the dictionary and the signal analyzed.

The second problem is that we must also consider estimating a good-enough value for the squared distance between the correct signal and the approximated one to stop the algorithm at the right moment. We define the maximum error on the estimated squared distance as $\epsilon_f \in \mathbb{R}^+$.

We want to highlight that we have considered an error $\epsilon_f$ on the squared estimation of the distance, which corresponds to an estimation of the squared residual norm $\|r\|^2$. It is possible to link the error on the squared norm of the residual with an error on the distance (and for this reason on $\|r\|$) called $\epsilon_d \in \mathbb{R}^+$ such that $|\overline{d} - d| \leq \epsilon_d$ noticing that

$$|\overline{d}^2 - d^2| = |\overline{d} - d||\overline{d} + d| \leq \epsilon_d|\overline{d} + d| \leq \epsilon_d|\overline{d} - d + 2d| \leq \epsilon_d(\epsilon_d + 2d) \tag{4.30}$$

Now, if we select $\epsilon_f = \epsilon_d^2 + 2d\epsilon_d$ we obtain that $\epsilon_d = -d \pm \sqrt{d^2 + \epsilon_f}$. We select the positive solution, and we bound the value obtaining $\epsilon_d = -d + \sqrt{d^2 + \epsilon_f} \leq -d + \sqrt{d^2} + \sqrt{\epsilon_f} = \sqrt{\epsilon_f}$. Selecting a value for the error on the squared distance $\epsilon_f$, we guarantee an error on the final residual norm of $\epsilon_d \leq \sqrt{\epsilon_f}$.

If we guarantee a low enough $\epsilon_z$ and the stop condition is correctly satisfied, we can obtain the same selection of the atoms of the classical algorithm. In Section 5.3.1, we provide some experimental results with different choices for these parameters.

We now analyze the error of the generic $t^{th}$ iteration. We consider the steps of the Algorithm 4.2 to explain where the error is introduced. Step number 1 and number 2 do not add any errors.

We consider steps 3 to 8 with the same error introduction as steps 4 to 9 of the Q-MP procedure described in Algorithm 4.1, with the same bound $\epsilon_z$ on the final approximation of the inner products. We report the values for the error parameters defined in Section

4.1.2 for the different routines. We can choose $\epsilon_{in_2} \leq \frac{1}{4\|\widetilde{\phi_t}\|}\epsilon_z$, $\epsilon_\phi \leq \frac{1}{8\|\phi_t\|}\epsilon_z$, $\eta \leq \frac{1}{8\|\phi_t\|}\epsilon_z$ and $\epsilon_{in_1} \leq \frac{1}{2\|s\|}\epsilon_z$ to guarantee an error on the final estimation of the inner products that is less or equal than $\epsilon_z$. We highlight that the parameters $\epsilon_\phi$ and $\eta$ are the precision parameters that are necessary for the definition of the unitary $U_{\phi_t}$.

Steps from number 9 to number 13 do not add any error.

Considering step number 14, we want to guarantee that the quantum state $|\overline{\phi_t}\rangle$, generated by the unitary $\overline{U_{\phi_t}}$, has a maximum error defined with the value $\overline{\epsilon_\phi} \in \mathbb{R}^+$. For this reason, we must have an error in the block-encoding of the matrix $A_t$ that satisfies the conditions of Lemma 3.6. We define the error $\overline{\zeta} \in \mathbb{R}^+$ in the block-encoding such that

$$\overline{\zeta} \leq \frac{\overline{\epsilon_\phi}\|A_t^T A_t\|}{64\sqrt{L}\,\kappa(A_t)^4 log^3(2\kappa(A_t)/\overline{\epsilon_\phi})} \tag{4.31}$$

In addition to that, we denote the relative error on the approximated norm $\overline{\|\phi_t\|}$ with the value $\overline{\eta} \in \mathbb{R}^+$.

Considering step 15, we want to bound the error on the final squared distance estimation $\widetilde{d}^2$ with the value $\epsilon_f$ as explained at the beginning of the section. For this reason, we must consider that some error came from the approximation of the vector $\phi_t$, obtained from the unitary $\overline{U_{\phi_t}}$, used in the computation of the distance. Consequently, we must find a value to bound the parameters $\overline{\epsilon_\phi}$ and $\overline{\eta}$ using the value $\epsilon_f$. In addition, it is necessary to compute a bound on the precision of the routine defined in Lemma 2.17 used to perform the distance estimation with the normalized states called $\epsilon_{in_3} \in \mathbb{R}^+$. To obtain the bound we can use Claim 3.4 with as input the signal $s$, its norm $\|s\|$, and the values $|\overline{\phi}\rangle$ and $\overline{\|\phi\|}$. In this case, if we consider $\epsilon_f$ as required bound, we can guarantee that $3\overline{\eta}\|\phi\|^2 + 2\|s\|\|\phi\|\overline{\eta} + 2\|s\|\|\phi\|\overline{\epsilon_\phi} + 2\|s\|\overline{\|\phi\|}\epsilon_{in_3} \leq \epsilon_f$.

Considering that the norm of the signal $s$ is an upper bound of the norm of the approximated signal $\phi_t$, we can choose the following bounds for the error considered. We define $\epsilon_{in_3} \leq \frac{1}{16\|s\|^2}\epsilon_f$, $\overline{\epsilon_\phi} \leq \frac{1}{8\|s\|^2}\epsilon_f$ and, $\overline{\eta} \leq \frac{1}{8\|s\|^2}\epsilon_f$.

In the same way as for step number 14, considering step 16, we must guarantee that the quantum state $|\widetilde{\phi_t}\rangle$, generated by the unitary $U_{\phi_t}$, has a maximum error defined with the value $\epsilon_\phi \in \mathbb{R}^+$. For this reason, we must have an error in the block-encoding of the matrix $A_t$ that satisfies the conditions of Lemma 3.6. We define the error $\zeta \in \mathbb{R}^+$ in the block-encoding such that

$$\zeta \leq \frac{\epsilon_\phi\|A_t^T A_t\|}{64\sqrt{L}\,\kappa(A_t)^4 log^3(2\kappa(A_t)/\epsilon_\phi)} \tag{4.32}$$

Finally, the last step in which some error is introduced is step 18. As we have already mentioned, this step is executed once in the algorithm. The user can select the errors $\epsilon_x$ and $\eta_x$ on the final solution trying to minimize the error as much as possible. The choice of these errors does not modify the overall procedure but just the quality of the quantum output. For this reason, we do not provide bounds on them. In addition, it is also possible to obtain a classical output composed of the selected atoms and solve just one OLS problem classically, obtaining a solution without approximation error.

## Summary of the errors

We here summarize all the errors analyzed with a dependency on $\epsilon_z$ or $\epsilon_f$. We will use these values in the estimation of the running time.

The errors that have a dependency on $\epsilon_z$ are:

- $\epsilon_{in_1} \leq \epsilon_z/(2\|s\|)$
- $\epsilon_{in_2} \leq \epsilon_z/(4\widetilde{\|\phi\|})$
- $\epsilon_\phi \leq \epsilon_z/(8\|s\|) \leq \epsilon_z/(8\|\phi\|)$
- $\eta \leq \epsilon_z/(8\|s\|) \leq \epsilon_z/(8\|\phi\|)$
- $\zeta \leq \frac{\epsilon_\phi \|A_t^T A_t\|}{64\sqrt{L}\,\kappa(A_t)^4 log^3(2\kappa(A_t)/\epsilon_\phi)}$

While the errors with a dependency on $\epsilon_f$ are:

- $\overline{\epsilon_\phi} \leq \epsilon_f/(8\|s\|^2)$
- $\overline{\eta} \leq \epsilon_f/(8\|s\|^2)$
- $\epsilon_{in_3} \leq \epsilon_f/(16\|s\|^2)$
- $\overline{\zeta} \leq \frac{\overline{\epsilon_\phi} \|A_t^T A_t\|}{64\sqrt{L}\,\kappa(A_t)^4 log^3(2\kappa(A_t)/\overline{\epsilon_\phi})}$

It is possible to see how the errors that are upper-bounded by values in which the unknown value $\|\phi\|$ is present can be easily bounded by substituting it with the norm $\|s\|$. We also want to highlight that it is possible to avoid the conditions that arise from the exit condition by solving the problem for a fixed amount of iterations. In that case, we do not consider all the errors that depend on $\epsilon_f$.

### 4.2.4. Bounds for running time parameters

Proving the running time in the next section, we will encounter the parameters $\mu(A_t)$ and $\kappa(A_t)$ related to the submatrices $A_t$ of $D$ for the different steps $t = [0, ..., k]$ where $k$ is the final number of iterations. The value of these two parameters varies during the different iterations of the algorithm, and a precise estimation of them is not possible because they depend on the atoms chosen at each step of the algorithm. For this reason, we want to find a bound for these values to obtain a running time that does not depend on them.

First, we find an upper bound for the value of $\mu(A_t)$.

**Claim 4.2.** *Let $D \in \mathbb{R}^{n \times m}$, with $n < m$, be a matrix with columns with unitary $l_2$-norm. Let $I_t \subset \{0, 1, ..., m-1\}$ be a subset of $t$ indices of the columns of the matrix $D$. We define the matrix $A_t \in \mathbb{R}^{n \times t}$ as the sub-matrix built using only the columns with index in the subset $I_t$. Let $k \in \mathbb{N}$, we can say that for each matrix $A_t$ with $t = 0, 1, ..., k$ we have that $\mu(A_t) \leq \sqrt{k}$.*

*Proof.* First of all, consider that for each matrix $A_t$ we have $\mu(A_t) \leq \|A_t\|_F$ and that $\|A_t\|_F = \sqrt{t}$ because the columns are unitary vectors. We know that $k$ is the greatest possible value of $t$ by definition and, for this reason, we have $\mu(A_t) \leq \sqrt{t} \leq \sqrt{k}$. □

Also, the parameter $\kappa(A_t)$ is easy to bound, as shown in the following claim.

**Claim 4.3.** *Let $D \in \mathbb{R}^{n \times m}$, with $n < m$, be a matrix with columns with unitary $l_2$-norm. Let be $I_t \subset \{0, 1, ..., m-1\}$ a subset of $t$ indices of the columns of the matrix $D$. We define the matrix $A_t \in \mathbb{R}^{n \times t}$ as the sub-matrix built using only the columns with the index in the subset $I_t$. We can say that $\kappa(A_t) \leq \kappa(D)$.*

*Proof.* We use the variational characterization of the singular values. We define the sets

$$S_1 = \left\{ \frac{\|Dx\|}{\|x\|} = 1 \right\}, \tag{4.33}$$

and

$$S_2 = \left\{ \frac{\|Dx\|}{\|x\|} = 1 \text{ and } x_i = 0 \text{ for all } i \notin I_t \right\}. \tag{4.34}$$

We consider $\sigma_{max}(D) = \max(S_1)$ as the maximum singular value of $D$. We know that the maximum singular value of a sub-matrix $A_t$, defined as $\sigma_{max}(A_t) = \max(S_2)$, is $\sigma_{max}(A_t) \leq \sigma_{max}(D)$. We also define $\sigma_{min}(D) = \min(S_1)$ as as the minimum singular value of $D$. We know that, given $\sigma_{min}(A_t) = \min(S_2)$ the minimum singular value of $A_t$,

we have $\sigma_{min}(A_t) \geq \sigma_{min}(D)$. As the condition number $\kappa$ is defined as $\kappa = \frac{\sigma_{max}}{\sigma_{min}}$, we have that $\kappa(D) = \frac{\sigma_{max}(D)}{\sigma_{min}(D)} \geq \frac{\sigma_{max}(A_t)}{\sigma_{min}(A_t)} = \kappa(A_t)$. □

For this reason, we can use as the upper bound of each instance of $\kappa(A_t)$ the value $\kappa(D)$. For well-conditioned dictionaries, the value $\kappa(D)$ will be a small integer number, as shown in Section 5.3.1.

We also define a claim to bound the norm of the solution vector that will appear in the running time.

**Claim 4.4.** *Let $A_t \in \mathbb{R}^{n\times t}$ be a submatrix of the dictionary $D \in \mathbb{R}^{n\times m}$ composed by $t$ atoms and $s \in \mathbb{R}^n$ a vector. Let $x_t \in \mathbb{R}^t$ be a vector such that $x_t = A_t^+ s$, we can considered that $\|x_t\| \leq \frac{\|s\|\kappa(D)}{\|D\|}$.*

*Proof.* The proof is trivial to obtain considering that $\|x_t\| = \|A_t^+ s\| \leq \|A_t^+\|\|s\| = \|s\|\frac{\kappa(A_t)}{\|A_t\|}$. We know that $\frac{\kappa(A_t)}{\|A_t\|} = \frac{1}{\sigma_{min}(A_t)}$ where $\sigma_{min}(A_t)$ is the smallest singular value of $A_t$ and $\|A_t^+\| = \sigma_{max}(A_t^+) = \frac{1}{\sigma_{min}(A_t)}$. Then, if we consider that, as explained in Claim 4.3, $\sigma_{min}(A_t) \geq \sigma_{min}(D)$ and $\frac{\kappa(D)}{\|D\|} = \frac{1}{\sigma_{min}(D)}$, we can say that $\frac{\kappa(A_t)}{\|A_t\|} \leq \frac{\kappa(D)}{\|D\|}$. For this reason, we obtain $\|x_t\| \leq \|s\|\frac{\kappa(A_t)}{\|A_t\|} \leq \|s\|\frac{\kappa(D)}{\|D\|}$ □

### 4.2.5. Running Time

**Theorem 4.5** (Quantum Orthogonal Matching Pursuit)**.** *Let $D \in \mathbb{R}^{n\times m}$ be the dictionary matrix stored in an efficient data structure such that we have efficient quantum access to its submatrices. Let $U_s$ be a unitary that generates the state $|s\rangle$ where $s \in \mathbb{R}^n$ is the signal to analyze, in time $O(T_s)$. Let $\|s\|$ be the norm of the signal stored classically and let $L \in \mathbb{N}$ be a sparsity threshold such that $L \leq m$. Let $\epsilon_z \in \mathbb{R}^+$ be a precision parameter on the estimation of the inner products. Let $\epsilon \in \mathbb{R}^+$ be the error reconstruction tolerance and let $\epsilon_f \in \mathbb{R}^+$ be the maximum error on the value of the final squared norm of the residual vector. Then, there exists a procedure that obtains an approximated solution similar to the one of the orthogonal matching pursuit algorithm, with high probability, in time*

$$\widetilde{O}\left(k^{3/2}\|s\|\frac{\kappa(D)^2}{\|D\|}\left(\frac{\sqrt{m}}{\epsilon_z} + \frac{\|s\|}{\epsilon_f}\right)T_s\right). \tag{4.35}$$

*where $k \leq L$ is the total number of iterations.*

If we also consider that $\epsilon_f \geq \frac{\|s\|}{\sqrt{m}}\epsilon_z$, we obtain a running time of

$$\widetilde{O}\left(k^{3/2}\|s\|\frac{\kappa(D)^2}{\|D\|}\frac{\sqrt{m}}{\epsilon_z}T_s\right). \tag{4.36}$$

*Proof.* The proof consists in proving the running time of the algorithm.

From Corollary 2.16.1, we know that the cost of step 3 is $\widetilde{O}\left(\frac{T_s}{\epsilon_{in_1}}\right)$ because we can create the superposition of the columns of the matrix $D_t$ in $\widetilde{O}(1)$ and the time to generate the state $|s\rangle$ is $O(T_s)$. Therefore, we can prepare the state $|\varphi_1\rangle$ in time $\widetilde{O}\left(\frac{T_s}{\epsilon_{in_1}}\right) \sim \widetilde{O}\left(\frac{\|s\|T_s}{\epsilon_z}\right)$.

The following step needs access to the columns of the matrix, obtained in $\widetilde{O}(1)$ as discussed previously, and the unitary that generates the state $|\widetilde{\phi_t}\rangle$. For the moment, we define the overall required time to obtain $|\widetilde{\phi_t}\rangle$ as $\widetilde{O}(T)$, and we will explain its value later in this section. Also the norm $\widetilde{\|\phi_t\|}$ is needed, but it is enough to estimate it just once for each iteration of the algorithm. We obtain this result in time $O(T_{norm})$.

Given that the cost of preparing the necessary inputs of the procedure can be bounded by $O(T)$, we know that the cost of step 4, from Corollary 2.16.1, is $\widetilde{O}\left(\frac{T}{\epsilon_{in_2}}\right)$. Therefore, we can prepare the state $|\varphi_2\rangle$ in time $\widetilde{O}\left(\frac{T}{\epsilon_{in_2}}\right) \sim \widetilde{O}\left(\frac{\widetilde{\|\phi_t\|}T}{\epsilon_z}\right)$.

The subtraction of the second register of the two quantum states $|\varphi_1\rangle$ and $|\varphi_2\rangle$ can be done in time $\widetilde{O}(1)$ and we obtain the state $|\varphi\rangle$. The total complexity of these first three steps is $\widetilde{O}\left(\frac{\widetilde{\|\phi_t\|}T+\|s\|T_s}{\epsilon_z}\right)$.

Using Corollary 2.18.1, we can find the index $j^*$ of the greatest value of $|\varphi\rangle$ in time $\widetilde{O}\left(\frac{\sqrt{m}}{\epsilon_z}(\widetilde{\|\phi_t\|}T + \|s\|T_s)\right)$. We update the iteration counter $t$ and of the matrices $A_t$ and $D_t$ in $\widetilde{O}(1)$.

The last step is the computation of the squared norm of the residual used to check if the algorithm must continue the iterations or not. For the moment, we denote this time with $\widetilde{O}(T_{dis})$, and we will explain later in the section its value.

The cost of the $i^{th}$ iteration of the loop, results in

$$O\left(\left(\frac{\widetilde{\|\phi_t\|}\sqrt{m}}{\epsilon_z}\right)T + \left(\frac{\|s\|\sqrt{m}}{\epsilon_z}\right)T_s + T_{norm} + T_{dis}\right). \tag{4.37}$$

Assuming that we perform $k$ iterations of the loop, the complexity of our algorithm is

$$O\left(k\left(\left(\frac{\widetilde{\|\phi_t\|}\sqrt{m}}{\epsilon_z}\right)T + \left(\frac{\|s\|\sqrt{m}}{\epsilon_z}\right)T_s + T_{norm} + T_{dis}\right)\right). \tag{4.38}$$

We consider that it is possible to create the block encoding of the matrix $A_t$ in $\widetilde{O}(1)$ following Theorem 2.9. The state $|s\rangle$ and its norm $\|s\|$ can be obtained in $\widetilde{O}(T_s)$. With these variables, we can now proceed in defining the value of $\widetilde{O}(T_{norm})$ and $\widetilde{O}(T)$.

The time needed to compute the norm of the vector of the approximated signal, namely $\widetilde{\|\phi_t\|}$, is $\widetilde{O}(T_{norm})$ and corresponds to the one obtained using Lemma 3.7

$$O\left(T_{norm}\right) = \widetilde{O}\left(\frac{1}{\eta}\left(\frac{\kappa(A_t)^2\mu(A_t)}{\|A_t\|}\right)T_s\right) \sim \widetilde{O}\left(\frac{\|s\|}{\epsilon_z}\left(\frac{\kappa(A_t)^2\mu(A_t)}{\|A_t\|}\right)T_s\right). \tag{4.39}$$

The preparation of the state $|\widetilde{\phi_t}\rangle$ denoted till now by $O(T)$ corresponds to a projection, and we can obtain its complexity following Lemma 3.6

$$O\left(T\right) = \widetilde{O}\left(\frac{\|x\|\kappa(A_t)\mu(A_t)}{\widetilde{\|\phi_t\|}}T_s\right). \tag{4.40}$$

Considering than that $\kappa(A_t) \leq \kappa(D)$, $\mu(A_t) \leq \sqrt{k}$ and $\frac{\kappa(A_t)}{\|A_t\|} \leq \frac{\kappa(D)}{\|D\|}$ as explained in Section 4.2.4, and substituting Equation 4.39 and Equation 4.40 in Equation 4.38, we can consider the running time as

$$\widetilde{O}\left(k\left(\frac{\|x\|k^{1/2}\kappa(D)}{\widetilde{\|\phi_t\|}}\left(\frac{\widetilde{\|\phi_t\|}\sqrt{m}}{\epsilon_z}\right)T_s + \frac{\|s\|}{\epsilon_z}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s + \left(\frac{\|s\|\sqrt{m}}{\epsilon_z}\right)T_s + T_{dis}\right)\right). \tag{4.41}$$

We can considered that $\|x\| \leq \frac{\|s\|\kappa(D)}{\|D\|}$, as explained in Claim 4.4, obtaining

$$\widetilde{O}\left(k\left(\left(\frac{\|s\|}{\epsilon_z}\frac{k^{1/2}\kappa(D)^2\sqrt{m}}{\|D\|} + \frac{\|s\|}{\epsilon_z}\frac{k^{1/2}\kappa(D)^2}{\|D\|} + \frac{\|s\|\sqrt{m}}{\epsilon_z}\right)T_s + T_{dis}\right)\right). \tag{4.42}$$

We can ignore the two middle terms in the final running time because they are upper-bounded by the left one.

For this reason, the final running time, without expanding the term that considers the estimation of the distance, corresponds to

$$\widetilde{O}\left(k\left(\|s\|\frac{\sqrt{m}}{\epsilon_z}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s + T_{dis}\right)\right). \tag{4.43}$$

We now analyze the value of $\widetilde{O}(T_{dis})$. This term corresponds to the time necessary to estimate the values $|\overline{\phi_t}\rangle$ and $\overline{\|\phi_t\|}$ and, after that, computing the distance between the

approximation of the signal and the correct value of $s$. We follow the same reasoning that we have used to define the time to create $|\widetilde{\phi_t}\rangle$ and $\widetilde{\|\phi_t\|}$. For this reason, we define the times $\widetilde{O}(\overline{T})$ and $\widetilde{O}(\overline{T_{norm}})$.

The value of the complexity $\widetilde{O}(\overline{T_{norm}})$, necessary to obtain the value $\overline{\|\phi_t\|}$, following Lemma 3.7 corresponds to

$$O\left(\overline{T_{norm}}\right) = \widetilde{O}\left(\frac{1}{\overline{\eta}}\left(\frac{\kappa(A_t)^2\mu(A_t)}{\|A_t\|}\right)T_s\right) \sim \widetilde{O}\left(\frac{\|s\|^2}{\epsilon_f}\left(\frac{\kappa(A_t)^2\mu(A_t)}{\|A_t\|}\right)T_s\right). \tag{4.44}$$

The preparation of the state $|\overline{\phi_t}\rangle$ denoted by $O(\overline{T})$ corresponds to a projection, and its complexity can be obtained following Lemma 3.6, as done for $\widetilde{O}(T)$

$$O\left(\overline{T}\right) = \widetilde{O}\left(\frac{\|x\|\kappa(A_t)\mu(A_t)}{\overline{\|\phi_t\|}}T_s\right). \tag{4.45}$$

Following Lemma 2.17, to compute the approximation of the squared distance called $\widetilde{d^2}$ we need

$$\widetilde{O}\left(\frac{\overline{T}+T_s}{\epsilon_{in_3}} + \overline{T_{norm}}\right) \sim \widetilde{O}\left(\frac{\|s\|\overline{\|\phi_t\|}}{\epsilon_f}(\overline{T}+T_s) + \overline{T_{norm}}\right). \tag{4.46}$$

Considering again than that $\kappa(A_t) \leq \kappa(D)$, $\mu(A_t) \leq \sqrt{k}$ and $\frac{\kappa(A_t)}{\|A_t\|} \leq \frac{\kappa(D)}{\|D\|}$ and substituting the results of Equation 4.45 and Equation 4.44 inside Equation 4.46 we obtain

$$\widetilde{O}(T_{dis}) = \widetilde{O}\left(\frac{\|s\|\overline{\|\phi_t\|}}{\epsilon_f}\left(\frac{\|x\|k^{1/2}\kappa(D)}{\overline{\|\phi_t\|}}T_s + T_s\right) + \frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\,\kappa(D)^2}{\|D\|}\right)T_s\right). \tag{4.47}$$

Starting from the equation above, we can considered that $\|x\| \leq \frac{\|s\|\kappa(D)}{\|D\|}$ as explained in Claim 4.4 obtaining

$$\widetilde{O}\left(\frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s + \frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\,\kappa(D)^2}{\|D\|}\right)T_s\right) \sim \widetilde{O}\left(\frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s\right). \tag{4.48}$$

Substituting the value of $\widetilde{O}(T_{dis})$ of Equation 4.48 in Equation 4.43, we obtain

$$\widetilde{O}\left(k\left(\|s\|\frac{\sqrt{m}}{\epsilon_z}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s + \frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\kappa(D)^2}{\|D\|}\right)T_s\right)\right). \tag{4.49}$$

Grouping the terms in a better way, we obtain

$$\widetilde{O}\left(k^{3/2}\,\|s\|\frac{\kappa(D)^2}{\|D\|}\left(\frac{\sqrt{m}}{\epsilon_z}+\frac{\|s\|}{\epsilon_f}\right)T_s\right). \tag{4.50}$$

If we choose a value of $\epsilon_f$ such that

$$\epsilon_f \geq \frac{\|s\|\epsilon_z}{\sqrt{m}}, \tag{4.51}$$

the final complexity of the algorithm corresponds to

$$\widetilde{O}\left(k^{3/2}\,\|s\|\frac{\kappa(D)^2}{\|D\|}\frac{\sqrt{m}}{\epsilon_z}T_s\right). \tag{4.52}$$

□

### 4.2.6. Theoretical Properties

We want to point out that if the norm of the solution vector $x$ is not too far from the norm of the approximated signal $\phi_t$, we can consider a better running time. This happens in the case of the presence of the Restricted Isometry Property for the dictionary used, as already introduced in Section 1.2.3. We want to highlight that many fast algorithms [58, 59] had been created exploiting this property with a low running time. We want to show how this property helps to lower the dependency on the value $\kappa(D)$ for our algorithm.

**Theorem 4.6** (Quantum Orthogonal Matching Pursuit with RIP property). *Let $D \in \mathbb{R}^{n\times m}$ be the dictionary matrix with Restricted Isometry Property stored in an efficient data structure such that we have efficient quantum access to its submatrices in polylogarithmic time and*

$$\sqrt{m} \geq \frac{\kappa(D)}{\|D\|}. \tag{4.53}$$

*Let $U_s$ be a unitary that generates the state $|s\rangle$ where $s \in \mathbb{R}^n$ is the signal to analyze in time $O(T_s)$. Let $\|s\|$ be the known norm of the signal stored classically and let $L \in \mathbb{N}$ be a sparsity threshold such that $L \leq m$. Let $\epsilon_z \in \mathbb{R}^+$ be a precision parameter on the estimation of the inner products. Let $\epsilon \in \mathbb{R}^+$ be the error reconstruction tolerance and let $\epsilon_f \in \mathbb{R}^+$ be the maximum error on the value of the final squared norm of the residual vector such that*

$$\epsilon_f \geq \|s\|\epsilon_z. \tag{4.54}$$

*Then, there exists a procedure that simulates the orthogonal matching pursuit algorithm*

*with high probability in running time*

$$\widetilde{O}\left(k^{3/2}\,\|s\|\kappa(D)\frac{\sqrt{m}}{\epsilon_z}T_s\right). \tag{4.55}$$

*where $k \leq L$ is the total number of iterations.*

*Proof.* The proof consists in proving the running time of the algorithm.

The proof follows strictly the one of Theorem 4.5 with the only difference that, starting from Equation 4.41, and considering that $\widetilde{O}(\|x\|) = \widetilde{O}(\|A_t x\|) = \widetilde{O}(\|\phi_t\|)$ for the RIP we can consider

$$\widetilde{O}\left(k\left(\left(\|\phi_t\|k^{1/2}\kappa(D)\frac{\sqrt{m}}{\epsilon_z} + \frac{\|s\|}{\epsilon_z}\frac{k^{1/2}\kappa(D)^2}{\|D\|} + \frac{\|s\|\sqrt{m}}{\epsilon_z}\right)T_s + T_{dis}\right)\right). \tag{4.56}$$

Knowing that $\|\phi_t\| \leq \|s\|$ because $\phi_t$ is an approximation of $s$, and considering $\sqrt{m} \geq \frac{\kappa(D)}{\|D\|}$, we can say that the complexity is

$$\widetilde{O}\left(k\left(k^{1/2}\kappa(D)\frac{\|s\|\sqrt{m}}{\epsilon_z}T_s + T_{dis}\right)\right). \tag{4.57}$$

Now we must analyze the running time necessary to estimate the distance. For this reason we start from Equation 4.47 and, again, we substitute the value $\|x\|$ with $\|s\|$

$$\widetilde{O}(T_{dis}) = \widetilde{O}\left(\frac{\|s\|^2}{\epsilon_f}\left(k^{1/2}\kappa(D)\right)T_s + \frac{\|s\|^2}{\epsilon_f}\left(\frac{k^{1/2}\,\kappa(D)^2}{\|D\|}\right)T_s\right) \tag{4.58}$$

At this point we impose that $\epsilon_f \geq \|s\|\epsilon_z$ and we obtain a final running time of

$$\widetilde{O}\left(k^{3/2}\,\|s\|\kappa(D)\frac{\sqrt{m}}{\epsilon_z}T_s\right). \tag{4.59}$$

□

As already shown in Section 1.2.3, exploiting the mutual incoherence value of a dictionary, it is possible to obtain interesting results from Theorem 1.4 and Theorem 1.5. We recall that it is possible to find a bound on the number of iterations that the algorithm needs to exactly recover a sparse signal and have a condition on the uniqueness of the solution. As a general intuition, the reader can think that if we introduce a small amount of error in the quantum procedure, we obtain the same theoretical properties as the classical one. We analyze this intuition more in detail, obtaining a condition on the uniqueness of the

solution in the quantum case.

**Theorem 4.7** (Quantum condition on the uniqueness of solution). *Let $D \in \mathbb{R}^{n\times m}$ be a matrix with $l_2$-normalized columns and $m \gg n$ and let $s \in \mathbb{R}^n$. Given that, during the procedure, we can guarantee an absolute error on the inner products between the atoms of maximum value $\epsilon_z \in \mathbb{R}^+$, if $Dx = s$ with $x \in \mathbb{R}^m$, we know that the sparsity of the solution is:*

$$\|x\|_0 < \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)}, \tag{4.60}$$

*where $\gamma$ is the mutual incoherence of the column vectors of $D$, and that $x$ is the unique solution of Problem $\mathcal{P}_0^0$ (Def. 1.1).*

*Proof.* The proof proceeds by contradiction and is based on the one of [4].

We define the solution $x$ with sparsity $\|x\|_0 \leq \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)}$. Let $y$ be a different solution with the same bound on the sparsity of $x$, $\|y\|_0 \leq \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)}$. If we consider the difference between the two solutions, we obtain a value $z$ such that

$$\|z\|_0 = \|x-y\|_0 \leq \|x\|_0 + \|y\|_0 < \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)} + \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)} = \frac{1-\epsilon_z}{\gamma+\epsilon_z} \tag{4.61}$$

For this reason, we can define the sparsity of $z$ as $\|z\|_0 < \frac{1-\epsilon_z}{\gamma+\epsilon_z}$. Since we know that both $x$ and $y$ are exact solutions, we can say that $Dx - Dy = 0$ and $D(x-y) = Dz = 0$. It is trivial to notice that if we have $Dz = 0$, we can say that $\langle Dz, Dz\rangle = z^T D^T Dz = \langle 0, 0\rangle = 0$. If we define the matrix $M = D^T D$, we know that $z^T Mz = 0$.

Let us define $S = supp(z)$, the indices of the components of $z$ that are not 0. We can consider the condition $z_s^T M_s z_s = 0$ instead of $z^T Mz = 0$, where in $z_s$ and $M_s$ we consider only the indices in $S$. Considering $z_s^T M_s z_s$, we know that $z_s$ is a vector without zero elements. For this reason, to guarantee that $z_s^T M_s z_s = 0$, we must consider the columns or the rows of the matrix $M_s$ linear dependent. For this reason, the matrix $M_s$ is **singular**.

Now, we proceed with the second part of the proof, where we show how it is impossible to have a singular matrix with the assumption of the theorem. It is easy to see how the matrix $M_s$ is a sub-block matrix of $D^T D$ in which the elements are the inner products between the atoms $\langle d_i, d_j\rangle$ for some $i, j \in [m]$. We know, by assumption, that D is $\gamma$-incoherent and that the inner products obtained by the procedure have maximum error $\epsilon_z \in \mathbb{R}^+$. At this point, the elements of the matrix $M_s$ on the diagonal are upper bounded

by $1+\epsilon_z$ and lower bounded by $1-\epsilon_z$, while the others are upper bounded by $\gamma+\epsilon_z$.

$$M_s = \begin{bmatrix} d & e & e & ... \\ e & d & e & ... \\ e & e & d & ... \\ ... & ... & ... & ... \end{bmatrix}, \text{ with } |e| \leq \gamma+\epsilon_z \text{ and } 1-\epsilon_z \leq |d| \leq 1+\epsilon_z$$

At this point, we must apply the Gershgorin Circle Theorem [6] that states that every eigenvalue of a matrix lies in one of the Gerhsgorin discs. These discs are bounded by circles, with the center in one of the diagonal elements of the matrix and a radius equal to the sum of the absolute values of the non-diagonal entries in the same row. Our objective is to avoid the value zero for the eigenvalues. For this reason, we want a disc with a positive center and radius such that it does not pass below the origin point (0,0). We define a disc with the center in the worst possible point $(1-\epsilon_z, 0)$. We impose the radius less than $1-\epsilon_z$ to guarantee that all the eigenvalues of the matrix $M_s$ will be different from zero following the theorem. This result can be obtained if we satisfy the following

$$\|z\|_0(\gamma+\epsilon_z) < 1-\epsilon_z. \tag{4.62}$$

From this condition, it is easy to define a bound on the sparsity of $z$ as

$$\|z\|_0 < \frac{1-\epsilon_z}{\gamma+\epsilon_z} \tag{4.63}$$

If the condition is satisfied, we can guarantee that the matrix $M_s$ is **non-singular** and creates a contradiction with the previous result.

For this reason, we can conclude that there is a unique solution if the condition $\|x\|_0 < \frac{1-\epsilon_z}{2(\gamma+\epsilon_z)}$ is satisfied. □

### 4.2.7. Success Probability

For this analysis, as for Q-MP in Section 4.1.4, we keep the same style as the one from Bellante and Zanero [7] due to the similarity of the algorithms.

Our algorithm has a certain probability of failure that comes from the fact that the majority of the routines are not guaranteed to output the correct value. In this section, we study the run-time overhead necessary to guarantee a good output with a high-enough probability.

The computation of the inner products in steps 3 and 7, the search in step 10, and the

distance computation in step 15 generate the probability of failure. In addition to them, also the computation of the approximations of the state $|\phi\rangle$ adds a possible error both in the computation of steps 14 and 16 due to the amplitude estimation and amplification routines used in the routine that performs the projection.

To keep the analysis clear, we consider running the algorithm for a fixed amount of iterations $L$ and avoiding the check on the exit condition.

We consider, for simplicity, the same parameter to describe the probabilities of failure $\delta$ for all the considered routines. Corollary 2.16.1, necessary for the inner product estimation, fails with a probability smaller than $2\delta$ and has a running time overhead of $O(\log(\frac{m}{\delta}))$. Corollary 2.18.1, necessary to identify the maximum absolute value, fails with a probability smaller than $\delta$ and has a running time overhead of $O(\log(\frac{1}{\delta}))$. The computation of the approximation of the vector $|\phi_t\rangle$ following Lemma 3.6 fails with probability $O(\log(\frac{1}{\delta}))$ considering that we want to estimate the norm of the vector.

In this case, the probability that the $t^{th}$ loop iteration would be successful is equal to

$$1 - p_f \geq (1-\delta)(1-\delta)(1-2\delta)(1-2\delta)(1-\delta) \geq 1 - 7\delta + 19\delta^2 - 25\delta^3 + 16\delta^4 - 4\delta^5. \tag{4.64}$$

where $p_f$ is the total probability of failure. That, with some steps of approximation, if we suppose $\delta^2$ small enough, corresponds to

$$1 - p_f \geq 1 - 7\delta, \tag{4.65}$$

with a run-time overhead of $O\left(\log(\frac{m}{\delta'})^2 \log(\frac{1}{\delta'})^3\right)$ if we do not consider any optimization but to boost all the routines one at a time. Therefore, the probability of obtaining the incorrect result during the $t^{th}$ iteration of the loop is constrained by $7\delta$.

If we want to bound the overall success probability, we must consider $k$ different iterations. The total probability of failure corresponds to

$$p(\cup_i^k p_f) \leq \sum_i^k p_f \leq 7k\delta \tag{4.66}$$

exploiting the use of the union bound.

Finally, we can define the probability of success as $1 - \delta'$ adjusting the running time overhead to $O(\text{polylog}(\frac{7km}{\delta'})) \sim \widetilde{O}(1)$.

## 4.3. Q-ECR

This section presents a quantum version of the Eigenvectors Classification/Recognition algorithm. We have explained the classical version of the algorithm in Section 1.2.4.

This version of the algorithm exploits the use of quantum routines to speed up the computation of the matrix-vector product and the distance computation.

In the following sections, we provide a detailed explanation of how the algorithm works, the running time, and an analysis of the errors introduced in the algorithm due to the use of quantum procedures.

### 4.3.1. Pre-processing

As in the classical algorithm, we must subtract the mean vector of the dataset, called $\mu \in \mathbb{R}^n$, from the sample $s \in \mathbb{R}^n$ that the algorithm wants to analyze before the start of the execution. For this reason, we must guarantee access to the centered sample defined as

$$\overline{s} = s - \mu. \tag{4.67}$$

We want, for this reason, a unitary $U_s$ that generates a states that represents $\overline{s}$ such that

$$U_s : |0^{\otimes \log_2(n)}\rangle \rightarrow |\overline{s}\rangle\,,\ |\overline{s}\rangle = \frac{1}{\|\overline{s}\|} \sum_i^n \overline{s}_i\, |i\rangle\,. \tag{4.68}$$

A solution when the inputs are provided as quantum states is to perform a subtraction between the two states. It is possible to do it following the methods explained in Section 3 of [66] and Section 2.3 of [67] with additional complexity and error.

To keep this analysis simple, we suppose to have the unitary $U_s$ as defined above.

### 4.3.2. Algorithm

The procedure is summarized in Algorithm 4.3.

**Algorithm 4.3** Quantum Eigenvectors Classification/Recognition

**Input**: Unitary $U_{A^T}$ that is a block-encoding of the transposed matrix of eigenvectors $A^T \in \mathbb{R}^{m \times n}$. Unitary $U_v$ that generates the superposition of $p$ weights vectors $v_j \in \mathbb{R}^m$ for $j \in [p]$ of set $V = \{v_1, ..., v_p\}$. Unitary $U_s$ that creates $|\overline{s}\rangle$ where $\overline{s}$ is the centered sample to classify.

$\delta_1, \delta_2 \in \mathbb{R}^+$ threshold values used to distinguish the outputs. $\epsilon_\delta \in \mathbb{R}^+$ precision parameter on the squared distance.

**Output**: if $\widetilde{d^2} \leq \delta_1^2$: *recognized in the same class of* $v_{j^*}$,
if $\delta_1^2 < \widetilde{d^2} < \delta_2^2$: *recognized as similar element,*
if $\widetilde{d^2} \geq \delta_2^2$: *not a similar element.*

1: Apply the block-encoding of $A^T$ to $|\overline{s}\rangle$ obtaining:
$|\widetilde{w}\rangle$ such that $|w\rangle = |A^T\overline{s}\rangle$ and $\| \, |\widetilde{w}\rangle - |w\rangle \, \| \leq \frac{1}{4}\epsilon_\delta$

2: Use distance estimation to create:
$|\varphi\rangle = \frac{1}{\sqrt{p}} \sum_{j=1}^{p} |j\rangle \, |\overline{d}_j^2\rangle$ where $d_j^2 = \|w - v_j\|^2$ and $|\overline{d}_j^2 - d_j^2| \leq \epsilon_\delta$

3: Apply finding the minimum absolute value on the unitary that implements $|\varphi\rangle$, to extract the closest element $j^*$ and the squared distance $\widetilde{d^2}$

4: Output: if $\widetilde{d^2} \leq \delta_1^2$: *recognized in the same class of* $v_{j^*}$,
if $\delta_1^2 < \widetilde{d^2} < \delta_2^2$: *recognized as similar element,*
if $\widetilde{d^2} \geq \delta_2^2$: *not a similar element.*

In this initial explanation, we put more emphasis on how the algorithm works than on the value of the errors. For the moment, it is enough to know for the reader that we use the error $\epsilon_\delta \in \mathbb{R}^+$ to define the other errors involved. There will be a detailed explanation of the error requirements for the different routines in Section 4.3.3.

The first step of the algorithm consists in apply $U_{A^T}$, a block-encoding of the transposed eigenvectors matrix $A^T$, to the vector-state that describes the sample to analyze $|\overline{s}\rangle$ obtained using the unitary $U_s$. Following Lemma 2.11 we define the unitary $U_w$ that generates a state $|\widetilde{w}\rangle$ such that $\| \, |\widetilde{w}\rangle - |w\rangle \, \| \leq \frac{1}{4}\epsilon_\delta$ with $|w\rangle = |A^T\overline{s}\rangle$.

We consider a set $V = \{v_1, ..., v_p\}$ with $p$ weights vectors $v_j \in \mathbb{R}^m$ of other samples as elements. Each of the samples described using these weights vectors belongs to a certain class. To understand if our test sample $s$ is in the same class as one of the samples described by the weights vectors in $V$, we check the distance between their weights vectors.

At this point, we use Corollary 2.17.1 with as input the unitary $U_v$ that creates a quantum state that is the superposition of the elements of $V$ and the unitary $U_w$ that computes $|\widetilde{w}\rangle$. In this way, we define the unitary that computes the squared distance between the weights vector obtained in the first step and the previously computed ones $|v_j\rangle$ for all $j \in [p]$. We obtain the unitary that generates

$$|\varphi\rangle = \frac{1}{\sqrt{p}} \sum_{j=1}^{p} |j\rangle \, |\bar{d}_j^{\,2}\rangle \tag{4.69}$$

where $d_j^2 = \|w - v_j\|^2$ and $|\bar{d}_j^{\,2} - d_j^2| \leq \epsilon_\delta$.

We obtain the minimum value of the squared distance $\widetilde{d^2}$ and the index of the corresponding element $j^*$ using the finding the minimum procedure of Lemma 2.18 on the unitary that computes the state $|\varphi\rangle$.

We compare the value $\widetilde{d^2}$ with the squared value of two different threshold values $\delta_1 \in \mathbb{R}^+$ and $\delta_2 \in \mathbb{R}^+$ with the following outcomes:

1. $\widetilde{d^2} \leq \delta_1^2$ means that the sample $s$ is recognized as an element of the same class of the sample $v_{j^*}$,
2. $\delta_1^2 < \widetilde{d^2} < \delta_2^2$ means that the sample $s$ is not recognized in a class already defined by the elements of $V$, but it is similar to them,
3. $\widetilde{d^2} \geq \delta_2^2$ mean that the sample $s$ is not similar to the stored samples.

The algorithm performs the comparisons with classical computation. We want to point out that we use the squared distance for the comparisons, while in the classical algorithm, we use the distance. We made this choice to keep the error propagation clear.

As a secondary application, it is also possible to perform tomography of the state $|\widetilde{w}\rangle$ following Lemma 2.19. We obtain an approximation of the vector in the classical form $\overline{w}$ with $\|\overline{w} - w\| \leq \frac{1}{4}\epsilon_\delta + \epsilon_t$ with $\epsilon_t \in \mathbb{R}^+$ chosen by the user. This step can be done to populate a certain dataset. In that case, the procedure terminates without performing the classification step.

### 4.3.3. Error Propagation

First of all, we must highlight that due to the presence of error in the squared distance estimation, it is possible to have two different kinds of problems. The first problem is the choice of a minimum squared distance that does not correspond to the one selected classically due to the error presence. This problem can cause the misclassification of an

element. This happens because the algorithm chooses as the closest sample an element different from the one selected by the classical procedure. If these samples are classified into different classes, our test sample will be classified wrongly.

Another problem is that, even if we have selected the closer sample correctly, we can have some error in the estimation of the squared distance that can cause a wrong output in relation to the comparison with the thresholds $\delta_1$ and $\delta_2$.

Both these problems are related to the error value of the squared distance. In general, we reduce the frequency of both problems by minimizing this error. In general, we define this error with the variable $\epsilon_\delta \in \mathbb{R}^+$, and we will show some experimental results on this value in Section 5.4.1 and 5.4.2. For a low enough error $\epsilon_\delta$, depending on the problem, we can guarantee very similar performances to the classical method.

For this reason, we can consider the condition to satisfy as

$$\left|\overline{d}_j^2 - d_j^2\right| \leq \epsilon_\delta. \tag{4.70}$$

If we consider that $d_j^2 = 2 - 2\langle w|v_j\rangle$, since the vectors are normalized, we must bound the precision just on the inner product computation. We want to obtain a bound for the value $\epsilon_w \in \mathbb{R}^+$ that corresponds to the error obtained due to the application of the block-encoding in Step 1. In addition, we want to define a bound for the value of the precision on the estimation of the inner product used to compute the squared distance in Step 2, which we call $\epsilon_{in} \in \mathbb{R}^+$.

We can obtain the required bounds for these values by applying Claim 3.4 with as input $\|w\| = 1$, $\|v_j\| = 1$, the state $|\widetilde{w}\rangle$ such that $\| \, |\widetilde{w}\rangle - |w\rangle \, \| \leq \epsilon_w$ and an error bound of $\frac{1}{2}\epsilon_\delta$. The bound corresponds to $\frac{1}{2}\epsilon_\delta$ because the inner product is multiplied by two while we use it to estimate the squared distance. We obtain $\epsilon_w + \epsilon_{in} \leq \frac{1}{2}\epsilon_\delta$. We can choose the values $\epsilon_w \leq \frac{1}{4}\epsilon_\delta$, $\epsilon_{in} \leq \frac{1}{4}\epsilon_\delta$ to satisfy the condition.

Considering the first step of the algorithm, if we want to guarantee that the approximation of the $w$ vector has an error of $\epsilon_w \leq \frac{1}{4}\epsilon_\delta$, we must have an error in the block-encoding of the matrix $A^T$ that satisfies the conditions of Lemma 2.11. We define this error with a value $\zeta \in \mathbb{R}^+$ such that

$$\zeta \leq \frac{\epsilon_w \|A\|}{4\,\kappa(A)} \leq \frac{\epsilon_\delta \|A\|}{16\,\kappa(A)}. \tag{4.71}$$

Finally, we want to highlight that if the objective is to populate the dataset with other weights vectors, the error on the approximated value of the classical representation of the vector $w$ is $\epsilon_t + \frac{1}{4}\epsilon_\delta$. This condition is trivial to check because Lemma 2.11 guarantees

that $\| |\widetilde{w}\rangle - |w\rangle \| \leq \frac{1}{4}\epsilon_\delta$ while the tomography step, following Lemma 2.19 guarantees that $\|\overline{w} - \widetilde{w}\| \leq \epsilon_t$. Considering that we are working with the normalized versions of the weights vectors, the two errors sum, and we obtain a final error on the classical representation of the vector such that $\|\overline{w} - w\| \leq \epsilon_t + \frac{1}{4}\epsilon_\delta$ in the following way:

$$\|\overline{w} - w\| \leq \|\overline{w} - \widetilde{w} + \widetilde{w} - w\| \leq \|\overline{w} - \widetilde{w}\| + \|\widetilde{w} - w\| \leq \epsilon_t + \frac{1}{4}\epsilon_\delta \tag{4.72}$$

### 4.3.4. Running Time

**Theorem 4.8** (Quantum Eigenvectors Classification/Recognition). *Let $A \in \mathbb{R}^{m\times n}$ be a matrix and let $U_{A^T}$ be a $(\alpha, a, \zeta)$-block-encoding of $A^T$ such that*

$$\zeta \leq \frac{\epsilon_\delta \|A\|}{16\,\kappa(A)}. \tag{4.73}$$

*that can be generated in time $O(T_A)$ where $\epsilon_\delta \in \mathbb{R}^+$ is a precision parameter on the value of the squared distances that will be estimated. Let $U_v$ be a unitary that generates a superposition of the elements of the set $V = \{v_1, ..., v_p\}$ that contains the already computed weights $v_j \in \mathbb{R}^m$ of $p$ elements in time $O(T_v)$. Let $U_s$ be a unitary that generates the state $|\overline{s}\rangle$ in time $O(T_s)$ where $\overline{s} = s - \mu$, $s \in \mathbb{R}^n$ is the test sample, and $\mu \in \mathbb{R}^n$ is the mean vector of the training dataset. Let $\delta_1, \delta_2 \in \mathbb{R}^+$, be two classification parameters such that $\delta_1 < \delta_2$. Then there exists a procedure that obtains similar results to eigenvectors classification/recognition of the sample $s$ with relation to the classes of the elements in $V$ in running time*

$$\widetilde{O}\left(\frac{\sqrt{p}}{\epsilon_\delta}\left(\frac{\alpha\,\kappa(A)(T_A + T_s)}{\|A\|} + T_v\right)\right). \tag{4.74}$$

*It is also possible to save an approximation $\overline{w} \in \mathbb{R}^m$ of the weights vector $w \in \mathbb{R}^m$ of the sample $s$ in running time $\widetilde{O}\left(\frac{\alpha\,\kappa(A)}{\|A\|}\frac{m\log(m)}{\epsilon_t^2}(T_s + T_A)\right)$ such that $\|\overline{w} - w\| \leq \frac{1}{4}\epsilon_\delta + \epsilon_t$ where $\epsilon_t \in \mathbb{R}^+$ is a chosen precision parameter.*

*Proof.* The proof consists in proving the running time of the algorithm.

The first step is defining the time necessary to obtain the approximated weights vector $|\widetilde{w}\rangle$. To obtain this state it is enough to apply Lemma 2.11 considering that we obtain the unitary $U_{A^T}$ in $\widetilde{O}(T_A)$ and the unitary $U_s$ generates $|\overline{s}\rangle$ in $\widetilde{O}(T_s)$. In this way, we have a complexity of

$$\widetilde{O}(T_w) = \widetilde{O}\left(\frac{\alpha\,\kappa(A)}{\|A\|}(T_s + T_A)\right). \tag{4.75}$$

Once we have performed this step, we can decide to perform tomography of the weights vector to populate our dataset and, in that case, the complexity of performing tomography, following Lemma 2.19 is

$$\widetilde{O}\left(T_w \frac{m\log(m)}{\epsilon_t^2}\right) \tag{4.76}$$

obtaining a classic representation $\overline{w}$ with $\|\overline{w} - w\| \leq \epsilon_t + \frac{1}{4}\epsilon_\delta$.

So the final complexity to generate and save the weights vector is

$$\widetilde{O}\left(\frac{\alpha\,\kappa(A)}{\|A\|}\frac{m\log(m)}{\epsilon_t^2}(T_s + T_A)\right). \tag{4.77}$$

Instead, if we want to classify the sample in the same class of an element of the dataset $V$, we perform the distance estimation procedure of Corollary 2.17.1 with as input the unitaries $U_v$ and $U_w$, that has complexity $\widetilde{O}\left(\frac{1}{\epsilon_{in}}(T_w + T_v)\right) \sim \widetilde{O}\left(\frac{1}{\epsilon_\delta}(T_w + T_v)\right)$.

Then, we apply the Finding the minimum procedure of Lemma 2.18 of complexity

$$\widetilde{O}\left(\frac{T_w + T_v}{\epsilon_\delta}\sqrt{p}\right), \tag{4.78}$$

and we obtain both the value $\widetilde{d^2}$ and the index $j^*$. The algorithm performs the last comparisons classically in $\widetilde{O}(1)$. The overall complexity corresponds to

$$\widetilde{O}\left(\frac{\sqrt{p}}{\epsilon_\delta}\left(\frac{\alpha\,\kappa(A)(T_A + T_s)}{\|A\|} + T_v\right)\right). \tag{4.79}$$

□

If we consider having the data stored in a QRAM data structure, as described in Section 2.2.1, or to have efficient quantum access to the unitaries, we can delete the dependencies on $T_s$, $T_A$ and $T_v$. In this case, the complexities became

$$\widetilde{O}\left(\frac{\sqrt{p}}{\epsilon_\delta}\frac{\mu(A)\kappa(A)}{\|A\|}\right) \tag{4.80}$$

for the classification task and

$$\widetilde{O}\left(\frac{\mu(A)\kappa(A)}{\|A\|}\frac{m\log(m)}{\epsilon_t^2}\right). \tag{4.81}$$

for the population of the dataset.

### 4.3.5. Success probability

Since most of the routines we use do not always produce the desired results, our method has a certain probability of failing. The computation of the weights in step 1, the computation of the distance in step 2, and the search in step 3 introduce a certain probability of failure in the algorithm.

We consider the same probability of failure $\delta$ for each routine to simplify the analysis. Corollary 2.11.1, used in step 1, fails with probability $\delta$ with a running time overhead of $O(\log(\frac{1}{\delta}))$. The distance estimation in step 2 follows the routine of Corollary 2.17.1 and fails with probability $2\delta$ with an overhead of $O(\log(\frac{p}{\delta}))$. Finally, Corollary 2.18.1 used in step 3 to find the minimum fails with a probability of $\delta$ with a run-time overhead of $O(\log(\frac{1}{\delta}))$.

Instead, if we consider storing the classical representation of the weights vector, we must consider the probability of failure of step 1 and of the additional tomography step. Lemma 2.19 fails with a probability of $\delta$ with an overhead of $O(\log(\frac{1}{\delta}))$.

At this point, we can compute the overall probability of success defined as $1 - p_f$, where $p_f$ is the probability of failure, and the running time overhead of the two different cases.

#### Classification of test sample

The probability of success of the classification of a sample is

$$1 - p_f \geq (1 - \delta)(1 - 2\delta)(1 - \delta) = 1 - 4\delta + 5\delta^2 - 2\delta^3. \tag{4.82}$$

That, if we consider the terms with a degree greater than one low enough, corresponds to

$$1 - p_f \geq 1 - 4\delta \tag{4.83}$$

with a run-time overhead of $O\left(\log\left(\frac{p}{\delta}\right)\log\left(\frac{1}{\delta}\right)^2\right)$. For this reason, the probability of failure is bounded by $4\delta$.

#### Creation and storing of weights vector

The probability of success of the creation and storing of a new weights vector is

$$1 - p_f \geq (1 - \delta)(1 - \delta) = 1 - 2\delta + \delta^2 \geq 1 - 2\delta. \tag{4.84}$$

with a run-time overhead of $O\left(\log\left(\frac{1}{\delta}\right)^2\right)$. The probability of getting the wrong solution is bounded by $2\delta$.

# 5 | Experiments

In this chapter, we provide some experimental results on the simulation of the quantum algorithms that we have presented. We have performed the experiments using Python3. It is possible to find all the code used for the experiments in two GitHub repositories [79, 80] to allow the reader to repeat them.

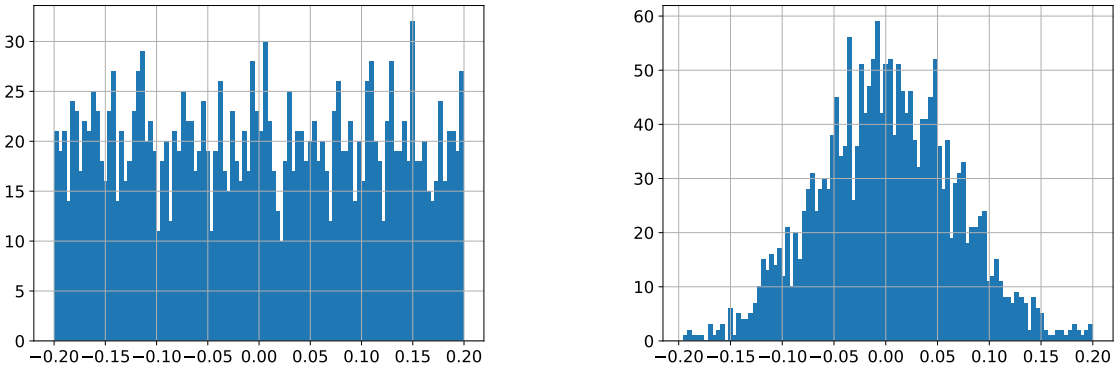
## 5.1. Introduction

We want to highlight that the aim of our experiments is to check if the introduction of error in the algorithms used to emulate the error generated by quantum routines would affect the performances. Our objective is to find suitable values for the error in relation to the tasks analyzed. In addition to that, we also estimate the parameters $\mu(A_t)$ and $\kappa(A_t)$, as defined in Section 4.2.4, that enable a more complete view of the running time for the Q-OMP algorithm.

### 5.1.1. Simulation of errors

We have considered two different cases for the type of error that we have introduced in the routines. The first case consists of a value extracted from a Gaussian distribution with zero mean and standard deviation equal to $\frac{1}{3}\epsilon$ with $\epsilon \in \mathbb{R}^+$, truncated on the interval $[-\epsilon, \epsilon]$. The second case consists of a value sampled from a uniform distribution, again in the range $[-\epsilon, \epsilon]$. Whenever an approximated vector presents an error on the $l_2$-*norm*, we spread that error accordingly on each entry.

In Figure 5.1, we report the frequency of the error for 2000 scalars extracted from the two distributions with $\epsilon = 0.2$. Figure 5.1a shows the frequency for the uniform distribution, while Figure 5.1b shows the frequency for the truncated normal.




(a) Uniform distribution

(b) Truncated Normal distribution

Figure 5.1: Frequency of measures of the various errors with $\epsilon = 0.2$ for the two error distributions for 2000 scalars

### 5.1.2. Metrics

First of all, we define the following variables

- TP (True Positive): defined as the samples that we classify as anomalies and are anomalies,
- TN (True Negative): defined as the samples that we classify as benign samples and are benign samples,
- FP (False Positive): defined as the samples that we classify as anomalies and instead are benign samples,
- FN (False Negative): defined as the samples that we classify as benign samples and instead are anomalies.

From these four variables, we define the metrics that we use in this section to evaluate some of our experiments

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN},$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \qquad F1\,score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}.$$

## 5.2. Q-MP

### 5.2.1. Artificial Dataset Experiments

We have decided to run a version of the algorithm without error, which we identify as the classical version, and a version in which the quantum error is simulated, which we identify as the quantum version.

We have analyzed both the **Single Error (SE)** and the **Double Error (DE)** versions of the quantum algorithm explained in Section 4.1.2. We recall that in the SE version, the error can guide us to the selection of the wrong atom but after that, the inner product used to update the solution is re-computed classically. On the other hand, in the DE version, the update of the solution is performed with an inner product that contains some error.

#### Dataset

For these experiments, we have generated our dataset in the following way. First of all, we have used the function *make_sparse_coded_signal* of the library *sklearn.datasets* [63]. This procedure takes as input the following values:

- number of components of the signals,
- number of atoms of the dictionary,
- sparsity of the solution,
- number of signals required.

The procedure generates the signals, the dictionary, and some $l_0$-*sparse* solutions.

We have defined 20 triplets (*number of components, number of atoms, sparsity of the solution*) with values variable from $(50, 100, 10)$ to $(1000, 2000, 200)$ with a step of $(50, 100, 10)$. For each triplet, we have generated 100 samples and a dictionary, for a total of 2000 samples and 20 dictionaries.

In each of the components of these samples, we have added a Gaussian noise $\delta \in [-0.05, 0.05]$ to increase the difficulty in the recovery for both the classical and the quantum algorithms.

We have considered an error reconstruction tolerance $\epsilon \in \mathbb{R}^+$ variable with the dimension of the signals. We have defined its value as $\epsilon = 0.05 \cdot \sqrt{number\ of\ components}$ that corresponds to an absolute error of value 0.05 on each component. This would be enough to enable finding the noiseless signal.

## Error Introduction

For this experiment, we have followed the definition of error of Theorem 4.1, which we now recall. We have considered a bound on the maximum value of the absolute error in the final inner products $\overline{z_j}$, with the value $\epsilon_z \in \mathbb{R}^+$. We have partitioned this error between the estimated inner products $\overline{z_{1j}}$, $\overline{z_{2j}}$, and the approximation of the vector $\phi_t$. In this way, the overall error of the final inner products corresponds to $\epsilon_z$, and we have also verified that the introduction of error in the vector $\phi_t$ follows our expectations. We have divided the error in the following way.

We have considered a quantum procedure that computes the inner products of two vectors with a specific accuracy. Referring to the variables used in Section 4.1.2 we supposed to have $\|s\|\epsilon_{in_1} \leq \epsilon_1/2 \leq \epsilon_z/2$ and $\widetilde{\|\phi_t\|}\epsilon_{in_2} \leq \epsilon_z/4$. In addition to that, we supposed to have a bound on the error of the $l_2$-norm of the approximation of $\phi_t$ as $\|\widetilde{\|\phi_t\|}\,|\widetilde{\phi_t}\rangle - \phi_t\| \leq \|\phi_t\|\epsilon_\phi + \|\phi_t\|\eta \leq \epsilon_z/4$. If we combine these bounds to obtain the final inner products' error, we obtain

$$\|s\|\epsilon_{in_1} + \|\phi_t\|\epsilon_{in_2} + \|\phi_t\|\epsilon_\phi + \|\phi_t\|\eta \leq \epsilon_z. \tag{5.1}$$

For this reason, we have introduced an error in the range $[-\epsilon_z/2, \epsilon_z/2]$ in the first inner product, in the range $[-\epsilon_z/4, \epsilon_z/4]$ in the second and $[-\epsilon_z/4, \epsilon_z/4]$ in the value of $\phi_t$.

We have defined and tested 10 possible values for $\epsilon_z$ in the range $[0.005, 0.05]$ with a step of 0.005. The error introduced in each of the variables corresponds to a Gaussian or a uniform error obtained using the methodology explained in Section 5.1.1.

For signals with a greater norm, this absolute error will be less relevant but, consequently, the running time grows as it is possible to see from Theorem 4.1.

## Goal

The main goal of the experiment is to understand if the two versions are robust to the introduction of noise and compare the classical and the quantum versions.

We have analyzed different elements; if the number of atoms selected for the solution would change increasing the error, if the stop condition is satisfied, and if it is present some sort of relationship between the goodness of the solution and the dimension of the dataset.

## Parameters

We define the following parameters that we use in this analysis:

- *error*: error that we add in the inner products, corresponds to the value $\epsilon_z$
- *residual*: the value of the norm of the final residual vector
- *similarity*: the ratio between the sparsity of the quantum solution and the sparsity of the classical solution. It is equal to 1 if the two solutions have the same sparsity. It assumes values lower than 1 if the quantum solution is more sparse than the classical one and greater than 1 in the opposite case.

## Results

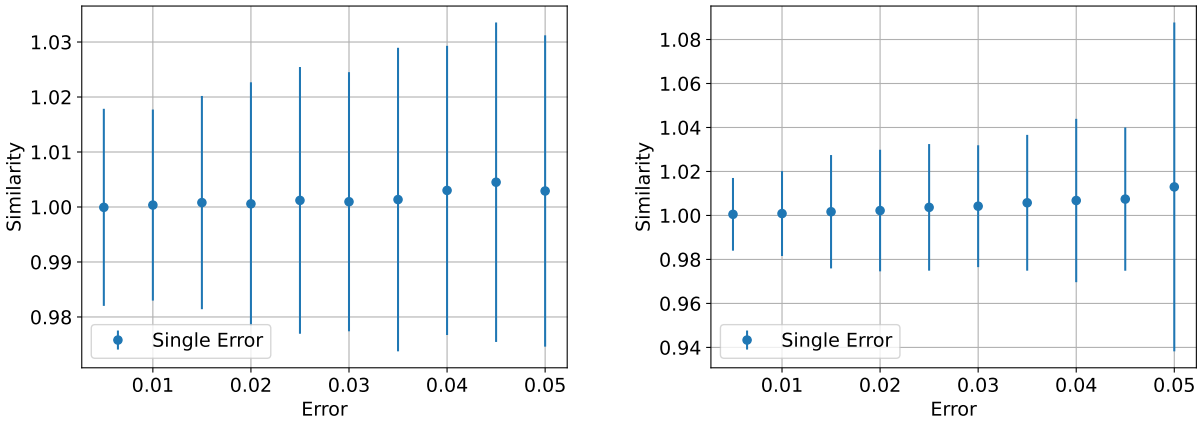
We executed both the classical version of the Matching Pursuit algorithm and the quantum versions for each of the 10 different values of $\epsilon_z$ on the 2000 signals that we have created. The signals were divided into 20 groups, and for each group, we used the dictionary generated for that specific batch of samples.

**Relationship between similarity and error**

We analyzed how the similarity of the solution would change in the SE and in the DE version if we increase the error. We first provide an analysis for each version of the algorithm. Then we compare the two versions together.

We run the classical and the quantum procedure for each one of the 2000 samples, and we have computed the ratio between the sparsity of the quantum solution and the sparsity of the classical one obtaining the *similarities*. We repeated this procedure for each of the 10 error levels considered. Finally, we computed the mean value and the standard deviation of the similarities obtained for each error.

Figure 5.2 shows how the similarity changes when the error increase for the SE version. Figure 5.2a shows the case of Gaussian error, and Figure 5.2b the case of uniform error.

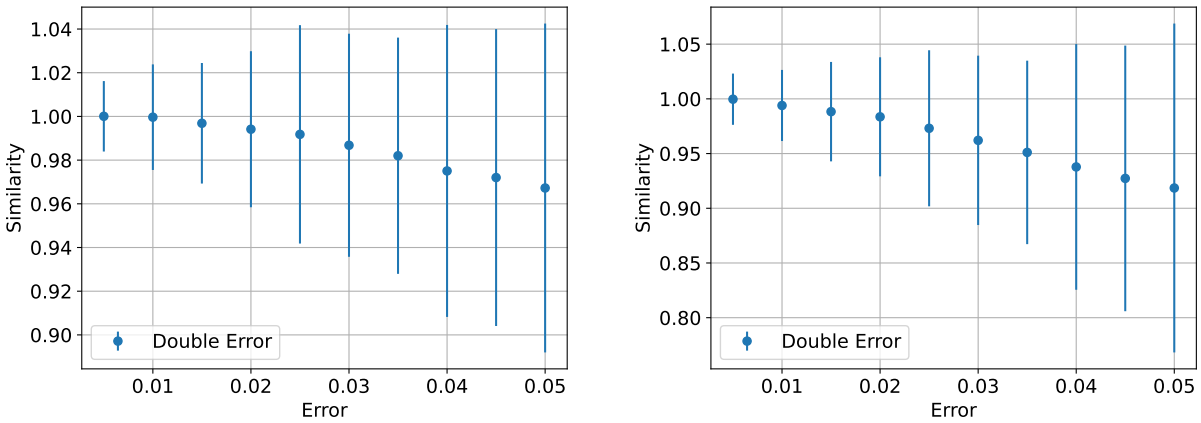



(a) Gaussian Error (b) Uniform Error

Figure 5.2: Similarity with relation to error, SE version

In both cases, it is possible to see how the algorithm is robust to the tested noise. We can see how the similarity can assume values greater or lower than 1. This happens because the selection of different atoms from the classical ones would sometimes help the algorithm to converge faster and sometimes would slow down the resolution. It is also important to notice how increasing the error increases the standard deviation. With errors in the range of $10^{-2}$, the solution is quite close to the classical one.

Figure 5.3 shows the analysis of the similarity of the DE version. Figure 5.3a and Figure 5.3b show the results with Gaussian and uniform error, respectively.




(a) Gaussian Error (b) Uniform Error

Figure 5.3: Similarity with relation to error, DE version

These results remark how the introduction of a higher error would have relevant consequences on the final similarity. With the DE version of the algorithm, we obtain quantum

solutions that are more sparse than the classical ones. For this reason, the similarity drastically decreases.

In the DE version of the algorithm, we have decided to use the alternative exit condition (see Section 1.2.2) to avoid the computation of the residual vector. As is possible to see in Algorithm 4.1, at each iteration, the algorithm updates the value $\xi$ that contains the energy of the signal described until that moment. When the value of the energy described is close to the energy of the initial signal, the algorithm stops. We perform the update of $\xi$ by adding the squared value of the highest inner product that we select as the next component of the solution. This inner product has an error generated by the routine that estimates it. At this point, we must highlight that the algorithm selects the greatest inner product without knowing the amount of error in it. For this reason, it is easy to add a lot of error in the final value of the energy $\xi$ and, consequently, the algorithm stops too soon and obtains, for this reason, a better sparsity. This phenomenon will be evident also in the analysis concerning the residual.

In general, we can state that it is simple to produce solutions with a better sparsity compared to the classical ones by introducing a high error, whereas we produce similar solutions when a small error is taken into account.

Finally, in Figure 5.4, it is possible to see the comparison between the SE and the DE versions with Gaussian noise.

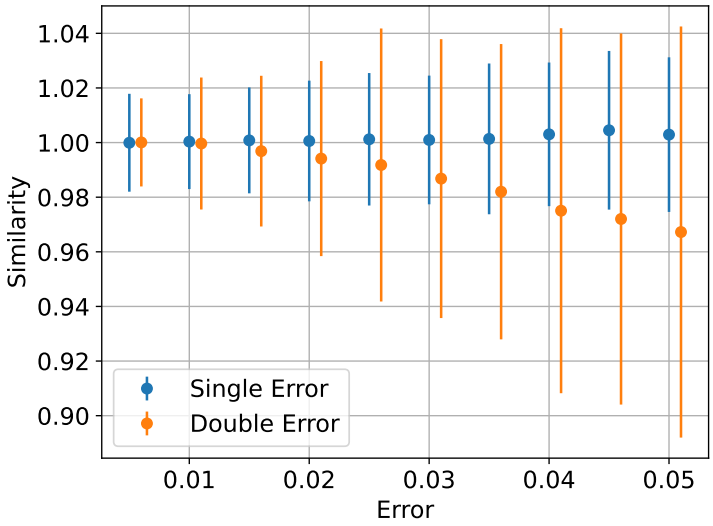



Figure 5.4: Comparison between SE and DE similarity with Gaussian Error

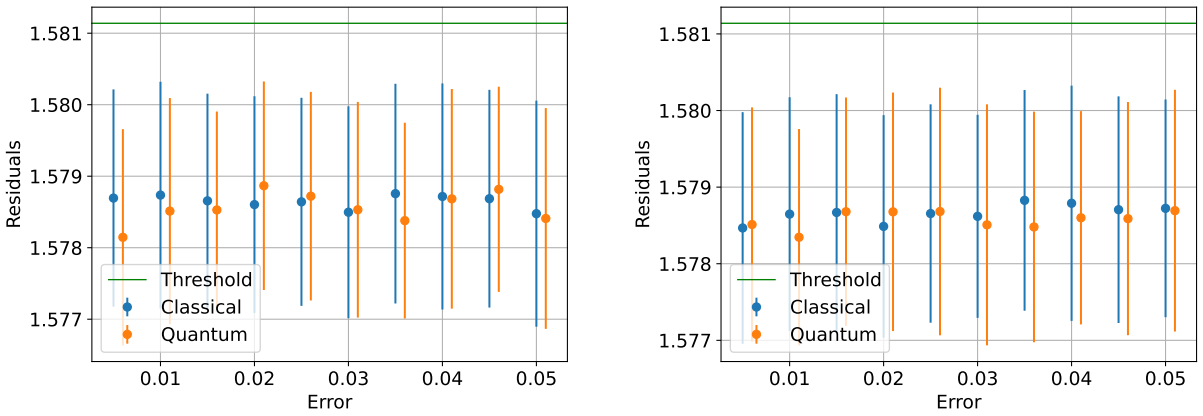
We can notice how the mean similarity of the DE version moves away from the value 1 way faster than the SE one. This was expected by the theoretical results presented in

Section 4.1.2, in which it is possible to see how the error in the DE version fast increases due to the propagation through different iterations and how the exit condition is verified. As explained above, the presence of errors forces the algorithm to stop too soon. Instead, in the SE version, thanks to the classical re-computation of the inner products, the error introduced is limited to one single iteration. It is also interesting to notice how the standard deviation increases a lot in the DE case in comparison with the SE one. This happens because, in the DE version, different errors at each iteration are combined, increasing the variability of the error in the final result.

**Relationship between residual values and error**

We analyzed the value of the residuals left by the different versions of the algorithm. We performed this analysis on a group of 100 test samples with dimensions (1000, 2000, 200). We consider all the 10 possible values for $\epsilon_z$. The error reconstruction tolerance was $\epsilon = 0.05\sqrt{1000} \simeq 1.581$, and the mean norm of the signals analyzed was $\simeq 14.18$. We have performed the analysis for both types of error, also in this case.

Figure 5.5 shows the value of the residuals for the SE version. Figure 5.5a shows the case of Gaussian error, and Figure 5.5b shows the one of uniform error.



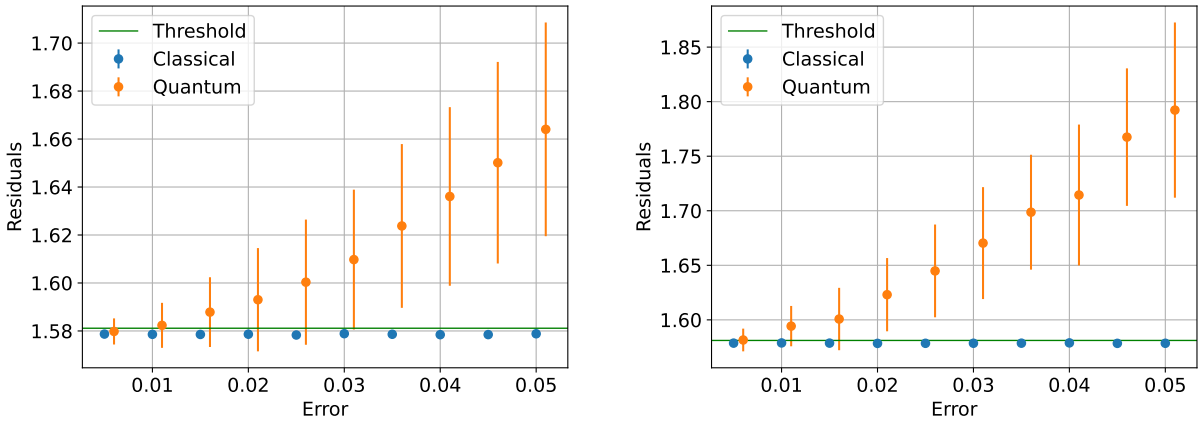

(a) Gaussian Error

(b) Uniform Error

Figure 5.5: Residuals with relation to error, SE version

As it is possible to see in the graphs, the stop condition is perfectly satisfied. The values of the quantum and classical solutions differ a little just because of the possible selection of different atoms during the computation. This result shows how, if the inner products are re-computed classically, the stop condition based on the energy works properly.

Figure 5.6 shows the value of the residuals for the DE version. Figure 5.6a and Figure 5.6b show the results with Gaussian and uniform error, respectively.




(a) Gaussian Error

(b) Uniform Error

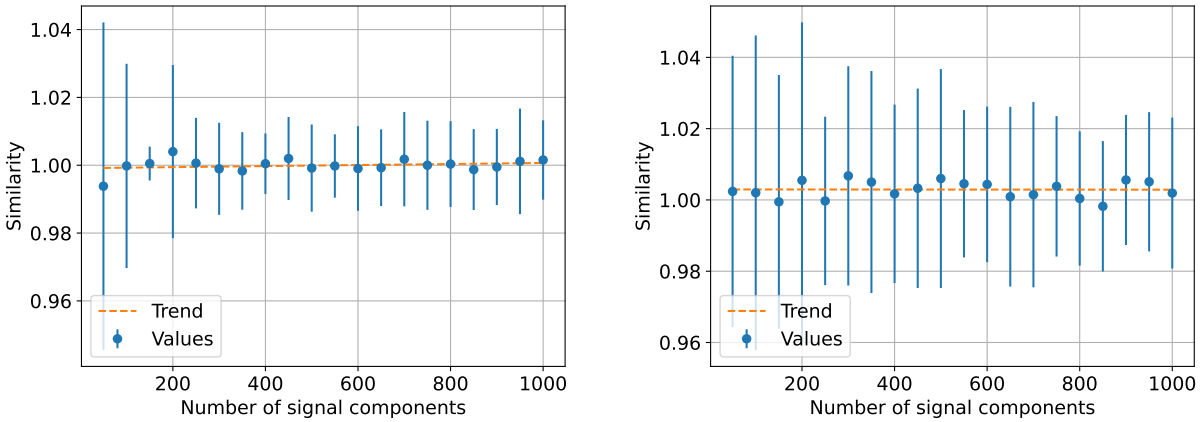
Figure 5.6: Similarity with relation to error, DE version

The stopping of this version is not precise due to the use of our exit condition. This happens because the algorithm terminates too soon, since the estimation of the energy contains some error. For this reason, the threshold value is not reached and it is possible to obtain greater values for the final residual. The algorithm guarantees better performances for small-enough errors.

### Relationship between similarity and dimensions

We analyzed if there is some dependency between the similarity and the dimension of the signal. We have decided to show this analysis with just Gaussian error, because the result does not differ in a relevant way from the case with uniform error. We have analyzed two different values of $\epsilon_z$, 0.005 and 0.05, for both versions of the algorithm.

Figure 5.7 shows the results for the SE version. Figure 5.7a shows the case with 0.005 Gaussian error, and Figure 5.7b shows the case with 0.05 Gaussian error.



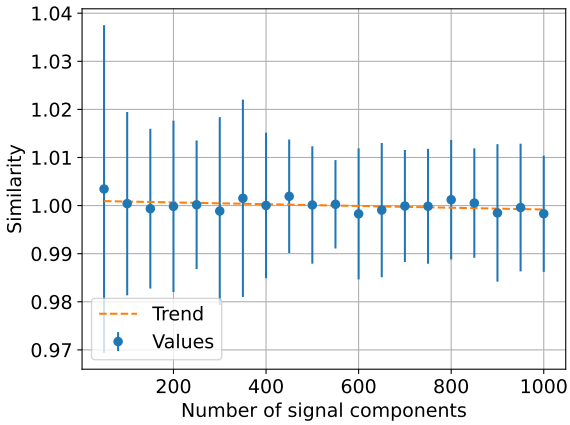

(a) Relationship between dimension and similarity with error 0.005

(b) Relationship between dimension and similarity with error 0.05

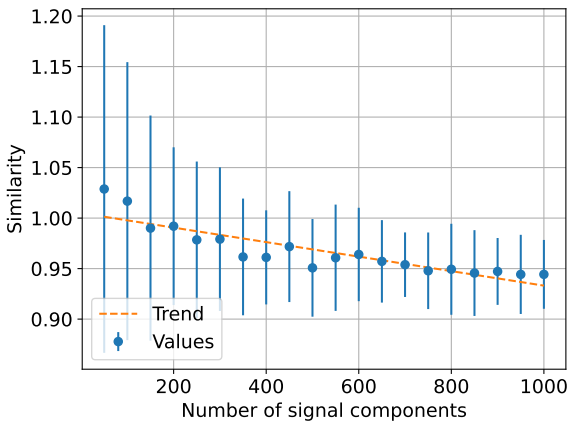
Figure 5.7: Relationship between dimension and similarity for SE version

The trend line represents the linear function that best fits the relationship. We can notice a lower standard deviation for signals with more components for both the errors analyzed. This can be explained since, with bigger signals, the magnitude of the sparsity of the classical and the quantum solutions is higher. For this reason, it is easier to have a ratio close to 1 if the two solutions differ for a few extra components. If we consider, instead, the value of the mean similarity it does not change increasing the dimensions as shown by the trend line for both errors. With a higher error, the mean value and its standard deviations are slightly higher. This result clearly shows how the classical recomputing of the inner products guarantees a correct exit condition for the algorithm and how increasing dimensions, the similarity remains more or less stable even introducing a relevant amount of error in the procedure.

Figure 5.8 shows the results for the DE version. Figure 5.8a and Figure 5.8b show the results with 0.005 and 0.05 Gaussian errors, respectively.




(a) Relationship between dimension and similarity with error 0.005




(b) Relationship between dimension and similarity with error 0.05

Figure 5.8: Relationship between dimension and similarity for DE version

These results highlight how, with a higher number of components, we obtain a lower variability on the final value. In Section 4.1.2, we have analyzed how the error on the final solution for the DE version depends both on the dimension of the sample and on the value of the error introduced. It is possible to see this trend clearly in the analysis with the higher error. In this case, the decrease in the value of the similarity is evident for higher dimensions. If we consider the case with the low 0.005 error, this trend is barely visible, but it is present.

## 5.3. Q-OMP

For the experiments on the Quantum Orthogonal Matching Pursuit algorithm (defined in Section 4.2), we have decided to use both an artificially generated dataset and a real one.

### 5.3.1. Artificial Dataset Experiments

In this section, we denote the version of the algorithm without error as the classical version and the version in which the quantum error is simulated as the quantum version, as in the previous one. We use the same dataset and the same parameters defined for the experiment of Section 5.2.1.

#### Error Introduction

We have introduced errors in the inner product estimation procedures and in the vector $\phi_t$ in the same way as in the experiment of Section 5.2.1 for a total maximum value of $\epsilon_z \in \mathbb{R}^+$.

In addition to these errors, we have considered another error in the distance estimation used in the exit condition. To keep the analysis simple, we have selected the error on the distance that corresponds to the value $\epsilon_d \in \mathbb{R}^+$ explained in Section 4.2.3, in the range $[-\epsilon_z/4, \epsilon_z/4]$.

As before, we have defined and tested 10 possible values for $\epsilon_z$ in the range $[0.005, 0.05]$ with a step of 0.005.

#### Goal

The main goal of this experiment is to analyze how the final solution would change if we increase the error in the different routines.

As in the experiment of Q-MP we have analyzed; if the number of atoms selected for the solution would change increasing the error, if these different atoms would modify the stop condition, and if a relationship between the goodness of the solution and the dimension of the dataset was present.

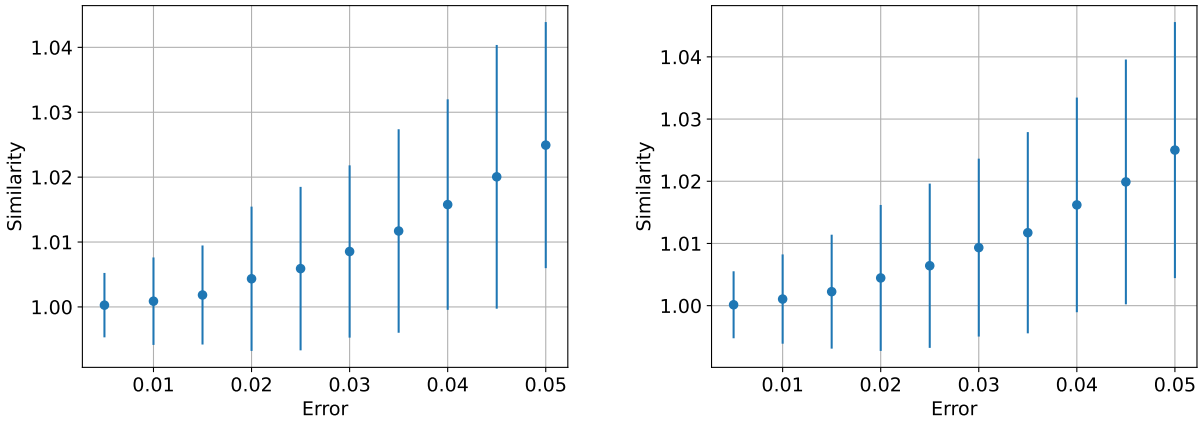
Finally, we provide an estimation of the parameters $\mu(A_t)$ and $\kappa(A_t)$ to show the bounds $\mu(A_t) \leq k^{3/2}$ and $\kappa(A_t) \leq \kappa(D)$ (as explained in Section 4.2.4).

## Results

### Relationship between similarity and error

First, we analyzed how the similarity would change if we increased the algorithm's error. We ran the classical and quantum procedures on each of the 2000 samples. We calculated the ratio of the sparsity of the quantum solution to the sparsity of the classical solution to find the similarities. We have repeated this procedure for each of the 10 errors examined. In the figures, we show the mean values and the standard deviations for each level of error.

Figure 5.9 shows the analysis of the similarity. Figure 5.9a and Figure 5.9b show the results with Gaussian and uniform error, respectively.




(a) Gaussian Error

(b) Uniform Error

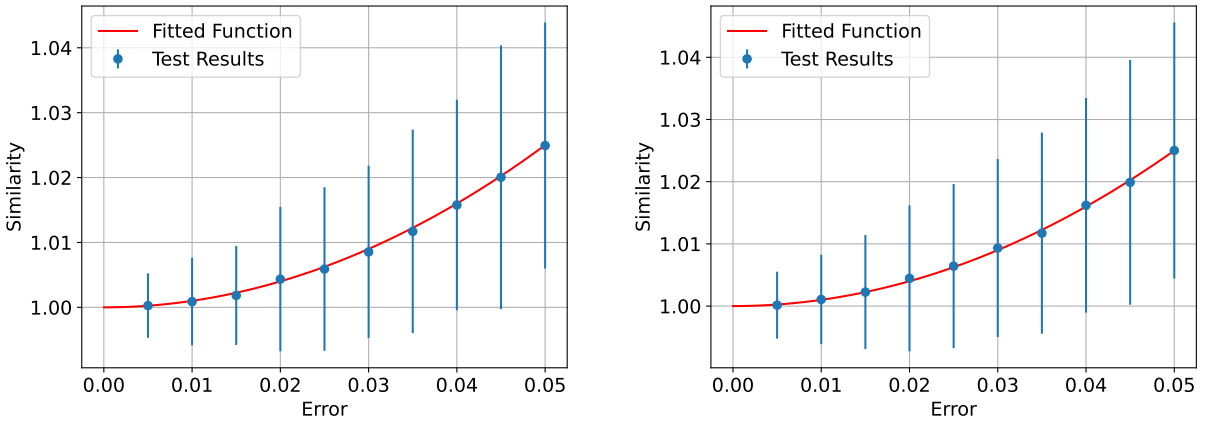
Figure 5.9: Similarity with relation to error

It is possible to notice how increasing the error, the similarity increases. This happens because with more error, it is easier to select the wrong atoms and, in this way, the algorithm needs more iteration to reach a small-enough residual. It is interesting to point out how the sparsity of the quantum version can also be better than the one of the classical version for low error values. This is visible since the similarity can assume values lower than 1. This result follows our expectations.

Another interesting consideration about the similarity is that it seems to follow the law

$$similarity = 1 + 10\,(\epsilon_z)^2 \tag{5.2}$$

as shown in Figure 5.10, both for the version with Gaussian error in Figure 5.10a and with Uniform error in Figure 5.10b.




(a) Gaussian Error

(b) Uniform Error

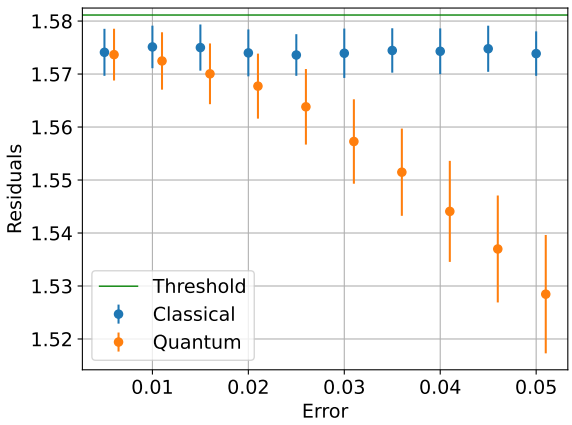
Figure 5.10: Similarity with relation to the error with fitted function

Fitting this polynomial, we have an almost perfect match, and it is interesting to notice how it is possible to estimate the final average similarity with enough confidence if we know the amount of error introduced beforehand.

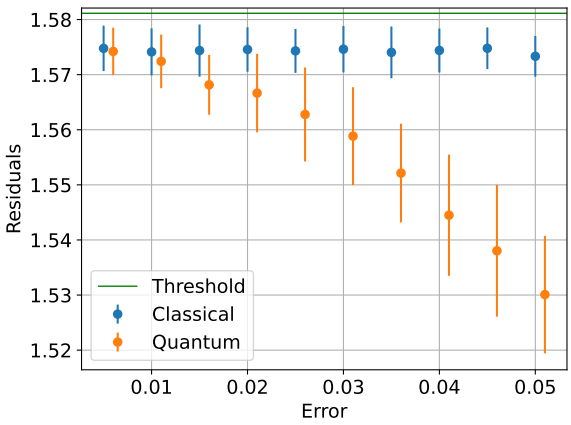
### Relationship between residual values and error

We have considered this second analysis to check if the algorithm, also with some noise, terminates giving good-enough solutions. In this case, we have used 100 test samples with the dimensions $(1000, 2000, 200)$ for the analysis. The mean norm of the signal is $\simeq 14.18$, and the error reconstruction tolerance is $\epsilon \simeq 1.581$. To obtain the value of the residual norm, we have used the atoms selected by the algorithm and classically recomputed the solution. We have made this choice following Algorithm 4.2 to avoid the selection of precision parameters ($\epsilon_x$ and $\eta_x$) on the quantum solution, that can be chosen arbitrarily by the user.

Figure 5.11 shows the residual value. Figure 5.11a shows the case of Gaussian error, and Figure 5.11b the case of uniform error.




(a) Gaussian Error

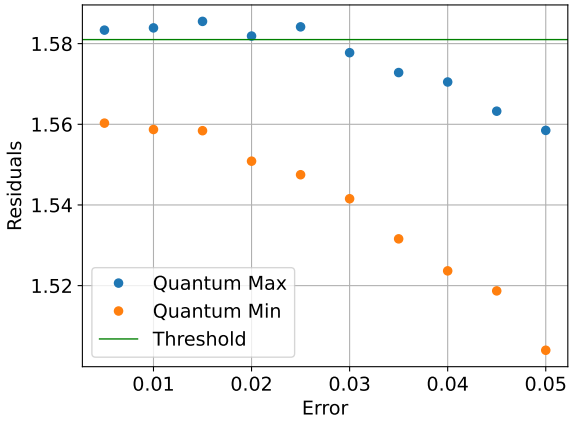



(b) Uniform Error

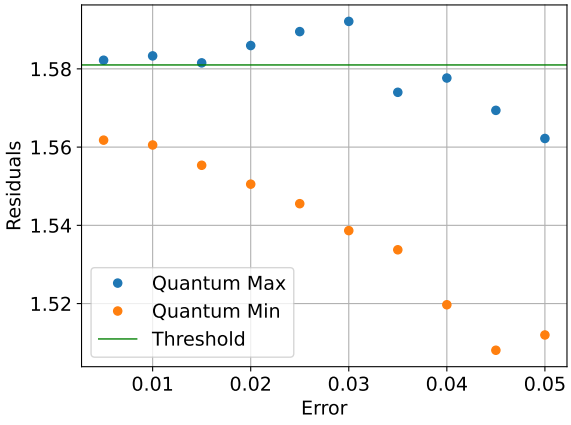
Figure 5.11: Residuals with relation to error

We can observe that the quantum algorithm leaves a lower residual than the classical one. We obtain this result by recomputing classically, without the addition of error, the final residual using the atoms selected using the quantum procedure. As it is possible to imagine, the quantum procedure selects more atoms than the classical one. For this reason, it is easy to compute a more precise approximation of the signal than the one obtained classically.

We also analyzed the maximum and the minimum value of each residual for the different choices of error. We show the results in Figure 5.12. Figure 5.12a shows the results with Gaussian error, and Figure 5.12b with uniform error.




(a) Gaussian Error




(b) Uniform Error

Figure 5.12: Maximum and minimum residual for each error

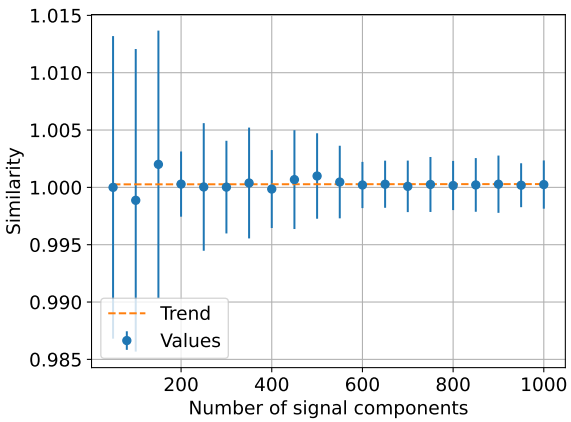
We observe that some of the values of the maximum residual exceed the threshold for

small error introduction, while for bigger errors, these values are below. The reason why this result is obtained is related to the correlation of two distinct phenomena. First of all, the estimation of the distance in the exit condition has some error. For that reason, it is possible to exit from the algorithm too soon and obtain a residual slightly above the limit. Secondly, with a greater amount of error in the algorithm, it is easier to select the wrong atoms and, for this reason, more atoms. In this way, the final approximation is less sparse and the value of the final residual decreases. In the end, if we recompute the solution classically, we have more precise but less sparse solutions for higher errors.

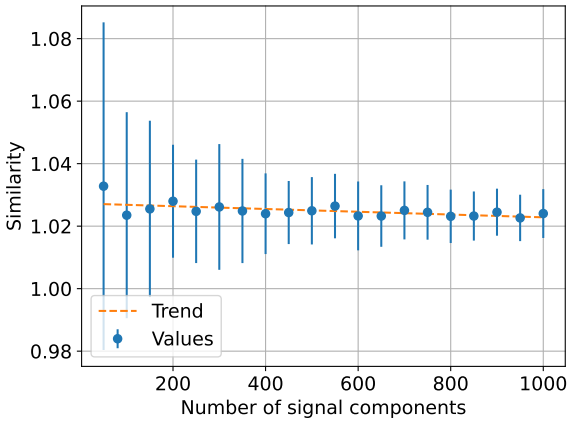
### Relationship between similarity and dimensions

Even for Q-OMP, we checked if there exists a dependency between the similarity and the dimension of the data. The results with the two error types (Gaussian and uniform) do not differ in a relevant way. For brevity, we show the results with two values of the Gaussian error.

Figure 5.13 shows the results with two different errors. Figure 5.13a shows the result with a Gaussian error of 0.005, and Figure 5.13b shows the result with a Gaussian error of 0.05.




(a) Relationship between dimension and similarity with error 0.005




(b) Relationship between dimension and similarity with error 0.05

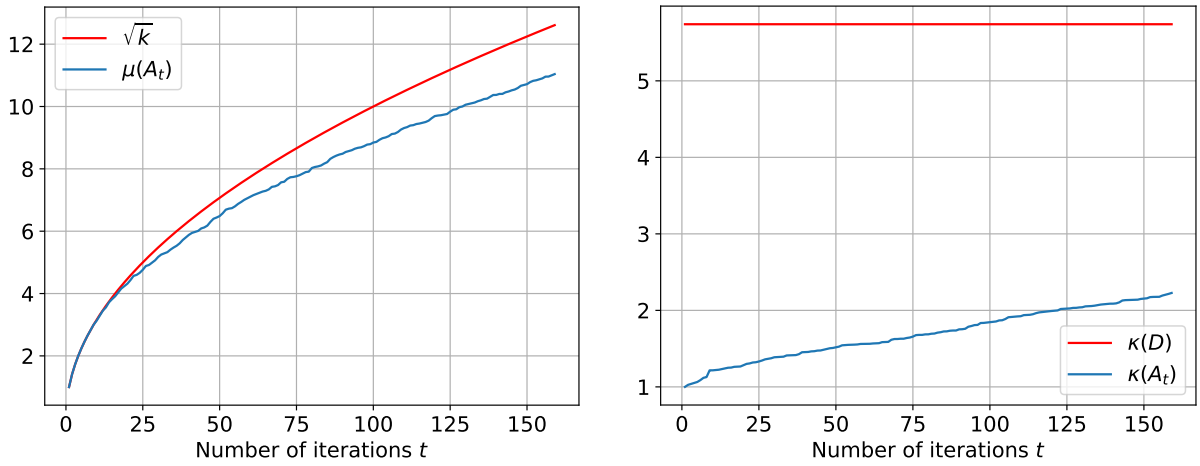
Figure 5.13: Relationship between dimension and similarity

It is possible to see that signals with more components have lower standard deviations. This phenomenon is caused by the fact that, as explained in the Q-MP experiment, it is easier to have a value of similarity close to 1 if we have sparsities with greater magnitude. As the trend line indicates, the average similarity remains stable as the dimensions increase while the standard deviation decreases.

### Parameters bounds

We have analysed the value of the parameters $\kappa(A_t)$ and $\mu(A_t)$, for $t = [0, .., k]$ where $k$ iterations of the algorithm are performed. We want to recall that the parameters $\mu(A_t)$ and $\kappa(A_t)$ are related to $A_t \in \mathbb{R}^{n \times t}$ that is a submatrix composed by $t$ atoms of the dictionary $D \in \mathbb{R}^{n \times m}$. To keep the analysis straightforward, we studied the parameter values for a sample of dimension 1000.

We show the results in Figure 5.14. Figure 5.14a shows results for the parameter $\mu(A_t)$ and Figure 5.14b shows the results for the value of the parameter $\kappa(A_t)$.




(a) Value of $\mu(A_t)$ during the computation

(b) Value of $\kappa(A_t)$ during the computation

Figure 5.14: Value of the parameters during the computation and their bounds

This result confirms that $\mu(A_t) \leq k^{1/2}$ and $\kappa(A_t) \leq \kappa(D)$, as analyzed theoretically in Section 4.2.4. In addition, we have shown how the condition number of the matrices that we create remains a low number for well-formed dictionaries as the ones generated by the function *make_sparse_coded_signal*.

## 5.3.2. Real Dataset Experiments

We have also performed some tests on a real dataset with the specific task of Sparse Representation Classification [38, 85]. We have decided to use the description of some softwares through the use of $n$-grams and try to distinguish between benign samples and malware.

First, we introduce what an $n$-gram is and why we can use it for this task. A $n$-gram is a contiguous sample of $n$ elements of text in a certain sequence. For example, in the

sentence "I am very hungry", considering the words as elements, there are 4 1-grams ("I", "am", "very", "hungry") and 3 2-grams ("I am", "am very", "very hungry"). This method is useful for various tasks such as Natural Language Processing [69].

For our application, this framework is helpful in describing hexadecimal instructions. Our objective is to identify specific kinds of software (malware) in which the frequency of specific $n$-grams is greater [1].

## Classification Algorithm

Algorithm 5.1 Sparse Representation Binary Classification

**Input:** a matrix of training samples $D \in \mathbb{R}^{n \times m}$ for 2 different classes such that $D = [D_1\, D_2]$ with $D_1 \in \mathbb{R}^{n \times m_1}$ with $m_1$ samples with $n$ features for one class and $D_2 \in \mathbb{R}^{n \times m_2}$ with $m_2$ samples with $n$ features for the other class, $m = m_1 + m_2$. Test sample $s \in \mathbb{R}^n$, $\epsilon \in \mathbb{R}^+$ error reconstruction tolerance, $\tau \in [0, 1]$ threshold on SCI parameter, $L \in \mathbb{N}$ sparsity required for OMP

**Output:** *class* or *sample refused*

1: Normalize the columns of $D$ to have unitary $l_2$-*norm*
2: Solve the $l_0$-minimization problem using OMP:
$x_0 = \arg\min_x \|x\|_0 \;\text{ subject to }\; \|Dx - s\| \leq \epsilon$
3: Compute the residual norms for each class: $\|r_i\| = \|s - D\delta_i(x)\|$ for $i = 1, 2$
4: Apply find the minimum and extract the label of the corresponding class:
$class = \arg\min_i \|r_i\|$.
5: Check if $SCI(x) \leq \tau$, if true refuse the sample
6: Output *class* or *sample refused*

We first compute the sparse representation that describes the signal and, after that, we select the class that is more related to it.

First, we initialize the matrix $D \in \mathbb{R}^{n \times m}$ that is obtained by concatenating a matrix $D_1 \in \mathbb{R}^{n \times m_1}$, that contains as columns $m_1$ samples of the first class of dimension $n$, and a matrix $D_2 \in \mathbb{R}^{n \times m_2}$, that contains as columns $m_2$ samples of the second class. Clearly, $m = m_1 + m_2$. We also initialize the test vector $s \in \mathbb{R}^n$. Before the computation, we normalize the column vectors of the matrix, $\|d_i\| = 1\, \forall i = 1, ..., m$.

The second step is the resolution of the minimization problem. Classically it is solved with a $l_1$-minimization approach, but it is possible to solve it also with a $l_0$-minimization approach, such as OMP, using as dictionary the matrix $D$ and the value $\epsilon \in \mathbb{R}^+$ as error

reconstruction tolerance.

Once we obtain the solution vector $x \in \mathbb{R}^m$, the following step is the computation of the norm of the residuals used to select the class that leaves less residual. The function $\delta_i(x)$ sets to zero the coefficients not related to the $i^{th}$ class. In this way, it is possible to select only the coefficients related to one class and obtain the norms of the residuals as

$$\|r_i\| = \|s - D\delta_i(x)\| \tag{5.3}$$

for $i = 1, 2$. We then select the index of the class with the lowest residual norm.

To validate the result, we check how much the coefficients of the solution are spread between the two classes. We use the $SCI$ parameter for this task.

**Definition 5.1** (Sparsity Concentration Index (SCI) [85]). *The SCI of a coefficient vector $x \in \mathbb{R}^n$ is defined as*

$$SCI(x) = \frac{k \cdot \max_i \|\delta_i(x)\|_1 / \|x\|_1 - 1}{k - 1} \tag{5.4}$$

*for a solution $\tilde{x}$, if SCI($\tilde{x}$)=1, then the test sample is represented from a single class, otherwise if SCI($\tilde{x}$)=0, it is spread evenly over all classes. We consider k different classes.*

If the value $SCI(x)$ is lower than a certain threshold $\tau \in [0, 1]$, we consider the sample $s$ as not possible to be correctly classified and excluded by the algorithm.

Algorithm 5.1 summarizes the Sparse Representation Binary Classification.

## Dataset

We have used the dataset created by Kumar [45]. This dataset had been created by downloading malware samples from different websites [22, 74, 81]. These malware were 600 samples of various types, including Viruses, Trojans, Worms, and Ransomware, obtained from 2014 to 2016. Malware were downloaded and disassembled in a virtual machine. A total of 504 Windows Applications/Software were taken from the same machine to be used as benign files. Also, the benign files had been disassembled and the duplicates were removed.

We have used 1-grams hexadecimal instructions to keep the analysis simple.

The overall HEX-dataset, provided in [45], is formed by 955 samples. Each sample corresponds to the 1-gram description of a software stored in a vector of length 256. Each component of the vector corresponds to the frequency of appearance of each hexadecimal instruction in the code. We have used the same split of [45] of the samples into a training

set of 716 elements (347 malware and 369 benign software) and a test set of 239 elements (114 malware and 125 benign software) to compare our results to theirs.

We have performed a 5-fold cross-validation method on the training set to understand the best values for the hyper-parameters of our algorithm and avoid over-fitting. In this step, we have excluded some samples to make the dataset perfectly balanced between bad and good samples (340 good samples and 340 bad samples).

We have used 5 different folds of dimension 136, each one with 68 malware and 68 benign softwares. We have used the 5 folds in the following way. During each one of the 5 iterations, we have used 3 folds to build the matrix of training sample $D$, one fold to tune the hyper-parameters, and one fold for testing. Each fold is used only once for validation and testing. We have used each fold three times to create the matrix.

## Hyperparameters

The two hyper-parameters to tune correspond to the threshold to assign to the OMP algorithm $\epsilon \in \mathbb{R}^+$ and the minimum value of SCI under which we must exclude the sample from the analysis called $\tau \in (0, 1)$. In this experiment, we consider instead of tuning the value $\epsilon$ to tune the value $\xi$ such that $\epsilon = \frac{\|s\|}{\xi}$ where $\|s\|$ is the norm of the sample to analyze. In this way, we describe the threshold of the OMP algorithm with a fraction of the sample norm. We have used a high value for the parameter $L = 300$ such that all the samples are described by OMP, focusing only on the value of $\epsilon$.

We have tuned the parameters optimizing the value

$$p = (1 - \lambda) * F1 + \lambda * (1 - \zeta) \tag{5.5}$$

where $F1$ corresponds to the $F1$-*score* and $\zeta$ corresponds to the fraction of the excluded samples. The parameter $\lambda$ describes how much we consider important to exclude fewer samples possible even if the performances decrease. We have tried different values for the value $\lambda$, as it is possible to see in the following section.

## Results

Table 5.1 shows the results obtained running the 5-fold cross-validation test. The parameters $\tau$ and $\xi$ were tuned using the validation set, while the parameter $\lambda$ was chosen by the user. We have varied the value of $\tau$ in the range $[0, 1]$ with a step of 0.05 during this procedure. Instead, for the value $\xi$, we have selected values in the range [1.5, 6] with step 0.5. We have chosen these values considering that, selecting values lower than 1.5, the

algorithm describes less than $\frac{1}{3}$ of the norm of the initial signal obtaining, in the majority of cases, a sample that is described by only one column of the matrix $D$. While for values greater than 6, which correspond to a description of more than $\frac{5}{6}$ of the initial norm of the signal, we obtain solutions with a very big value of sparsity because we overfit the sample.

The metrics that we have decided to analyze are the mean value of *Accuracy*, *Precision*, *Recall*, *F1 score* and the percentage of *Excluded samples* for the 5 test folds. In addition, we have also computed the mean value $\overline{\tau}$ of the parameter $\tau$ for the five folds and the mean value of $\xi$ called $\overline{\xi}$.

| $\lambda$ | **Acc.** | **Pre.** | **Rec.** | **F1** | **Excl.** | $\overline{\tau}$ | $\overline{\xi}$ |
|---|---|---|---|---|---|---|---|
| **0.0** | 83.93% | 84.44% | 80.07% | 82.00% | 18.99% | 0.99 | 5.00 |
| **0.1** | 81.69% | 83.43% | 77.89% | 80.44% | 4.35% | 0.84 | 2.20 |
| **0.2** | 81.69% | 83.43% | 77.89% | 80.44% | 4.35% | 0.84 | 2.20 |
| **0.5** | 81.72% | 84.01% | 78.22% | 80.92% | 0.87% | 0.54 | 1.50 |

Table 5.1: Results for hexadecimal 1-grams, varying $\lambda$

Without any limit in excluding samples, with value $\lambda = 0$, the algorithm excludes a big percentage of them, achieving an *Accuracy* and a $F1$ score that are the best values found in this step of the experiment. For the other values of $\lambda$, it is possible to see how the performances are very similar, but the percentage of excluded samples decreases. The amount of excluded samples is close to zero for $\lambda \geq 0.5$. It is also interesting to notice how both the mean values $\overline{\tau}$ and $\overline{\xi}$ decrease. The value $\overline{\tau}$ decreases since increasing the value of $\lambda$ the procedure excludes fewer samples from the analysis, so the algorithm lowers its bound on the *SCI* parameter. The value $\overline{\xi}$ instead decreases because describing a sample as a combination of fewer atoms it is easier to have a sparse representation concentrated toward one class. Consequently, the *SCI* parameter is greater and fewer samples are excluded.

At this point we have selected the value $\lambda = 0.5$, and its hyper-parameters $\tau = 0.54$ and $\epsilon = \frac{2}{3}\|s\|$. We have tested the algorithm on the same test sets as before adding errors to the OMP procedure as explained in Section 5.3.1 of the previous experiment, to simulate the use of a quantum algorithm. In Table 5.2, it is possible to see the mean results of the 5 test sets of the different folds.

| Error | Acc. | Pre. | Rec. | F1 | Excl. |
|---|---|---|---|---|---|
| **0.00** | 81.72% | 84.01% | 78.22% | 80.92% | 0.87% |
| **0.01** | 82.21% | 84.61% | 78.67% | 81.47% | 0.58% |
| **0.10** | 77.00% | 79.60% | 72.61% | 75.79% | 1.02% |
| **0.15** | 76.16% | 77.31% | 72.03% | 74.54% | 5.22% |
| **0.16** | 74.68% | 79.49% | 64.23% | 70.75% | 11.02% |
| **0.17** | 54.02% | 60.91% | 34.59% | 43.38% | 67.10% |

Table 5.2: Results for Hexadecimal 1-grams, varying $\epsilon_z$

As it is possible to notice, the algorithm maintains similar performances for errors in the range $[0.00, 0.1]$. It is interesting to notice how, by introducing a small amount of error, the performance can increase. This phenomenon can be considered just a local fluctuation, while the general trend that is possible to see is the decrease in the performance if we increase the error. It is also important to highlight how, after the value of error 0.1, if we increase more and more the error, the number of samples excluded increases very fast. This happens because if we add too much error, the sparse representation of the sample will be less sparse and less concentrated on one class. Consequently, also the value of the $SCI$ parameter decreases. The algorithm excludes the samples with a $SCI$ parameter lower than $\tau$ and, for this reason, excludes more samples.

## Comparison with other methods

We have decided to compare our method with other ones provided by [45]. We have considered the accuracy parameter to compare the performance since the training and the test set are balanced. We used the same train set to create the matrix $D$ and used the parameter $\epsilon = \frac{2}{3}\|s\|$ that we have found using the cross-validation method. We have considered different values for the parameter $\tau$.

It is important to highlight that with our method, modifying the parameter $\tau$, it is possible to choose a balance between how much we want to be sure about our predictions and the amount of excluded samples. If we want to select only the samples for which the algorithm is completely sure about their classification, it is enough to select $\tau = 1$. While choosing $\tau = 0$, we avoid the exclusion of samples to be completely comparable with the other methods. In this experiment, we have obtained the same results for $\tau = 0.54$ and $\tau = 1$.

We use the notation SRC-0 to denote the classical version of the algorithm with $\tau = 0$

while SCI-1 denotes the version with $\tau = 1$. The notation QSRC-0 indicates the quantum SRC method with error 0.01 and $\tau = 0$. Finally, the notation QSRC-1 denotes the quantum SRC method with error 0.01 and $\tau = 1$.

| Method | Accuracy | Excluded |
|---|---|---|
| **Random Forest** | 89.95% | 0.00% |
| **SVC('Linear' Kernel)** | 88.28% | 0.00% |
| **SRC-1** | 88.24% | 0.42% |
| **SRC-0** | 87.87% | 0.00% |
| **QSRC-1** | 86.56% | 0.42% |
| **QSRC-0** | 86.19% | 0.00% |
| **Decision Trees** | 85.77% | 0.00% |
| **KNN** | 79.91% | 0.00% |
| **Naive Bayes** | 66.94% | 0.00% |

Table 5.3: Comparison using Hexadecimal 1-grams

As it is possible to see in Table 5.3, if we set the $\tau$ parameter to 1, we exclude some samples, but we achieve greater accuracy. In addition, the general performances of our method are comparable with the ones of the other algorithms, also for the simulation of a quantum version in which we consider the addition of some error.

## 5.4. Q-ECR

We provide experiments related to the Quantum Eigenvectors Classification/Recognition algorithm defined in Section 4.3. We run two different experiments. The first one with the objective of face recognition, and the other one to perform anomaly detection in internet traffic.

### 5.4.1. Face Recognition

For this experiment, we modified the code of Singhal [70] to manage a different dataset and simulate the quantum algorithm errors.

#### Dataset

To perform this experiment, we used the Extended Yale Face Database B [32, 47]. This dataset contains 21888 images of 38 human subjects under 9 poses and 64 illumination conditions. We have decided to train the model using 132 images of 22 different subjects with the same 6 conditions of pose and illumination for each of them. Figure 5.15 shows the faces of the training set.

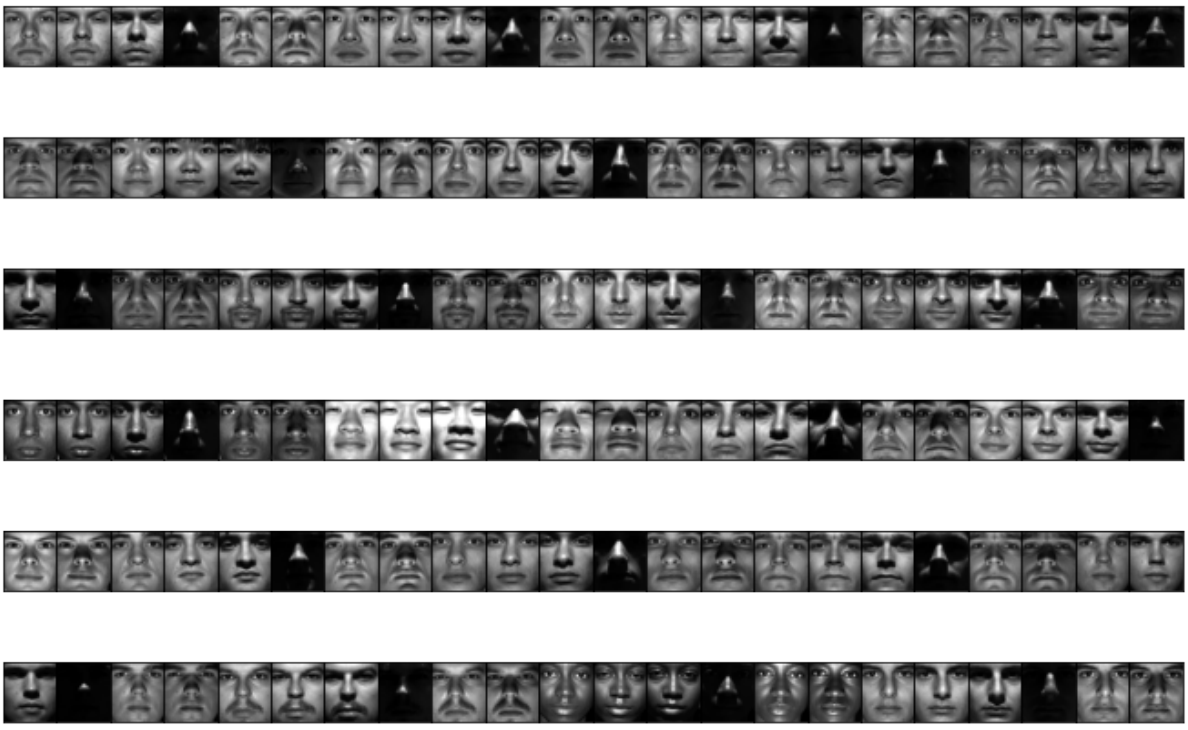


Figure 5.15: Faces of the Extended Yale B Dataset used for training

Instead, the validation set consisted of 19 images of 19 subjects with only one light condition, different from the ones used in the training set. 11 of the participants in the validation set were also in the training set, but the remaining 8 subjects were not. Figure 5.16 shows the faces of the validation set.

Figure 5.16: Faces of the Extended Yale B Dataset used for validation

The test set was composed of 19 images of 19 subjects with only one light condition, different from the ones used in the training set. 11 subjects were also in the training set, while the 8 others were not. The validation and the test set have no subjects in common. Figure 5.17 shows the images in the test set.

Figure 5.17: Faces of the Extended Yale B Dataset used for test

## Error introduction

We have introduced the error in the following way. As explained in Theorem 4.8, we have added some error in the final value of the distance. In Algorithm 4.3, we considered an error $\epsilon_\delta \in \mathbb{R}^+$ on the squared value of the distance such that $|\overline{d}^2 - d^2| \leq \epsilon_\delta$. First of all, for this experiment, we have considered a relative error instead of an absolute one because we have used not normalized samples. In this case, we can describe the error on the squared distance as $|\overline{d}^2 - d^2| \leq \eta\, d^2$ with $\eta \in \mathbb{R}^+$, just selecting $\epsilon_\delta = \eta\, d^2$. In addition to that, to make the experiments easier to understand, we have considered an error in the distance and not in its squared value. For this reason we have defined $|\overline{d} - d| \leq \eta_d\, d$ with $\eta_d \in (0, 1)$. It is easy to connect the values $\eta$ and $\eta_d$ just noticing that

$$\begin{aligned} |\overline{d}^2 - d^2| = |\overline{d} - d||\overline{d} + d| \leq \eta_d\, d|\overline{d} + d| = \eta_d\, d|\overline{d} - d + 2d| \\ \leq \eta_d\, d(\eta_d\, d + 2d) = \eta_d^2 d^2 + 2d^2\eta_d = (\eta_d^2 + 2\eta_d)d^2 \end{aligned}$$

At this point, as a consequence of $\eta_d \in (0, 1)$, we can say that $\eta_d^2 + 2\eta_d \leq 3\eta_d$. We select $\eta = 3\eta_d$ to satisfy the bound. For this reason, guaranteeing a relative error between $[-\eta_d, \eta_d]$ for the distance, we guarantee a relative error in $[-3\eta_d, 3\eta_d]$ for the squared distance.

We proceed with the description of the experiment using the value $\eta_d$ to define the error. For example, with a distance of $d = 2500$ with $\eta_d = 0.01$ we have considered an approximated distance $\overline{d} \in [2475, 2525]$.

## Goal

The goal of this experiment is to show how it is possible to perform facial recognition using Eigenvector Classification/Recognition and how robust to error it is using real data. This noise emulates the use of quantum routines in which the values obtained are not precise due to the introduction of error in the distance estimated, as explained in Section 4.3. Another interesting part of the experiment is related to the tuning of the hyper-parameter $\delta_1 \in \mathbb{R}^+$ used to decide if a face is already in the dataset or not. The hyper-parameter $\delta_2 \in \mathbb{R}^+$, used to understand if the sample is a face, has not been tuned since the dataset was composed only of face images.

## Results

We have carried out the experiment in the following way. First of all, we have used the images contained in the training set to create the matrix of the eigenvectors using the *PCA* method of *sklearn.decomposition* [63]. We have selected the main 11 components describing 80% of the total variance. After that, using the validation set, we have found the correct parameter $\delta_1$ to maximize the *Accuracy* and the *F1 score*. If the distance is lower than $\delta_1$, the sample is a known face, while if it is not, we consider it a different face.

In Figure 5.18, it is possible to see how the *Accuracy* and the *F1 score* would vary by changing the parameter $\delta_1$.

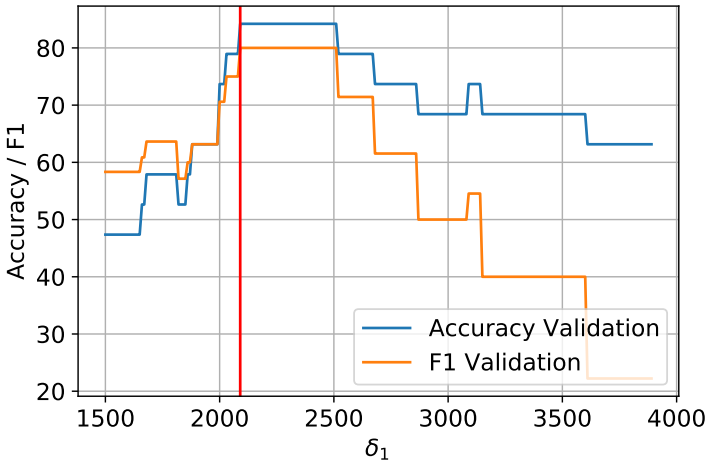



Figure 5.18: Accuracy and F1 score for different $\delta_1$ values

It is possible to notice how the best value for the *Accuracy* parameter is also the best for the *F1 score*. This value corresponds to $\delta_1 = 2090$.

At this point, we have classified the images in the test set. Table 5.4 shows the results introducing different amounts of error in the distances computed. We used the trained parameter $\delta_1 = 2090$ as threshold.

| Error | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| **0.0** | 89.47% | 100.00% | 83.33% | 90.91% |
| **0.1** | 89.47% | 100.00% | 83.33% | 90.91% |
| **0.3** | 78.95% | 80.00% | 80.00% | 80.00% |
| **0.5** | 73.68% | 70.00% | 77.78% | 73.68% |
| **0.8** | 47.37% | 40.00% | 50.00% | 44.45% |

Table 5.4: Results for Face Recognition with different error

It is possible to see that this procedure is robust to the introduction of error. If we introduce an error between 0.0 and 0.3, the final result maintains good performance. Increasing more and more the error it is possible to observe how the performance decrease importantly. As it is easy to notice, both the *Precision* and the *Recall* decrease. The precision parameter decreases significantly, showing a high number of false positives. This means that we recognize faces even if they are not present in the dataset due to the error in the distance. It is easy to understand the reason why this happens. Considering that we add a relative error on the distance and that this value is low for the known faces, it is clear that these samples are less sensitive to the introduction of error. For the samples of unknown faces, we have the opposite situation with a higher distance and, consequently, a higher relative error.

### 5.4.2. Anomaly Detection in Network Traffic

For this experiment, we have decided to use the same code as the previous one adapting it to the new data. Since it is possible to describe very different kinds of data as simple vectors, we have decided to try to use the same method used for facial recognition with vectors that describe network traffic. This shows the flexibility of the general method.

#### Dataset

For this experiment, we have used the Darknet dataset [46]. To be more specific, the dataset we have used in our experiments is a part of the one used in the master thesis work of Fioravanti [29]. The dataset is made by an ensemble of samples that describe regular traffic, VPN traffic, and Tor traffic. Seven different categories had been considered: Browsing, Chat, Email, File Transfer, FTP over SSL, Streaming, Voip, and P2P. We have used a total of 80,000 samples, of which 70,000 (87.5%) benign samples and 10,000 (12.5%) darknet samples which are considered anomalous ones (to be more specific, the ones labeled as TOR or VPN).

We have decided to split our dataset into three parts, train, validation, and test. The training dataset consists of 60,000 (75%) benign samples. The validation set consists of 10,000 samples, of which 5,000 are benign samples and 5,000 are anomalies. Finally, we used the test set to check the performance of the algorithm. For this dataset, we have used 10,000 samples, of which 5,000 are benign samples and 5,000 are anomalies.

The dataset does not contain duplicates. The benign samples and the anomalies had been shuffled randomly before performing the division.

#### Error introduction

We have introduced the error as in the previous experiment, with a relative error $\epsilon_d \in \mathbb{R}^+$ on the distance.

#### Goal

The goal is to show how to use this algorithm to implement a straightforward method for anomaly recognition with good performances. In addition, we want to show how the introduction of some error that simulates a quantum version of the algorithm does not affect the overall performance of the classifier.

## Results

As in the previous experiment, we have used the training set to create the matrix of the eigenvectors using the PCA method of *sklearn.decomposition* [63]. We have described 80% of the total variance using 13 components. We have used the validation set to tune the parameter $\delta_1 \in \mathbb{R}^+$. If the distance is lower than this value, we can consider the sample in the same class of the training set (benign samples). Instead, if the distance is greater, we consider the sample an anomaly (see Section 1.2.4). To tune this parameter, we have tried to maximize the *Accuracy* and the *F1 score*.

Figure 5.19 shows how the *Accuracy* and the *F1 score* would vary by changing the parameter $\delta_1$ using the validation set.

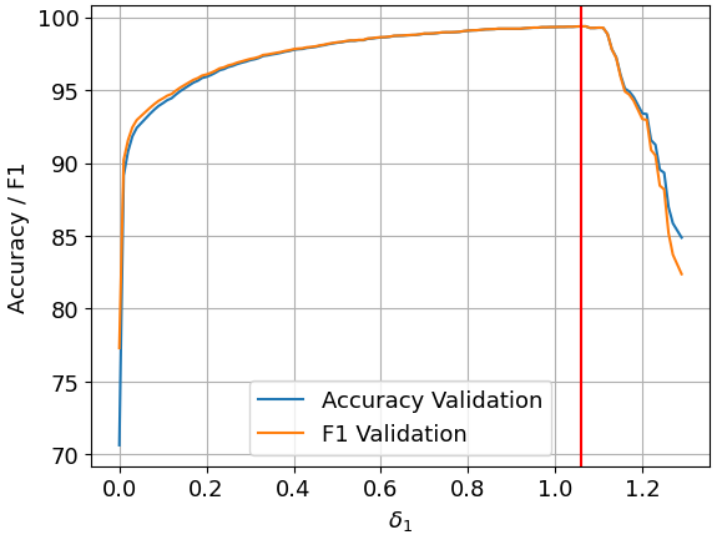



Figure 5.19: Accuracy and F1 score for different $\delta_1$ values

Using this evaluation, we have selected a parameter $\delta_1 = 1.06$. The best value for the *Accuracy* parameter is also the best value for the *F1 score*

At this point, we have used the test set to check the performances of the method with $\delta_1 = 1.06$. We have decided to try different values for the error parameter. In Table 5.5, it is possible to see the results.

| Error | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| **0.0** | 99.42% | 100.0% | 98.85% | 99.42% |
| **0.1** | 99.42% | 100.0% | 98.85% | 99.42% |
| **0.3** | 86.22% | 73.54% | 98.53% | 84.22% |
| **0.5** | 65.35% | 31.68% | 97.00% | 47.76% |
| **0.6** | 57.68% | 16.08% | 95.71% | 27.53% |

Table 5.5: Results for Anomalies Recognition with different errors

As in the previous experiment, it is possible to see that this procedure is robust to the introduction of error. If we introduce an error between 0.0 and 0.3, the final result slightly decreases. Obviously, with a high value of error, the performance decreases significantly. Also in this case, we have a significant decrease in the precision that shows how we classify as good samples some anomalies due to the presence of error.

It is interesting to notice how, even if the two datasets analyzed contain very different kinds of data, the performances decrease in a very similar way introducing the error.

# 6 | Limitations and Future Work

In this work, we formalized and tested different quantum algorithms for learning sparse representations. For each of them, we can identify some limitations and possible research directions that are worth examining.

**Limitations**

The main limitation of our algorithms probably is the running time dependency on the parameters $\epsilon_z$ for Theorem 4.1 and Theorem 4.5 and $\epsilon_\delta$ for Theorem 4.8, which are not bounded by an easy rule. If we want to achieve similar results to the classical algorithms, their value must be tuned using experiments.

The use of the QRAM in the algorithms of Q-MP and Q-OMP can also be considered a restriction since the opinions in the quantum community on the possibility of creating a fault-tolerant quantum memory that can act as required are discordant.

Considering the Double Error version of the Q-MP algorithm, an obstacle is not having a precise bound on the average case of error that can be propagated through iterations but just a worst-case bound. An average-case bound can be helpful to understand better the behavior of the algorithm when it is used and have a more clear vision of the results of the experiments performed.

A general limitation of our work is the fact that we have developed our algorithms based on procedures that were already created and studied classically. A way to potentially obtain relevant speed-ups in the computation of the same results can be to try changing our perspective on the problem. If we do not adapt classical algorithms to the quantum framework but, instead, develop new algorithms that can be defined using quantum-only procedures, it can be possible to achieve better results.

**Future work on new quantum algorithms**

A very interesting research direction can be the development of quantum versions of different algorithms based on OMP, such as ROMP [59] or CoSaMP [58]. With the additional constraint of the validity of the RIP [12], these procedures obtain low running

time classically, and it would be extremely interesting to see if these speed-ups would be present also in the quantum version of the algorithms and how much their running time would differ from the one obtained in this work.

One direction we have analyzed during the development of this work is the definition of algorithms for Dictionary Learning. Various methods to obtain a dictionary that can be used more efficiently for the description of sparse signals than a user-provided one are based on algorithms such as Orthogonal Matching Pursuit. Even if we do not formally present it in this work, we have studied how, in the block-encodings framework, it is possible to obtain a simple quantum version of the algorithm known as Method of Optimal Directions (MOD)[28]. To point out why we consider this direction promising, we want to highlight how, since the new dictionary is computed as a product of matrices, it is possible to use the block-encoding framework to compute this product efficiently. To continue the research in the direction of quantum algorithms for sparse representation, the formal development of the MOD algorithm or similar algorithms for dictionary learning can be an interesting way to apply our results about the Quantum Orthogonal Matching Pursuit.

**Other interesting research directions**

An additional possible future work is the development of more tests for the algorithms studied with real datasets. It would be interesting to analyze the performances of Q-MP with real data and increase the number of experiments of Q-OMP with the use, for example, of datasets with n-grams defined by a highier amount of consecutive strings.

Finally, another possible direction not studied in this work in detail but just as an application is the use of classical sparse representation algorithms for the classification task for cybersecurity. It would be interesting to check if these methods can achieve good results on other tasks related to the same field as, for example, the detection of intrusion and the detection of anomalies for fraud detection.

# 7 | Conclusions

**Contributions**

In this study, we have defined and analyzed three new quantum algorithms that can be used as a foundation for the application of sparse representation theory in the quantum context. We have created two distinct algorithms with different running times and final error guarantees, reported in Theorems 4.1 and Theorems 4.5, for describing a sample as a sparse combination of dictionary atoms. In addition, we have also examined a classification algorithm with running time reported in Theorem 4.8. We have analyzed carefully the value of the errors introduced into the various routines and have highlighted some technical results that may be useful for future research in the same area. We have proven the upper bounds on the running times of these procedures achieving a polynomial speed-up with relation to the classical algorithms thanks to the use of novel quantum routines defined for the block-encoding framework. We have analyzed the probability of success of each algorithm and the additional polylogarithmic running time necessary to obtain good enough results.

**Experiments**

In this study, we have demonstrated how the analyzed algorithms are impacted by the introduction of error, which is typical of quantum procedures. We have examined whether the use of algorithms in which some error is introduced would change the results of the procedures. To check it, we have used both artificially generated and real datasets, such as the HEX-dataset [45] for the Q-OMP case, and we have found that it does not happen for reasonably low values of error. Additionally, we have shown how the Sparse Representation Classification algorithm can be used to identify malware using their n-grams description, achieving results similar to other state-of-the-art methods. We have also shown how a quantum reduced-representation-based classification methods, such as the Q-ECR algorithm, can be used for a variety of tasks, from Face Recognition using the Extended Yale Face Dataset b [32, 47] to Anomalies Detection using the Darknet dataset [46]. This method obtains excellent results and a similar decrease in performances related to error introduction for both tasks. We have shown how the bounds we identified

in the theoretical definition of our algorithms match up with the outcomes of our tests.

### Conclusion

With this work, we have added another brick to the wall which is the development and comprehension of quantum computing-based algorithms. We have defined procedures that may also be beneficial for other quantum machine learning algorithms. We have defined quantum algorithms that can be used to obtain approximated sparse representations from signals with guarantees on the final error. Finally, we have also developed a quantum algorithm for classification that can be used for very different tasks also related to cybersecurity. We expect that it will be possible to create many other algorithms with good performances related to this field in the future.

# Bibliography

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. N-gram-based detection of new malicious code. In *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, volume 2, pages 41–42 vol.2, 2004. doi: 10.1109/CMPSAC.2004.1342667.

[2] A. Adler, M. Elad, Y. Hel-Or, and E. Rivlin. Sparse coding with anomaly detection. *J. Signal Process. Syst.*, 79(2):179–188, may 2015. ISSN 1939-8018. doi: 10.1007/s11265-014-0913-0. URL https://doi.org/10.1007/s11265-014-0913-0.

[3] A. Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations, 2010. URL https://arxiv.org/abs/1010.4458.

[4] A. Ang. Uniqueness of solution on sparse recovery problem. Lecture Notes, 2020.

[5] M. S. ANIS, Abby-Mitchell, and et al. Qiskit: An open-source framework for quantum computing, 2021.

[6] H. E. Bell. Gershgorin's theorem and the zeros of polynomials. *American Mathematical Monthly*, 72:292, 1965.

[7] A. Bellante and S. Zanero. Quantum matching pursuit: A quantum algorithm for sparse representations. *Physical Review A*, 105(2), feb 2022. doi: 10.1103/physreva.105.022414. URL https://doi.org/10.1103%2Fphysreva.105.022414.

[8] F. Bergeaud and S. Mallat. Matching pursuit of images. In *Proceedings., International Conference on Image Processing*, volume 1, pages 53–56 vol.1, 1995. doi: 10.1109/ICIP.1995.529037.

[9] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.*, 94:015004, Feb 2022. doi: 10.1103/RevModPhys.94.015004. URL https://link.aps.org/doi/10.1103/RevModPhys.94.015004.

[10] N. Bhatia and Vandana. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8, 07 2010.

[11] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation, 2002. URL `https://doi.org/10.1090%2Fconm%2F305%2F05215`.

[12] E. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005. doi: 10.1109/TIT.2005.858979.

[13] E. J. Candes and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008. doi: 10.1109/MSP.2007.914731.

[14] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59, 08 2006. doi: 10.1002/cpa.20124.

[15] S. Chakraborty, A. Gilyén, and S. Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation. In *International Colloquium on Automata, Languages and Programming*, 2019.

[16] S. Chakraborty, A. Morolia, and A. Peduri. Quantum regularized least squares, 2022. URL `https://arxiv.org/abs/2206.13143`.

[17] Y. Chen and R. de Wolf. Quantum algorithms and lower bounds for linear regression with norm constraints, 2021. URL `https://arxiv.org/abs/2110.13086`.

[18] P. Courrieu. Fast computation of moore-penrose inverse matrices. *Neural Information Processing-Letters and Reviews*, 8, 05 2008.

[19] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

[20] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009. doi: 10.1109/TIT.2009.2016006.

[21] Y. Dang, N. Jiang, H. Hu, Z. Ji, and W. Zhang. Image classification based on quantum k-nearest-neighbor algorithm. *Quantum Information Processing*, 17, 08 2018. doi: 10.1007/s11128-018-2004-9.

[22] Dasmalwerk. Dasmalwerk malware dataset, 2022. URL `https://dasmalwerk.eu/`.

[23] M. Davenport and M. Wakin. Analysis of orthogonal matching pursuit using the restricted isometry property. *Information Theory, IEEE Transactions on*, 56:4395 – 4401, 10 2010. doi: 10.1109/TIT.2010.2054653.

[24] D. Donoho. For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, 59:907 – 934, 07 2006. doi: 10.1002/cpa.20131.

[25] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001. doi: 10.1109/18.959265.

[26] D. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58:1094–1121, 02 2012. doi: 10.1109/TIT.2011.2173241.

[27] C. Dürr and P. Hoyer. A quantum algorithm for finding the minimum. *CoRR*, quant-ph/9607014, 07 1996.

[28] K. Engan, S. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 5, pages 2443–2446 vol.5, 1999. doi: 10.1109/ICASSP.1999.760624.

[29] T. Fioravanti. Evaluation of quantum machine learning algorithms for cybersecurity. Master's thesis, School of Industrial Engineering and Information, 2022.

[30] T. Fioravanti. Sqlearn-experiments, 2022. URL `https://github.com/tommasofioravanti/sq-learn-experiments`.

[31] E. Fix and J. Hodges. *Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties*. USAF School of Aviation Medicine, 1951. URL `https://books.google.it/books?id=4XwytAEACAAJ`.

[32] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[33] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, jun 2019. doi: 10.1145/3313276.3316366. URL `https://doi.org/10.1145%2F3313276.3316366`.

[34] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), apr 2008. doi: 10.1103/physrevlett.100.160501. URL `https://doi.org/10.1103%2Fphysrevlett.100.160501`.

[35] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237866. URL https://doi.org/10.1145/237814.237866.

[36] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), oct 2009. doi: 10.1103/physrevlett.103.150502. URL https://doi.org/10.1103%2Fphysrevlett.103.150502.

[37] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936. ISSN 00063444. URL http://www.jstor.org/stable/2333955.

[38] K. Huang and S. Aviyente. Sparse representation for signal classification. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/file/c922de9e01cba8a4684f6c3471130e4c-Paper.pdf.

[39] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2006. ISBN 978-0-19-857000-4.

[40] I. Kerenidis and A. Luongo. Classification of the MNIST data set with quantum slow feature analysis. *Physical Review A*, 101(6), jun 2020. doi: 10.1103/physreva.101.062327. URL https://doi.org/10.1103%2Fphysreva.101.062327.

[41] I. Kerenidis and A. Prakash. Quantum Recommendation Systems. In C. H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.49. URL http://drops.dagstuhl.de/opus/volltexte/2017/8154.

[42] I. Kerenidis and A. Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2), feb 2020. doi: 10.1103/physreva.101.022316. URL https://doi.org/10.1103%2Fphysreva.101.022316.

[43] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash. q-means: A quantum algorithm for unsupervised machine learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/16026d60ff9b54410b3435b403afd226-Paper.pdf.

[44] S. Krstulovic and R. Gribonval. Mptk: Matching pursuit made tractable. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 3, pages III–III, 2006. doi: 10.1109/ICASSP.2006.1660699.

[45] A. Kumar. Malware-detection-using-n-gram-frequency, 2021. URL `https://github.com/Arun152k/Malware-Detection-using-N-Gram-Frequency`.

[46] A. H. Lashkari, G. Kaur, and A. Rahali. Didarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning. *10th International Conference on Communication and Network Security*, nov 2020.

[47] K. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(5):684–698, 2005.

[48] Y. Liao and R. Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers and Security*, 21:439–448, 10 2002. doi: 10.1016/S0167-4048(02)00514-X.

[49] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, jul 2014. doi: 10.1038/nphys3029. URL `https://doi.org/10.1038%2Fnphys3029`.

[50] G. H. Low and I. L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, jul 2019. doi: 10.22331/q-2019-07-12-163. URL `https://doi.org/10.22331%2Fq-2019-07-12-163`.

[51] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008. doi: 10.1109/TIP.2007.911828.

[52] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. doi: 10.1109/78.258082.

[53] J. M. Martyn, Z. M. Rossi, A. K. Tan, and I. L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4), dec 2021. doi: 10.1103/prxquantum.2.040203. URL `https://doi.org/10.1103%2Fprxquantum.2.040203`.

[54] A. Maćkiewicz and W. Ratajczak. Principal components analysis (pca). *Computers and Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R. URL `https://www.sciencedirect.com/science/article/pii/009830049390090R`.

[55] Q. Mo. A sharp restricted isometry constant bound of orthogonal matching pursuit. *CoRR*, abs/1501.01708, 2015. URL http://arxiv.org/abs/1501.01708.

[56] Q. Mo and y. Shen. A remark on the restricted isometry property in orthogonal matching pursuit. *IEEE Transactions on Information Theory - TIT*, 58, 01 2012. doi: 10.1109/TIT.2012.2185923.

[57] A. Montanaro and C. Shao. Quantum communication complexity of linear regression, 2022. URL https://arxiv.org/abs/2210.01601.

[58] D. Needell and J. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2008.07.002. URL https://www.sciencedirect.com/science/article/pii/S1063520308000638.

[59] D. Needell and R. Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9, 08 2007. doi: 10.1007/s10208-008-9031-3.

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

[61] Y. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, 1993.

[62] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901. doi: 10.1080/14786440109462720. URL https://doi.org/10.1080/14786440109462720.

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[64] W. C. Peng Li. Signal recovery under mutual incoherence property and oracle inequalities. *Frontiers of Mathematics in China*, 13(6):1369, 2018. doi: 10.1007/s11464-018-0733-9. URL https://journal.hep.com.cn/fmc/EN/abstract/article_24101.shtml.

[65] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13), sep 2014. doi:

10.1103/physrevlett.113.130503. URL https://doi.org/10.1103%2Fphysrevlett.113.130503.

[66] C. Shao. From linear combination of quantum states to grover's searching algorithm, 2018. URL https://arxiv.org/abs/1807.09693.

[67] C. Shao. Quantum arnoldi and conjugate gradient iteration algorithm, 2018. URL https://arxiv.org/abs/1807.07820.

[68] B. Shen, W. Hu, Y. Zhang, and Y.-J. Zhang. Image inpainting via sparse representation. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 697–700, 2009. doi: 10.1109/ICASSP.2009.4959679.

[69] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.08.015. URL https://www.sciencedirect.com/science/article/pii/S0957417413006271. Methods and Applications of Artificial and Computational Intelligence.

[70] V. Singhal. Eigenfaces recognition, 2017. URL https://github.com/vutsalsinghal/EigenFace.

[71] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America. A, Optics and image science*, 4:519–24, 04 1987. doi: 10.1364/JOSAA.4.000519.

[72] B. L. Sturm and M. G. Christensen. Comparison of orthogonal matching pursuit implementations. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 220–224, 2012.

[73] M.-Y. Su. Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications*, 38(4):3492–3498, 2011. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2010.08.137. URL https://www.sciencedirect.com/science/article/pii/S0957417410009450.

[74] TekDefense. TekDefense malware dataset, 2022. URL http://www.tekdefense.com/downloads/malware-samples/.

[75] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL http://www.jstor.org/stable/2346178.

[76] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991. doi: 10.1109/CVPR.1991.139758.

[77] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991. doi: 10.1109/CVPR.1991.139758.

[78] J. van Apeldoorn, A. Cornelissen, A. Gilyén, and G. Nannicini. Quantum tomography using state-preparation unitaries, 2022. URL `https://arxiv.org/abs/2207.08800`.

[79] S. Vanerio. Quantum sparse algorithms, 2022. URL `https://github.com/Stefano-Vanerio/quantumSparseAlgorithms`.

[80] S. Vanerio. Quantum eigenvector classification/eigenface recognition, 2022. URL `https://github.com/Stefano-Vanerio/Quantum-Eigenvector-Classification`.

[81] VirusShare. VirusShare malware dataset, 2022. URL `https://virusshare.com/`.

[82] J. Wang and B. Shim. A simple proof of the mutual incoherence condition for orthogonal matching pursuit. *Computing Research Repository - CORR*, 05 2011.

[83] N. Wiebe, A. Kapoor, and K. M. Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Info. Comput.*, 15(3–4): 316–356, mar 2015. ISSN 1533-7146.

[84] Wikipedia. Bloch sphere, 2022. URL `https://en.wikipedia.org/wiki/Bloch_sphere`.

[85] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009. doi: 10.1109/TPAMI.2008.79.

[86] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009. doi: 10.1109/TPAMI.2008.79.

[87] Y. Wu and V. S. Batista. Matching-pursuit for simulations of quantum processes. *The Journal of Chemical Physics*, 118(15):6720–6724, 2003. doi: 10.1063/1.1560636. URL `https://doi.org/10.1063/1.1560636`.

[88] Y. Wu and V. S. Batista. Quantum tunneling dynamics in multidimensional systems: A matching-pursuit description. *The Journal of Chemical Physics*, 121(4):1676–1680, 2004. doi: 10.1063/1.1766298. URL `https://doi.org/10.1063/1.1766298`.

[89] W. Yambor, B. Draper, and J. Beveridge. Analyzing pca-based face recognition algorithms: Eigenvector selection and distance measures. *Empirical Evaluation Methods in Computer Vision*, 05 2002. doi: 10.1142/9789812777423_0003.

[90] T. J. Yoder, G. H. Low, and I. L. Chuang. Fixed-point quantum search with an optimal number of queries. *Physical Review Letters*, 113(21), nov 2014. doi: 10.1103/physrevlett.113.210501. URL https://doi.org/10.1103%2Fphysrevlett.113.210501.

[91] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: Algorithms and applications. *IEEE Access*, 3:490–530, 2015. doi: 10.1109/access.2015.2430359. URL https://doi.org/10.1109%2Faccess.2015.2430359.

[92] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: Algorithms and applications. *IEEE Access*, 3:490–530, 2015. doi: 10.1109/ACCESS.2015.2430359.

[93] M. Zhao, J. Chen, and Y. Li. A review of anomaly detection techniques based on nearest neighbor. In *Proceedings of the 2018 International Conference on Computer Modeling, Simulation and Algorithm (CMSA 2018)*, pages 290–292. Atlantis Press, 2018/04. ISBN 978-94-6252-523-8. doi: https://doi.org/10.2991/cmsa-18.2018.65. URL https://doi.org/10.2991/cmsa-18.2018.65.

[94] H. Zhu, G. Yang, and W. Chen. Efficient implementations of orthogonal matching pursuit based on inverse cholesky factorization. In *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, pages 1–5, 2013. doi: 10.1109/VTCFall.2013.6692175.

# A | Appendix A

## A.1. Quantum Projection Norm Error

We report here the extensive proof of Lemma 3.7.

*Proof.* We can say, following Lemma 2.15 and applying amplitude estimation of Lemma 2.10, that the norm of the solution vector of the OLS problem is estimated, considering a value $\overline{\eta} \in [-\eta, \eta]$ and $\eta \in (0, 1)$, as

$$\widetilde{\|A^+ s\|} = (\|A^+ |s\rangle\| + \overline{\eta}\|A^+ |s\rangle\|)\|s\| = \|A^+ s\| + \overline{\eta}\|A^+ s\|. \tag{A.1}$$

From the proof of Lemma 2.12, we use the following result

$$\|A |A^+ s\rangle\| \simeq \|A |\widetilde{A^+ s}\rangle\| \text{ if } \delta \ll \kappa \tag{A.2}$$

and we highlight that the value of $\delta$ can be decreased without a big impact on the running time because the increment related to it is just a polylogarithmic term and does not change the final complexity. We must also highlight that because of the Equation A.2 simplification, we can consider

$$\begin{aligned} \widetilde{\|AA^+ s\|} &= \widetilde{\|A|\widetilde{A^+ s}\rangle\|}\, \widetilde{\|A^+ |s\rangle\|}\, \|s\| \\ &= \|A |\widetilde{A^+ s}\rangle\|\, \widetilde{\|A^+ s\|} + \overline{\eta}\|A |\widetilde{A^+ s}\rangle\|\, \widetilde{\|A^+ s\|} \\ &\simeq \|A |A^+ s\rangle\|\, \widetilde{\|A^+ s\|} + \overline{\eta}\|A |A^+ s\rangle\|\, \widetilde{\|A^+ s\|} \end{aligned} \tag{A.3}$$

But the correct norm of the result is $\|AA^+ s\| = \|A_t |A^+ s\rangle\|\, \|A^+ s\|$.

Basically we know, starting from A.3 and substituting the result of A.1, that

$$\begin{aligned}\widetilde{\|AA^+s\|} &= \|A_t\,|A^+s\rangle\|\,(\|A^+s\| + \overline{\eta}\|A^+s\|) + \overline{\eta}\|A_t\,|A^+s\rangle\|\,(\|A^+s\| + \overline{\eta}\|A^+s\|) \\ &= (1+\overline{\eta})\|A_t\,|A^+s\rangle\|\,(1+\overline{\eta})\|A^+s\| \\ &= (1+\overline{\eta})^2\|AA^+s\|\end{aligned} \tag{A.4}$$

The error on the estimation of the norm, considering that $\overline{\eta}^2 \leq |\overline{\eta}| \leq \eta$, and, considering that we assume the worst-case for the error, is

$$\left|\widetilde{\|AA^+s\|} - \|AA^+s\|\right| \leq 3\eta\|AA^+s\| \tag{A.5}$$

□

## A.2. Classical Inner Product

In this section, we explain how it is possible to re-compute the inner products classically for the Single Error version of the Q-MP algorithm described in Section 4.1.2.

To re-compute the inner product classically, we must consider computing two different values as done quantumly. First of all, the inner product between the select atom $d_{j^*} \in \mathbb{R}^n$ and the signal $s \in \mathbb{R}^n$ can be done trivially in $O(n)$. If we consider repeating this process for $k$ iterations, we need $O(kn)$ total operations.

The second inner product, between $\phi_t \in \mathbb{R}^n$ and again the vector $d_{j^*}$, is less trivial because we do not have a classical representation of the approximated signal vector $\phi_t$ but only of the solution vector $x_t \in \mathbb{R}^m$. To obtain $\phi_t$ is possible to multiply the vector $x_t$ to the dictionary $D \in \mathbb{R}^{n\times m}$, but this step has a complexity of $O(nm)$. To avoid this high complexity, considering that at each step only one coefficient of the solution $x_t$ is updated, we can act more efficiently.

First of all, let us highlight that we can describe the vector $\phi_t$ as a linear combination of some atoms considering as coefficients of the combination the components of $x_t$. Let us denote the atom selected in the $i^{th}$ iteration as $d^{(i)}$ and its coefficient as $x^{(i)}$. We can describe $\phi_t$ as

$$\phi_t = x^{(1)}d^{(1)} + ... + x^{(t)}d^{(t)},\ i \in [1,t]. \tag{A.6}$$

For this reason, we can consider the inner product of $\phi_t$ with a newly selected atom $d_{j^*}$

as

$$\langle d_{j^*}, \phi_t \rangle = \langle d_{j^*}, x^{(1)} d^{(1)} + ... + x^{(t)} d^{(t)} \rangle = \sum_{i=1}^{t} x^{(i)} \langle d_{j^*}, d^{(i)} \rangle \tag{A.7}$$

for the properties of the inner product. At this point, we can consider the problem of creating the sum defined in Equation A.7 instead of the problem of computing the inner product using $\phi_t$.

We can store all the inner products between the atoms of the dictionary in a tree structure as a pre-computation step. In this way, access to the value of the inner product between two atoms is possible in time $O(\log(m^2))$.

We can also efficiently store all the coefficients $x^{(i)}$ found at each iteration of the algorithm in another tree data structure with an update and access time of $O(\log t)$.

At this point, to obtain the value of the inner product between the vector $\phi_t$ and the newly selected atom, it is enough to sum all the inner products of the already selected atoms to the newly selected one multiplied by the stored coefficients. This step is possible in $O(t(\log m^2 + \log t))$ considering accessing the data for each inner product and performing the sum.

We need an extra pre-computation time of $O(nm)$ and an extra space of $O(m(m+1)/2)$ but, after that, we can obtain the value of the sum, and equivalently of the inner product, with a bound of $\widetilde{O}(k)$ where $k$ is the number of iterations of the algorithm. In addition to that, we can use the precomputed inner products for different input signals. If we consider repeating this process for $k$ iterations, we need $\widetilde{O}(k^2)$ total operations.

The overall complexity of re-computing all the inner products is bounded by $\widetilde{O}(kn)$.

## A.3. Error Propagation of Double Error version

In this section, we explain the bound on the error required for the Double Error version of the Q-MP algorithm of Section 4.1.2.

We recall that $\|\phi_t\|\epsilon_\phi + \|\phi_t\|\eta + \widetilde{\|\phi_t\|}\epsilon_{in_2} \leq \epsilon_2$ for the Single Error version. It is possible to notice how this error is composed by a part that depends on the approximation of $\phi_t$ that we can define as $\epsilon_\alpha = \|\phi_t\|(\epsilon_\phi + \eta)$ and a part, the term $\widetilde{\|\phi_t\|}\epsilon_{in_2}$, related to the inner product procedure. In the case of an iteration-varying error in the solution $x \in \mathbb{R}^m$, the term $\epsilon_\alpha$ is the component that changes over time. For this reason, we define the iteration-varying version of the error $\epsilon_\alpha$ as $\epsilon_{\alpha_t}$.

Remembering that we want to bound the final error by $\epsilon_z$, the value needed to not miss-

select an atom, we can define an error $\epsilon_{3_t}$ as

$$\epsilon_{3_t} = \epsilon_1 + \epsilon_{2_t} = \|s\|\epsilon_{in_1} + \widetilde{\|\phi_t\|}\epsilon_{in_2} + \epsilon_{\alpha_t} \leq \epsilon_z \tag{A.8}$$

where $\epsilon_{2_t}$ is the iteration-varying version of the error $\epsilon_2$. The value $\epsilon_{3_t}$ also corresponds to the error that is added in a component of the solution $x$ at iteration $t$, considering that it is present in the result of the higher inner product estimated.

We now find a bound for the value $\epsilon_{3_t}$.

## Error on the approximated signal

First of all, we analyze how the error related to the approximated solution $\phi_t$ changes. We define the value $\epsilon_{\alpha_t}$ as

$$\left\|\phi_t - \widetilde{\phi_t}\right\| \leq \epsilon_{\alpha_t}, \tag{A.9}$$

that is the bound for the error between the vector $\phi_t$ and the approximated version $\widetilde{\phi_t}$ that we obtain during the algorithm.

Our starting point is

$$\left\||\widetilde{\phi_t}\rangle - |\overline{\phi_t}\rangle\right\| \leq \epsilon_\phi, \tag{A.10}$$

where we define $\overline{\phi_t}$ as the approximated value of $\phi_t$ considering that the solution vector $x_t$ has some error from the previous iterations. More formally, we define

$$\overline{\phi_t} = D(x_t + \vec{\epsilon_{3_{t-1}}}) = \phi_t + D\vec{\epsilon_{3_{t-1}}}, \tag{A.11}$$

where the vector $\vec{\epsilon_{3_{t-1}}}$ represents a vector such that, in the best case, each component of the vector contains the error of one iteration $\epsilon_{3_i}$ with $i \in [0, t-1]$. In the worst case, instead, all the errors summed together are present in one component of the error vector $\vec{\epsilon_{3_{t-1}}}$.

Using Claim 3.2, considering that $\left\||\widetilde{\phi_t}\rangle - |\overline{\phi_t}\rangle\right\| \leq \epsilon_\phi$ and that we can estimate the norm $\widetilde{\|\phi_t\|}$ such that $\left|\widetilde{\|\phi_t\|} - \|\overline{\phi_t}\|\right| \leq \|\overline{\phi_t}\|\eta$, we can say that

$$\left\|\widetilde{\phi_t} - \overline{\phi_t}\right\| \leq \|\overline{\phi_t}\|\epsilon_\phi + \|\overline{\phi_t}\|\eta. \tag{A.12}$$

Now, if we want to consider the error of our estimation not in relation to $\overline{\phi_t}$ but with the

correct value $\phi_t$ we obtain, thanks to the triangular inequality that

$$\left\|\widetilde{\phi_t} - \phi_t\right\| \leq \|\overline{\phi_t}\|\epsilon_\phi + \|\overline{\phi_t}\|\eta + \|D\epsilon_{3_{t-1}}^{\rightarrow}\|. \tag{A.13}$$

We can consider this value as the bound $\epsilon_{\alpha_t}$

$$\|\overline{\phi_t}\|\epsilon_\phi + \|\overline{\phi_t}\|\eta + \|D\epsilon_{3_{t-1}}^{\rightarrow}\| = \epsilon_{\alpha_t}. \tag{A.14}$$

If we substitute back this value in Equation A.8 we obtain

$$\epsilon_{3_t} = \|s\|\epsilon_{in_1} + \|\widetilde{\phi_t}\|\epsilon_{in_2} + \|\overline{\phi_t}\|\epsilon_\phi + \|\overline{\phi_t}\|\eta + \|D\epsilon_{3_{t-1}}^{\rightarrow}\|. \tag{A.15}$$

We can group the non-iteration-varying terms and consider $\|s\| \geq \|\phi_t\|$, $\|s\| \geq \|\overline{\phi_t}\|$ and $\|s\| \geq \|\widetilde{\phi_t}\|$ to obtain

$$\epsilon_{3_t} = \|s\|(\epsilon_{in_1} + \epsilon_{in_2} + \epsilon_\phi + \eta) + \|D\epsilon_{3_{t-1}}^{\rightarrow}\|. \tag{A.16}$$

We define the non-iteration-varying term $\epsilon_{cos} = \|s\|(\epsilon_{in_1} + \epsilon_{in_2} + \epsilon_\phi + \eta)$ to be more clear in this explanation

$$\epsilon_{3_t} = \epsilon_{cos} + \|D\epsilon_{3_{t-1}}^{\rightarrow}\|. \tag{A.17}$$

In this way, we have found a relationship between $\epsilon_{3_t}$, the value that is added at iteration $t$ in one component of the solution, and $\|D\epsilon_{3_{t-1}}^{\rightarrow}\|$. Now we must connect the value of this error vector to the error values $\epsilon_{3_i}$ with $i \in [0, t-1]$ of the previous iterations.

## Error on the solution

First of all, we must find what is the start condition of this error. It is easy to see that the error introduced in the first iteration is just the error of the first IPE procedure, so $\epsilon_1$. For this reason, we can say that $\epsilon_{3_0} = \epsilon_1$.

For the next step, we consider the worst case in which the algorithm selects the same atom at each update, and the error is added every time in the same component

$$\vec{\epsilon_{3_t}} = \sum_{i=0}^{t-1} \epsilon_{3_i} * e_c, \tag{A.18}$$

with a constant $c \in [m]$. If we compute the norm $\|D\vec{\epsilon_{3_t}}\|$ we obtain

$$\|D\vec{\epsilon_{3_t}}\| = \left\| D \sum_{i=0}^{t-1} \epsilon_{3_i} * e_c \right\| = \left\| \sum_{i=0}^{t-1} \epsilon_{3_i} d_c \right\| \leq \sum_{i=0}^{t-1} |\epsilon_{3_i}| \|d_c\| = \sum_{i=0}^{t-1} |\epsilon_{3_i}| \tag{A.19}$$

and considering that $\epsilon_{3_t} = \epsilon_{cos} + \|D\vec{\epsilon_{3_{t-1}}}\|$ we obtain

$$\epsilon_{3_t} \leq \epsilon_{cos} + \sum_{i=0}^{t-1} |\epsilon_{3_i}|, \text{ for } t \geq 1. \tag{A.20}$$

As already explained we consider $\epsilon_{3_0} = \epsilon_1$ and we obtain

$$\epsilon_{3_t} \leq \begin{cases} \epsilon_1 & t = 0 \\ 2^{(t-1)}(\epsilon_1 + \epsilon_{cos}) & t \geq 1 \end{cases} \tag{A.21}$$

For this reason, if we want that the maximum error on the single component will not be more than the value $\epsilon_z$, we need to know an estimate of the number of iterations.

Once we have fixed the value $L$, we can fix the values of the other errors as

$$2^{(L-1)}(\epsilon_1 + \epsilon_{cos}) \leq \epsilon_z \tag{A.22}$$

Finally, we obtain the bound

$$2\epsilon_{in_1} + \epsilon_{in_2} + \epsilon_\phi + \eta \leq \frac{\epsilon_z}{2^{(L-1)}\|s\|}. \tag{A.23}$$

# List of Figures

# List of Tables