Exploration of Quantum Algorithms for Singular Value Decomposition Problems

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Jezer Jojo

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2024

Supervisor: Dr. M. Girish Chandra

Certificate

This is to certify that this dissertation entitled Exploration of Quantum Algorithms for Singular Value Decomposition Problems towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Jezer Jojo at Indian Institute of Science Education and Research under the supervision of Dr. M. Girish Chandra, Principal Scientist, TCS Research and Innovation Lab , during the academic year 2023-2024.

Dr. M. Girish Chandra

Committee:

Dr. M. Girish Chandra

Professor M.S. Santhanam

This thesis is dedicated to my family -
To my keen-eyed and sharp-witted father, who has always had my utmost respect,
To my amiable mother, with her other-worldly ability to fill any room with mirth and laughter,
And to my incredibly hard-working sister, whose sense of ambition and drive inspires me more than she knows.

Declaration

I hereby declare that the matter embodied in the report entitled Exploration of Quantum Algorithms for Singular Value Decomposition Problems are the results of the work carried out by me at the TCS Research and Innovation Lab, Indian Institute of Science Education and Research, Pune, under the supervision of Dr. M. Girish Chandra and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

Jezer Jojo

Acknowledgments

I extend my deepest gratitude to my supervisor, Dr. M Girish Chandra, for his mentorship and aid throughout the course of this research, and for simply believing in me. His support and encouragement have been instrumental in shaping this thesis.

I am also indebted to my expert, Professor M.S. Santhanam, for his expertise and guidance throughout my time at IISER Pune. The projects and course I did under his tutelage have helped me significantly in my understanding of quantum computing.

I'm especially thankful to Ankit Khandelwal for his assistance, encouragement, and friendship. This thesis simply would not have been possible without his constant support and feedback.

I'd like to thank Tata Consultancy Services and the Indian Institute of Science Education and Research Pune for supporting my work and providing me the resources I needed to get through this journey.
I am extremely thankful for these 5 years I've spent as an IISER Pune student, and to the friends I've had the privilege of sharing this time with.
I'm grateful to my family whose love and patience have been a constant source of encouragement and support, and I'd also like to specifically thank my family members and friends in Bangalore who took care of me during this final year.

I am truly thankful to all those who have supported and inspired me along the way. Your collective efforts have played a significant role in the successful completion of this thesis, and for that, I am grateful[1].

[1] I'd also like to thank those of you who have taken the time to open this work. Yes, you. Thanks for stopping by!

Abstract

This thesis explores quantum algorithms for Singular Value Decomposition (SVD) problems, and focuses on both theoretical developments and practical implementations. First, a detailed description of a known purely quantum algorithm for SVD is provided, along with its application to Latent Semantic Analysis (LSA). Following this, we take a known variational quantum approach to SVD and identify a drawback in its objective function. A solution is presented in the form of a modification to this objective function and an application of this modified algorithm for LSA is also proposed. Comparative simulations between the original and modified algorithms are conducted, alongside experimental validation of the LSA algorithm on quantum hardware. We then study the Quantum Singular Value Transform (QSVT) and understand its relation to Quantum Signal Processing with the aid of an example. Various block encodings are explored, including a novel proposal. Simulations employing QSVT for solving linear systems and Topological Data Analysis are carried out for various block encodings and the results are presented. Finally, we extend our study to tensors, proposing both a purely quantum algorithm and a hybrid variational quantum algorithm to find the t-SVD of a third-order tensor. Simulations are conducted to validate their efficacy.

Contents

List of Figures

Introduction

The Singular Value Decomposition (SVD) is a form of matrix decomposition that finds many uses in a variety of fields. Its use is ubiquitous in the field of Data Science where the SVD is often used to reduce the dimensionality of input data [1]. This specific use of SVD finds itself in applications like Principal Component Analysis (PCA) and recommendation systems via matrix completion [2]. Finding the truncated SVD of an image is also known to serve as a method of image compression [3].
Quantum computing is a new paradigm of computing that makes use of the principles of quantum mechanics [4]. It is known to offer advantage over classical computers in certain problems such as unstructured search [5] and factoring primes [6].
The inherent linearity of quantum mechanics suggests the existence of efficient quantum algorithms to solve problems in linear algebra like computing the SVD of a matrix.

We begin by briefly going through a few preliminaries in Chapter 1. There we will describe the SVD in detail and then go through a brief introduction to quantum computing.

Chapter 2 then explores a purely quantum algorithm (as opposed to a hybrid algorithm) for finding a matrix's SVD, as described in [7, 8]. This method employs the use of a data structure with which a walk operator can be constructed whose eigendecomposition explicitly contains the SVD information of the given matrix. The main ideas behind this approach were fleshed out in [7] for the purpose of implementing a Quantum Recommendation System. Ref. [8] explored the use of these ideas for other applications in linear algebra. Chapter 2 of this work first describes this algorithm in enough detail for it to be directly implemented by the reader. The concluding section of this chapter presents the results obtained from running the algorithm on randomly generated matrices, as well as images of MNIST handwritten digits [9]. We then go through how this algorithm can be adapted for Latent Semantic Analysis. While an algorithm was proposed in [8] for the same purpose, their method involves applying tomography to retrieve vector representations of words which can then

be used however one wishes. This work proposes using a Hadamard test to compare words using quantum circuits to estimate their cosine similarity.
Other purely quantum approaches for SVD problems not explored in this thesis include the qPCA in [10] and a closely related quantum SVD algorithm in [11]. The algorithm in [10] involves the application of $e^{-iS\Delta t}$ on a target register and an auxiliary register containing a mixed state whose density matrix is given by A. Here S represents the SWAP operator. It is then demonstrated that this is approximately equivalent to applying $e^{-iA\Delta t}$ on the target register. With n copies of the auxiliary register, one can apply $e^{-iAn\Delta t}$ on the target register. Quantum Phase Estimation (QPE) can then be used to obtain the eigendecomposition of A. Ref. [11] employs a similar approach, but here the matrix A is not stored in the density matrix of a mixed state, but is embedded in the entries of a modified SWAP operator that is used in place of S. This work also puts forward the idea of using an 'extended matrix' $\tilde{A}$ whose eigendecomposition explicitly contains the SVD information of A, and can be retrieved through QPE.

Generally, these algorithms that use purely quantum approaches require large circuits with too many qubits to be feasibly implemented in the near term. On top of that, [12] dequantizes the algorithm in [7] and challenges its claim to exponential speedups for recommendation systems. This motivates us to look to variational circuits in the next chapter.

In Chapter 3, we discuss a hybrid variational algorithm [13] that computes the SVD of a given square matrix A. This algorithm uses both classical and quantum computers and requires less qubits than the purely quantum approach. This chapter proposes a modification to the algorithm in the form of a change to the objective function. The new objective function makes stronger theoretical guarantees than the existing one, and upon running the algorithm on the MNIST Handwritten Digits dataset, it heuristically appears to give better results for ansätze of lower depth. This chapter also puts forward a method to use this routine for the same LSA problem from Chapter 2. The algorithm is run on hardware for a simple LSA problem and results are presented. There are other variational quantum algorithms for SVD that this thesis doesn't explore, including [14] and [15]. The former employs a different approach from the algorithm we explore in this chapter; it takes the matrix A as an amplitude encoded state $|A\rangle$ and trains the ansätze U and V, applied on the row register and column register of $|A\rangle$ respectively, such that the correlation in their measured outputs is maximized. This effectively obtains the Schmidt decomposition of $|A\rangle$ which is equivalent to the SVD of A.

Chapter 4 then introduces and explains the Quantum Singular Value Transform [16]. The QSVT is a powerful framework that has many applications in a variety of problems, as demonstrated in [17]. This chapter begins by explaining how it works as a generalization of the Quantum Signal Processing procedure, and it demonstrates why this generalization holds with the help of a simple example and visualizations. Various known block encodings are then described and a new block encoding method is also proposed. The QSVT approach to solving linear systems of equations and a QSVT approach to Topological Data Analysis are described. These algorithms are simulated for different choices of block encodings and the results are presented.

In the final chapter, we introduce two new quantum t-SVD (tensor-SVD) [18] algorithms. The first algorithm is largely based on [19] which itself proposed a quantum t-SVD algorithm for recommendation systems. The new algorithm however seeks to address and fix certain drawbacks to the original, and is fundamentally different in its approach compared to the existing work. We discuss how it could be used to construct a certain type of recommendation system that is exponentially faster than any known classical algorithm that could do the same. The second algorithm proposed uses a hybrid variational approach largely based on the variational quantum SVD algorithm from Chapter 3, while making use of the new block encoding method proposed in Chapter 4.
A related algorithm that won't be explored in this thesis can be found in [20]. It also uses quantum computing to tackle a tensor generalization of SVD, namely the HOSVD.

Chapter 1

Preliminaries

We begin by introducing ourselves to a few important preliminary notions, definitions, and notations that will be used throughout this thesis. We start by defining and explaining the SVD of a given matrix, before then going into a primer on quantum computing.

1.1 Singular Value Decomposition

1.1.1 Definition

The SVD of a matrix A is a matrix decomposition of the following form:

$$A = U\Sigma V^\dagger \tag{1.1}$$

Here, U and V are unitary matrices and the i-th column of U (V), denoted u_i (v_i), is referred to as the i-th left (right) singular vector of A. Σ is a diagonal matrix whose i-th diagonal element, denoted σ_i, is referred to as the i-th singular value of A. The singular values of a matrix are positive and real.

1.1.2 Understanding the SVD

To understand what an SVD is, we observe that

$$Au_i = \sigma_i v_i. \tag{1.2}$$

In plain words, we can say that the SVD identifies a set of orthonormal vectors $\{v_i\}$ (called the right singular vectors) in the input space of A, and a set of orthonormal vectors $\{u_i\}$ (called the left singular vectors) in the output space of A, such that each right singular vector v_i maps to the corresponding left singular vector u_i scaled up by an associated factor σ_i called the singular value. To put it even more succinctly, the SVD of a matrix A identifies a set of orthonormal vectors spanning the input space of A such that A maps them to orthogonal vectors in the output space of A.

The SVD can be seen as a generalization of the eigendecomposition and they are equivalent for positive semidefinite matrices. Unlike the eigendecomposition, every matrix (not necessarily square) has an SVD.

The number of non-zero singular values of a matrix is called its rank, which we denote as T.

The Frobenius norm of a matrix A is given by

$$||A||_F = \sqrt{\sum_i^T \sigma_i^2} = \sqrt{\sum_{i,j} |a_{ij}|^2}. \tag{1.3}$$

1.1.3 Flexibility of SVD representation

The SVD of a matrix is not unique. There are multiple ways of factoring a matrix in this form. For example, given some fixed permutation $i \to j$, we can reorder the columns of U and V to get U' and V' whose i-th columns are $u'_i = u_j$ and $v'_i = v_j$ respectively, and reorder the diagonal elements of Σ to get Σ' such that $\Sigma'_{ii} = \Sigma_{jj}$. And this will still be a valid SVD of A:

$$A = U'\Sigma'V'^{\dagger} \tag{1.4}$$

This degree of freedom can be standardized by picking an ordering such that $\sigma_i \geq \sigma_j$ if $i < j$. This does not however tame all our degrees of freedom. Our matrix could have multiple singular values that are equal, in which case the associated left singular vectors can be replaced by any other set of orthonormal singular vectors that span the same space, as long as the corresponding right singular vectors are also modified accordingly. Another degree of freedom we possess is that we can replace any pair of columns u_i and v_i with $e^{i\theta}u_i$ and $e^{i\theta}v_i$.

1.2 Quantum Computing

1.2.1 Representing states

Just as classical computers use bits that can assume a 0 state or a 1 state, quantum computers use qubits that can be measured in one of two quantum states - $|0\rangle$ and $|1\rangle$. Before measurement however, a qubit can also be in a linear combination (or 'superposition') of these states (over the field of complex numbers). So the state of a qubit will have the form:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.5}$$

The coefficient associated with one of these two 'computational basis states' is called its 'probability amplitude'. This is because the norm squared of this value is the probability of measuring the associated basis state. This means that

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.6}$$

When dealing with multiple qubits or a 'register' of qubits, each possible outcome upon measuring all the qubits will have an associated probability amplitude. If we have n qubits, each of which can be measured to be $|0\rangle$ or $|1\rangle$, then we will have 2^n possible outcomes. The combined state of n qubits then will have the form:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i|i\rangle, \tag{1.7}$$

such that

$$\sum_{i=0}^{2^n-1} |c_i|^2 = 1, c_i \in \mathbb{C}. \tag{1.8}$$

Here, $|i\rangle$ represents the state where the n qubits assume computational basis states whose bit representations make up the n-bit binary representation of i. So, if we have three qubits in the states $|0\rangle$, $|1\rangle$, and $|0\rangle$; we represent their combined state as $|2\rangle$, since 010 is the binary representation of 2. We can use Two's Complement representation for negative numbers just as we do in classical computing.

To represent a general quantum state for n qubits, as seen in (1.7), we use a vector whose i-th component is c_i - we call this a 'statevector'. The constraint given in (1.8) tells us that a statevector will have a norm of 1.

Given a vector v, $|v\rangle$ will denote the quantum state whose statevector is v (or $v/||v||$, if v is unnormalized).

In this work we sometimes use $|\ldots\rangle$ to represent a general normalized state when we don't want to explicitly label or describe the state specifically. We use this for error terms or states we will end up throwing away through postselection.

Given two registers of qubits with states $|\psi_1\rangle$ and $|\psi_2\rangle$, the combined state of both these registers $|\Psi\rangle$ is given by their tensor product:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \tag{1.9}$$

1.2.2 Representating operations

We've seen how we can represent the state of our qubits; but if we want to do computations, we'll have to study how we can apply operations on these qubits to change their state.

In quantum mechanics, the operations that can be applied to a quantum state have to be linear. This means any operation that we can apply on our qubits can be represented as a matrix O, such that the final state of our qubits is represented by a statevector $v_f = Ov_i$, where v_i is the initial statevector of our qubits.

We also observe that a valid statevector has to satisfy (1.8), and so a valid operation has to be one that preserves the norm of any qubit state it may act on. Matrices that are norm-preserving are

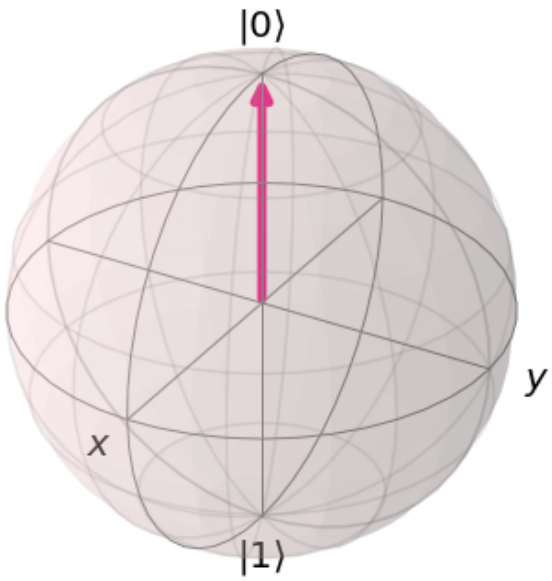


Figure 1.1: Bloch sphere

called ‘unitary matrices’, and these are the matrices that represent valid operations.
There are a few other terms that are almost synonymous to ‘quantum operator’ that we should acquaint ourselves with. The word ‘gate’ is used - in the same vein as in ‘NOT gate’ and ‘OR gate’ in classical computing - to refer to an operation that is elementary. ‘Quantum operator’ is a term from quantum mechanics which, for our purposes, will be used synonymously with ‘quantum operation’, though generally a quantum operator does not have to refer to a valid unitary operation. A ‘quantum circuit’ is a set of qubits and a sequence of gates applied to those qubits. A ‘quantum algorithm’ simply refers to an algorithm that makes use of quantum circuits in its procedure.

1.2.3 Bloch sphere

Any single-qubit quantum state

$$|\psi(\theta,\phi)\rangle = \cos(\frac{\theta}{2})|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})|1\rangle \tag{1.10}$$

can be mapped onto a unit sphere onto the point $(1,\theta,\phi)$ in polar coordinates. This sphere whose surface contains all valid single-qubit quantum states is called the ‘Bloch sphere’ as depicted in Fig. 1.1.

Single-qubit operators can be visualized as rotations on the Bloch sphere. The $RX(\delta)$, $RY(\delta)$, and $RZ(\delta)$ gates for example are named as they are because they represent rotations around the X, Y, and Z axes respectively by an angle of δ.

1.2.4 A few well known gates and operations

Here, we will go through a few well known quantum gates and operations that will be used or referred to in this work.

- $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
- $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
- $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- $RX(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$
- $RY(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$
- $RZ(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$
- The CNOT gate is a “controlled NOT”. It acts on two qubits - a ‘control’ and a ‘target’. A NOT gate is applied on the target qubit if the control qubit is in the $|1\rangle$ state. If the control

qubit is in the $|0\rangle$ state, nothing happens.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.11}$$

When we say an operator U is controlled on the $|x\rangle$ state of a qubit or a register of qubits, we mean that U is applied only when the qubit/register is in the $|x\rangle$ state and that it is not applied if the qubit/register is in a state orthogonal to $|x\rangle$.

- The Quantum Fourier Transform, often denoted QFT, applies an inverse Discrete Fourier Transform on the statevector of the given register of n qubits.

$$QFT|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n}xy}|y\rangle \tag{1.12}$$

- Quantum Phase Estimation (QPE) is an algorithm that makes use of a given operator U controlled on a single qubit and can be applied to an eigenstate of U, $|\psi\rangle$ to retrieve the associated eigenphase θ as a basis state in a separate register of n qubits which we will often refer to as the 'phase register'.

$$QPE(U)|\psi\rangle|0\rangle = |\psi\rangle|\frac{2^n}{2\pi}\theta\rangle, \tag{1.13}$$

where $U|\psi\rangle = e^{i\theta}|\psi\rangle$.

- The Hadamard test is an algorithm used to compare two states $|\psi\rangle$ and $|\phi\rangle$ and retrieve the value $\mathrm{Re}(\langle\phi|\psi\rangle)$. The way it works is we use a quantum circuit to prepare the state

$$|\Psi\rangle = \frac{1}{2}\left(|0\rangle(|\psi\rangle + |\phi\rangle) + |1\rangle(|\psi\rangle - |\phi\rangle)\right). \tag{1.14}$$

Having prepared this state, we measure the first qubit and estimate its expectation value (unless stated otherwise, this way of phrase refers to the expectation value of the Z operator

applied on the given qubit(s)):

$$\Pr(|0\rangle) - \Pr(|1\rangle) \tag{1.15}$$

$$= \frac{1}{4}\left((\langle\psi| + \langle\phi|)(|\psi\rangle + |\phi\rangle) - (\langle\psi| - \langle\phi|)(|\psi\rangle - |\phi\rangle)\right) \tag{1.16}$$

$$= \frac{1}{4}(2\langle\psi|\phi\rangle + 2\langle\phi|\psi\rangle) \tag{1.17}$$

$$= \mathrm{Re}(\langle\phi|\psi\rangle) \tag{1.18}$$

1.2.5 Jordan's Lemma

Jordan's Lemma [21] is an important result that is used to study the effect of applying two reflection operators $2\Delta - 1$ and $2\Pi - 1$ one after the other. The result is as follows:

Given projection operators (operators whose each eigenvalue is either a 0 or a 1) Δ and Π, the eigenvectors of $\Delta\Pi\Delta$, with eigenvalue 0 or 1, correspond to 1D invariant subspaces on which $(2\Pi - 1)(2\Delta - 1)$ is just ± 1.

An eigenvector $|v\rangle$ of $\Delta\Pi\Delta$ with eigenvalue $\lambda \in (0,1)$ corresponds to a 2D subspace spanned by $|v\rangle$ and

$$|v^{\perp}\rangle := \frac{(1-\Delta)\Pi|v\rangle}{||(1-\Delta)\Pi|v\rangle||}. \tag{1.19}$$

$\Pi|v\rangle$, when normalized, can be written as

$$\frac{\Pi|v\rangle}{||\Pi|v\rangle||} = \cos\frac{\theta}{2}|v\rangle + \sin\frac{\theta}{2}|v^{\perp}\rangle, \tag{1.20}$$

where

$$\cos(\frac{\theta}{2}) = \sqrt{\lambda}. \tag{1.21}$$

Associated with every such eigenvector $|v\rangle$, are two eigenvectors of $(2\Pi - 1)(2\Delta - 1)$, $|\Psi_{\pm}\rangle$

given by

$$|\Psi_{\pm}\rangle = \frac{|v\rangle \mp i|v^{\perp}\rangle}{\sqrt{2}}, \tag{1.22}$$

with corresponding eigenvalues $e^{\pm i\theta}$.

Chapter 2

Purely quantum algorithm for Singular Value Decomposition

In this chapter we acquaint ourselves with an algorithm based on the Quantum Singular Value Estimation (QSVE) routine introduced in [7] and further fleshed out for various other applications in [8]. We first go through a detailed step-by-step description of two quantum circuits. The first of these two circuits uses this QSVE approach to output a final quantum state which, if read, gives a full description of the singular value decomposition of a given matrix. The second circuit is based on the first, and is used to construct a quantum algorithm that tackles a problem in Latent Semantic Analysis (LSA). This algorithm has a time complexity that is polylogarithmic in the size of the input matrix.

2.1 Notation

- A is an $m \times n$ matrix we want the SVD of.
- a_i is the i-th row of A written out as a vector.
- a_{ij} is the j-th element of a_i, or the entry on the i-th row and j-th column of A.
- $\tilde{A}$ is a vector whose i-th element is $|a_i|$.
- $||A||_F$ is the Frobenius norm of A.

- σ_i, u_i, and v_i are the i-th singular value, left singular vector, and right singular vector respectively. We also define $\theta_i := 2\arccos\left(\frac{\sigma_i}{||A||_F}\right)$

2.2 Data structure requirements

To run this algorithm, we need to input the matrix we want the SVD of. To do so, we assume we possess the ability to apply operators P' and Q', where

$$P'|i\rangle|0\rangle = \frac{1}{|a_i|}|i\rangle|a_i\rangle, \tag{2.1}$$

$$Q'|0\rangle|j\rangle = \frac{1}{||A||_F}|\tilde{A}\rangle|j\rangle. \tag{2.2}$$

Appendix A of [7] proposes a data structure that allows for the efficient application of these operators.

2.3 Theoretical underpinnings

The strategy behind this algorithm is to construct an operator W whose eigendecomposition contains information about our matrix's SVD. Then we will use quantum phase estimation to extract that information out.

W can be written as the product of two reflection operators U and V

$$W = UV \tag{2.3}$$

where $U = 2PP^T - \mathbb{1}_{mn}$ and $V = 2QQ^T - \mathbb{1}_{mn}$.

And P and Q are defined as follows:

$$P := \sum_i \frac{1}{|a_i|}(|i\rangle|a_i\rangle)(\langle i|) \tag{2.4}$$

$$Q := \frac{1}{||A||_F}\sum_j (|\tilde{A}\rangle|j\rangle)(\langle j|) \tag{2.5}$$

What's special about this construction is that P and Q have the following properties:

$$P^T P = \mathbb{1} \tag{2.6}$$

$$Q^T Q = \mathbb{1} \tag{2.7}$$

$$P^T Q = |0\rangle\langle 0| \otimes \frac{A}{||A||_F} \tag{2.8}$$

$U = 2PP^T - \mathbb{1}_{mn}$ can be practically realized by

$$U = P'(2|0\rangle\langle 0| - \mathbb{1})P'^{\dagger} \tag{2.9}$$

$V = 2QQ^T - \mathbb{1}_{mn}$ can be practically realized by

$$V = Q'(2|0\rangle\langle 0| - \mathbb{1})Q'^{\dagger} \tag{2.10}$$

Now that we know what W is, we will use Jordan's lemma (Section 1.2.5) to learn about its eigen-decomposition. Setting $\Delta = QQ^T$ and $\Pi = PP^T$, it can be seen that $Q|v_i\rangle|0\rangle$ is an eigenvector of $\Delta\Pi\Delta = QQ^T PP^T QQ^T$ with eigenvalue $\lambda = \frac{\sigma^2}{||A||_F^2}$.

Jordan's lemma would then tell us that the eigenvectors of W are given by

$$|\Psi_{\pm}\rangle = \frac{Q|v_i\rangle|0\rangle \mp i(Q|v_i\rangle|0\rangle)^{\perp}}{\sqrt{2}} \tag{2.11}$$

with eigenvalues $e^{\pm i\theta_i}$, where $\cos\frac{\theta_i}{2} = \frac{\sigma_i}{||A||_F}$.

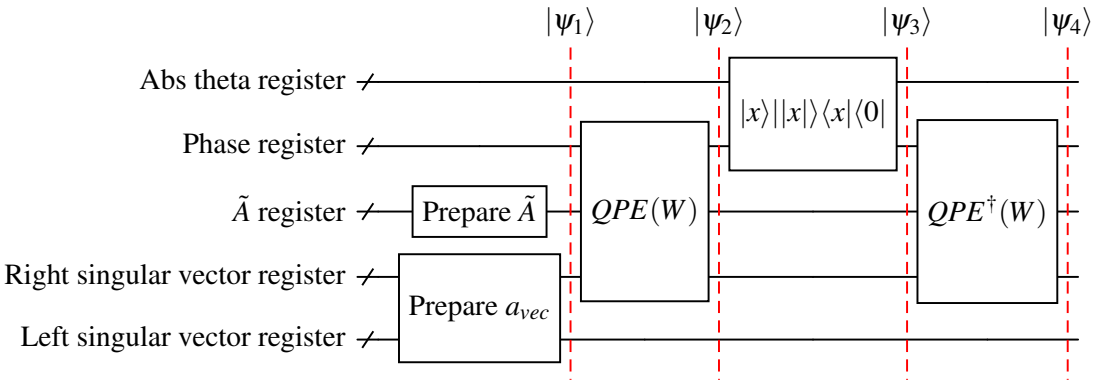


Figure 2.1: Circuit diagram of the quantum algorithm for full SVD

2.4 Getting the full SVD

Now that we have an operator W whose eigendecomposition contains information about the SVD of A, we will see how we can extract that information. We describe a quantum circuit defined in [8], that can output the following state:

$$|\psi\rangle = \sum_i \frac{\sigma_i}{||A||_F} |u_i\rangle|v_i\rangle|\sigma_i\rangle \tag{2.12}$$

While implementing this algorithm is not unfeasible, in practice it is simpler to implement one that returns the following state instead

$$|\psi\rangle = \sum_i \frac{\sigma_i}{||A||_F} |u_i\rangle|v_i\rangle|\theta_i\rangle \tag{2.13}$$

Getting σ_i from θ_i can be done classically after the measurements. This circuit is depicted in Fig. 2.1.

2.4.1 Circuit registers

Our circuit has 5 registers:

1. Left singular vector register (lsv)
2. Right singular vector register (rsv)
3. $\tilde{A}$ register (a)
4. Phase register (p)
5. Abs theta register (t)

2.4.2 Algorithm procedure

Step 1a: We begin with our first step of preparing the initial state

$$|\psi_1\rangle_{lsv,rsv} = |a_{vec}\rangle = PQ|0\rangle_{lsv}|0\rangle_{rsv} \tag{2.14}$$

$$= \frac{1}{||A||_F}\sum_{ij} a_{ij}|i\rangle|j\rangle = \sum_i \frac{\sigma_i}{||A||_F}|u_i\rangle|v_i\rangle \tag{2.15}$$

Step 1b: At the same time, we apply Q to prepare the $|\tilde{A}\rangle$ state in the $\tilde{A}$ register.

$$|\psi_1\rangle = \sum_i \frac{\sigma_i}{||A||_F^2}|u_i\rangle_{lsv}|\tilde{A}\rangle_a|v_i\rangle_{rsv} \tag{2.16}$$

$$= \sum_i \frac{\sigma_i}{||A||_F^2}|u_i\rangle_{lsv}(Q|0\rangle_a|v_i\rangle_{rsv}) \tag{2.17}$$

$$= \sum_i \frac{\sigma_i}{||A||_F^2}|u_i\rangle_{lsv}(\frac{|\Psi_{i+}\rangle + |\Psi_{i-}\rangle}{\sqrt{2}})_{a,rsv} \tag{2.18}$$

Step 2: Next we apply Quantum Phase Estimation with respect to W on the $\tilde{A}$ and Right singular vector registers, and store the eigenphases in the Phase register.

$$|\psi_2\rangle = \sum_i \frac{\sigma_i}{||A||_F^2}|u_i\rangle_{lsv}\left(\frac{|\Psi_{i+}\rangle|\theta_i\rangle_p + |\Psi_{i-}\rangle|-\theta_i\rangle_p}{\sqrt{2}}\right) \tag{2.19}$$

Step 3: Then we use an operator that takes $|x\rangle|0\rangle$ to $|x\rangle||x|\rangle$. We construct such an operator by first applying Toffoli gates to copy the first register onto the second if the most significant qubit is $|0\rangle$ (i.e., if $x \geq 0$, the second register would be $|x\rangle = ||x|\rangle$).
Then we use a quantum fourier transform adder [22] to subtract the first register from the second if the most significant qubit is $|1\rangle$ (i.e., if $x < 0$, the second register would be $|0-x\rangle = |-x\rangle = ||x|\rangle$). This operator is applied on the Phase register and the Abs Theta register to give us the state

$$|\psi_3\rangle = \sum_i \frac{\sigma_i}{||A||_F^2} |u_i\rangle_{lsv} \left(\frac{|\Psi_{i+}\rangle|\theta_i\rangle_p|\theta_i\rangle_t + |\Psi_{i-}\rangle|-\theta_i\rangle_p|\theta_i\rangle_t}{\sqrt{2}} \right). \tag{2.20}$$

Step 4: Finally, we undo the QPE step to get our final output state

$$|\psi_4\rangle = \sum_i \frac{\sigma_i}{||A||_F^2} |u_i\rangle_{lsv}|\tilde{A}\rangle_a|v_i\rangle_{rsv}|\theta_i\rangle_t. \tag{2.21}$$

The $\tilde{A}$ register is unentangled with the other qubits and can be ignored to get the promised state

$$|\psi_4\rangle_{lsv,rsv,t} = \sum_i \frac{\sigma_i}{||A||_F} |u_i\rangle_{lsv}|v_i\rangle_{rsv}|\theta_i\rangle_t. \tag{2.22}$$

2.5 Application in Latent Semantic Analysis

2.5.1 Latent Semantic Analysis

Latent Semantic Analysis is a method to study relations between words by analyzing their frequency of occurrence in various documents from a large enough corpus. We begin this section by describing how it works.

Given N documents and M unique words, we can construct an $N \times M$ "Term-Document matrix" A whose (m,n)-th element is the number of occurrences of the word m in document n. Let the singular value decomposition of A be given by $A = U\Sigma V^\dagger$. Let $A^{\geq\tau}$ represent the truncated version of A, given by $U\Sigma^{\geq\tau}V^\dagger$, where $\Sigma^{\geq\tau}$ is obtained by setting to 0 all entries of Σ that are less than τ.

The cosine similarity between the i-th and j-th rows of $A^{\geq\tau}$ is used as a metric to gauge the similarity of those words. The cosine similarity between two vectors $|u\rangle$ and $|v\rangle$ is defined as

$\frac{\langle u|v\rangle}{\sqrt{\langle u|u\rangle\langle v|v\rangle}}$. This technique is called Latent Semantic Analysis [23]. For $\tau = 0$ (trivial case with no truncation), this gauges the co-occurrence of the given words in the set of documents. Choosing a smaller value of τ, given a large enough corpus, would possibly reveal higher order relations between words even if they don't occur together in many documents. In this way, LSA can be seen as a noise reduction technique [24]. Our Term-Document matrix can be seen as a noisy approximation of an underlying matrix whose each entry measures how semantically similar the entry's associated term is to the entry's document, and truncating the Term-Document matrix can be seen as a method of reducing the noise to obtain a better approximation to this underlying matrix.

This section describes a way to use the QSVE approach for an application in latent semantic analysis. We construct a circuit that uses QSVE with respect to the given Term-Document (entered into the circuit as discussed in Section 2.2) to get the inner product of $(A^{\geq\tau})^T|i\rangle$ and $(A^{\geq\tau})^T|j\rangle$. The output is retrieved as the expectation value of a certain 'output qubit' in the circuit. To get the cosine similarity between these two vectors, we also construct a simpler circuit to estimate the norm of $(A^{\geq\tau})^T|l\rangle$ for a given l.

2.5.2 Circuit to retrieve $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle$

This circuit is depicted in Fig. 2.2.

It makes use of 6 registers:

1. Input register (inp)
2. Right singular vector register (rsv)
3. $\tilde{A}$ register (a)
4. Phase register (p)
5. Truncation qubit $(trunc)$
6. Output qubit (out)

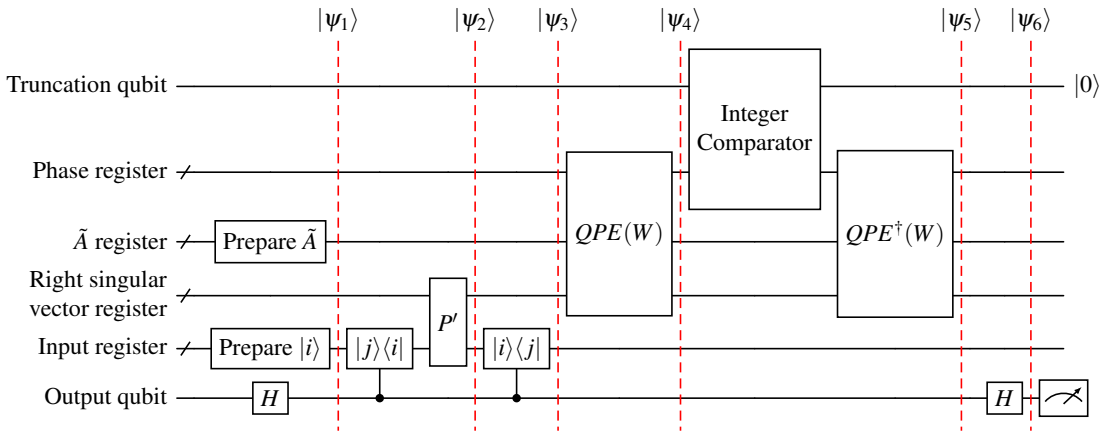


Figure 2.2: Circuit diagram of the purely quantum algorithm for LSA

2.5.3 Algorithm procedure

Step 1: First we apply a Hadamard gate to the output qubit to get the state

$$|\psi_1\rangle_{out} = \frac{1}{\sqrt{2}}(|0\rangle_{out} + |1\rangle_{out}) \tag{2.23}$$

Step 2: We prepare basis states $|i\rangle$ and $|j\rangle$ on the input register, controlled on the $|0\rangle$ state and the $|1\rangle$ state of the output qubit respectively. And then we apply P' on the input register and the right singular vector register to retrieve the corresponding rows of A.

$$|\psi_2\rangle_{out,inp,rsv} = \frac{1}{\sqrt{2}}(|0\rangle_{out}|i\rangle_{inp}A^T|i\rangle_{rsv} + |1\rangle_{out}|j\rangle_{inp}A^T|j\rangle_{rsv}) \tag{2.24}$$

Step 3: Now we undo our basis state preparation step to set the input register back to the $|0\rangle$ state (or any other state as long as we unentangle it from our other qubits).

$$|\psi_3\rangle_{out,rsv} = \frac{1}{\sqrt{2}}(|0\rangle_{out}A^T|i\rangle_{rsv} + |1\rangle_{out}A^T|j\rangle_{rsv}) \tag{2.25}$$

Step 4: Now we need to apply QSVE on the right singular vector register. To do this, we first apply Q' to prepare $|\tilde{A}\rangle$ on the $\tilde{A}$ register and then we apply QPE with respect to W on the $\tilde{A}$ register and the right singular vector register. This gives us the state

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}|0\rangle_{out}\sum_l\left(\sigma_l\left(\frac{|\Psi_{l+}\rangle_{a,rsv}|\theta_l\rangle_p + |\Psi_{l-}\rangle_{a,rsv}|-\theta_l\rangle_p}{\sqrt{2}}\right)\langle l|U^\dagger|i\rangle\right) + \frac{1}{\sqrt{2}}|1\rangle_{out}\sum_l\left(\sigma_l\left(\frac{|\Psi_{l+}\rangle_{a,rsv}|\theta_l\rangle_p + |\Psi_{l-}\rangle_{a,rsv}|-\theta_l\rangle_p}{\sqrt{2}}\right)\langle l|U^\dagger|j\rangle\right) \tag{2.26}$$

Step 5: Our next step is truncation. We use an integer comparator [25] to flip the first truncation qubit if the phase register carries a basis state less than $2\arccos(\tau)$, and then again for basis states less than $2\pi - 2\arccos(\tau)$. This has the effect of flipping the truncation qubit when $\theta \in (2\arccos(\tau), 2\pi - 2\arccos(\tau))$, i.e., when $\sigma < \tau$. We then undo QPE to get the following state.

$$|\psi_5\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_{out}\sum_{l,\sigma_l\geq\tau}\left(\sigma_l V|l\rangle\langle l|U^\dagger|i\rangle\right) + |1\rangle_{out}\sum_{l,\sigma_l\geq\tau}\left(\sigma_l V|l\rangle\langle l|U^\dagger|j\rangle\right)\right)|0\rangle_{trunc} + |\ldots\rangle|1\rangle_{trunc} \tag{2.27}$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle_{out}(A^{\geq\tau})^T|i\rangle + |1\rangle_{out}(A^{\geq\tau})^T|j\rangle\right)|0\rangle_{trunc} + |\ldots\rangle|1\rangle_{trunc} \tag{2.28}$$

Step 6: Finally, we apply another Hadamard gate to the output qubit and measure its expectation value.

$$|\psi_6\rangle_{out,inp} = \frac{1}{2}\left(|0\rangle_{out}\left((A^{\geq\tau})^T|i\rangle + (A^{\geq\tau})^T|j\rangle\right) + |1\rangle_{out}\left((A^{\geq\tau})^T|i\rangle - (A^{\geq\tau})^T|j\rangle\right)\right|0\rangle_{trunc} + |\ldots\rangle|1\rangle_{trunc} \tag{2.29}$$

We can now measure the output and truncation qubits to estimate $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle$ using the following relation.

$$\Pr(|0\rangle_{out}|0\rangle_{trunc}) - \Pr(|1\rangle_{out}|0\rangle_{trunc}) = \langle i|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle \tag{2.30}$$

2.5.4 Circuit to retrieve $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|i\rangle$

We use essentially the same circuit to retrieve $\langle i|(A^{\geq\tau})^T A^{\geq\tau}|i\rangle$ and $\langle j|(A^{\geq\tau})^T A^{\geq\tau}|j\rangle$, except we don't need a Hadamard test.

So this circuit makes use of 5 registers:

1. Input register (inp)
2. Right singular vector register (rsv)
3. $\tilde{A}$ register (a)
4. Phase register (p)
5. Truncation qubit $(trunc)$

Step 1: We first prepare the basis state $|i\rangle$ on the input register and then apply P' on the input register and the right singular value register to get the following state.

$$|\psi_1\rangle = |i\rangle_{inp} A^T |i\rangle_{rsv} \tag{2.31}$$

The input register can be ignored from here on out.

Step 2: Then we apply steps 4 and 5 from the previous circuit to get the final state.

$$|\psi_2\rangle = (A^{\geq\tau})^T |i\rangle |0\rangle_{trunc} + |\ldots\rangle |1\rangle_{trunc} \tag{2.32}$$

From here we can measure the truncation qubit and estimate the probability of observing the $|0\rangle$ state. This gives us the quantity we were looking for, $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|i\rangle$.

2.5.5 Putting it all together

With LSA, we measure the similarity of two words i and j as the cosine similarity of their corresponding rows in $A^{\geq\tau}$.

$$\text{Cosine Similarity} = \frac{\langle i|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle}{\sqrt{\langle i|A^{\geq\tau}(A^{\geq\tau})^T|i\rangle\langle j|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle}} \tag{2.33}$$

We can use the first circuit we described to estimate $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle$, and then the second circuit to estimate $\langle i|A^{\geq\tau}(A^{\geq\tau})^T|i\rangle$ and $\langle j|A^{\geq\tau}(A^{\geq\tau})^T|j\rangle$. We can obtain the required cosine similarity by putting these values in (2.33). The depths of these circuits are polylogarithmic in the size of A, and the number of times these circuits need to be run does not scale with the size of A. Therefore, the time complexity of this algorithm is polylogarithmic in the size of the input matrix A.

2.6 Simulations

Simulations were run for the first algorithm on hundred 4×4 random matrices. The entries of each matrix are sampled from a uniform distribution after which each matrix is normalized by dividing it by its Frobenius norm. The algorithm is run using PennyLane [26] on each of these matrices to produce its SVD, which is then used to try and reconstruct the original matrix. The error is then measured as the Frobenius distance between the original matrix and the reconstruction. This is done multiple times for different numbers of phase qubits. Fig. 2.3 depicts the average error vs the number of phase qubits used in the circuit.

The same experiment is then run on hundred 8×8 matrices to produce the Fig. 2.4, and then on hundred 8×8 images taken from the MNIST hand-drawn digits database to produce Fig. 2.5.

In all three figures, the average error decreases when we use more qubits in the phase register. We see a tradeoff between the accuracy of the result against the depth and number of qubits used in the circuit, which is not unexpected. By comparing Fig. 2.4 and Fig. 2.5, we see that while the average errors in both figures are around 0.5 when we have 2 phase register qubits, the error in the MNIST case decreases a lot more rapidly, going below 0.1 at 7 qubits, while for the randomly

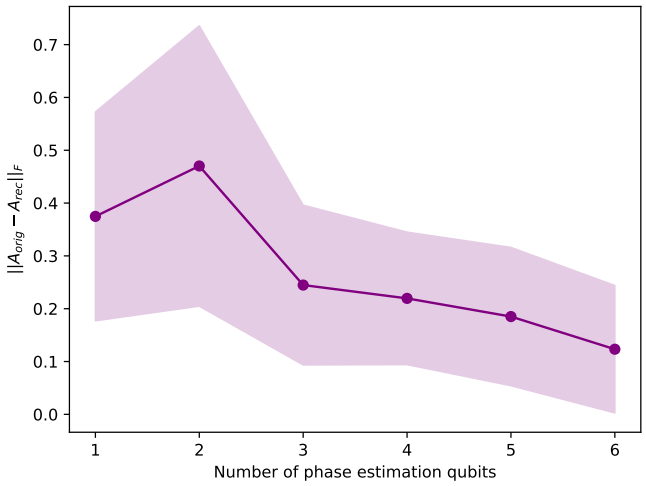


Figure 2.3: Quantum SVD results on hundred 4×4 randomly generated matrices. The shaded region above and below a data point depicts ± 1 standard deviation around the mean.

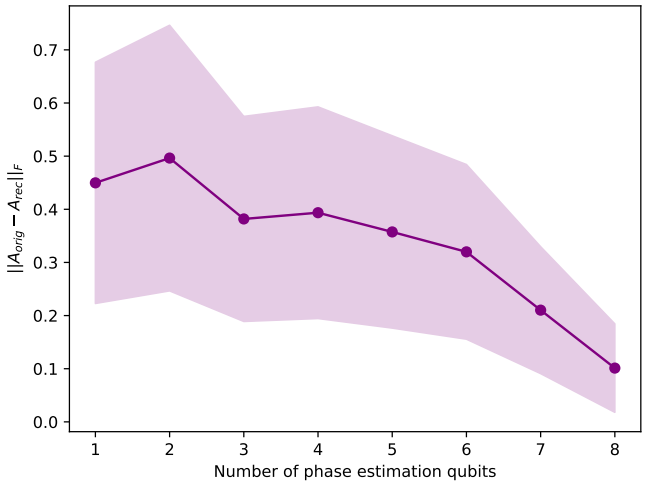


Figure 2.4: Quantum SVD results on hundred 8×8 randomly generated matrices. The shaded region above and below a data point depicts ± 1 standard deviation around the mean.

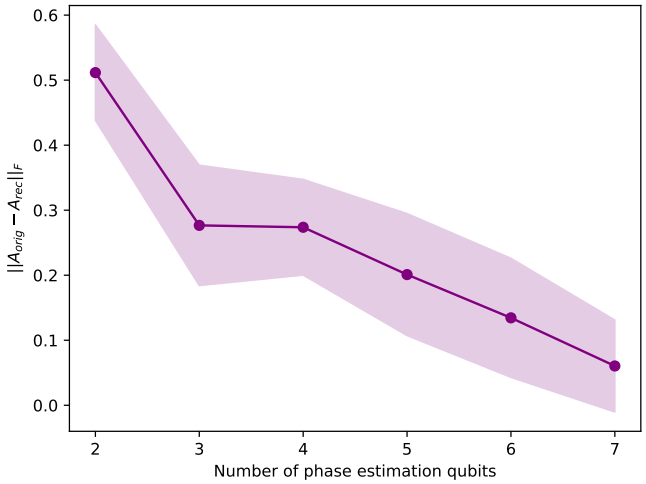


Figure 2.5: Quantum SVD results on hundred 8×8 images from MNIST dataset. The shaded region above and below a data point depicts ± 1 standard deviation around the mean.

generated matrices the average error is above 0.2 for 7 qubits. This indicates that the significant singular values are easier to single out through QPE for the MNIST images than for the randomly generated matrices (the larger singular values are more spread out from each other and from other smaller singular values in the MNIST images than they are in the randomly generated matrices).

See Section 6.1 for further discussion and future works.

Chapter 3

Variational Algorithm for Singular Value Decomposition

This chapter is based on a publication made by the author and collaborators during the writing of this thesis [29].

3.1 Notation

- Let A be a $2^n \times 2^n$ square matrix that we want the SVD of.
- The SVD of A is given by $A = U\Sigma V^\dagger$ where U and V are unitary matrices and Σ is a diagonal matrix of positive real values arranged in descending order.
- Let $\sigma_i = \Sigma_{ii}$ denote the i-th largest singular value of A.
- Just as $A = \sum_i^{2^n} \sigma_i U|i\rangle\langle i|V^\dagger$, we can define the truncated matrix, $A_T^{trunc} := \sum_i^T \sigma_i U|i\rangle\langle i|V^\dagger$.

3.2 Overview of the algorithm

This section describes the general strategy behind the existing algorithm proposed in [13]. This description is laid out in a way that makes it convenient to frame modifications that will be made later.

The algorithm takes the following inputs

1. A $2^n \times 2^n$ matrix A
2. A positive integer $1 \leq T \leq 2^n$

If the matrix we want the SVD of is not $2^n \times 2^n$, we can always pad it with zeros to make it so.

It then returns quantum circuits $\tilde{U}$ and $\tilde{V}$ along with a sequence of numbers $\tilde{\sigma}_1 \ldots \tilde{\sigma}_T$ such that the following holds:

$$A_T^{rec} := \sum_i^T \tilde{\sigma}_i \tilde{U}|i\rangle\langle i|\tilde{V}^\dagger \approx \sum_i^T \sigma_i U|i\rangle\langle i|V^\dagger \tag{3.1}$$

The way the algorithm does this is by having an ansatz W and two sets of parameters $\vec{\alpha}$ and $\vec{\beta}$ which are optimized using some objective function (which is elaborated on in the next section). This objective function is crafted such that for optimized parameters $\vec{\alpha}^*$ and $\vec{\beta}^*$, $W(\vec{\alpha}^*) = \tilde{U}$ and $W(\vec{\beta}^*) = \tilde{V}$. The quality of the approximation in (3.1) is dependent on how well this optimization process goes. To then get $\tilde{\sigma}_i$, we evaluate $\tilde{\sigma}_i = \langle i|\tilde{U}^\dagger A \tilde{V}|i\rangle$.

3.3 New objective function

The first modification to be made to the algorithm in [13] is a change in the objective function. The objective function used there by Xin Wang et al. was

$$f'_T(\vec{\alpha}, \vec{\beta}) = \sum_i^T q_i \mathrm{Re}(\langle i|W(\vec{\alpha})^\dagger A W(\vec{\beta})|i\rangle) \tag{3.2}$$

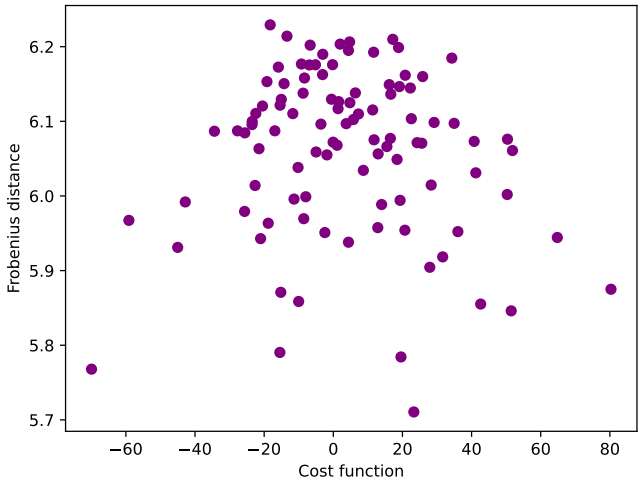


Figure 3.1: Error vs Cost of unmodified Variational SVD algorithm, evaluated on 100 instances of randomly sampled parameters for a fixed randomly generated 4×4 matrix A with $T = 4$. Here, error is evaluated as the Frobenius distance between the original matrix and the SVD matrix reconstruction.

where $q_i \in \mathbb{R}^+$ and $i < j \implies q_j < q_i$ and any such choice for $\{q_i\}$ works.

They then proved that the theoretical maximum of this objective function corresponded to $W(\vec{\alpha}^*) = U$ and $W(\vec{\beta}^*) = V$. In practice however we usually don't find the true theoretical maximum; instead we use a process such as gradient descent which will find one of possibly many local maxima with W often constrained to a limited subspace of all possible unitaries. What we would ideally want from an objective function then is some guarantee that a higher value of the objective function corresponds to a better solution. This however is not true for the original objective function as we can see in Fig. 3.1 and Fig. 3.2.

To address this issue, a new objective function is proposed:

$$f_T(\vec{\alpha}, \vec{\beta}) = \sum_i^T |\langle i|W(\vec{\alpha})^\dagger A W(\vec{\beta})|i\rangle|^2 \tag{3.3}$$

The advantage behind using this objective function is that a larger f_T corresponds to a better SVD solution. Of course we need to pick a metric to explain what is meant by "a better SVD solution". The metric we choose for our purpose is the Mean Square Error (MSE) of the entries of our reconstructed matrix A_T^{rec} with respect to A; i.e. $\frac{1}{2^{2n}}||A - A_T^{rec}||_F^2$, where $||.||_F$ denotes the Frobenius norm.

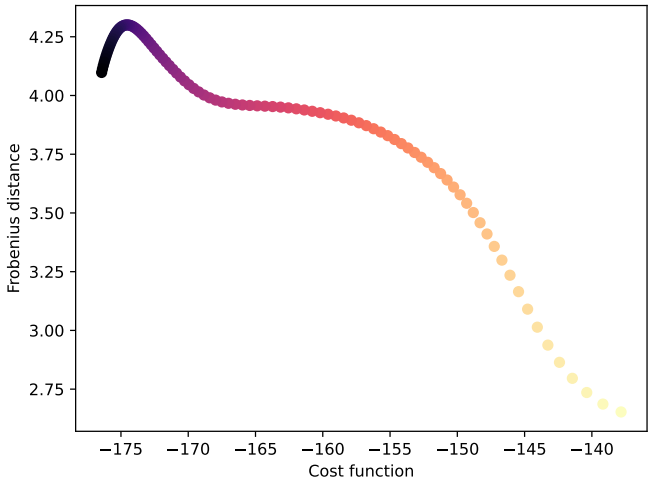


Figure 3.2: Error vs Cost of unmodified Variational SVD algorithm, evaluated on parameters returned after each of 100 iteration steps of optimization on a randomly generated 4×4 matrix A with $T = 4$. The points are colored darker the further along the optimization process they get. Here, error is evaluated as the Frobenius distance between the original matrix and the SVD matrix reconstruction.

So now, the claim to be made is formalized below:

$$\begin{aligned} & f_T(\vec{\alpha}_1, \vec{\beta}_1) > f_T(\vec{\alpha}_2, \vec{\beta}_2) \iff \\ & \text{MSE of } A_T^{rec}(\vec{\alpha}_1, \vec{\beta}_1) < \text{MSE of } A_T^{rec}(\vec{\alpha}_2, \vec{\beta}_2) \end{aligned} \tag{3.4}$$

The proof is as follows:

$$\begin{aligned} & \text{MSE of } A_T^{rec}(\vec{\alpha}, \vec{\beta}) \\ =& \frac{1}{2^{2n}} ||A - A_T^{rec}(\vec{\alpha}, \vec{\beta})||_F^2 \\ =& \frac{1}{2^{2n}} ||W^\dagger(\vec{\alpha})(A - A_T^{rec}(\vec{\alpha}, \vec{\beta}))W(\vec{\beta})||_F^2 \\ =& \frac{1}{2^{2n}} \sum_{l,m} |\langle l|W^\dagger(\vec{\alpha})(A - A_T^{rec}(\vec{\alpha}, \vec{\beta}))W(\vec{\beta})|m\rangle|^2 \end{aligned} \tag{3.5}$$

$$= \frac{1}{2^{2n}} \left(\sum_{l \neq m} |\langle l|W^\dagger(\vec{\alpha})AW(\vec{\beta})|m\rangle|^2 + \sum_{l \geq T} |\langle l|W^\dagger(\vec{\alpha})AW(\vec{\beta})|l\rangle|^2 \right) \tag{3.6}$$

$$= \frac{1}{2^{2n}} (||A||_F^2 - f_T(\vec{\alpha}, \vec{\beta})) \tag{3.7}$$

To get from (3.5) to (3.6), we use the fact that $A_T^{rec} = \sum_i^T (\langle i|W^\dagger(\vec{\alpha})AW(\vec{\beta})|i\rangle) W(\vec{\alpha})|i\rangle\langle i|W^\dagger(\vec{\beta})$. Since $||A||_F$ is constant in $(\vec{\alpha}, \vec{\beta})$, we see in (3.7) that any increase in f_T is directly associated with a corresponding decrease in the MSE and vice versa. This can be seen in Fig. 3.3 and Fig. 3.4.

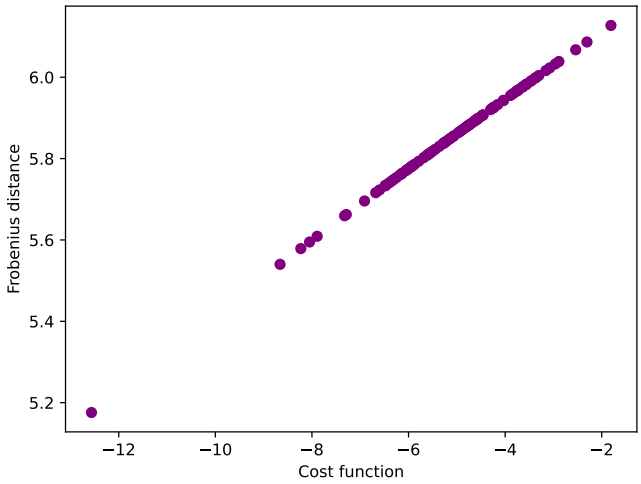


Figure 3.3: Error vs Cost of new Variational SVD algorithm, evaluated on 100 instances of randomly sampled parameters for a fixed randomly generated 4×4 matrix A with $T = 4$. Here, error is evaluated as the Frobenius distance between the original matrix and the SVD matrix reconstruction.

A corollary of (3.4) when combined with the Eckart–Young–Mirsky theorem [30] is that at the theoretical optimum of f_T, $A_T^{rec} = A_T^{trunc}$.

3.4 Algorithm implementation

To compute our objective function, we need to evaluate $|\langle i|W^\dagger(\vec{\alpha})AW(\vec{\beta})|i\rangle|^2$ for all $0 \leq i \leq T$. To do so, we use a block encoding of A (See Section 4.2 and Section 4.4), U_A.

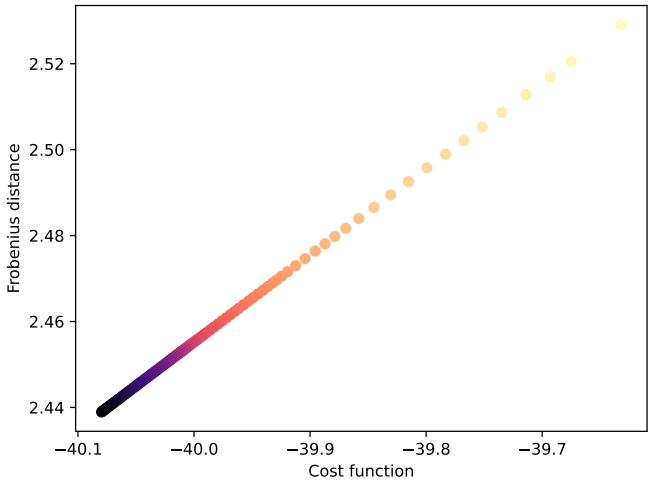


Figure 3.4: Error vs Cost of new Variational SVD algorithm, evaluated on parameters returned after each of 100 iteration steps of optimization on a randomly generated 4×4 matrix A with $T = 4$. The points are colored darker the further along the optimization process they get. Here, error is evaluated as the Frobenius distance between the original matrix and the SVD matrix reconstruction.

$$U_A = \frac{A}{\lambda} \otimes |0\rangle_F\langle 0|_F + G_1 \otimes |0\rangle_F\langle 0^\perp|_F + G_2 \otimes |0^\perp\rangle_F\langle 0|_F + G_3 \otimes |0^\perp\rangle_F\langle 0^\perp|_F \tag{3.8}$$

where G_1, G_2, and G_3 represent blocks of U_A that are not important to us.

We prepare the state $|i\rangle|0\rangle_F$ and then apply $(W^\dagger(\vec{\alpha}) \otimes I_F)U_A(W(\vec{\beta}) \otimes I_F)$ to get

$$(W^\dagger(\vec{\alpha}) \otimes I_F)U_A(W(\vec{\beta}) \otimes I_F)|i\rangle|0\rangle_F = \frac{1}{\lambda}W^\dagger(\vec{\alpha})AW(\vec{\beta})|i\rangle \otimes |0\rangle_F + W^\dagger(\vec{\alpha})G_2W(\vec{\beta})|i\rangle \otimes |0^\perp\rangle_F \tag{3.9}$$

The probability of measuring $|i\rangle|0\rangle_F$ from this state is given by $\frac{1}{\lambda^2}|\langle i|W^\dagger(\vec{\alpha})AW(\vec{\beta})|i\rangle|^2$ which when multiplied by λ^2 produces the required quantity.

We use T such circuits to evaluate the objective function when optimizing for $\vec{\alpha}$ and $\vec{\beta}$. At the end of the optimization we get our optimal parameters $\vec{\alpha}^*$ and $\vec{\beta}^*$. We set $\tilde{U} = W(\vec{\alpha}^*)$ and $\tilde{V} = W(\vec{\beta}^*)$. We then use Hadamard tests comparing $|\Phi\rangle$ and $|+\rangle_{dat}^{\otimes 2n}|0\rangle_{aux}|\psi\rangle_{isr}$ to get the real and imaginary parts of $\frac{1}{(2^n)^{3/2}}\langle i|\tilde{U}^\dagger A\tilde{V}|i\rangle$. We set $\tilde{\sigma}_i = \langle i|\tilde{U}^\dagger A\tilde{V}|i\rangle$. The algorithm returns the two circuits $\tilde{U}$ and $\tilde{V}$ as well as a set of values $\{\tilde{\sigma}_i\}$

The parameterized circuit W used for the code run in Section 3.6 is depicted in Fig. 3.5.

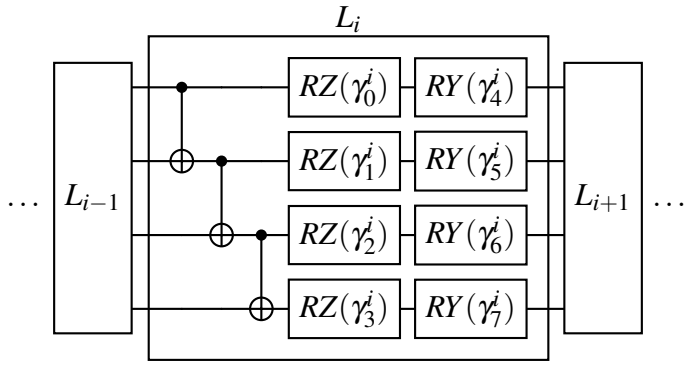


Figure 3.5: Ansatz for $W(\vec{\gamma})$ for a 16×16 matrix ($n = 4$ qubits). If the ansatz has l layers, then $\vec{\gamma}$ will consist of $2nl = 8l$ individual parameters

Ideally we want an ansatz that should be able to assume any general unitary operator. The ansatz we're using may not be able to take the form of any general unitary; but as the number of layers increases, the closer it gets to that ideal.
However, the tradeoff is that the number of layers, while increasing the depth of the circuits, also increases the number of parameters and therefore the expected number of circuit runs it takes to get to the optimum.

For classical optimization, the PaddlePaddle [31] implementation of `ADAM` [32] is used for the code run in Section 3.6.

3.5 Application in Latent Semantic Analysis

In this section we will solve the same LSA problem discussed in Section 2.5.1, but by using the variational algorithm discussed in this chapter. As discussed, the problem boils down to estimating $\langle i|(A^{trunc})^T A^{trunc}|j\rangle$, $\langle i|(A^{trunc})^T A^{trunc}|i\rangle$, and $\langle j|(A^{trunc})^T A^{trunc}|j\rangle$ for a given Term-Document matrix A and indices of words i and j.
The first step to applying our variational quantum SVD algorithm to this problem is recognizing that

$$\langle i|(A^{trunc})^T A^{trunc}|j\rangle = \langle i|V\Sigma_{trunc}U^\dagger U\Sigma_{trunc}V^\dagger|j\rangle = \langle i|V\Sigma^2_{trunc}V^\dagger|j\rangle, \tag{3.10}$$

where Σ_{trunc} represents a diagonal matrix containing the top T (T determines the level of truncation) singular values (in the order corresponding to the columns of V) followed by zeros. After running the Variational Quantum SVD algorithm on A, we will have access to the circuit V whose first T columns will be accurate (to the extent determined by the number of iterations of training done and the ansatz used). To get an estimate of $\langle i|V\Sigma^2_{trunc}V^\dagger|j\rangle$, we need some way of applying Σ^2_{trunc}. This task is solved by constructing a unitary W_Σ such that

$$\mathrm{Re}(W_\Sigma) = \Sigma^2_{trunc}/\sigma_1, \tag{3.11}$$

where σ_1 is the largest singular value. Thus we can use $W_\Sigma = e^{i\arccos\Sigma^2_{trunc}/\sigma_1}$. A procedure for implementing diagonal unitaries efficiently is given in [33]. The expression we need to now estimate becomes

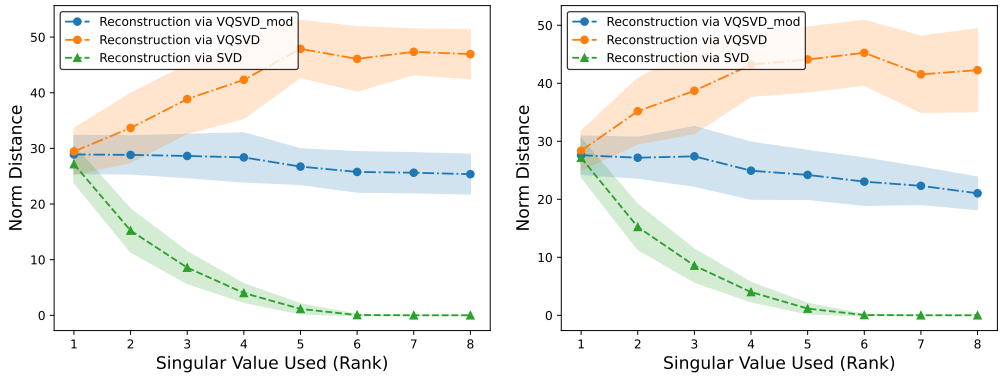
$$\langle i|V\Sigma^2_{trunc}V^\dagger|j\rangle = \sigma_1\mathrm{Re}(\langle i|VW_\Sigma V^\dagger|j\rangle). \tag{3.12}$$

We can use the Hadamard test to get this value by comparing the states $W_\Sigma V^\dagger|j\rangle$ and $V^\dagger|i\rangle$. We can use this same procedure to get estimates of $\langle i|(A^{trunc})^T A^{trunc}|i\rangle$ and $\langle j|(A^{trunc})^T A^{trunc}|j\rangle$.

3.6 Results

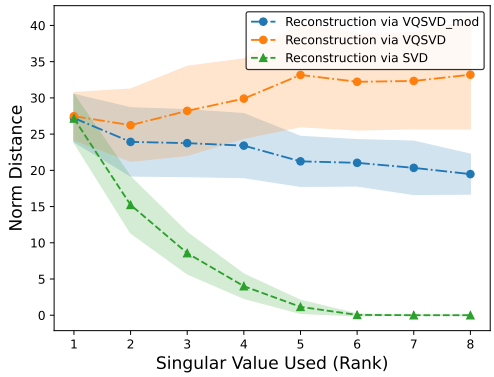
3.6.1 MNIST

In this subsection, the performance of the algorithm is measured on 8×8 MNIST images. For each $1 \leq T \leq 8$, and for each of 20 images (2 of each digit), simulations are run with 5 randomly generated initial parameters. For each run of the algorithm, the returned solution is used to reconstruct the input matrix and the Frobenius distance of the reconstructed matrix is evaluated. The average Frobenius distance vs T is plotted for the reconstructions outputted by the unmodified and modified Variational Quantum SVD algorithms and classical SVD computations in Fig. 3.6.

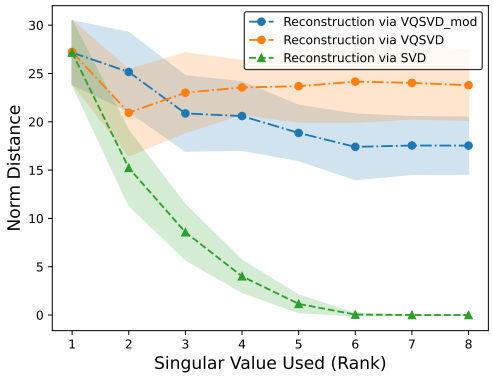


(a) 2 layers

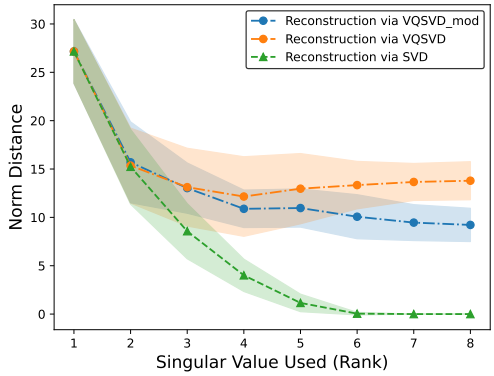
(b) 3 layers



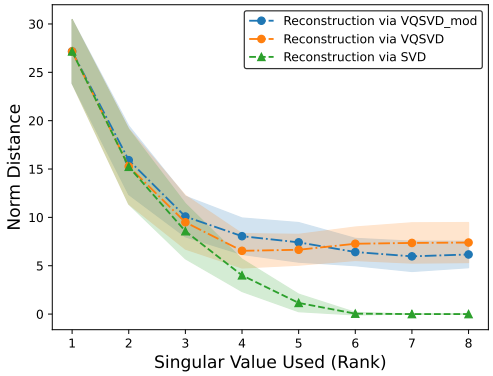
(c) 4 layers



(d) 5 layers



(e) 10 layers



(f) 15 layers

Figure 3.6: Variational Quantum SVD results on MNIST data. The different figures depict results for different layer counts in the ansatz used. The shaded region above and below a data point depicts ± 1 standard deviation around the mean.

We see from Fig. 3.6 that for low layer counts, the errors are much higher in the unmodified algorithm and they in fact tend to increase with rank. This unanticipated trend is not however seen with the modified algorithm, as expected from (3.4). The unmodified algorithm does however seem to outperform the modified version in a few cases, namely in the 5 layer case for rank 2, the 10 layer case for rank 2, and the 15 layer case for ranks 2 to 5. It could be the case that different algorithms are better in different regimes. More simulations would need to be run for more layers to better study these algorithms and how they scale.

3.6.2 LSA

To demonstrate the use of this algorithm for LSA a convenient example problem was generated. We have 4 documents generated by ChatGPT [34] and the terms we want to compare are "science", "mathematics", "fish", and "water".

Document 1

In the vast landscape of human knowledge, science stands as a beacon of curiosity and inquiry. Rooted in observation and experimentation, science seeks to unravel the mysteries of the universe through systematic exploration. From the smallest particles to the grandest galaxies, science endeavors to understand the fundamental laws that govern our existence. Through the lens of science, we peer into the cosmos and delve into the intricacies of life itself. The pursuit of scientific understanding drives innovation and progress, shaping the course of human history and illuminating the path towards a brighter future. With each discovery, science expands the boundaries of what is known and challenges the limits of our imagination. Across diverse disciplines, from biology to physics, chemistry to astronomy, science offers a framework for understanding the world around us and unlocking its secrets. Mathematics, as a vital tool of scientific inquiry, underpins many of the discoveries and advancements that propel science forward. From the elegant equations of quantum mechanics to the precise calculations of celestial mechanics, mathematics provides the language through which scientists communicate and explore the wonders of the universe. In the pursuit of scientific knowledge, mathematics serves as both a guiding light and a powerful tool, enabling scientists to unravel complex phenomena and unlock the mysteries of the natural world.

Document 2

Water, the universal solvent, holds a paramount role in sustaining life and facilitating countless human activities. From quenching our thirst to irrigating fields, water is an indispensable resource woven into the fabric of our existence. Its versatility knows no bounds, serving as a vital component in industries ranging from agriculture to manufacturing, while also providing a medium for transportation and recreation. Beyond its practical applications, water holds cultural and spiritual significance, symbolizing purity, renewal, and tranquility across diverse civilizations. The presence of water, whether in rivers, lakes, or oceans, fundamentally influences the development and sustainability of societies across the globe. In aquatic ecosystems, water serves as a habitat for an array of life forms, including fish, which thrive in its depths. Fish, with their graceful movements and diverse species, contribute to the ecological balance of water bodies, playing roles as both predators and prey. Their presence not only enriches the biodiversity of aquatic environments but also provides sustenance for human populations, supporting livelihoods and culinary traditions around the globe. The presence of fish in aquatic environments also underscores the profound relationship between these creatures and the vital resource of water, reflecting the intricate balance of life within freshwater and marine ecosystems.

Document 3

In the rich tapestry of aquatic life, fish stand as captivating symbols of grace, diversity, and resilience. From the majestic marlin slicing through the ocean depths to the nimble trout dancing in mountain streams, fish inhabit a vast array of habitats, adapting to the unique challenges posed by their watery realms. Their remarkable diversity spans from the colorful coral reefs of tropical seas to the frigid depths of polar oceans, showcasing the adaptability of life to the ever-changing dynamics of water environments. As guardians of underwater ecosystems, fish play pivotal roles in regulating populations of prey and predators, maintaining the delicate balance of aquatic food webs. Amidst the shimmering currents and tranquil depths, fish navigate their watery domains with finesse and precision, utilizing an array of sensory adaptations to detect prey, evade predators, and navigate their surroundings. Whether darting through dense vegetation in freshwater lakes or gliding gracefully amidst the vibrant corals of tropical seas,

fish demonstrate a profound connection to the element of water that shapes their existence. Yet, the fate of fish is intricately linked to the quality and availability of water, as these vital habitats face increasing threats from pollution, habitat destruction, and climate change. Rising temperatures, altered flow regimes, and contaminants pose significant challenges to fish populations worldwide, highlighting the urgent need for conservation efforts to safeguard their habitats and ensure their continued survival.

Document 4

Mathematics, often hailed as the language of the universe, is a discipline of pure abstraction and elegant logic. From the simplicity of arithmetic to the complexity of advanced calculus, mathematics unveils the hidden structures underlying the fabric of reality. The applications of mathematics transcend boundaries, shaping not only the realms of science and engineering but also economics, art, and philosophy. Through mathematics and reasoning, we navigate the complexities of our world, from predicting the trajectory of celestial bodies to optimizing the flow of traffic in urban centers. In essence, mathematics is the cornerstone of human civilization, providing a framework for understanding, problem-solving, and innovation. Amidst its beauty and complexity, mathematics finds a kindred spirit in science, as both disciplines share a common quest for understanding the intricacies of our universe. While science employs experimentation and observation to explore the natural world, mathematics provides the indispensable tools for quantifying and interpreting these phenomena, enriching our comprehension and driving progress.

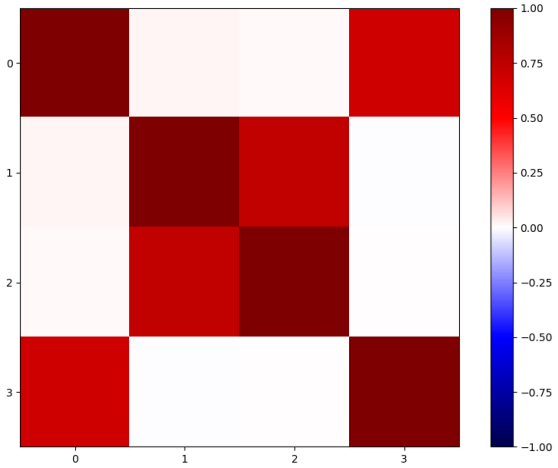


Figure 3.7: LSA simulator result

The Term-Document matrix of these 4 documents for the chosen words has been reverse engineered to be the following matrix:

$$\begin{array}{c} \\ science \\ water \\ fish \\ mathematics \end{array} \begin{array}{c} \begin{array}{cccc} Document1 & Document2 & Document3 & Document4 \end{array} \\ \begin{bmatrix} 7 & 0 & 0 & 3 \\ 0 & 7 & 3 & 0 \\ 0 & 3 & 7 & 0 \\ 3 & 0 & 0 & 7 \end{bmatrix} \end{array}$$

Qiskit [35] is used to run the LSA algorithm described in Section 3.5. First the variational algorithm was run using the LCU block encoding (see Section 4.4.5) to get circuits U and V as well as the singular values, using SPSA [36] as the classical optimizer. Then the circuits described

in Section 3.5 were run on Qiskit's Aer simulator with 20000 shots each, to get the matrix in Fig. 3.7. The (i, j)-th element of this matrix corresponds to the cosine similarity of word i and word j.

The same circuits are then run on IBM cloud hardware [37] to obtain Fig. 3.8. We used `ibm_brisbane`, which is one of the IBM Quantum Eagle r3 processors. This output looks very similar in structure to the simulation results; however the outputted similarities for Documents 1 and 4, and Documents 2 and 3 have gone negative. One possible explanation may be dephasing errors that creep in when running the algorithm on actual hardware. Such errors could cause a shift in relative phase between the two states that we compare using the Hadamard test, thus explaining the results we see in Fig. 3.8.

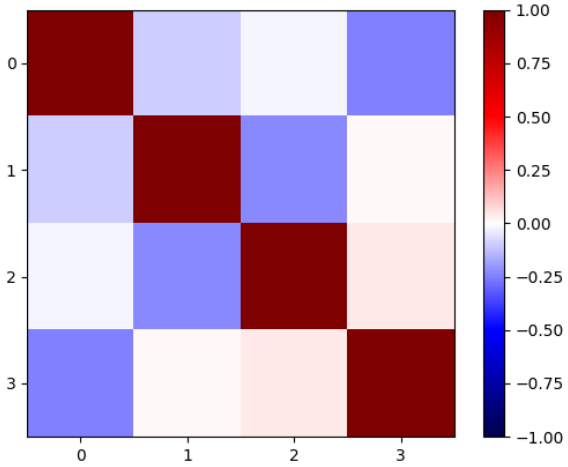


Figure 3.8: LSA IBM hardware result

The utility function is also calculated using IBM hardware for 8 sets of parameters that were evaluated during the simulated optimization process. The results are shown in Fig. 3.9. Here, we used `ibm_osaka`, which is one of the IBM Quantum Eagle r3 processors. This figure gives us a sense of how trainable our ansatzes are when running the algorithm on real hardware. In this figure, we can see that the utility function calculated with hardware without mitigation does increase with increase in the true value of the utility function, but only up to a point. Looking at parameter sets 6, 7, and 8, we see that the calculated utility function plateaus, indicating that the ansatzes can be trained up to a point after which the calculated utility function can't be increased any more. This flattening of the utility function is solved however when using error mitigation. The utility function calculated with hardware with mitigation does seem to increase consistently with increase in the true value of the utility function. While it may not be cost-effective, these results indicate

the possibility of successfully running this variational algorithm on existing hardware.

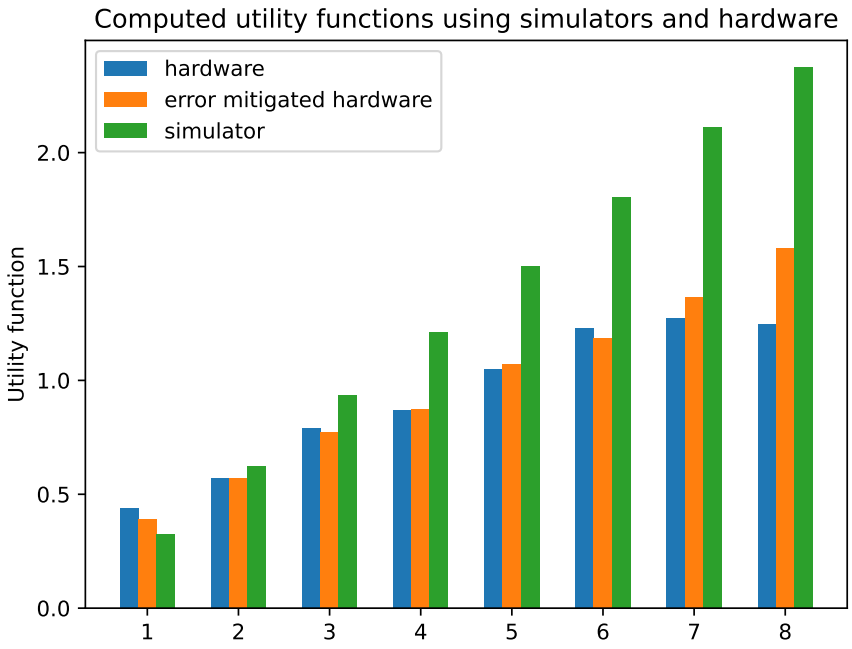


Figure 3.9: Depicted in green are LSA training losses for 8 handpicked sets of parameters that were evaluated during training on a simulator. Blue bars show the losses evaluated using circuits run on IBM hardware. Orange bars show the losses evaluated using circuits run on hardware enhanced with error mitigation.

See Section 6.2 for discussion and future works.

I acknowledge the use of IBM Quantum services for this work. The views expressed are those of the author, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Chapter 4

Quantum Singular Value Transformation

The Quantum Singular Value Transform (QSVT) is a powerful new framework for quantum algorithms put forward by Gilyén et al. [16] in 2018 and further popularized by Martyn et al. [17] in 2021. The latter work demonstrated this framework's ubiquity in quantum computing and its many applications including amplitude amplification, phase estimation, solving linear systems of equations (LSE), and Hamiltonian simulation. In this chapter, we start by exploring what this framework is, along with how and why it works. To do this, we begin by acquainting ourselves with Quantum Signal Processing (QSP). Then, we study how the QSVT circuit's behavior can be expressed as the operation of multiple QSP circuits acting on disjoint 2D subspaces of the Hilbert space, and provide an example demonstration. We then go through the details of various block encodings that can be used to implement the QSVT in practice, including a novel block encoding scheme. Lastly, we run the QSVT circuit for LSE problems as well as for Topological Data Analysis (TDA) and measure its performance for various block encodings.

4.1 Quantum Signal Processing

While there is a more common formulation used in most introductions of QSP [38, 16], for our purposes, QSP is a single qubit operation of the following form [16]:

$$QSP_{\vec{\phi}} = \prod_{j=1}^{d} \left(e^{i\phi_j Z} R(x) \right) \tag{4.1}$$

where Z is the Pauli-Z matrix and $R(x)$ is the following reflection:

$$R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}$$

The significance of this operation is revealed to us in the following theorem (Corollary 8 from [16]):

Theorem 4.1.1. *Let $P \in \mathbb{C}[x]$ be a degree-d polynomial, such that*

- *P has parity-$(d \mod 2)$*
- $\forall x \in [-1,1] : |P(x)| \leq 1$
- $\forall x \in (-\infty,-1] \bigcup [1,\infty) : |P(x)| \geq 1$
- *if d is even, then $\forall x \in \mathbb{R} : P(ix)P^*(ix) \geq 1$*

If all of these conditions are met, there exists $\vec{\phi} \in \mathbb{R}^d$ such that

$$\prod_{j=1}^{d} \left(e^{i\phi_j Z} R(x) \right) = \begin{bmatrix} P(x) & . \\ . & . \end{bmatrix} \tag{4.2}$$

4.2 Quantum Singular Value Transformation

A block encoding of an $M \times N$ matrix A is a unitary U_A such that $A/\alpha = \tilde{\Pi} U \Pi$ where $\tilde{\Pi}$ and Π are projection operators and $\alpha > ||A||_F$ is what we will call the subnormalization factor. The way block encodings are defined in the literature [16, 39], $\Pi = \sum_{i=0}^{N-1} |i\rangle\langle i|$ and $\tilde{\Pi} = \sum_{i=0}^{M-1} |i\rangle\langle i|$. This means that the matrix representation of U has A/α embedded in its top left corner.

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

The choice of Π and $\tilde{\Pi}$ is, however, ultimately arbitrary when working with the QSVT.

The QSVT [16, 17], $U_A^{\vec{\phi}}$, for a given block encoding U_A and sequence of angles $\vec{\phi}$ with length d is an operation that has the following form when d is odd

$$U_A^{\vec{\phi}} = e^{i\phi_1(2\tilde{\Pi}-1)}U_A\left(\prod_{k=1}^{(d-1)/2} e^{i\phi_{2k}(2\Pi-1)}U_A^\dagger e^{i\phi_{2k+1}(2\tilde{\Pi}-1)}U_A\right) \tag{4.3}$$

and the below form when d is even.

$$U_A^{\vec{\phi}} = \prod_{k=1}^{d/2} e^{i\phi_{2k-1}(2\Pi-1)}U_A^\dagger e^{i\phi_{2k}(2\tilde{\Pi}-1)}U_A \tag{4.4}$$

The significance of this operation is explained in Theorem 17 of [16] which states that for a given sequence of angles $\vec{\phi}$ and its corresponding polynomial P from Theorem 4.1.1,

$$U_A^{\vec{\phi}} = \begin{bmatrix} P(A/\alpha) & . \\ . & . \end{bmatrix} \tag{4.5}$$

Here P is acting on the singular values of A/α. If the SVD of A/α is given by $A/\alpha = USV^\dagger = \sum_k \sigma_k U|k\rangle\langle k|V^\dagger$, then $P(A/\alpha) = \sum_k P(\sigma_k)U|k\rangle\langle k|V^\dagger$ when d is odd, and $P(A/\alpha) = \sum_k P(\sigma_k)V|k\rangle\langle k|V^\dagger$ when d is even.

4.2.1 Why is QSVT related to QSP in this way?

The singular values of A/α have to be less than or equal to 1 to ensure unitarity of U_A. Let $|v\rangle$ be a right singular vector of A/α with corresponding left singular vector $|u\rangle$ and singular value $\sigma < 1$.

Let $|\psi\rangle = |v\rangle \oplus |0\rangle$ be a vector in the subspace of Π. This means the first N entries of the vector $|\psi\rangle$ are from $|v\rangle$ and the rest are zeros. Similarly let $|\tilde{\psi}\rangle = |u\rangle \oplus |0\rangle$ be a vector in the subspace of $\tilde{\Pi}$. This means the first M entries of the vector $|\tilde{\psi}\rangle$ are from $|u\rangle$ and the rest are zeros.

While a more complete treatment can be found in [16], we will see why (4.5) holds by studying how the QSVT acts on $|\psi\rangle$.

We start by defining $|\tilde{\psi}^{\perp}\rangle$ such that $U_A|\psi\rangle = \sigma|\tilde{\psi}\rangle + \sqrt{1-\sigma^2}|\tilde{\psi}^{\perp}\rangle$. Only the first N columns of U_A contribute to this operation since only the first N entries of $|\psi\rangle$ are non-zero. The first M rows make up the matrix A and contribute to taking $|v\rangle \oplus |0\rangle$ to $\sigma|u\rangle \oplus |0\rangle$. The remaining rows contribute to taking $|\psi\rangle$ to $\sqrt{1-\sigma^2}|\tilde{\psi}^{\perp}\rangle$ whose first M rows are zeros.
We similarly define $|\psi^{\perp}\rangle$ such that $U_A^{\dagger}|\tilde{\psi}\rangle = \sigma|\psi\rangle + \sqrt{1-\sigma^2}|\psi^{\perp}\rangle$. A similar understanding of how this vector interacts with the different blocks of U_A shows that the first N rows of $|\psi^{\perp}\rangle$ are zeros.

The first point we want to prove is that U_A takes the subspace $s_1 = \{|\psi\rangle, |\psi^{\perp}\rangle\}$ in its entirety to the subspace $s_2 = \{|\tilde{\psi}\rangle, |\tilde{\psi}^{\perp}\rangle\}$ (and vice versa for $U_A^{\dagger}$). We already know that U_A takes $|\psi\rangle$ to s_2. We need to show that it does the same to $|\psi^{\perp}\rangle$.

$$\begin{aligned}
\sqrt{1-\sigma^2}U_A|\psi^{\perp}\rangle &= U_A(U_A^{\dagger}|\tilde{\psi} - \sigma|\psi\rangle) \\
&= |\tilde{\psi}\rangle - \sigma U_A|\psi\rangle \\
&= |\tilde{\psi}\rangle\sigma(\sigma|\tilde{\psi} + \sqrt{1-\sigma^2}|\tilde{\psi}^{\perp}\rangle) \\
&= (1-\sigma^2)|\tilde{\psi}\rangle - \sigma\sqrt{1-\sigma^2}|\tilde{\psi}^{\perp}\rangle
\end{aligned}$$

Dividing by $\sqrt{1-\sigma^2}$ on both sides, we get

$$U_A|\psi^{\perp}\rangle = \sqrt{1-\sigma^2}|\tilde{\psi}\rangle - \sigma|\tilde{\psi}^{\perp}\rangle. \tag{4.6}$$

From here, we can represent the action of U_A on s_1 to s_2 as a 2×2 matrix:

$$U_A = \begin{array}{c} \\ |\tilde{\psi}\rangle \\ |\tilde{\psi}^{\perp}\rangle \end{array} \begin{array}{c} \begin{array}{cc} |\psi\rangle & |\psi^{\perp}\rangle \end{array} \\ \begin{bmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma \end{bmatrix} \end{array}$$

We can also represent $e^{i\phi(2\Pi-1)}$ and $e^{i\phi(2\tilde{\Pi}-1)}$ in a similar fashion.

$$e^{i\phi(2\Pi-1)} = \begin{matrix} & \begin{matrix} |\psi\rangle & |\psi^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\psi\rangle \\ |\psi^{\perp}\rangle \end{matrix} & \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix} \end{matrix}$$

$$e^{i\phi(2\tilde{\Pi}-1)} = \begin{matrix} & \begin{matrix} |\tilde{\psi}\rangle & |\tilde{\psi}^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\tilde{\psi}\rangle \\ |\tilde{\psi}^{\perp}\rangle \end{matrix} & \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix} \end{matrix}$$

Putting all this together, we can see how the QSVT circuit acts on $\{|\psi\rangle, |\psi^{\perp}\rangle\}$. For even d, the operation becomes:

$$\begin{aligned} U_A^{\vec{\phi}} &= \prod_{k=1}^{d/2} e^{i\phi_{2k-1}(2\Pi-1)} U_A^{\dagger} e^{i\phi_{2k}(2\tilde{\Pi}-1)} U_A \\ &= \prod_{k=1}^{d/2} \begin{matrix} & \begin{matrix} |\psi\rangle & |\psi^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\psi\rangle \\ |\psi^{\perp}\rangle \end{matrix} & \begin{bmatrix} e^{i\phi_{2k-1}} & 0 \\ 0 & e^{-i\phi_{2k-1}} \end{bmatrix} \end{matrix} \begin{matrix} & \begin{matrix} |\tilde{\psi}\rangle & |\tilde{\psi}^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\psi\rangle \\ |\psi^{\perp}\rangle \end{matrix} & \begin{bmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma \end{bmatrix} \end{matrix} \\ &\qquad \begin{matrix} & \begin{matrix} |\tilde{\psi}\rangle & |\tilde{\psi}^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\tilde{\psi}\rangle \\ |\tilde{\psi}^{\perp}\rangle \end{matrix} & \begin{bmatrix} e^{i\phi_{2k}} & 0 \\ 0 & e^{-i\phi_{2k}} \end{bmatrix} \end{matrix} \begin{matrix} & \begin{matrix} |\psi\rangle & |\psi^{\perp}\rangle \end{matrix} \\ \begin{matrix} |\tilde{\psi}\rangle \\ |\tilde{\psi}^{\perp}\rangle \end{matrix} & \begin{bmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma \end{bmatrix} \end{matrix} \end{aligned}$$

To understand why this substitution is valid, note how the row labels of a matrix coincide with the column labels of the matrix to its immediate left. Since we are having this whole operator act on s_1, we know that we can write the right-most U_A operator in its 2×2 matrix representation that acts on s_1. Since the output of that operation lies in s_2, we know that we can express the operator to its immediate left - $e^{i\phi_{2k+1}(2\tilde{\Pi}-1)}$ - in its 2×2 matrix representation that acts on s_2. In this manner, we find that we are allowed to replace every operation in this product with its 2×2 matrix representation. Thus we have taken an operator of arbitrarily high dimension and expressed its

behavior as that of a single qubit operator when acting on a specific subspace of the larger Hilbert space. This is called qubitization [40].
Now that we're confident in the consistency of these shuffling bases, we'll shift our focus away from the distracting basis changes and take a closer look at these matrices.

$$
\begin{aligned}
&\left(\prod_{k=1}^{d/2}\begin{bmatrix} e^{i\phi_{2k-1}} & 0 \\ 0 & e^{-i\phi_{2k-1}}\end{bmatrix}\begin{bmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma\end{bmatrix}\begin{bmatrix} e^{i\phi_{2k}} & 0 \\ 0 & e^{-i\phi_{2k}}\end{bmatrix}\begin{bmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma\end{bmatrix}\right)^{s_1}_{s_1} \\
&= \left(\prod_{k=1}^{d/2} e^{i\phi_{2k-1}Z}R(\sigma)e^{i\phi_{2k}Z}R(\sigma)\right)^{s_1}_{s_1} \\
&= \left(\prod_{k=1}^{d} e^{i\phi_k Z}R(\sigma)\right)^{s_1}_{s_1} \\
&= \begin{bmatrix} P(\sigma) & . \\ . & .\end{bmatrix}^{s_1}_{s_2}
\end{aligned}
$$

That explains why QSVT behaves like (4.5) on vectors of the form that $|\psi\rangle$ takes. [16] does a more complete treatment and also considers all other possible cases and shows that the QSVT circuit satisfies (4.5) for all vectors in the Hilbert space, thereby proving (4.5).

4.3 An example demonstration

Consider the following simple QSP circuit:

$$QSP = e^{i\frac{\pi}{4}Z}R(\frac{1}{\sqrt{2}})e^{-i\frac{\pi}{4}Z}R(\frac{1}{\sqrt{2}}) \tag{4.7}$$

We can study the action of this circuit on the Bloch sphere - one operator at a time - to produce Fig. 4.1.

Now, let's study a QSVT circuit related to this QSP operation. Our matrix A is given by:

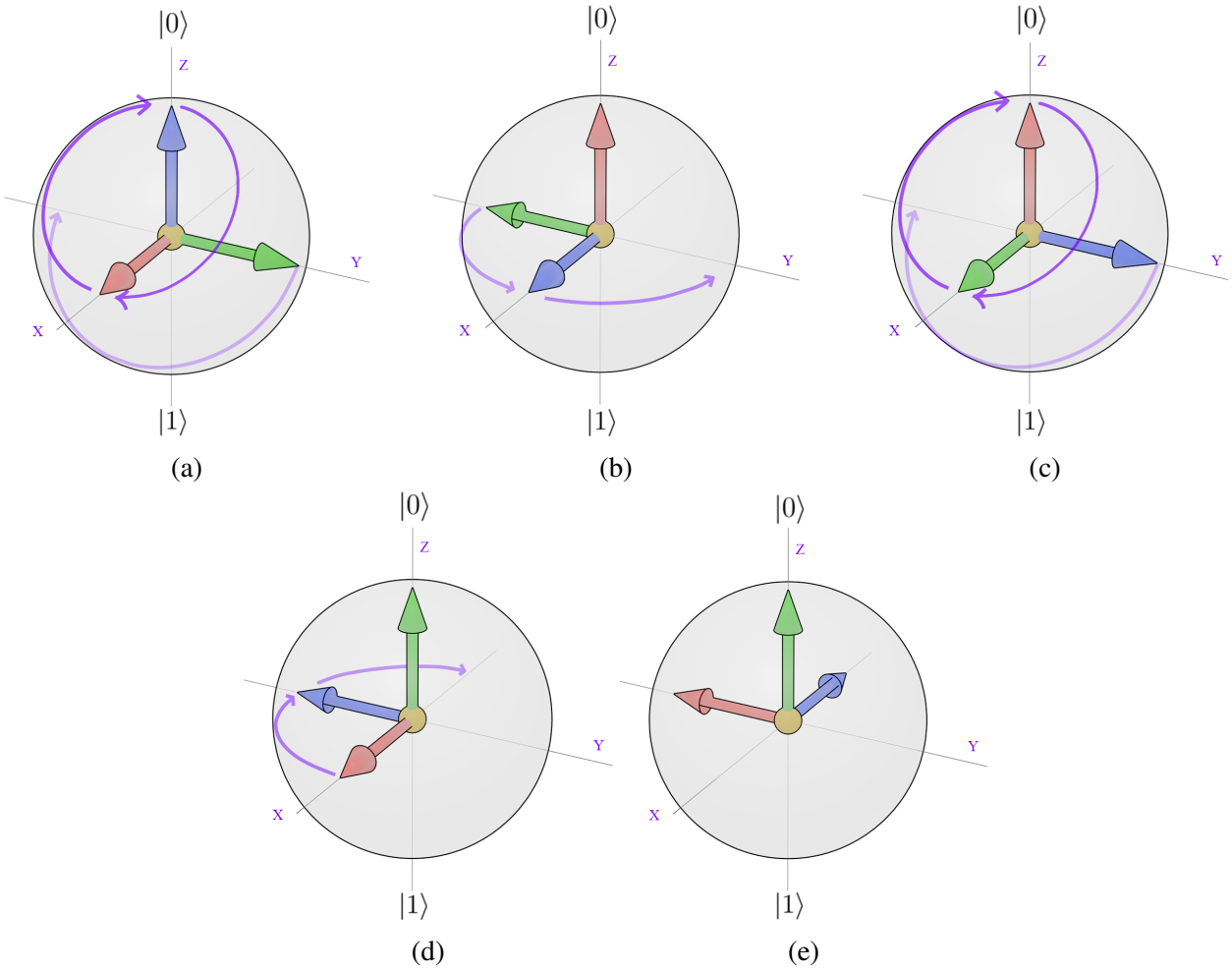


Figure 4.1: Quantum Signal Processing example from (4.7) depicted on the Bloch Sphere. We start with three vectors $|0\rangle$, $|+\rangle$, and $|+i\rangle$ represented with blue, red and green vectors respectively in (a). (a) also shows the first (right-most in (4.7)) operator, $R(\frac{1}{\sqrt{2}})$, acting on these vectors (depicted using the purple arrows). This results in a new orientation seen in (b). (b) also shows the next operator $e^{-i\frac{\pi}{4}Z}$ acting on these vectors to produce the orientation seen in (c). (c) depicts the next operator $R(\frac{1}{\sqrt{2}})$, producing the orientation seen in (d). And lastly, (d) shows the left-most operator $e^{i\frac{\pi}{4}Z}$, yielding the final orientation of the vectors seen in (e).

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & -\frac{1}{3} \end{bmatrix}$$

The SVD of A is given by

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{\sqrt{2}}{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We produce a block encoding of A given by U:

$$U = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{3} & \frac{1}{2} & -\frac{\sqrt{7}}{3\sqrt{2}} \\ \frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} & -\frac{1}{2} & -\frac{1}{3} \\ -\frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} & \frac{1}{2} & -\frac{1}{3} \end{bmatrix}$$

We will focus our attention on the singular vectors corresponding to the singular value $\frac{1}{\sqrt{2}}$ and the subspaces associated with them. Thus, we get

$$|\psi\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad |\psi^{\perp}\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \qquad |\tilde{\psi}\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}, \qquad |\tilde{\psi}^{\perp}\rangle = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

We got $|\psi\rangle$ and $|\tilde{\psi}\rangle$ from the right and left singular vectors of A. $|\psi^{\perp}\rangle$ and $|\tilde{\psi}^{\perp}\rangle$ are then evaluated from the following relations.

$$U|\psi\rangle = \sigma|\tilde{\psi}\rangle + \sqrt{1-\sigma^2}|\tilde{\psi}^{\perp}\rangle$$
$$U^{\dagger}|\tilde{\psi}\rangle = \sigma|\psi\rangle + \sqrt{1-\sigma^2}|\psi^{\perp}\rangle$$

The QSVT circuit associated with the QSP circuit in (4.7) is given by

$$e^{i\frac{\pi}{4}(2\Pi-1)}U^\dagger e^{-i\frac{\pi}{4}(2\tilde{\Pi}-1)}U \tag{4.8}$$

$$= \begin{bmatrix} e^{i\frac{\pi}{4}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\pi}{4}} & 0 & 0 \\ 0 & 0 & e^{-i\frac{\pi}{4}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{3} & -\frac{1}{3} & \frac{\sqrt{7}}{3\sqrt{2}} & \frac{\sqrt{7}}{3\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{\sqrt{7}}{3\sqrt{2}} & -\frac{\sqrt{7}}{3\sqrt{2}} & -\frac{1}{3} & -\frac{1}{3} \end{bmatrix}$$

$$\begin{bmatrix} e^{-i\frac{\pi}{4}} & 0 & 0 & 0 \\ 0 & e^{-i\frac{\pi}{4}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\pi}{4}} & 0 \\ 0 & 0 & 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{3} & \frac{1}{2} & -\frac{\sqrt{7}}{3\sqrt{2}} \\ \frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} & -\frac{1}{2} & -\frac{1}{3} \\ -\frac{1}{2} & \frac{\sqrt{7}}{3\sqrt{2}} & \frac{1}{2} & -\frac{1}{3} \end{bmatrix} \tag{4.9}$$

$$= \begin{bmatrix} \frac{1+i}{2} & 0 & \frac{1-i}{2} & 0 \\ 0 & \frac{2+7i}{9} & 0 & \frac{\sqrt{14}-\sqrt{14}i}{9} \\ \frac{-1-i}{2} & 0 & \frac{1-i}{2} & 0 \\ 0 & \frac{-\sqrt{14}-\sqrt{14}i}{9} & 0 & \frac{2-7i}{9} \end{bmatrix} \tag{4.10}$$

Fig. 4.2 depicts how this circuit acts on any vector in the subspace spanned by $|\psi\rangle$ and $|\psi^\perp\rangle$, one operator at a time. Note for example how the blue vector, which was initially on $|\psi\rangle$, ends up on $\frac{|\psi\rangle-|\psi^\perp\rangle}{2}$. We can confirm that (4.10) does indeed take $|\psi\rangle$ to $\frac{|\psi\rangle-|\psi^\perp\rangle}{2}$, up to a global phase.

4.4 Block encodings

In this section, we explore a few known block encoding methods and one novel method, and elaborate on their advantages and disadvantages. Given A is an $M \times N$ matrix, let $s = \lceil \log_2 \max\{M,N\} \rceil$. Let U_A be a block encoding of A with subnormalization factor α. If U_A acts on $s+a$ qubits, we will call the first s qubits the 'target qubits', and the other a qubits the 'block determining qubits'. Let $A' = A/\alpha$.

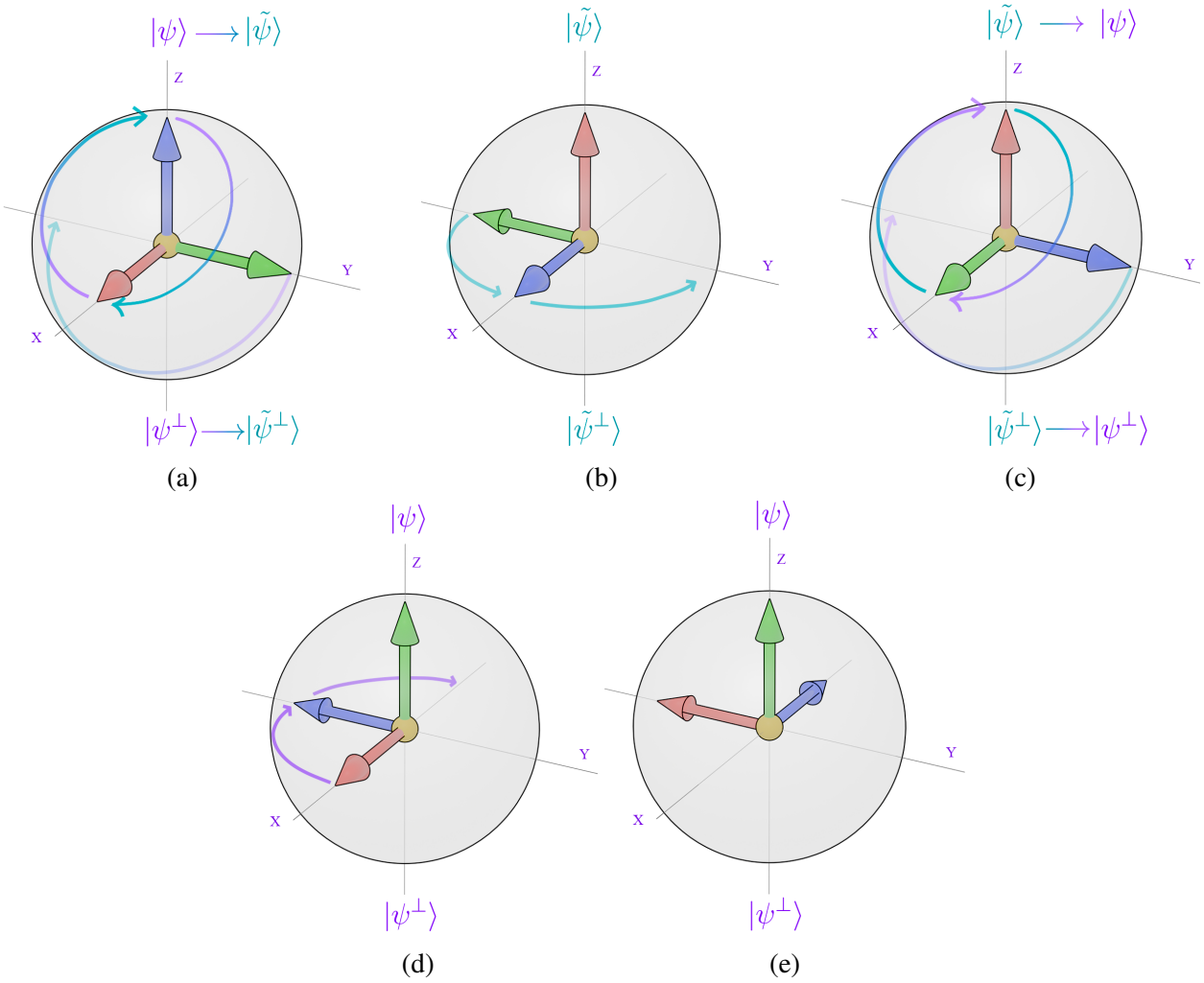


Figure 4.2: QSVT example from (4.9) depicted on the Bloch Sphere. Note how the figures look identical to Fig. 4.1, except the Bloch spheres here represent different subspaces at different steps. In (a),(d), and (e), we are in the subspace spanned by $|\psi\rangle$ and $|\psi^{\perp}\rangle$; whereas in (b) and (c), the Bloch sphere represents the subspace spanned by $|\tilde{\psi}\rangle$ and $|\tilde{\psi}^{\perp}\rangle$.

4.4.1 Naive block encoding

This block encoding implementation was given as a pedagogical example in [17] and [41]. We construct the following unitary matrix.

$$U_A = \begin{bmatrix} A' & \sqrt{I-A'^2} \\ \sqrt{I-A'^2} & A' \end{bmatrix} \tag{4.11}$$

Here, $\sqrt{I-A'^2}$ is defined as

$$\sqrt{I-A'^2} := \sum_k \sigma_k |w_k\rangle\langle v_k|, \tag{4.12}$$

where σ_k, $|w_k\rangle$ and $|v_k\rangle$ are the k-th singular value, left singular vector, and right singular vector of A' respectively. This is not a block encoding method that would be used in practice since evaluating the off-diagonal blocks of this matrix requires one to compute the SVD of A. The strength of QSVT typically comes from the ability to manipulate the singular values of a matrix without having to compute its SVD. If the SVD of A is already computed classically, it would presumably be easier to manipulate the singular values directly without having to use QSVT at all. Furthermore, a true implementation of this block encoding would require unitary synthesis [42], which could be costly. On the other hand, this is a very convenient block encoding method for demonstration purposes since it only requires 1 block determining qubit and α can be set to be as low as $\alpha = \max_k \{\sigma_k\}$. It is straightforward to simulate since we aren't given a procedure to construct a quantum circuit, but are instead given a direct matrix representation of the unitary that needs to be applied.

4.4.2 FABLE

The FABLE block encoding was proposed in [43] and is realized by using QPIXL [44] in conjunction with the block encoding scheme for sparse-access matrices in [16] (Section 4.2, Lemma 48). This block encoding method, in stark contrast to the previous method, can be directly implemented and, unlike many other block encoding schemes, it makes no additional assumptions of the user's abilities - it does not assume unfounded oracle access, a convenient data structure, or the availability of a QRAM in general. However, it has a subnormalization factor of $\alpha = 2^s \max_{i,j} \{A_{i,j}\}$ where s is the number of target qubits and $A_{i,j}$ refers to the (i,j)-th entry of A. In practice, we see that this subnormalization factor ends up being significantly higher than those of other methods. This matters because a larger subnormalization factor implies a shorter range containing the distribution

of singular values of A'. And, in practice, this usually means one would require more layers in their QSVT circuit to do the same transformation on A, than if they used a block encoding with a smaller α. This method requires $s+1$ number of block determining qubits.

4.4.3 Sparse FABLE

While FABLE used a similar block encoding scheme as the one proposed in [16] for sparse-access matrices, the FABLE method itself does not take sparsity into account. Here, we go through how the exact block encoding proposed in Lemma 48 of [16] can be implemented, but in a simpler manner by using FABLE/QPIXL.

Let $A_{max} = \max_{i,j}\{A_{i,j}\}$.
Let the row sparsity of A be denoted by S_r and the column sparsity by S_c.
Let $m = \lceil \log_2(M) \rceil$, $n = \lceil \log_2(N) \rceil$, $m'_s = \lceil \log_2(S_c) \rceil$, and $n'_s = \lceil \log_2(S_r) \rceil$.
Let $t = \max\{m+n'_s, n+m'_s\}$.
Let $n_s = t-m$ and $m_s = t-n$.

Given indices (i,j) of some non-zero entry of A, let s_c represent the number of non-zero entries in column j among the first $i-1$ rows, and let s_r represent the number of non-zero entries in row i among the first $j-1$ columns. We construct an oracle O_s such that

$$O_s|j\rangle_n|s_c\rangle_{m_s} = |i\rangle_m|s_r\rangle_{n_s} \tag{4.13}$$

for all non-zero entries. O_s acts on t qubits. The input qubits are split into two registers of sizes n and m_s, whereas the output qubits are split into registers of sizes m and n_s. Given a non-zero entry of our matrix, one can either refer to it as the s_c-th non-zero entry of column j or the s_r-th non-zero entry of row i. This oracle, O_s, switches between these two representations. Let D be a $2^m \times 2^{n_s}$ matrix. The i-th row of this matrix consists of the non-zero entries of the i-th row of A in order, padded by zeros at the end. The last $2^m - M$ rows are filled with zeros. We then construct the FABLE matrix from [43] with respect to this matrix D to get the oracle O_D.

$$O_D|i\rangle_m|s_r\rangle_{n_s}(|0\rangle_F) = |i\rangle_m|s_r\rangle_{n_s}\left(\frac{A_{i,j}}{A_{max}}|0\rangle_F + \sqrt{1-\frac{A_{i,j}}{A_{max}}^2}|1\rangle_F\right) \tag{4.14}$$

We use these two oracles to construct the circuit depicted in Fig. 4.3. This circuit gives us a block encoding of A with subnormalization factor $\alpha = \sqrt{2^{m_s}2^{n_s}}A_{max}$. For sparse matrices, this factor will be lower than that of the straightforward FABLE scheme.

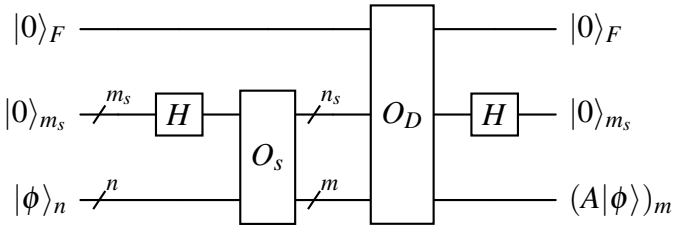


Figure 4.3: Sparse FABLE circuit

4.4.4 Quantum-accessible data structure

Section 2.2 of [39] introduces a block encoding scheme using the same data structure used in [7]. This method has a subnormalization factor of $\alpha = ||A||_F$ and can be implemented if one has access to operators P and Q such that:

$$P|i\rangle|0\rangle = \frac{1}{|A_i|}|i\rangle|A_i\rangle, \tag{4.15}$$

$$Q|0\rangle|j\rangle = \frac{1}{||A||_F}|\tilde{A}\rangle|j\rangle. \tag{4.16}$$

Here A_i represents the i-th row of A as a vector, and $\tilde{A}$ represents the vector whose i-th element is $|A_i|$.

4.4.5 Linear Combination of Unitaries

The LCU (Linear Combination of Unitaries) approach was introduced in [45]. If A can be written as $A = \sum_k c_k U_k$, where $\{U_k\}$ are unitaries, then this block encoding will have a subnormalization factor of $\alpha = \sum_k |c_k|$. The procedure to prepare this block encoding requires two operators *PREP* and *SELECT*.

$$PREP|0\rangle_1 := \sum_k \sqrt{\frac{c_k}{\alpha}}|k\rangle_1 \tag{4.17}$$

$$SELECT|k\rangle_1|\psi\rangle_2 := |k\rangle_1 U_k|\psi\rangle_2 \tag{4.18}$$

From these definitions it can be shown that

$$\langle 0|_1 PREP^\dagger SELECT PREP|0\rangle_1|\psi\rangle_2 = \frac{A}{\alpha}|\psi\rangle_2 \tag{4.19}$$

This method may find much use in physics and chemistry problems where Hamiltonians are often expressed as linear combinations of Pauli strings.

4.4.6 Novel method

This section is based on a publication made by the author and collaborators during the writing of this thesis [29].

Our goal in this section is to create a circuit U that takes in $|A_{flat}\rangle_1|\psi\rangle_2$ and returns a state $\frac{1}{\sqrt{2^n}}|0\rangle_1 \frac{A}{||A||_F}|\psi\rangle_2$, where A is a square matrix with 2^n rows and columns (if A does not already have this form, we can pad it with zeros accordingly) and $|A_{flat}\rangle$ is an amplitude encoding of $A/||A||_F$. Given a preparation operator O_P that takes $|0\rangle_1$ to $|A_{flat}\rangle_2$, UO_P will be a block encoding of A with $\alpha = ||A||_F$. This method of block encoding has not been proposed before in prior literature to the best of our knowledge.

Notation

1. We define the following matrices based on the Pauli basis matrices:

$$P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$
$$P_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad P_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2. We define $P_{s_1,\ldots,s_n} := P_{s_1} \otimes \cdots \otimes P_{s_n}$.

3. There exists a unique set $\{c_{s_1,\ldots,s_n}\}$ such that $A = \sum_{s_1,\ldots,s_n} c_{s_1,\ldots,s_n} P_{s_1,\ldots,s_n}$. Or, for convenience

$$A = \sum_{\vec{s}} c_{\vec{s}} P_{\vec{s}} \tag{4.20}$$

Circuit registers

Our circuit will have a data register of $2n$ block-determining qubits denoted *dat* and we will use them to store the entries of M in amplitude encoding. We will also have an input state register of n target qubits denoted *isr* that will be used to store the input state $|\psi\rangle$.

Encoding our matrix as a statevector

First we need to reshape our matrix into a one dimensional vector. The intuitive way to flatten a matrix A into vector A_{flat} for amplitude encoding is by ordering the matrix elements row by row. The problem with this for our purposes is that this ordering does not preserve tensor products. So instead we order our matrix elements using a Z-order space filling curve as shown in Fig. 4.4.

Now, if $X \otimes Y = Z$, then $X_{flat} \otimes Y_{flat} = Z_{flat}$. To prove this, let G map square matrices to their Z-order flattened vectors and let l be a function that maps (i, j) to k such that

$$\text{revbin}(k)[p] = \begin{cases} \text{revbin}(i)[p//2] & \text{if } p \text{ is odd} \\ \text{revbin}(j)[p//2] & \text{if } p \text{ is even} \end{cases}$$

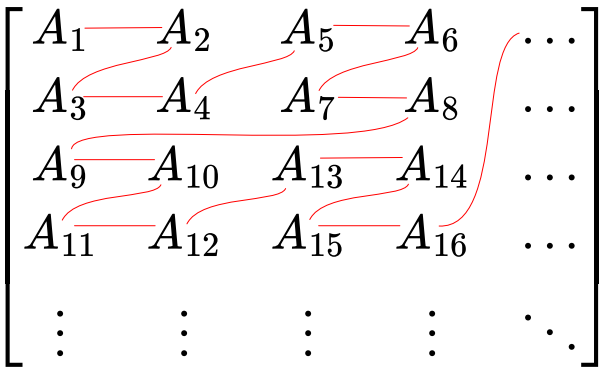


Figure 4.4: Ordering of A_{flat}

Here, revbin$(x)[p]$ is the p-th element (starting from 0) of the reversed binary bitstring representing x and $//$ represents floored division. Essentially, l takes two numbers and interleaves the bits in their binary representations to generate a new number. Given how Z-ordering is defined, it can be seen that $(G(X))_{l(i,j)} = X_{ij}$ (Note that this means that a row-by-row amplitude encoding of the matrix can be converted to a Z-ordered amplitude encoding by just reordering the qubits). What we need to prove now is that $(G(X \otimes Y))_m = (G(X) \otimes G(Y))_m$ for all m. Let $(i, j) = l^{-1}(m)$ and let the dimensions of Y be $2^q \times 2^q$.

$$\begin{aligned} LHS &= (G(X \otimes Y))_m = (X \otimes Y)_{ij} \\ &= X_{i//2^q, j//2^q} Y_{i\%2^q, j\%2^q} \end{aligned} \tag{4.21}$$

The last equality comes from the definition of tensor products. Here, the $\%$ symbol represents the remainder operator.

$$\begin{aligned} RHS &= (G(X) \otimes G(Y))_{l(i,j)} \\ &= (G(X))_{l(i,j)//2^{2q}} (G(Y))_{l(i,j)\%2^{2q}} && (4.22) \\ &= (G(X))_{l(i//2^q, j//2^q)} (G(Y))_{l(i\%2^q, j\%2^q)} && (4.23) \\ &= X_{i//2^q, j//2^q} Y_{i\%2^q, j\%2^q} \end{aligned}$$

To explain how we get from (4.22) to (4.23), we note that $l(i, j)//2^{2q}$ is what we get when we interleave the bitstring representations of i and j and then get rid of the $2q$ least significant bits. We can get the same result by first getting rid of the q least significant bits from both i and j and then interleaving their bits. This shows that $l(i, j)//2^{2q} = l(i//2^q, j//2^q)$. A similar explanation holds for $l(i, j)\%2^{2q} = l(i//2^q, j\%2^q)$.

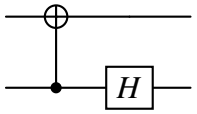
Let $|A_{flat}\rangle$ represent a normalized statevector that's an amplitude encoding of A_{flat}.

$$\begin{aligned} |A_{flat}\rangle_{dat} &= \sum_{\vec{s}} c^A_{\vec{s}} |P_{\vec{s},flat}\rangle_{dat} \\ &= \sum_{\vec{s}} c^A_{\vec{s}} (|P_{s_1,flat}\rangle \otimes \cdots \otimes |P_{s_n,flat}\rangle)_{dat} \end{aligned} \tag{4.24}$$

where $|P_{\vec{s},flat}\rangle$ and $|P_{i,flat}\rangle$ represent flattened unnormalized statevectors of $P_{\vec{s}}$ and P_i respectively; and $\frac{A}{||A||_F} = \sum_{\vec{s}} c^A_{\vec{s}} P_{\vec{s}}$.

Bell states and $\{|P_{i,flat}\rangle\}$

Closer inspection of $\{|P_{i,flat}\rangle\}$ reveals that these are the Bell states up to a scaling factor. We can use circuit B from Fig. 4.5 to go from the Bell basis to the computational basis.



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

Figure 4.5: Circuit B

$$\begin{aligned} B|P_{0,flat}\rangle &= \sqrt{2}|00\rangle \equiv \sqrt{2}|0\rangle \\ B|P_{1,flat}\rangle &= \sqrt{2}|01\rangle \equiv \sqrt{2}|1\rangle \\ B|P_{2,flat}\rangle &= \sqrt{2}|11\rangle \equiv \sqrt{2}|3\rangle \\ B|P_{3,flat}\rangle &= \sqrt{2}|10\rangle \equiv \sqrt{2}|2\rangle \end{aligned}$$

Let g be the map that takes $0,1,2,3$ to $0,1,3,2$ respectively.

$$B|P_{i,flat}\rangle = \sqrt{2}|g(i)\rangle \tag{4.25}$$

Taking the amplitude encoded state that we had prepared in (4.24), we apply B to every consecutive pair of qubits in dat to get the following state:

$$\sum_{\vec{s}} c^A_{\vec{s}} (B|P_{s_1,flat}\rangle \otimes \cdots \otimes B|P_{s_n,flat}\rangle)_{dat} = \sqrt{2^n} \sum_{\vec{s}} c^A_{\vec{s}} (|g(s_1)\rangle \otimes \cdots \otimes |g(s_n)\rangle)_{dat} \tag{4.26}$$

Applying $P_{\vec{s}}$ on our input state

The circuit K in Fig. 4.6 takes a state $|g(i)\rangle|\phi\rangle$ and returns the state $|g(i)\rangle(P_i|\phi\rangle)$.

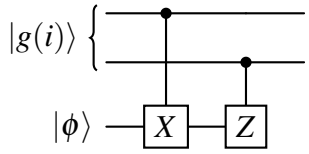


Figure 4.6: Circuit K

Now, let the isr qubits be in some state $|\psi\rangle$. We apply K on each pair of qubits in dat and the corresponding qubit in isr to produce the following state

$$\begin{aligned} &\sqrt{2^n} \sum_{\vec{s}} c^A_{\vec{s}} (|g(s_1)\rangle \otimes \cdots \otimes |g(s_n)\rangle)_{dat} (P_{s_1} \otimes \cdots \otimes P_{s_n}) |\psi\rangle_{isr} \\ &= \sqrt{2^n} \sum_{\vec{s}} c^A_{\vec{s}} |g(s_1) \ldots g(s_n)\rangle_{dat} P_{\vec{s}} |\psi\rangle_{isr} =: |\Phi\rangle \end{aligned} \tag{4.27}$$

Applying Hadamard gates on the dat qubits yields us the final state we required:

$$\frac{1}{\sqrt{2^n}} \sum_{\vec{s}} c^A_{\vec{s}} |0\rangle_{dat} P_{\vec{s}} |\psi\rangle_{isr} = \frac{1}{\sqrt{2^n}||A||_F} |0\rangle_{dat} A|\psi\rangle_{isr} \tag{4.28}$$

The full circuit is depicted in Fig. 4.7.

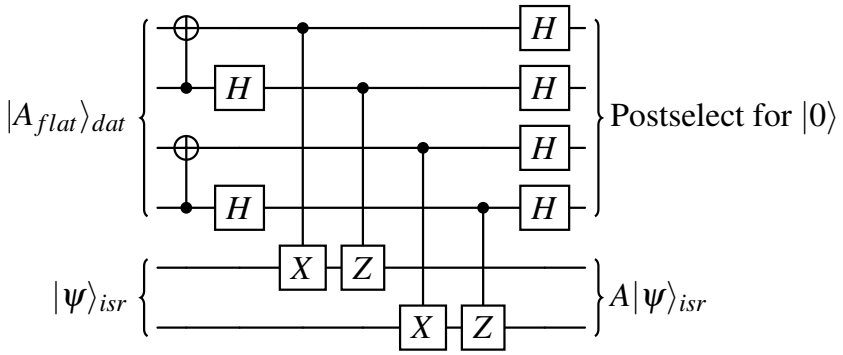


Figure 4.7: Full block encoding circuit for a 4×4 matrix A using the new method.

4.5 Running simulations

In this section, PennyLane is used to run simulations to test two applications of QSVT on the Naive block encoding implementation, FABLE, Sparse FABLE, the accessible Data Structure block encoding, and LCU. PennyLane's Adagrad optimizer [46] is used to obtain the optimal angleset $\vec{\phi}$.

4.5.1 Solving Linear Systems of Equations with QSVT

One of the applications of QSVT addressed in [17] is solving linear systems of equations (LSE). Other quantum algorithms for solving LSE problems exist, one of which is HHL [47]. HHL however requires Hamiltonian simulation to be used with a phase estimation step. The QSVT approach is arguably more straightforward. Given a general matrix A (not necessarily invertible, or even square for that matter) and b, we can find x such that $Ax = b$ using the Moore-Penrose Pseudoinverse [48]. If such an x doesn't exist, this method will return a 'best fit' solution (least squares).
If the SVD of A is given by $U\Sigma V^\dagger$, the pseudoinverse of A is given by $A^+ = V\Sigma^{-1}U^\dagger$.
Let P be a polynomial such that

$$\mathrm{Re}(P(y)) \approx f(y) = \frac{a}{y} \tag{4.29}$$

for some appropriate a. Given a block encoding of A, U_A, we can construct $QSVT_P(U_A^\dagger)$, a block encoding of $P(A) = VP(\Sigma)U^\dagger$ using QSVT. We can similarly construct $QSVT_P(U_A)^\dagger$ to get

a block encoding of $(P(A^\dagger))^\dagger = (UP(\Sigma)V^\dagger)^\dagger = VP^*(\Sigma)U^\dagger$. We can use the LCU method to combine these two unitaries in equal parts to obtain our final operator $W(A)$, a block encoding of $V(Re)(P(\Sigma))U^\dagger \approx A^+$.

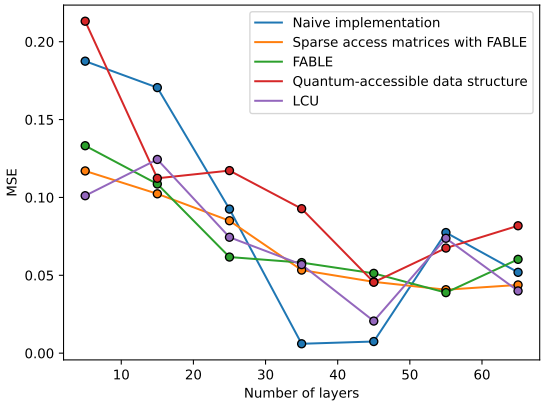


Figure 4.8: MSE vs Layer count for various block encodings when using QSVT to solve linear systems of equations

To test this method, 100 problem instances are generated. Each problem instance consists of a 4×4 matrix A whose each entry is sampled from a uniform distribution after which the matrix is normalized, setting its Frobenius norm to 1; a vector x of length 4, whose each entry is also sampled from a uniform distribution; and the vector $b = Ax$.
For a given block encoding, the lowest subnormalization factor α_{min} among all the matrices is noted. Since the Frobenius norm of every matrix A is set to 1, we know that the singular values encountered by P will be upper bounded by $\frac{1}{\alpha_{min}}$. We also have to pick a positive lower bound on the range of values for which we want (4.29) to hold, since the inverse function tends to infinity as we approach 0. The lower bound is chosen to be $a = \frac{0.1}{\alpha_{min}}$ and this value of a is plugged into (4.29). This ensures that the optimization happens for

$$f(y) \leq 1 \forall y \in [\frac{0.1}{\alpha_{min}}, \frac{1}{\alpha_{min}}]. \tag{4.30}$$

As mentioned earlier, PennyLane's Adagrad optimizer is used to get the optimal P (200 optimization steps). To evaluate the loss function for a given angleset $\vec{\phi}$, a QSP circuit is constructed with these angles to get the associated polynomial P'. 100 points are sampled uniformly from $[\frac{0.1}{\alpha_{min}}, \frac{1}{\alpha_{min}}]$ and the loss is evaluated as

$$\text{Loss} = \sum_{i=1}^{100} (P'(y_i) - f(y_i))^2. \tag{4.31}$$

Once the optimal angles are obtained, the circuit $W(A)$ is applied on a normalized amplitude encoding of b, $|0\rangle \otimes |b\rangle$ to obtain $|0\rangle \otimes |x\rangle + |\ldots\rangle$ from which an estimate of x is produced, $\bar{x}$. To measure the error associated with our computation, one should refrain from setting Error $= |x - \bar{x}|$, because A may not be invertible and could have multiple valid solutions for x. Instead the error is set to be Error $= |b - A\bar{x}|$. This procedure can be run over all 100 problem instances and the MSE can be calculated. The MSE for various block encodings and various layer counts has been plotted in Fig. 4.8. We can see the MSE generally decreasing as the layer count increases, as expected. But there are a few unexpected cases where this trend doesn't hold (for example, from 45 layers to 55 layers with the Naive block encoding implementation). It is not obvious why this is the case, though one explanation may be that we're not using enough optimization steps for the bigger ansatz sizes. More simulations need to be run with more optimization steps. Another unexpected observation we make from this figure is the level of overlap with graphs of different block encodings. This is in contrast with the TDA results we'll see soon in Fig. 4.9.

4.5.2 Topological Data Analysis with QSVT

Given the ubiquity of data science in the modern age, it is important to have the tools necessary to extract features out of large amounts of data. Topological Data Analysis [49, 50] provides an approach to analyze and extract topological features from a given dataset.

A plethora of work exists on the subject of using quantum algorithms for TDA [51, 52, 53].

The way TDA works is fairly straightforward. We are given data in the form of a point cloud. We treat each point as the vertex of a graph G. An edge joins two vertices if the distance between the points is less than some threshold ε. Having constructed a graph from a point cloud, we then further convert the graph G into a simplicial complex by converting every k-clique in our graph into a corresponding $k-1$-simplex. Topological features of this simplicial complex can then be studied to reveal information about our data.

One important set of topological features takes the form of Betti numbers. The k-th Betti number β_k^G is the number of k-dimensional holes in the complex constructed from G. The problem of finding the k-th Betti number of a graph can be reduced to the problem of finding the nullity of the Combinatorial Laplacian Δ_k^G [54] of the graph.

Let A be the matrix representation of Δ_k^G, padded with zeros to make it a $2^s \times 2^s$ matrix with 1s on the extra diagonal entries (The 1s are necessary to ensure that the nullity of A is the same as that of Δ_k^G). Let P be a polynomial such that

$$|P(x)| \approx \Theta(x) \tag{4.32}$$

where Θ is the signum function. If U_A is a block encoding of A, then we can use QSVT to construct $QSVT_P(U_A)$, a block encoding of $P(A)$, whose Frobenius norm squared is $\sum_{i=1}^{2^s} |P(\sigma_i)|^2 \approx \text{rank}(A)$. Nullity of Δ_k^G can therefore be estimated as

$$\text{nullity}(\Delta_k^G) = 2^s - ||P(A)||_F^2. \tag{4.33}$$

So now we need to find $||P(A)||_F^2$ given $QSVT_P(U_A)$. This can be done by preparing a maximally mixed state in the target qubits (and leaving the block determining qubits in the $|0\rangle$ state) and then applying $QSVT_P(U_A)$. The probability of measuring the block determining qubits in the $|0\rangle$ state is equal to $\frac{||P(A)||_F^2}{2^s}$.
Estimating this probability and plugging the obtained value of $||P(A)||_F^2$ into (4.33) will give us an estimate for $\beta_k^G = \text{nullity}(\Delta_k^G)$.

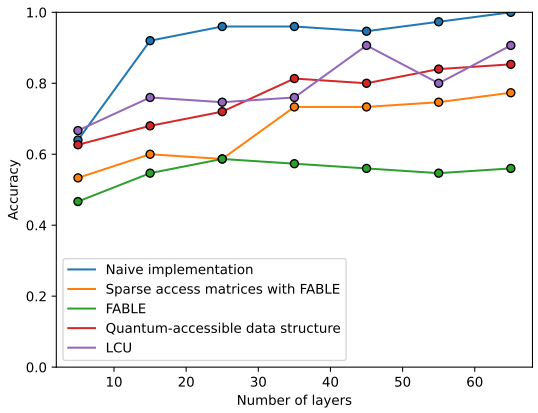
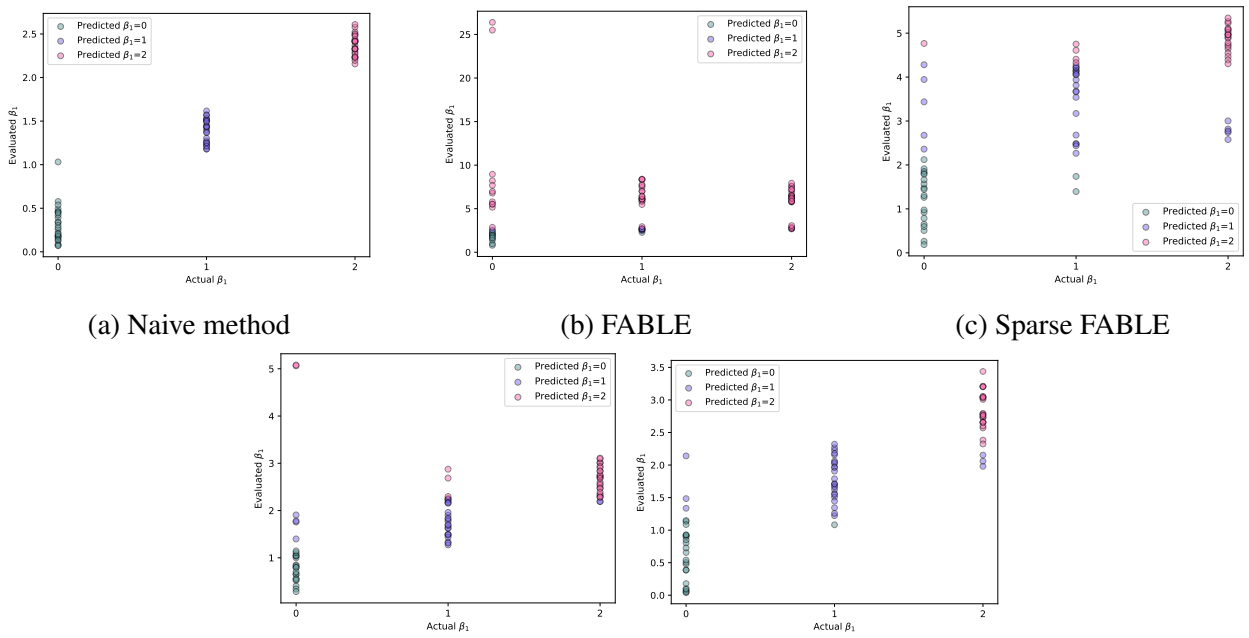


Figure 4.9: Accuracy vs Layer count for various block encodings when using QSVT for TDA

Simulations of this method were run on PennyLane to find the first Betti number, β_1, of randomly generated datasets with 7 points distributed uniformly in a 3-dimensional space of size $100 \times 100 \times 100$ with $\varepsilon = 65$. Random datasets were generated until 25 instances were obtained with $\beta_1 = 0$, 25 with $\beta_1 = 1$, and 25 with $\beta_1 = 2$. Our goal to solve with this toy problem is



(a) Naive method (b) FABLE (c) Sparse FABLE

(d) Accessible Data Structure (e) LCU

Figure 4.10: Estimated β_1 from QSVT vs True value of β_1 for various block encodings with 65 QSVT layers

classifying these 3 kinds of data through our QSVT-based estimates of their first Betti numbers. Following the same strategy used in the LSE simulations, PennyLane's Adagrad optimizer is used on QSP circuits to find the optimal $\vec{\phi}$ that approximates the signum function $\Theta(x)$ for $x \in [0, \frac{1}{\alpha_{min}}]$ (200 optimization steps). For a given block encoding scheme and layer count, the algorithm is run and a Betti number estimate $\bar{\beta}_1$ is produced for each dataset. For given thresholds τ_1 and τ_2, datasets with $\bar{\beta}_1 \leq \tau_1$ are placed in the $\beta_1 = 0$ class, datasets with $\bar{\beta}_1 > \tau_2$ in the $\beta_1 = 2$ class, and the rest of the datasets in the $\beta_1 = 1$ class. The accuracy is calculated as the fraction of accurately classified datasets. For a given block encoding scheme and layer count, τ_1 and τ_2 are picked to maximize this accuracy. This maximum accuracy is plotted against layer counts for various block encodings in Fig. 4.9.

Unlike the results of LSE, here we see a clear difference in performance using different block encodings. With each block encoding, we observe the general downward trend of MSE with respect to layer count. However, just like in our LSE results, the trend is not consistent. More simulations with more optimization steps need to be run.

Fig. 4.10 depicts how the classification of the datasets is done using different block encodings

with 65 layers. We see that the Naive block encoding implementation classifies the datasets into very neat clusters and performs better than all others. It is also the only case where all the datasets are classified correctly. The Accessible Data Structure block encoding and the LCU block encoding also seem to perform pretty well. Sparse FABLE block encoding is not great but still performs better than FABLE, which displays the worst performance. This is, of course, all in line with what we see in Fig. 4.9 when the layer count is 65.

See Section 6.3 for discussion and future works.

Chapter 5

Quantum Algorithms to compute t-SVD

The t-SVD [18] is one of many generalizations of SVD for third-order tensors. It has various applications such as video denoising, context-aware recommendation systems, and image classification. One of the things that make a quantum approach to t-SVD promising [19] is that the t-SVD involves computations in the Fourier space, and applying a Fourier Transform is something quantum computers are inherently good at. In this chapter, we will first acquaint ourselves with notation and define what a t-SVD is in Section 5.1. Then in the next two sections we will go through two new quantum algorithms to obtain the t-SVD of a third-order tensor that are analogous to the SVD algorithms we saw in Chapter 2 and Chapter 3. The first quantum t-SVD algorithm in Section 5.2 is based on the Quantum SVE routine in [7]. We'll discuss how we can use this to create a type of recommendation system which is exponentially faster than any known classical algorithms for the same. The second one in Section 5.3 is a variational algorithm based on the quantum SVD algorithm from Chapter 3 and our new block encoding from Section 4.4.6.

5.1 Notation

Consider an order-3 tensor $\mathcal{A} \in \mathbb{R}^{M\times N\times L}$ with elements a_{ijk}; $i < M, j < N, k < L$.
$A^{(k)}$ or $\mathcal{A}[:,:,k]$ will denote the k-th face of $\mathcal{A}$ which is defined as the matrix whose (i,j)-th element is a_{ijk}.
Similarly we can define the (i,j)-th tube of $\mathcal{A}$, denoted by a^{ij} or $\mathcal{A}[i,j,:]$, as the vector whose k-th entry is a_{ijk}.

Let $\tilde{\mathcal{A}} \in \mathbb{R}^{M\times N\times L}$ be the tensor whose (i,j)-th tube $\tilde{\mathcal{A}}[i,j,:]$ is the Fourier transform of a^{ij}, denoted $\tilde{a}^{ij}$. The tilde will generally denote a Fourier transform applied along this third dimension. The k-th face of $\tilde{\mathcal{A}}$ is denoted by $\tilde{A}^{(k)}$ or $\tilde{\mathcal{A}}[:,:,k]$.

To find the t-SVD of an order-3 tensor $\mathcal{A} \in \mathbb{R}^{M\times N\times L}$ is to find order-3 tensors $\mathcal{U} \in \mathbb{R}^{M\times M\times L}$, $\mathcal{S} \in \mathbb{R}^{M\times N\times L}$, and $\mathcal{V} \in \mathbb{R}^{N\times N\times L}$ such that the SVD of $\tilde{A}^{(k)}$ is $U^{(k)}S^{(k)}V^{(k)\dagger}$ for all $0 \leq k < L$ [18].

In this chapter, we will choose to refer to the entries of $\tilde{\mathcal{S}}$ as the singular values of $\mathcal{A}$, denoted as

$$\tilde{\sigma}_i^{(k)} := \tilde{\mathcal{S}}[i,i,k]. \tag{5.1}$$

We will also define $\tilde{u_i}^{(k)}$ and $\tilde{v_i}^{(k)}$ as follows:

$$\tilde{u_i}^{(k)} := \tilde{\mathcal{U}}[:,i,k] \tag{5.2}$$
$$\tilde{v_i}^{(k)} := \tilde{\mathcal{V}}[:,i,k] \tag{5.3}$$

5.2 Purely quantum algorithm for tensor completion using t-SVD

In this section, we will first explore the limitations of an existing quantum t-SVD algorithm, thus indicating the opportunity for an improved solution. We then go through a procedure that serves the same purpose as the Quantum SVE from [7] but adapted for the t-SVD. Lastly, we will see how this routine can be used for tensor completion and then for recommendation systems.

5.2.1 Drawbacks of existing algorithm

Introduced in [19] is a quantum t-SVD recommendation algorithm based on the Quantum Singular Value Estimation approach we used in Chapter 2. That work however uses a linear combination

of unitaries (See Section 4.4.5) approach to create a block encoding of the required walk operator that they then need to perform Quantum Phase Estimation on (Appendix B of [19]). This step is fairly nontrivial and its most straightforward application would require an auxiliary register of qubits whose size grows exponentially with the number of phase register qubits (Lemma 4 (Product of block-encoded matrices) in [39]). It also makes strong implicit assumptions of the tensor in question. Appendix B of [19] defines U_{P_k} and $U_{\hat{P}_m}$ as follows:

$$U_{P_k}|i\rangle|0\rangle = \frac{1}{||\mathcal{A}(i,:,k)||_2}\sum_{j=0}^{N-1} a_{ijk}|i\rangle|j\rangle \tag{5.4}$$

$$U_{\hat{P}_m} \triangleq \frac{1}{L}\sum_{k=0}^{L-1} \omega^{km} U_{P_k} \tag{5.5}$$

It is then asserted that $U_{\hat{P}_m}|i\rangle|0\rangle = |i\rangle|\tilde{\mathcal{A}}[i,:,m]\rangle$. For a given i and up to a scaling factor, this can only hold under the assumption that the norm $||\mathcal{A}(i,:,k)||_2$ is the same for all values of k.

The quantum algorithm proposed in this section does not contain these drawbacks. It is fundamentally different in its procedure and works by modifying the Quantum SVE routine itself.

5.2.2 Quantum t-SVE

As proposed in [7], the Quantum SVE with respect to a matrix A is used to extract the singular value corresponding to a given amplitude encoded state of the left singular vector of A. Similarly, the Quantum t-SVE algorithm proposed here, when applied with respect to a tensor $\mathcal{A}$, can be used to extract the singular value $\tilde{\sigma}_i^{(k)}$ from the corresponding statevector $|\tilde{v}_i^{(k)}\rangle|\tilde{k}\rangle$.

Circuit registers

This procedure makes use of 6 registers:

1. Walk operator left singular vector register (1), with $n_m := \lceil \log_2(M) \rceil$ qubits
2. Right singular vector register (2), with $n_n := \lceil \log_2(N) \rceil$ qubits
3. Walk operator face register (3), with $n_l := \lceil \log_2(L) \rceil$ qubits
4. Face register (4), with n_l qubits
5. Phase register $(phase)$, with n_p qubits
6. Abs Phase register (abs), with d qubits

Data structure requirements

As input to this procedure we need access to operators P' and Q' such that

$$\begin{aligned} & P'_{1234}|i\rangle_1|0\rangle_2|\tilde{k}\rangle_3|0\rangle_4 \\ =& \frac{1}{||\mathcal{A}||_F}\sum_j ||\mathcal{A}[:,j,:]||_F|i\rangle_1|j\rangle_2|\tilde{k}\rangle_3|\tilde{k}\rangle_4 \qquad (5.6) \\ & Q'_{1234}|0\rangle_1|j\rangle_2|0\rangle_3|\tilde{l}\rangle_4 \\ =& \frac{1}{||\mathcal{A}[:,j,:]||_F}\sum_{i,k} a_{ijk}|i\rangle_1|j\rangle_2|k\rangle_3|\tilde{l}\rangle_4. \qquad (5.7) \end{aligned}$$

P' can of course be further decomposed as an operator P'' acting on register 2, and an operation copying register 3 to register 4 in the Fourier basis. P'' is defined such that

$$P''|0\rangle = \frac{1}{||\mathcal{A}||_F}\sum_j ||\mathcal{A}[:,j,:]||_F|j\rangle. \qquad (5.8)$$

The same data structure proposed in [7] can be used to store a matrix whose $(j, i\times 2^{n_l}+k)$-th entry is $\mathcal{A}[i,j,k]$ and, with such a data structure, the operators P'' and Q' can be applied with an associated circuit depth that's polylog(LMN). The circuit depth associated with copying register 3 to register 4 in the Fourier basis is $O(n_l)$ which means the circuit depth spent in applying P' is also polylog(LMN).

Algorithm Input-Output and Complexity

Let $\mathcal{A}$ be the given tensor stored in the data structure described in Section 5.2.2. Define $\theta_i^{(k)}$ as follows:

$$\theta_i^{(k)} = 2\arccos\frac{\tilde{\sigma}_i^{(k)}}{||\mathcal{A}||_F} \tag{5.9}$$

Let $\delta > 0$ be the smallest difference between θ values.

$$\delta := \min_{i,j,k,l}(|\theta_i^{(k)} - \theta_j^{(l)}|) \tag{5.10}$$

We pick d to be an integer such that

$$2^{-d} < \frac{\delta}{2\pi}. \tag{5.11}$$

Let $\varepsilon > 0$ be the precision factor.

Then, we can construct a circuit $QtSVE$ such that

$$\begin{aligned} &QtSVE|\tilde{v}_i^{(k)}\rangle_2|\tilde{k}\rangle_4 \\ =&|\tilde{v}_i^{(k)}\rangle_2|\tilde{k}\rangle_4|\theta_i^{(k)}\rangle_{abs} + \sqrt{2}\varepsilon|\ldots\rangle. \end{aligned} \tag{5.12}$$

Here, $|\theta_i^{(k)}\rangle$ represents a state whose first d qubits form a basis state approximation of $2^d \times \theta_i^{(k)}/(2\pi)$.

We can justify the approximation in Eq. (5.12) by taking the binomial expansion of $\sqrt{1-\varepsilon^2}$ and discarding the higher power terms, since $\varepsilon << 1$ ideally.

While we could employ quantum arithmetic to further compute $|\tilde{\sigma}_i^{(k)}\rangle$, it is not a necessary step for the algorithms discussed in this paper. And if we wanted to use this routine directly to compute

and measure the singular values of a tensor, it would be more practical to measure $|\theta_i^{(k)}\rangle$ and then use it to compute $\tilde{\sigma}_i^{(k)}$ classically.

The time complexity of this procedure is given by

$$\text{Circuit depth} = O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.13}$$

Theoretical underpinnings

Our aim would be to construct operators $2PP^\dagger - \mathbb{1}$ and $2QQ^\dagger - \mathbb{1}$ for a P and Q that satisfy the following properties

$$P^\dagger P = \mathbb{1} \tag{5.14}$$
$$Q^\dagger Q = \mathbb{1} \tag{5.15}$$
$$P^\dagger Q = \sum_k A^{(k)} |\tilde{k}\rangle\langle\tilde{k}| \tag{5.16}$$

where $|\tilde{k}\rangle = QFT|k\rangle$.

To do this, we define matrices P and Q like so

$$P = \frac{1}{||\mathcal{A}||_F} \sum_{i,j,k} ||\mathcal{A}[:,j,:]||_F (|i\rangle|j\rangle|\tilde{k}\rangle|\tilde{k}\rangle)(\langle i|\langle\tilde{k}|) \tag{5.17}$$
$$Q = \frac{1}{||\mathcal{A}[:,j,:]||_F} \sum_{i,j,k} a_{ijk} (|i\rangle|j\rangle|k\rangle|\tilde{l}\rangle)(\langle j|\langle\tilde{l}|) \tag{5.18}$$

To implement $W = (2PP^\dagger - \mathbb{1})(2QQ^\dagger - \mathbb{1})$, we can use our operations P' and Q'.

$$2PP^\dagger - \mathbb{1} = P'_{1234}(2|0\rangle_2|0\rangle_4\langle 0|_2\langle 0|_4 - \mathbb{1})P'^\dagger_{1234} \tag{5.19}$$

Similarly,

$$2QQ^{\dagger} - \mathbb{1} = Q'_{1234}(2|0\rangle_1|0\rangle_3\langle 0|_1\langle 0|_3 - \mathbb{1})Q'^{\dagger}_{1234} \tag{5.20}$$

Just like in the Quantum SVE, we create a walk operator

$$W = (2PP^{\dagger} - \mathbb{1})(2QQ^{\dagger} - \mathbb{1}). \tag{5.21}$$

Then, from Jordan's lemma [21], we can then say that the following states are eigenvectors of W:

$$|\Psi^{(k)}_{i\pm}\rangle_{24} = \frac{Q|\tilde{v}_i{}^{(k)}\rangle_2|\tilde{k}\rangle_4 \mp i(Q|\tilde{v}_i{}^{(k)}\rangle_2|\tilde{k}\rangle_4)^{\perp}}{\sqrt{2}} \tag{5.22}$$

with eigenvalues $\exp(\pm\theta_i^{(k)})$.

Algorithm procedure

Now we will see how we can run the Quantum t-SVE routine on our state $|\tilde{v}_i{}^{(k)}\rangle|\tilde{k}\rangle$.

Step 1: First we apply Q' on $|\tilde{v}_i{}^{(k)}\rangle|\tilde{k}\rangle$ and two other empty registers to get

$$|\psi_1\rangle = Q'_{1234}|0\rangle_1|\tilde{v}_i{}^{(k)}\rangle_2|0\rangle_3|\tilde{k}\rangle_4 = Q|\tilde{v}_i{}^{(k)}\rangle|\tilde{k}\rangle. \tag{5.23}$$

This step contributes a circuit depth of:

$$\text{Circuit depth} = O(\text{polylog}(LMN)) \tag{5.24}$$

Step 2: Then we apply Quantum Phase Estimation with respect to W on this state with

$$n_p = d + \lceil \log_2(2 + \frac{1}{2\varepsilon^2}) \rceil. \tag{5.25}$$

This yields us the following state:

$$|\psi_2\rangle = \frac{|\Psi_{i+}^{(k)}\rangle_{24}|\Theta_i^{(k)}\rangle_{phase} + |\Psi_{i-}^{(k)}\rangle_{24}|-\Theta_i^{(k)}\rangle_{phase}}{\sqrt{2}} \tag{5.26}$$

Here we define $|\Theta_i^{(k)}\rangle$ as follows:

$$|\Theta_i^{(k)}\rangle = \sqrt{1-\varepsilon^2}|\theta_i^{(k)}\rangle + \varepsilon|\ldots\rangle \tag{5.27}$$

The state $|\theta_i^{(k)}\rangle$ is defined in the same way that it was in (5.12).

This step contributes a circuit depth of:

$$\text{Circuit depth} = O(\text{polylog}(LMN) \times 2^{n_p}) \tag{5.28}$$

$$= O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right) \tag{5.29}$$

Step 3: Next we copy the absolute value of $\theta_i^{(k)}$ to a new register and uncompute the QPE step. Now we get the state

$$|\psi_3\rangle = Q'_{1234}|0\rangle_1|\tilde{v_i}^{(k)}\rangle_2|0\rangle_3|\tilde{k}\rangle_4|\tilde{\theta}_i^{(k)}\rangle_{abs} + \sqrt{2}\varepsilon|\ldots\rangle, \tag{5.30}$$

where $|\tilde{\theta}_i^{(k)}\rangle$ represents the basis state approximation of $2^d \times \theta_i^{(k)}/(2\pi)$.

To do this, we need an operator K such that

$$K|x\rangle|0\rangle = |x\rangle||x|\rangle \tag{5.31}$$

To implement this, we can first use Toffoli gates to copy the first d qubits of the phase register to the absolute phase register controlled for when the most significant qubit is $|0\rangle$ ($O(n_p)$), i.e., when the phase register is in some basis state $|x\rangle$ where $x \geq 0$. Then we can use the QFT weighted

adder from [22] to set the absolute phase register to $|-x\rangle$ controlled on when the most significant qubit is in the $|1\rangle$ state ($O(n_p^2)$). The circuit depth of K is thus $O(n_p^2)$. This depth is insignificant compared to uncomputing the QPE step, so the time complexity of this step again comes out to be

$$\text{Circuit depth} = O(\text{polylog}(LMN) \times 2^{n_p}) \tag{5.32}$$

$$= O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.33}$$

Step 4: Applying $Q'^{\dagger}$, we finally get the output state we wanted:

$$|\psi_4\rangle = |\tilde{v}_i^{(k)}\rangle_2 |\tilde{k}\rangle_4 |\tilde{\theta}_i^{(k)}\rangle_{abs} + \sqrt{2}\varepsilon|\dots\rangle \tag{5.34}$$

This step contributes to a circuit depth of

$$\text{Circuit depth} = O(\text{polylog}(LMN)) \tag{5.35}$$

We will represent this whole procedure as an operator denoted by $QtSVE_{1234}$. The leading contribution to the circuit depth comes from the QPE step, and so the time complexity of this operation is given by

$$\text{Circuit depth} = O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.36}$$

5.2.3 Tensor truncation with t-SVD

We define $\mathcal{A}$, $\theta_i^{(k)}$, δ, d, ε as they were defined in Section 5.2.2 Let $\mathcal{A}_{trunc}^{\tau}$ be the truncated tensor obtained by setting all the singular values of $\mathcal{A}$ less than τ to 0.

The output of this algorithm will be the state

$$|\mathcal{A}^{\tau}_{trunc}\rangle + 2\varepsilon|\ldots\rangle \tag{5.37}$$

with the probability of success being α^2. We define α as

$$\alpha := \frac{||\mathcal{A}^{\tau}_{trunc}||_F}{||\mathcal{A}||_F} \tag{5.38}$$

Circuit registers

This circuit is depicted in Fig. 5.1 and makes use of 8 registers:

1. Left singular vector register (1), with $n_m := \lceil \log_2(M) \rceil$ qubits
2. Right singular vector register (2), with $n_n := \lceil \log_2(N) \rceil$ qubits
3. Face register (3), with $n_l := \lceil \log_2(L) \rceil$ qubits
4. Walk operator face register (4), with n_l qubits
5. Walk operator left singular vector register 2 (5), with n_m qubits
6. Phase register (6), with n_p qubits
7. Abs Phase register (7), with d qubits
8. Flag qubit (F)

Algorithm procedure

We begin with all qubits initialized to the $|0\rangle$ state. The procedure is as follows.

Step 1: Prepare the state

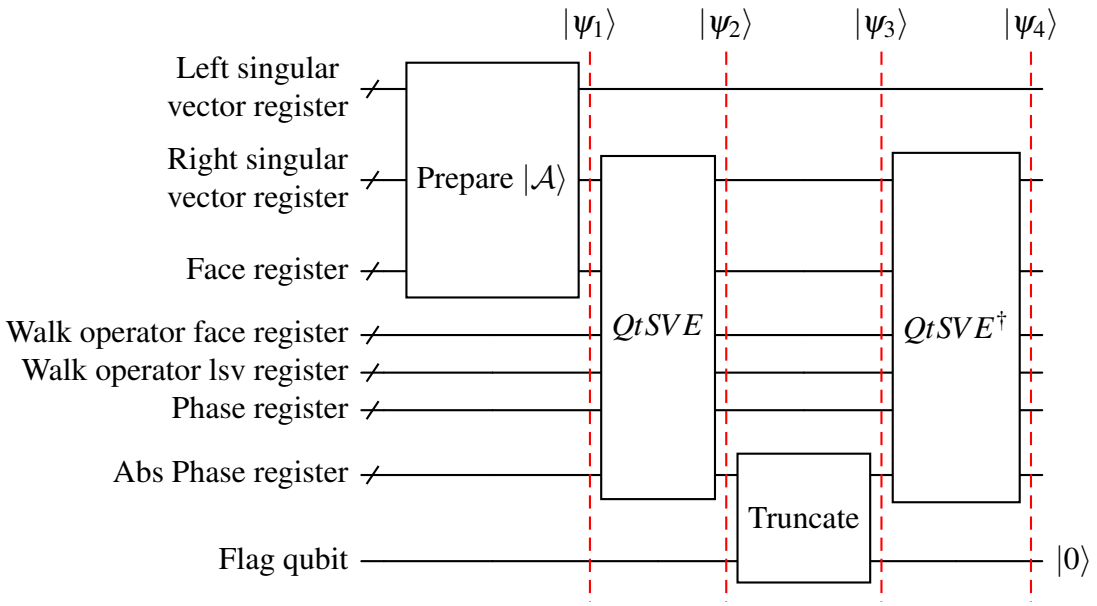


Figure 5.1: Circuit diagram of the purely quantum t-SVD algorithm

$$|\psi_1\rangle = \frac{1}{||\mathcal{A}||_F}\sum_{ijk} a_{ijk}|i\rangle_1|j\rangle_2|k\rangle_3 \tag{5.39}$$

$$= \frac{1}{||\mathcal{A}||_F}\sum_{ijk} \tilde{a}_{ijk}|i\rangle_1|j\rangle_2|\tilde{k}\rangle_3$$

$$= \frac{1}{||\mathcal{A}||_F}\sum_{tk} \tilde{\sigma}_t^{(k)}|\tilde{u}_t^{(k)}\rangle_1|\tilde{v}_t^{(k)}\rangle_2|\tilde{k}\rangle_3. \tag{5.40}$$

To get this state, we start with $|0\rangle_1|0\rangle_2|0\rangle_3|0\rangle_4$ and apply Hadamard gates on all the qubits in register 3 to get $|0\rangle_1|0\rangle_2|\tilde{0}\rangle_3|0\rangle_4$. Then we apply P'_{1234} to get

$$P'_{1234}|0\rangle_1|0\rangle_2|\tilde{0}\rangle_3|0\rangle_4$$

$$= \frac{1}{||\mathcal{A}||_F}\sum_j ||\mathcal{A}[:,j,:]||_F|0\rangle_1|j\rangle_2|\tilde{0}\rangle_3|\tilde{0}\rangle_4 \tag{5.41}$$

Next, we apply Hadamard gates on registers 3 and 4 to reset those registers to $|0\rangle$ and then apply Q'_{1234} to finally get

$$|\psi_1\rangle = \frac{1}{||\mathcal{A}||_F}\sum_{ijk} a_{ijk}|i\rangle_1|j\rangle_2|k\rangle_3|0\rangle_4. \tag{5.42}$$

This step contributes a circuit depth of:

$$\text{Circuit depth} = \mathcal{O}(\text{polylog}(LMN)) \tag{5.43}$$

For **Step 2:** Apply the operator $QtSVE_{5243}$ to store $|\tilde{\theta}_t^{(k)}\rangle$ in a new register.

$$|\psi_2\rangle = \frac{1}{||\mathcal{A}||_F}\sum_{tk} \tilde{\sigma}_t^{(k)}|\tilde{u}_t^{(k)}\rangle_1|\tilde{v}_t^{(k)}\rangle_2|\tilde{k}\rangle_3|\tilde{\theta}_t^{(k)}\rangle_7 + \sqrt{2}\varepsilon|\dots\rangle \tag{5.44}$$

This step contributes a circuit depth of:

$$\text{Circuit depth} = O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.45}$$

Step 3: Use a flag qubit to separate the singular values above a given threshold τ.

$$\begin{aligned}|\psi_3\rangle = &\frac{1}{||\mathcal{A}||_F}\sum_{\sigma_t^{(k)}>\tau} \tilde{\sigma}_t^{(k)}|\tilde{u}_t^{(k)}\rangle_1|\tilde{v}_t^{(k)}\rangle_2|\tilde{k}\rangle_3|\tilde{\theta}_t^{(k)}\rangle_7|0\rangle_F \\ &+ \sqrt{1-\alpha^2}|\dots\rangle|1\rangle_F + \sqrt{2}\varepsilon|\dots\rangle\end{aligned} \tag{5.46}$$

For this, we need to construct an operator G such that

$$G|\theta\rangle_7|0\rangle_F = |\theta\rangle_7|f_{\tau/||\mathcal{A}||_F}(\theta)\rangle_F. \tag{5.47}$$

where we define

$$f_a(\theta) := \begin{cases} 1 & \cos\frac{\theta}{2} < a \\ 0 & \cos\frac{\theta}{2} \geq a \end{cases} \tag{5.48}$$

Since $\theta \in [0, \pi]$, the condition $\cos\frac{\theta}{2} < a$ is equivalent to $\theta > 2\arccos a$. This can be implemented using a simple integer comparator [25] on the first d qubits of the absolute phase register. This circuit has time complexity:

$$\text{Circuit depth} = O\left(\frac{1}{\delta}\right) \tag{5.49}$$

Step 4: Apply the inverse of the $QtSVE$ operator to get the output state:

$$|\psi_4\rangle = \alpha|\mathcal{A}^{\tau}_{trunc}\rangle|0\rangle_F + \sqrt{1-\alpha^2}|\ldots\rangle|1\rangle_F + 2\varepsilon|\ldots\rangle \tag{5.50}$$

Then we postselect for the flag qubit to be in the $|0\rangle_F$ state. The minimum probability of measuring this state can be calculated as:

$$\text{Probability of success} > (\alpha - 2\varepsilon)^2 \tag{5.51}$$

We now write down the form of the final output state after postselection, while making worst-case assumptions regarding the error terms:

$$|\psi\rangle \approx |\mathcal{A}^{\tau}_{trunc}\rangle + 2\varepsilon|\ldots\rangle. \tag{5.52}$$

Time complexity:

$$\text{Circuit depth} = O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.53}$$

5.2.4 Quantum t-SVD Recommendation System

We may turn this into a quantum recommendation system that can take in the index of an input entry, i, and recommend a primary output index, j, along with an associated value, k. For example, a potential use-case would be if a store wanted to email discount codes for a certain product (i) to a few customers. They would have to decide which customer (j) to target this discount to, and on what day of the week (k) the customer should be sent these emails (based on when the customer tends to do their shopping) in order to maximize incentive. To do so, we employ tensor truncation as a method of tensor completion (similar to [19], and to how [7] uses matrix truncation as a method of matrix completion). Given an incomplete preference tensor $\mathcal{A}$ whose unknown elements are set to 0, we assume our data is structured such that a t-SVD truncation of this incomplete tensor approximates the true complete preference tensor $\mathcal{A}^{\tau}_{trunc}$.

To run this algorithm, we need to follow the same procedure as in Section 5.2.3 except we change the initial state in Step 1. Here, we start by preparing the basis state $|i\rangle_1$ as input. Then we apply an operator R such that

$$R|i\rangle_1|0\rangle_2|0\rangle_3 = \frac{1}{||\mathcal{A}[i,:,:]||_F}\sum_{jk} a_{ijk}|i\rangle_1|j\rangle_2|k\rangle_3 \tag{5.54}$$

One way such an operator can be constructed is by following the same method with which Q' was constructed, using an additional data structure to store a matrix whose $(i, j \times 2^{n_l} + k)$-th entry is $\mathcal{A}[i,j,k]$.

After following the rest of the procedure from Section 5.2.3, we postselect for $|0\rangle_F$ and get the following state:

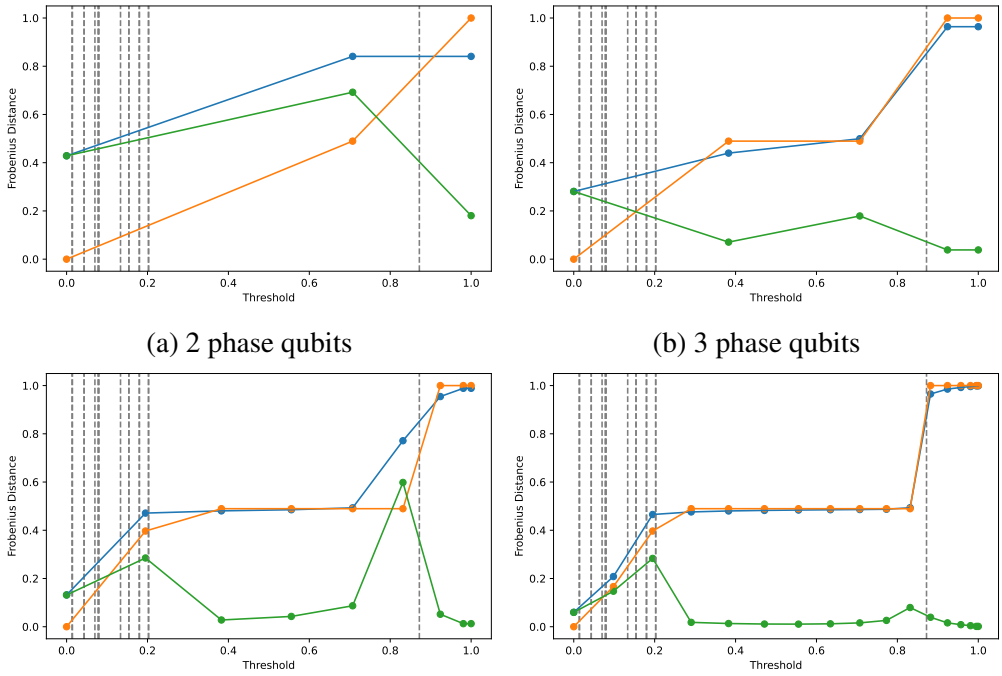
$$|\psi\rangle \approx |i\rangle|\mathcal{A}^{\tau}_{trunc}[i,:,:]\rangle + 2\varepsilon|\dots\rangle \tag{5.55}$$

We can then make a measurement on registers 2 and 3 to get our recommendations j and k respectively. We only need to make one successful measurement shot to get a recommendation, and the probability of measuring $|0\rangle_F$ does not directly depend on the size of $\mathcal{A}$. Therefore the time complexity of this algorithm is just the circuit depth:

$$\text{Circuit depth} = O\left(\frac{\text{polylog}(LMN)}{\delta\varepsilon^2}\right). \tag{5.56}$$

5.2.5 Running simulations

QuEST [55] is used to simulate the truncation algorithm for a randomly generated $4 \times 4 \times 4$ tensor (each entry sampled from a uniform distribution) which is then scaled to have a Frobenius norm of 1. These simulations are run for various values of τ and n_p and the results are depicted in Fig. 5.2.



(a) 2 phase qubits

(b) 3 phase qubits

(c) 4 phase qubits

(d) 5 phase qubits

Figure 5.2: Simulating the t-SVD tensor completion algorithm for a $4 \times 4 \times 4$ tensor whose each entry is randomly sampled from a uniform distribution. The x-axis represents different choices of threshold τ. The grey dashed lines represent the t-SVD singular values of the tensor.

The results are compared with classical methods of truncation by plotting the Frobenius distances between (i) The original tensor $\mathcal{A}$ and the tensor generated through the simulated quantum algorithm $\mathcal{A}^q_{trunc}$ (blue), (ii) The original tensor $\mathcal{A}$ and the tensor generated through a classical truncation procedure $\mathcal{A}^c_{trunc}$ (orange), and (iii) The tensor generated through the simulated quantum algorithm $\mathcal{A}^q_{trunc}$ and the one generated through classical truncation $\mathcal{A}^c_{trunc}$ (green). These Frobenius distances are plotted against the threshold τ on the x-axis. From plots (i) and (ii), we see

that the truncated tensors $\mathcal{A}^q_{trunc}$ and $\mathcal{A}^q_{trunc}$ stray further from the original tensor $\mathcal{A}$ in Frobenius distance as the threshold increases and more singular values are set to 0. As we can expect, we see that the quantum results in plot (i) fit more tightly onto the classically calculated results in plot (ii) as we increase the number of phase register qubits. The difference between their results is measured in plot (iii), and we can see that these differences show up around the singular values. The classical computation switches sharply when the threshold crosses a singular value, whereas the quantum computation interpolates smoothly. We therefore see spikes in (iii) around such points.

5.3 Variational algorithm for t-SVD

In Chapter 3, we acquainted ourselves with a variational quantum SVD algorithm. We take heavy inspiration from that algorithm and use it with our block encoding from Section 4.4.6 to create a new variational quantum t-SVD routine which is described in this section.

5.3.1 Broad overview

The algorithm takes the following inputs

1. A $2^n \times 2^n \times 2^k$ tensor $\mathcal{A}$
2. A positive integer $1 \leq T \leq 2^n$

If the tensor we want the t-SVD of is not $2^n \times 2^n \times 2^k$, we can always pad it with zeros to make it so.

Just as in Chapter 3, we will have have an ansatz W and two sets of parameters $\vec{\alpha}$ and $\vec{\beta}$ that we try to optimize with respect to an objective function. W is a special ansatz in that it takes in two registers 1 and 2 of size k and n respectively and leaves computational basis states in register 1 unaffected. That is to say, $W|m\rangle_1|\phi\rangle = |m\rangle_1|\psi\rangle$, for some ψ and ϕ. Another way of phrasing this is that W has a block diagonal matrix representation, where each $2^n \times 2^n$ block is given by $\langle m|_1 W |m\rangle_1$. Let $U'_m := \langle m|_1 W(\vec{\alpha})|m\rangle_1$ and $V'^{\dagger}_m := \langle m|_1 W(\vec{\beta})|m\rangle_1$

The algorithm will try to find unitaries $\{U_m'^{\dagger}\}$ and $\{V_m'\}$ that maximize the following expression:

$$\frac{1}{||\mathcal{A}||_F^2 2^n} \sum_m \sum_i |\langle i|U_m'^{\dagger} \tilde{A}^{(m)} V_m'|i\rangle|^2 \tag{5.57}$$

Since these inner sums are (ideally, for an expressive ansatz) independent of each other, maximizing the whole expression is equivalent to maximizing all the inner sums. And we know from Section 3.3 that when the inner sum is maximized,

$$\tilde{\mathcal{A}}^{rec}[:,:,m] := \sum_{i<T} \tilde{\sigma}_i' U_m' |i\rangle\langle i| V'^{\dagger_m} \approx \sum_{i<T} \tilde{\sigma}_i^{(m)} U^{(m)} |i\rangle\langle i| V^{(m)\dagger} \tag{5.58}$$

where $\left(\tilde{\sigma}_i'^{(m)}\right)^2 := |\langle i|U_m'^{\dagger} \tilde{A}^{(m)} V_m'|i\rangle|^2$ and $\tilde{\sigma}_i^{(m)}$ represents the i-th largest singular value of $\tilde{A}^{(m)}$. This can be seen in Fig. 5.3, which plots $||\tilde{\mathcal{A}} - \tilde{\mathcal{A}}^{rec}||_F$ against the cost (negative of the objective function) of 100 instances of $\tilde{\mathcal{A}}^{rec}$ produced with randomly sampled parameters $\vec{\alpha}$ and $\vec{\beta}$ for a randomly generated $4 \times 4 \times 4$ tensor $\tilde{\mathcal{A}}$ with $T = 4$.

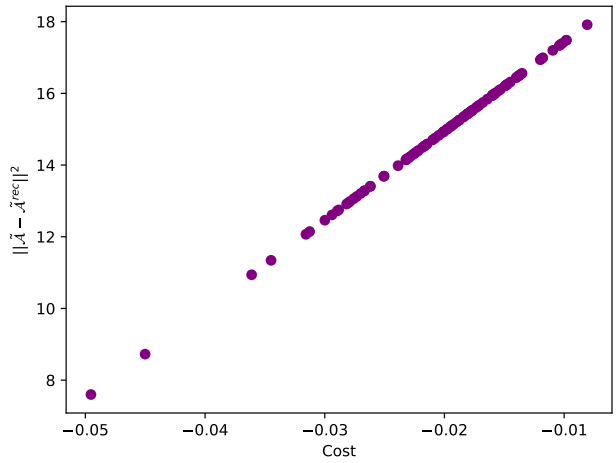


Figure 5.3: Error vs Cost of 100 instances of randomly sampled parameters for a fixed randomly generated $4 \times 4 \times 4$ tensor $\mathcal{A}$ with $T = 4$

5.3.2 Algorithm procedure

In this section, the algorithm is laid out in detail.

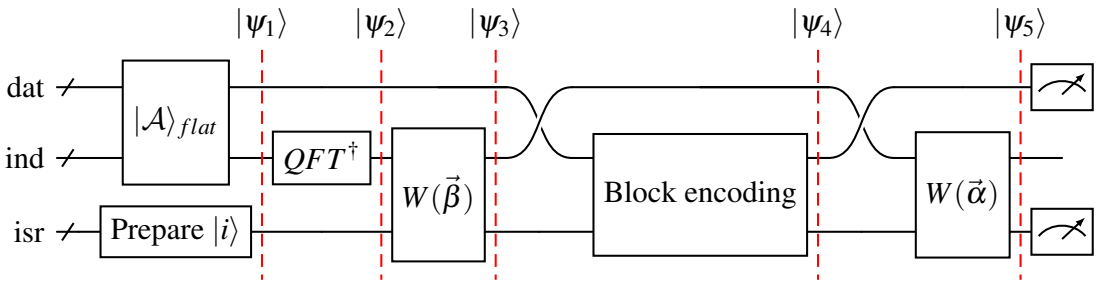


Figure 5.4: Circuit diagram of the variational quantum t-SVD algorithm

We start with a description of the procedure to obtain the objective function in (5.57) using quantum circuits.

The quantum circuit we use is depicted in Fig. 5.4 and has 3 registers:

1. Face index register with l qubits (ind)
2. Face data register with $2n$ qubits (dat)
3. Input state register with n qubits (isr)

The procedure to obtain $\frac{1}{||\mathcal{A}||_F 2^n} \sum_m ||\langle i|U_m'^\dagger \tilde{A}^{(m)} V_m'|i\rangle|^2$ for a given i is as follows:

Step 1a: Prepare $|i\rangle$ on the input state register.

Step 1b: Let $\mathcal{A}_{flat}$ represent the normalized vector obtained by concatenating the Z-order flattening of each face $A_{flat}^{(l)}$ one after the other. We prepare this statevector in the face index register and face data register to get the state:

$$|\psi_1\rangle = |\mathcal{A}_{flat}\rangle_{ind,dat}|i\rangle_{isr} = \sum_k^{2^l} |k\rangle_{ind}|A_{flat}^{(k)}\rangle_{dat}|i\rangle_{isr} \tag{5.59}$$

Step 2: Apply an inverse Quantum Fourier Transform on the face index register to obtain the state:

$$|\psi_2\rangle = \sum_m^{2^l} |m\rangle_{ind}|\tilde{A}_{flat}^{(m)}\rangle_{dat}|i\rangle_{isr} \tag{5.60}$$

Step 3: We apply ansatz $W(\vec{\beta})$ on the face index register and the input state register to get the following state:

$$|\psi_3\rangle = \sum_m^{2^l} |m\rangle_{ind} |\tilde{A}^{(m)}_{flat}\rangle_{dat} V'_m |i\rangle_{isr} \tag{5.61}$$

Step 4: We apply the block encoding from Section 4.4.6 on the face data register and input state register to produce the state:

$$|\psi_4\rangle = \frac{1}{||\mathcal{A}||_F \sqrt{2^n}} \sum_m^{2^l} |m\rangle_{ind} |0\rangle_{dat} \tilde{A}^{(m)} V'_m |i\rangle_{isr} + |0^\perp\rangle_{dat} |\dots\rangle_{ind,isr} \tag{5.62}$$

Step 5: We apply ansatz $W^\dagger(\vec{\alpha})$ on the face index register and the input state register to get the following state:

$$|\psi_5\rangle = \frac{1}{||\mathcal{A}||_F \sqrt{2^n}} \sum_m^{2^l} |m\rangle_{ind} |0\rangle_{dat} U'^\dagger_m \tilde{A}^{(m)} V'_m |i\rangle + |0^\perp\rangle_{dat} |\dots\rangle_{ind,isr} \tag{5.63}$$

Step 6: Estimate the probability of measuring $|0\rangle_{dat}|i\rangle_{isr}$. Born's rule dictates that this probability will be equal to

$$\Pr(|0\rangle_{dat}|i\rangle_{isr}) = \frac{1}{||\mathcal{A}||_F^2 2^n} \sum_m || \langle i| U'^\dagger_m \tilde{A}^{(m)} V'_m |i\rangle ||^2, \tag{5.64}$$

which is the required quantity.

The full objective function (5.57) can be evaluated by running this circuit for all $0 \leq i < T$ and summing all the results (5.64).

We use a classical optimizer to find the optimal parameters $\vec{\alpha}^*$ and $\vec{\beta}^*$ that maximize this objective function. As mentioned in Section 5.3.1, we then get $U'^\dagger_m \approx U^{(m)\dagger}$, $V'_m \approx V^{(m)}$, and $|\langle i| U'^\dagger_m \tilde{A}^{(m)} V'_m |i\rangle|^2 \approx |\tilde{\sigma}^{(m)}_i|^2$. The latter can be obtained by running the circuit for optimal parameters and estimating the probability of measuring $|m\rangle_{ind}|0\rangle_{dat}|i\rangle_{isr}$ from state (5.63). This probability comes out to be

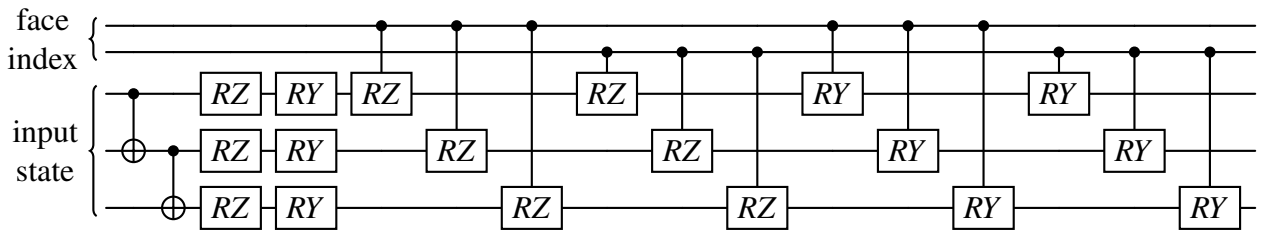


Figure 5.5: A single layer of the ansatz used in simulating the Variational t-SVD

$$\Pr(|m\rangle_{ind}|0\rangle_{dat}|i\rangle_{isr}) = \frac{1}{||\mathcal{A}||_F^2 2^n}||\langle i|U_m'^{\dagger}\tilde{A}^{(m)}V_m'|i\rangle||^2 \tag{5.65}$$

$$\approx |\tilde{\sigma}_i^{(m)}|^2. \tag{5.66}$$

5.3.3 Running simulations

This algorithm is implemented in PennyLane using the ansatz from Fig. 5.5 and simulations have been run on 60 randomly generated $4 \times 4 \times 4$ tensors whose entries have been sampled from a uniform distribution.

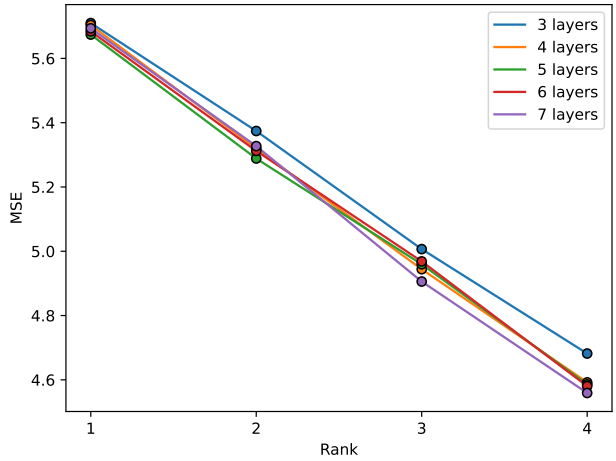


Figure 5.6: MSE vs T when simulating Variational t-SVD with different ansatz layer counts

The algorithm is run for each tensor $\mathcal{A}$, and the output tensor $\mathcal{A}^{rec}$ is produced, as defined in (5.58), for different values of T. Error is measured as the Frobenius distance between $\mathcal{A}^{rec}$ and $\mathcal{A}$. The MSE is plotted for different values of T and ansatz layer counts in Fig. 5.6. We observe that

the MSE decreases with rank, as expected. We also observe however, that the performance of the algorithm does not seem to change in any significant way as we increase the number of layers in our ansatz. Other ansatz options should be explored.

See Section 6.4 for discussion and future works.

Chapter 6

Discussion and Future Works

6.1 Chapter 2

In this chapter, we went through two algorithms that made use of the QSVE routine from [7]. A detailed practical description was laid out on how QSVE can be applied simply on an amplitude encoding of the input matrix A to return its SVD. This procedure lies in the heart of the algorithms in [8]. Simulations of this algorithm were run on various matrices and the performance was measured. A new LSA algorithm was proposed based on QSVE to measure the similarity between two words, given a Term-Document matrix that contains those words. This method works without us having to perform extensive tomography. It is worth noting, however, that there are two main arguments for the use of SVD in LSA.

The first argument is that SVD reduces the dimensionality of our data, which is useful because it decreases the amount of data we need to handle. This argument does not apply when using our LSA algorithm. It would be much easier for us to skip steps 4 (QSVE) and 5 (truncation) in the algorithm we laid out, and work with the row vectors of the untruncated Term-Document matrix.

The second argument is that the truncated Term-Document matrix may reveal higher order correlations between words. This is what our algorithm would hope to accomplish. For future works, we will have to run simulations to see how well this algorithm works in practice.

6.2 Chapter 3

In the third chapter, we looked at the Variational Quantum SVD algorithm from [13] and recognized a drawback in its objective function which was addressed by modifying it. Simulations were run for both the unmodified and modified versions of the algorithm on the MNIST dataset. We also saw how to use this algorithm for the same LSA problem that was discussed in the previous chapter. The argument for LSA as a compression algorithm could apply here depending on how the ansatz used will need to scale with the size and rank of the matrix. Say, for example, a low-rank approximation of a large matrix can be dealt with using a shallow ansatz that takes significantly fewer parameters than there are matrix entries. In this scenario, we can find these parameters once and, from then on, deal with shallow circuits instead of having to bring out a large matrix every time two words need to be compared. A thorough analysis should be done comparing various ansatz choices and studying how they perform as problem size increases. This LSA algorithm was also run for a convenient toy problem, both on a simulator and on hardware as proof of concept. More tests would need to be run on hardware using different kinds of error mitigation schemes.

6.3 Chapter 4

This chapter introduces us to the QSVT, a powerful framework with a surprisingly broad range of applications. We went through - in detail and with an example - how the QSVT generalizes from the QSP routine. We then explored various block encoding schemes. We went through what was referred to as the Sparse FABLE method, a simplified version of a known block encoding method for sparse-access matrices, and a completely new block encoding method was also proposed. This latter method allows one to apply any operator whose matrix representation is available as an amplitude encoded state. Simulations were run solving LSE and TDA problems using QSVT. The TDA problem for which simulations were run was an artificial toy problem. The QSVT algorithm for TDA needs to be tested on real classification problems and its performance should be measured accordingly with a meaningful metric.

6.4 Chapter 5

In this last chapter, we introduce ourselves to two new quantum algorithms for t-SVD. The first algorithm is a purely quantum approach based on modifying the QSVE procedure and adapting it for t-SVD. It is then demonstrated how this modified routine, referred to as Quantum t-SVE, can be used for a type of recommendation system that is polylogarithmic in the size of the tensor. This is exponentially faster than currently known classical t-SVD algorithms. For future works, dequantization efforts need to be explored to see if our algorithm truly is better than what can be done with classical computing. The second t-SVD algorithm is a variational approach based on the variational quantum SVD algorithm we saw in Chapter 3. This second algorithm makes use of our new block encoding and demonstrates the strength in being able to apply any operator that's encoded into the amplitudes of a statevector. Our ability to manipulate the amplitudes of a statevector can then directly translate to manipulations of operators, which is exactly what we do in this algorithm. The simulation results however indicate poor scaling in performance as the layer count of the ansatz increases. Different choices of ansätze and higher layer counts should be explored. In both algorithms, the QFT could be replaced by other transformations, like the Hadamard transform for example. It is worth exploring which transformation would lead to the best results for a given application.

References

[1] YongchangWang and Ligu Zhu. "Research and implementation of SVD in machine learning". In: *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. 2017, pp. 471–475. DOI: 10.1109/ICIS.2017.7960038.

[2] Badrul Munir Sarwar et al. "Application of Dimensionality Reduction in Recommender System - A Case Study". In: 2000. URL: https://api.semanticscholar.org/CorpusID:17065558.

[3] Mei Tian, Si-Wei Luo, and Ling-Zhi Liao. "An investigation into using singular value decomposition as a method of image compression". In: *2005 International Conference on Machine Learning and Cybernetics*. Vol. 8. IEEE. 2005, pp. 5200–5204.

[4] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (1982), pp. 467–488. DOI: 10.1007/bf02650179. URL: https://doi.org/10.1007/bf02650179.

[5] Lov 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. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[6] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[7] Iordanis Kerenidis and Anupam Prakash. *Quantum Recommendation Systems*. 2016. arXiv: 1603.08675 [quant-ph].

[8] Armando Bellante, Alessandro Luongo, and Stefano Zanero. "Quantum algorithms for SVD-based data representation and analysis". en. In: *Quantum Mach. Intell.* 4.2 (2022).

[9] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[10] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". en. In: *Nat. Phys.* 10.9 (Sept. 2014), pp. 631–633.

[11] Patrick Rebentrost et al. "Quantum singular-value decomposition of nonsparse low-rank matrices". In: *Phys. Rev. A (Coll. Park.)* 97.1 (Jan. 2018).

[12] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pp. 217–228. ISBN: 9781450367059. DOI: 10.1145/3313276.3316310. URL: https://doi.org/10.1145/3313276.3316310.

[13] Xin Wang, Zhixin Song, and Youle Wang. "Variational Quantum Singular Value Decomposition". In: *Quantum* 5 (2021), p. 483. ISSN: 2521-327X. DOI: 10.22331/q-2021-06-29-483. URL: https://doi.org/10.22331/q-2021-06-29-483.

[14] Carlos Bravo-Prieto, Diego García-Martín, and José I. Latorre. "Quantum singular value decomposer". In: *Phys. Rev. A* 101 (6 2020), p. 062310. DOI: 10.1103/PhysRevA.101.062310. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.062310.

[15] Ju–Young Ryu et al. "A Variational Quantum Algorithm for Ordered SVD". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 01–05. DOI: 10.1109/GLOBECOM46510.2021.9685404.

[16] András Gilyén et al. "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*. STOC '19. ACM, 2019. DOI: 10.1145/3313276.3316366. URL: http://dx.doi.org/10.1145/3313276.3316366.

[17] John M. Martyn et al. "Grand Unification of Quantum Algorithms". In: *PRX Quantum* 2 (4 2021), p. 040203. DOI: 10.1103/PRXQuantum.2.040203. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040203.

[18] Zemin Zhang and Shuchin Aeron. *Exact tensor completion using t-SVD*. 2015. arXiv: 1502.04689 [cs.LG].

[19] Xiaoqiang Wang et al. "Quantum tensor singular value decomposition*". In: *Journal of Physics Communications* 5.7 (2021), p. 075001. ISSN: 2399-6528. DOI: 10.1088/2399-6528/ac0d5f. URL: http://dx.doi.org/10.1088/2399-6528/ac0d5f.

[20] Lejia Gu, Xiaoqiang Wang, and Guofeng Zhang. "Quantum Higher Order Singular Value Decomposition". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019, pp. 1166–1171. DOI: 10.1109/SMC.2019.8914525.

[21] Camille Jordan. "Essai sur la géométrie à *n* dimensions". fr. In: *Bulletin de la Société Mathématique de France* 3 (1875), pp. 103–174. DOI: 10.24033/bsmf.90. URL: http://www.numdam.org/articles/10.24033/bsmf.90/.

[22] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. "Quantum arithmetic with the quantum Fourier transform". In: *Quantum Inf. Process.* 16.6 (June 2017).

[23] Susan T. Dumais. "Latent semantic analysis". In: *Annual Review of Information Science and Technology* 38.1 (2004), pp. 188–230. ISSN: 1550-8382. DOI: 10.1002/aris.1440380105. URL: http://dx.doi.org/10.1002/aris.1440380105.

[24] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407. DOI: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9.

[25] Yewei Yuan et al. "An improved QFT-based quantum comparator and extended modular arithmetic using one ancilla qubit". In: *New Journal of Physics* 25.10 (2023), p. 103011. ISSN: 1367-2630. DOI: 10.1088/1367-2630/acfd52. URL: http://dx.doi.org/10.1088/1367-2630/acfd52.

[26] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: 1811.04968 [quant-ph].

[27] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[28] W.N. Francis and H. Kučera. *Manual of Information to Accompany A Standard Corpus of Present-day Edited American English, for Use with Digital Computers*. Brown University, Department of Lingustics, 1979. URL: https://books.google.co.in/books?id=9dw5nQEACAAJ.

[29] Jezer Jojo, Ankit Khandelwal, and M Girish Chandra. *On Modifying the Variational Quantum Singular Value Decomposition Algorithm*. 2024. arXiv: 2310.19504 [quant-ph].

[30] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/bf02288367. URL: https://doi.org/10.1007/bf02288367.

[31] *Paddle Quantum*. 2020. URL: https://github.com/PaddlePaddle/Quantum.

[32] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[33] Jonathan Welch et al. "Efficient quantum circuits for diagonal unitaries without ancillas". In: *New Journal of Physics* 16.3 (2014), p. 033040. ISSN: 1367-2630. DOI: 10.1088/1367-2630/16/3/033040. URL: http://dx.doi.org/10.1088/1367-2630/16/3/033040.

[34] OpenAI. *ChatGPT*. https://openai.com/gpt. Accessed: March 10, 2024. 2022.

[35] Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2023. DOI: 10.5281/zenodo.2573505.

[36] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE Transactions on Automatic Control* 37.3 (1992), pp. 332–341. DOI: 10.1109/9.119632.

[37] IBM. *IBM Quantum*. https://quantum.ibm.com/. Accessed: February, 2024. 2021.

[38] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. "Methodology of Resonant Equiangular Composite Quantum Gates". In: *Phys. Rev. X* 6 (4 2016), p. 041067. DOI: 10.1103/PhysRevX.6.041067. URL: https://link.aps.org/doi/10.1103/PhysRevX.6.041067.

[39] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Ed. by Christel Baier et al. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 33:1–33:14. ISBN: 978-3-95977-109-2. DOI: 10.4230/LIPIcs.ICALP.2019.33. URL: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2019.33.

[40] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (2019), p. 163. ISSN: 2521-327X. DOI: 10.22331/q-2019-07-12-163. URL: https://doi.org/10.22331/q-2019-07-12-163.

[41] Daan Camps et al. *Explicit Quantum Circuits for Block Encodings of Certain Sparse Matrices*. 2023. arXiv: 2203.10236 [quant-ph].

[42] Seiseki Akibue, Go Kato, and Seiichiro Tani. *Probabilistic unitary synthesis with optimal accuracy*. 2023. arXiv: 2301.06307 [quant-ph].

[43] Daan Camps and Roel Van Beeumen. "FABLE: Fast Approximate Quantum Circuits for Block-Encodings". In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2022, pp. 104–113. DOI: 10.1109/QCE53715.2022.00029.

[44] Mercy G. Amankwah et al. "Quantum pixel representations and compression for N-dimensional images". In: *Scientific Reports* 12.1 (2022). DOI: 10.1038/s41598-022-11024-y. URL: https://doi.org/10.1038/s41598-022-11024-y.

[45] Andrew M. Childs and Nathan Wiebe. "Hamiltonian simulation using linear combinations of unitary operations". In: *Quantum Information and Computation* 12.11 & 12 (2012), pp. 901–924. ISSN: 1533-7146. DOI: 10.26421/qic12.11-12-1. URL: http://dx.doi.org/10.26421/QIC12.11-12-1.

[46] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[47] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009). ISSN: 1079-7114. DOI: 10.1103/physrevlett.103.150502. URL: http://dx.doi.org/10.1103/PhysRevLett.103.150502.

[48] Arnold Dresden. "The fourteenth western meeting of the American Mathematical Society". en. In: *Bull. Am. Math. Soc.* 26.9 (June 1920), pp. 385–397.

[49] Gunnar Carlsson. "Topology and data". en. In: *Bull. New Ser. Am. Math. Soc.* 46.2 (Jan. 2009), pp. 255–308.

[50] "Topological data analysis". In: *Inverse Probl.* 27.12 (Dec. 2011), p. 120201.

[51] Dominic W. Berry et al. "Analyzing Prospects for Quantum Advantage in Topological Data Analysis". In: *PRX Quantum* 5 (1 2024), p. 010319. DOI: 10.1103/PRXQuantum.5.010319. URL: https://link.aps.org/doi/10.1103/PRXQuantum.5.010319.

[52] Ankit Khandelwal and M Girish Chandra. “Quantum-Enhanced Topological Data Analysis: A Peep from an Implementation Perspective”. In: *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, May 2023. DOI: 10.1109/ipdpsw59300.2023.00093. URL: http://dx.doi.org/10.1109/IPDPSW59300.2023.00093.

[53] Ryu Hayakawa. “Quantum algorithm for persistent Betti numbers and topological data analysis”. In: *Quantum* 6 (2022), p. 873. ISSN: 2521-327X. DOI: 10.22331/q-2022-12-07-873. URL: https://doi.org/10.22331/q-2022-12-07-873.

[54] J Friedman. “Computing Betti numbers via combinatorial Laplacians”. en. In: *Algorithmica* 21.4 (1998), pp. 331–346.

[55] Tyson Jones et al. “QuEST and high performance simulation of quantum computers”. en. In: *Sci. Rep.* 9.1 (2019), p. 10736.