# Accelerating Linear Algebra and Machine Learning With Quantum Computers


Arthur Giuseppe Rattew

St. Edmund Hall

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Trinity 2025

# Acknowledgements

First, I would like to thank my supervisor, Simon Benjamin. I could not have asked for a better person to guide me through my DPhil journey: his brilliance, leadership, excellence as a researcher, and genuine care for his students have made me cherish my experience in his group. Next, I would like to thank Patrick Rebentrost, who became my *de facto* co-supervisor. He generously allowed me to repeatedly visit his group many times, and working with him was both a pleasure and a privilege. He brought me up to speed with modern quantum linear algebra research, and taught me a range of techniques which have become a foundational part of my toolkit as a theorist. I greatly enjoyed our conversations and his advice. Of equal importance, I would like to express my sincerest appreciation to Marco Pistoia. He brought me under his wing, and was an excellent example as a researcher and mentor. I am also grateful to Bálint Koczor whose guidance early during my theoretical research career gave me a valuable foundation to build on. Additionally, I extend my gratitude to Sam Jaques with whom I've greatly enjoyed countless discussions on QRAM and wider topics in computer science and physics. Moreover, I've had many entertaining and informative discussions and debates with the members of my group in Oxford, including with Po-Wei Huang, Tim Chan, Armands Strikis, Daniel Marti-Dafcik, Tyson Jones, Hamza Jnane, and Hans Chan, just to name a few. I additionally extend my thanks to my college, St. Edmund Hall, and in particular Belinda Huse, for their practical support.

From my visits to Singapore, I would like to thank many people, including my co-authors Naixu Guo and Lirandë Pira. Additionally, I would like to thank Alessandro Luongo, Armando Bellante, Hitomi Mori, Yixian Qiu.

From my undergraduate, I would like to thank I-Ting Angelina Lee, who was my first research advisor, Ron Cytron who supervised my independent study and with whom it was a pleasure to learn quantum computing, and Roman Garnett who supervised my master's thesis. They all helped to shape me into the researcher I am today.

Additionally, I sincerely thank all of my collaborators, some of whom were mentioned above.

On a professional note, I would like to thank the whole Global Technology Applied Research (GTAR) group at JPMorgan Chase for their support. It was a

privilege to have had multiple internships in GTAR, and I have many fond memories. I would also like to thank Shouvanik Chakrabarti for many fruitful conversations where he shared some of his brilliance, as well as Pradeep Niroula, Dylan Herman, Yue Sun, Pierre Minssen, Ruslan Shaydulin, Shree Hari Sureshbabu, Niraj Kumar, Shaohan Hu, and many others. I would also like to extend my gratitude to Quantum Motion (and in particular, Tom Bromley). Additionally, I would like to thank Erich Clementi for his mentorship – his wisdom is unparalleled and often prophetic.

I also extend my thanks to all of my highschool teachers, including Mr. Weekes (physics) and Mr. Chun (math).

On a personal note, I extend my thanks to my close friend Kyle Yamaguchi for his support through the years. I would also like to thank Joowon Kim for being an incredibly supportive and kind partner. I would also like to thank all my grandparents, but in particular my Nonna Giuseppina, for their continuous love and presence. Most importantly, I would like to thank my parents for their enduring love and support, and for providing me with every opportunity to explore my curiosity and interests as I grew up.

# Abstract

In this thesis, I develop the theory required to use quantum computers to accelerate linear algebra and machine learning, in a manner akin to how GPUs are used today. I begin by analysing quantum random access memory (QRAM), the mechanism by which quantum algorithms can access classical (non-quantum) data. When used for linear algebra applications, I show that QRAM has substantial challenges making it harder to construct than an error-corrected quantum computer. Should QRAM prove to be unrealistic, I ask what types of classical data can be efficiently loaded into quantum computers, and prove that discretized continuous functions are one such option. Next, I introduce a framework for manipulating vectors with quantum computers. The primary result of this framework is a technique allowing for the non-linear transformations of vectors to be enacted efficiently for a broad class of functions, despite the underlying unitary (and thus linear) nature of quantum algorithms. In the final chapter, utilizing the preceding results and further developing the vector encoding framework, I demonstrate that quantum computers can accelerate inference for multilayer neural networks under a range of scenarios of QRAM feasibility. This chapter also notes a plausible path towards constructing a practically useful QRAM.

# Contents

**Mathematical Notation** **xi**

**1 Introduction** **1**
1.1 Motivation . . . . . 1
1.2 Technical Introduction . . . . . 2
1.2.1 Quantum Computing Overview . . . . . 3
1.2.2 Technical Definitions and Extant Results . . . . . 3

**2 The Limitations of Quantum Random Access Memory** **9**
2.1 Introduction . . . . . 10
2.2 The Different Types of QRAM . . . . . 11
2.2.1 QRAM Categories Depending on Data Type . . . . . 11
2.2.2 Circuit QRAM Derivation and Complexity Analysis . . . . . 13
2.2.3 Passive and Active QRAM . . . . . 21
2.3 Limitations of QRAM Applied to Linear Algebra and Machine Learning 24
2.3.1 Input Model and Matrix Functions . . . . . 25
2.3.2 Dequantization . . . . . 26
2.3.3 QRAM Versus Parallel Classical Computation . . . . . 29
2.3.4 Quantum Linear Algebra with Noisy QRAM . . . . . 34

**3 Quantum State Preparation** **37**
3.1 Introduction . . . . . 38
3.1.1 Prior Work . . . . . 40
3.2 Algorithm Overview . . . . . 42
3.2.1 Efficient Implementation of the Time Evolution . . . . . 45
3.2.2 Limitations . . . . . 47
3.3 Numerical Demonstration . . . . . 49
3.4 Further Applications . . . . . 51
3.5 Discussion and Conclusion . . . . . 52

**4 Non-Linear Transformation of Quantum Amplitudes 55**
4.1 Introduction . . . 56
4.1.1 Prior Work . . . 57
4.1.2 Overview of Our Work . . . 59
4.1.3 Preliminaries . . . 60
4.2 New Algorithmic Subroutines . . . 61
4.2.1 Diagonal Block-Encoding of State Amplitudes . . . 62
4.2.2 Importance-Weighted Amplitude Transformation . . . 63
4.3 Applications . . . 67
4.3.1 Exponential Improvement in Applying tanh Activation Function 67
4.3.2 End-to-End Efficiency in Applying Common Functions . . . 68
4.3.3 Quantum Maximum Finding in the Unitary Input Model . . . 69
4.3.4 Generalizing Quantum State Preparation . . . 71
4.4 Conclusion . . . 72

**5 Quantum Processors as Accelerators for Machine Learning 75**
5.1 Introduction . . . 76
5.2 Quantum Matrix-Vector Arithmetic . . . 79
5.2.1 New Operations on Vector Encodings . . . 80
5.2.2 Matrix Vector Squared Product . . . 82
5.2.3 QRAM-Free Quantum Encoding of 2D Multi-Filter Convolutions . . . 83
5.3 Architectural Blocks . . . 83
5.4 Architectures . . . 86
5.4.1 Key Results Under Differing Quantum Data Access Assumptions 87
5.5 Conclusion . . . 89
5.6 Future Work . . . 90

**Appendices**

**A The Limitations of Quantum Random Access Memory 95**

**B Quantum State Preparation 99**
B.1 Preliminaries and Definitions . . . 99
B.1.1 Continuous and Efficiently Computable Functions . . . 99
B.1.2 Encoding Functions into Quantum States . . . 101
B.2 Hamiltonian Simulation Techniques . . . 102
B.2.1 1-Sparse Hamiltonian Simulation . . . 102
B.2.2 Low-Rank Hamiltonian Simulation . . . 106
B.2.3 Tighter Bounds on 1-Sparse Simulation Error . . . 107

B.2.4 Adiabatic Computation . . . 112
B.3 General Distribution Loading Through Adiabatic Evolution: Direct State Preparation on $n$ Qubits . . . 115
B.3.1 The Procedure . . . 118
B.3.2 Proofs for Finite Qubit Count $n$ . . . 121
B.4 Generalization to Multivariate Functions . . . 136
B.5 Further State-Preparation Techniques . . . 138
B.5.1 Summary of Prior Work on State Preparation . . . 138
B.5.2 Possible Implementation of Grover's State Preparation . . . 139
B.5.3 Possible Implementation of Grover-Rudolph State Preparation 141
B.6 Asymptotic Analysis . . . 142
B.6.1 Asymptotic Properties of the Rank-One Matrix $A$ . . . 144

**C Non-Linear Transformation of Quantum Amplitudes 147**
C.1 Preliminaries . . . 147
C.2 Diagonal Block-Encoding of Quantum State Amplitudes . . . 148
C.3 Importance-Weighted Amplitude Transform . . . 154
C.4 Non-Linear Transformations of State Amplitudes via Polynomial Approximations . . . 158
C.5 Application: End-to-End Complexities for Applying Various Functions to Arbitrary Quantum States . . . 161
C.6 Application: Exponential Improvement in Applying Tanh Activation Function in Machine Learning . . . 164
C.7 Application: Maximum Finding . . . 165
C.8 Application: Continuous State Preparation . . . 167
C.9 Comparison to [121] . . . 168
C.10 Vector Encodings . . . 171

**D Quantum Processors as Accelerators for Machine Learning 177**
D.1 Quantum Random Access Memory (QRAM) . . . 177
D.2 Quantum Matrix-Vector Arithmetic . . . 178
D.2.1 General Matrix-Vector-Squared Product . . . 190
D.2.2 Convolution Block-Encoding . . . 196
D.2.3 Non-Linear Transformation of Vector-Encodings . . . 202
D.3 General Architectural Blocks . . . 205
D.4 Feasibility of QRAM Assumptions . . . 212
D.4.1 Passive and Active QRAM . . . 212
D.4.2 Input Preparation via QRAM . . . 214
D.5 Architectures in Different Regimes . . . 215
D.5.1 Regime 1: Input and Network Use QRAM . . . 215

D.5.2 Regime 2: Network Stored in QRAM, Input Loaded Without QRAM . . . . . 218
D.5.3 Regime 3: No QRAM . . . . . 218
D.6 Technical Results . . . . . 221

**References** **225**

# Mathematical Notation

$\mathbb{Z}$ . . . . . The set of integers.

$\mathbb{Z}_{\geq 0}$ . . . . . The set of non-negative integers.

$\mathbb{Z}_{<0}$ . . . . . The set of negative integers.

$\mathbb{N}$ . . . . . The set of natural numbers (here, defined to include zero).

$[N]$ . . . . . The set of integers 0 through $N-1$.

$\oplus$ . . . . . Bit-wise addition modulo 2 of two bit strings (bit-wise XOR operation).

$\mathfrak{R}$ . . . . . Real-part function (also written as Re(.). Given a complex number $z = a + ib$, $\mathfrak{R}(z) = a$.

$\mathfrak{I}$ . . . . . Imaginary-part function (also written as Im(.). Given a complex number $z = a + ib$, $\mathfrak{I}(z) = b$.

## Asymptotic Notation

$O(\cdot)$ . . . . . Asymptotic upper-bound (inclusive). If $f(x) \in O(g(x))$ then $f(x)$ is asymptotically upper-bounded by $g(x)$.

$o(\cdot)$ . . . . . Asymptotic upper-bound (exclusive). If the function $f(x) \in o(g(x))$ then $\lim_{x \mapsto \infty} f(x)/g(x) = 0$.

$\Omega(\cdot)$ . . . . . Asymptotic lower-bound (inclusive). If $f(x) \in \Omega(g(x))$ then $f(x)$ is asymptotically lower-bounded by $g(x)$.

$\omega(\cdot)$ . . . . . Asymptotic lower-bound (exclusive). If the function $f(x) \in \omega(g(x))$ then $\lim_{x \mapsto \infty} g(x)/f(x) = 0$.

$\Theta(\cdot)$ . . . . . Asymptotic tight bound. If $f(x) \in O(g(x))$ and $f(x) \in \Omega(g(x))$ then $f(x) \in \Theta(g(x))$.

$\tilde{O}(\cdot)$ / $\tilde{\Omega}(\cdot)$ . . . Asymptotic upper-bound/ lower-bound hiding a polylog factor, e.g., $\tilde{O}(f(n)) = O(f(n) \cdot \operatorname{poly}\log(f(n)))$. This notation is also used with $\tilde{o}(\cdot)$ and $\tilde{\omega}(\cdot)$.

## Linear Algebra Notation

$\boldsymbol{x}$ . . . . . Vector are bolded.

$\boldsymbol{e}_i$, $|i\rangle$ . . . . . Basis vectors are represented by $\boldsymbol{e}_i = |i\rangle$. E.g., $\boldsymbol{e}_0 = \begin{pmatrix}1 & 0 & \dots & 0\end{pmatrix}^T$. In ket notation, alternate labels are sometimes used.

$\|\cdot\|_p$ . . . . . The standard $p$-norm.

$\|\cdot\|_\infty$ . . . . . The max norm. May sometimes be denoted as $\|\cdot\|_{max}$.

$|x\rangle_n$ . . . . . For some label $x$, and $n \in \mathbb{N}$, $|x\rangle_n$ represents an $n$-qubit quantum state, i.e., $|x\rangle_n \in \mathbb{C}^{2^n}$. When clear from the context, the subscript indicating the dimension will be dropped.

$\langle x|y\rangle$ . . . . . Inner product. If $\boldsymbol{x} = |x\rangle$ and $\boldsymbol{y} = |y\rangle$, then $\langle x|y\rangle = \boldsymbol{x}^\dagger \boldsymbol{y}$.

$|x\rangle\langle y|$ . . . . . Outer product. If $\boldsymbol{x} = |x\rangle$ and $\boldsymbol{y} = |y\rangle$, then $|x\rangle\langle y| = \boldsymbol{x}\boldsymbol{y}^\dagger$.

$\dagger$ . . . . . Conjugate transpose.

$\otimes$ . . . . . Kronecker product. Often, when writing the Kronecker product of two vectors, I drop the $\otimes$ symbol, i.e., $|a\rangle|b\rangle := |a\rangle \otimes |b\rangle$. Sometimes, this will be called a tensor product.

$f(\boldsymbol{x})$ . . . . . Given a function $f : \mathbb{C} \mapsto \mathbb{C}$ and a vector $\boldsymbol{x} \in \mathbb{C}^n$, then $f(\boldsymbol{x})$ is defined by element-wise application of the function to the elements of the vector, i.e., $f(\boldsymbol{x}) = \sum_{i=1}^n f(x_i)\boldsymbol{e}_i$.

# 1 Introduction

In this chapter, I first provide a brief high-level introduction motivating this thesis. Next, I define and motivate some of the basic mathematics and tools used repeatedly in different chapters. Some of the text in this chapter is taken verbatim from the papers corresponding to Chapters 2 to 5 of this thesis (with corresponding papers [1–4]). I was the primary author on any such text.

## Contents


**1.1 Motivation . . . . . 1**
**1.2 Technical Introduction . . . . . 2**
1.2.1 Quantum Computing Overview . . . . . 3
1.2.2 Technical Definitions and Extant Results . . . . . 3


## 1.1 Motivation

The progress in developing quantum computers has been rapid, and new records are being set every year [5–9]. With a rich history of theoretical developments beginning with Richard Feynman's original proposal for a quantum computer [10, 11], a range of quantum algorithms have been proposed. Famous examples include Shor's algorithm (an efficient integer factorization algorithm) [12], Grover search (an

algorithm for unstructured search) [13], Hamiltonian simulation (a type of physical simulation) [14], and HHL (matrix inversion) [15].

Concurrently, in recent years machine learning techniques have seen widespread adoption across a wide range of important problem domains [16–20]. Indeed, Demis Hassabis and John Jumper won the 2024 Nobel Prize for Chemistry, in part, for their contributions to AlphaFold2 [21] which can accurately predict protein structure. Such advances were partially enabled by the exponential rate of improvement of classical compute hardware. Indeed, Moore's law [22], which posits that the number of transistors on a given chip doubles roughly every year (or more recently, every two years), has primarily been fuelled by shrinking the size of transistors with each new chip generation. However, this halving in size cannot continue indefinitely, and so it seems likely that Moore's law will soon come to an end [23]. However, as observed in [24], the total compute required to train state-of-the-art machine learning models has been growing exponentially over the last decade. Combined with transistor-based hardware reaching their physical limits, to continue this rate of growth, an exponential increase in the amount of resources (chips, and input energy) would be required. Illustrating how soon we might start to see stalling advances in classical compute hardware (at least through the shrinking of transistors), Apple states that its latest generation (M3) of chips are produced using a 3nm fabrication technique. This is already approaching the scale of atoms, and it seems likely that quantum effects will soon begin to prevent further miniaturization [23].

Consequently, it is natural to wonder if these quantum effects could instead be directly leveraged to further accelerate linear algebra and machine learning workloads. Developing the machinery to enable this is the primary goal of this thesis.

## 1.2 Technical Introduction

Now, I will briefly provide an elementary technical summary of quantum computing, some useful definitions, results from prior work, and other miscellaneous details which are used across multiple chapters of this thesis.

### 1.2.1 Quantum Computing Overview

In this thesis, I assume familiarity with the basics of quantum computing. A standard introductory (and comprehensive) reference is [25]. Here, I provide an extremely brief summary of the absolute basics.

A quantum state is an $\ell_2$-normalized complex vector. Quantum computers operate on qubits, where a qubit is a 2 dimensional quantum state. A quantum algorithm is enacted by a unitary transformation acting upon an initial quantum state, and is specified by a quantum circuit, which consists of a sequence of single and two qubit gates. A quantum gate is a physically realizable unitary operation which a quantum computer can directly (or at least with some compilation) execute.

Four common single-qubit gates are the Hadamard matrix, and the Pauli X, Y and Z operators:

$$H := \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \quad X := \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \quad Y := \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \quad Z := \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}. \tag{1.1}$$

As a note, I often use $H$ to refer to arbitrary matrices, so when it refers to the Hadamard matrix it will be clear from the context. A common two-qubit gate is the controlled-not gate $CX$,

$$CX := \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}. \tag{1.2}$$

### 1.2.2 Technical Definitions and Extant Results

**Definition 1.2.1** (Matrix eigenvalue functions)**.** *Given a $N \times N$ normal matrix $H = \sum_{j=1}^{N} \lambda_j |\lambda_j\rangle\langle\lambda_j|$, with $H|\lambda_j\rangle = \lambda_j|\lambda_j\rangle$, and a function $f : \mathbb{C} \mapsto \mathbb{C}$, we define a matrix function on its eigenvalues, i.e. $f(H) := \sum_{j=1}^{N} f(\lambda_j)|\lambda_j\rangle\langle\lambda_j|$.*

**Definition 1.2.2** (Lipschitz constant and Lipschitz continuity)**.** *If a function $f$ is Lipschitz continuous on the interval $[a, b]$, there exists some constant $L \geq 0$ such that for all $x \neq y$ (with $x \in [a, b]$ and $y \in [a, b]$),*

$$\frac{|f(y) - f(x)|}{|y - x|} \leq L. \tag{1.3}$$

*Moreover, we call the smallest $L$ satisfying the preceding equation the Lipschitz constant; the Lipschitz constant $L$ is given by $L := \max_{x\in[a,b]} |f'(x)|$.*

**Definition 1.2.3** (Efficient Uniform Approximation)**.** *Let $f(x) : [-1,1] \to \mathbb{R}$ be a real-valued function in the domain $x \in [-1,1]$. With $k \in \mathbb{Z}_+$ and $\epsilon > 0$, we say that the function $f(x)$ is $(\epsilon, k)$-uniformly approximated by a polynomial if there exists a degree-$k$ polynomial $P_k : [-1,1] \to \mathbb{C}$ such that*

$$\max_{x\in[-1,1]} |f(x) - P_k(x)| \leq \epsilon. \tag{1.4}$$

*Equivalently, we say that $f(x)$ is an "$\epsilon$-approximable function with degree $k$", where $k$ can be any function of $\epsilon$.*

**Definition 1.2.4** (Efficiently Computable Functions)**.** *Given the above families of continuous functions, we consider $f$ to be efficiently computable if there exists an efficient quantum algorithm $O_f$ performing the mapping,*

$$O_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle, \tag{1.5}$$

*for any computational basis state $|x\rangle$ given a suitable binary encoding of the real and complex numbers $x$ and $f(x)$, respectively. Such an oracle may be efficiently constructed for any function where an efficient classical circuit allows the evaluation of $f$ for any possible input in the domain. Intuitively, if there is a classical algorithm allowing the evaluation of $f$, then it is possible to construct $O_f$.*

Another useful gate definition which will be frequently used is the following.

**Definition 1.2.5** ($R_Y(t)$ Gate)**.** *Let $t \in \mathbb{R}$, and let $Y$ be the standard single-qubit Pauli-$Y$ gate. Then, define*

$$R_Y(t) := e^{-itY} = \cos(t)I - i\sin(t)Y = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix}. \tag{1.6}$$

For completeness, I will now present a standard result allowing one to transfer digitally represented information to the amplitudes of a quantum state. Similar versions of this result are used throughout this thesis, see Lemma B.2.1 for the first usage.

**Lemma 1.2.1** ($CR_Y(t)$ Gate)**.** *Let $t \in \mathbb{R}$. Let $Y$ be a standard Pauli-$Y$ gate. Let $|a\rangle_d$ be a $d$-bit standard basis vector, and let $|\psi\rangle_1$ be an arbitrary single-qubit quantum state. Then, we can define the gate $CR_Y(t)$ by the following action,*

$$CR_Y(t)|\psi\rangle_1|a\rangle_d = (e^{-iatY}|\psi\rangle_1)|a\rangle_d. \tag{1.7}$$

*In the event that $|\psi\rangle_1 = |0\rangle_1$, this action can be simplified to*

$$CR_Y(t)|0\rangle_1|a\rangle_d = (\cos(at)|0\rangle_1 + \sin(at)|1\rangle_1)|a\rangle_d. \tag{1.8}$$

*Moreover, the $CR_Y(t)$ gate is implemented with $O(d)$ circuit depth.*

*Proof.* This is a standard result. This proof is included for completeness, and follows the one in [2]. Let $D = 2^d$. First, note that $CR_Y(t) = \sum_{a=0}^{D-1} e^{-iatY} \otimes |a\rangle\langle a|$. Additionally, let $a = a_{d-1}a_{d-2}\dots a_1a_0 = a_{d-1}2^{d-1} + ... + a_12 + a_0$. Then,

$$e^{-iatY} = e^{-i(a_{d-1}2^{d-1}+...+a_12+a_0)tY} = e^{-ia_{d-1}2^{d-1}tY} \cdot \dots \cdot e^{-ia_1tY}e^{-ia_0tY}. \tag{1.9}$$

Then, $CR_Y(t)$ can be implemented by applying a sequence of $d$ controlled $e^{-i2^jtY}$ gates (Definition 1.2.5), targeting the first register, controlled on the $j^{th}$ bit of the second register. □

Note that $CR_X$ and $CR_Z$ gates can be defined analogously; see Lemma B.2.1.

### Matrix Block-Encoding

A widely used tool in quantum algorithm design is the block-encoding [26], which can be viewed as a way to encode and manipulate matrices in quantum algorithms. A block-encoding is a unitary matrix $U$, specified by a quantum circuit, whose top left block contains a matrix $\tilde{A}$ (such that $\|\tilde{A}\|_2 \leq 1$) which is a scaled approximation to some matrix $A$. We give the formal definition in the following.

**Definition 1.2.6** (Block Encoding [26])**.** *Suppose that $A$ is a $2^s \times 2^s$ matrix, $\alpha, \epsilon \in \mathbb{R}_+$ and $a \in \mathbb{N}$, then we say that the $2^{s+a} \times 2^{s+a}$ unitary matrix $U$ is an $(\alpha, a, \epsilon)$-block-encoding of $A$, if*

$$\|A - \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)\| \leq \epsilon.$$

Essentially, noting that $\langle 0|^{\otimes a} \otimes I = \begin{pmatrix} I & 0 & \dots & 0 \end{pmatrix}$, we see that $\langle 0|^{\otimes a} \otimes I$ selects the first $2^s$ rows of $U$, and then $|0\rangle^{\otimes a} \otimes I$ selects the first $2^s$ columns of $(\langle 0|^{\otimes a} \otimes I)U$, meaning that $(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)$ is simply the top-left $2^s \times 2^s$ block of $U$. Indeed, if $\epsilon = 0$, then $A/\alpha = (\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I)$. Additionally, $\alpha$ can be viewed as an upper-bound on the normalization factor of $A$, e.g., if $\epsilon = 0$, then $\|A/\alpha\|_2 \leq 1$. Any matrix encoded in a sub-block of a unitary matrix cannot have norm exceeding 1.

**Lemma 1.2.2** (Product of Block Encodings [26]). *If $U$ is an $(\alpha, a, \delta)$-block-encoding of an $s$-qubit operator $A$, and $V$ is an $(\beta, b, \epsilon)$-block-encoding of an $s$-qubit operator $B$ then $(I_b \otimes U)(I_a \otimes V)$ is an $(\alpha\beta, a + b, \alpha\epsilon + \beta\delta)$-block-encoding of $AB$.*

In Lemma 1.2.2, following the convention of [26], it is assumed that $U$ acts trivially on the ancillas of $V$, and $V$ acts trivially on the ancillas of $U$ – this notation is not used anywhere else in this paper. I.e., the tensor products in $(I_b \otimes U)(I_a \otimes V)$ are not read in the usual sense in this lemma.

### Vector Encoding (VE)

In this subsection, I motivate and present the concept of a *vector encoding*, which I first introduced in (and is a contribution of) the paper covered in Chapter 4 (although I originally called it a state-preparation block-encoding (SPBE)), and further developed in Chapter 5.

Analogously to how a quantum block-encoding encodes a general matrix in the top left block of a unitary, we can embed arbitrary (sub-normalized) $N$-dimensional vectors in the first $N$ rows of a larger vector corresponding to a normalized quantum state.

This naturally leads to the following definition of quantum vector-encodings (VEs)

**Definition 1.2.7** (Vector-Encoding (VE) Chapter 4). *Let $\alpha \geq 1$, $a \in \mathbb{N}$, and $\epsilon \geq 0$. We call the $2^{a+n} \times 2^{a+n}$ unitary matrix $U_\psi$ an $(\alpha, a, \epsilon)-$VE for the $2^n$-dimensional quantum state $|\psi\rangle_n$, if*

$$\| |\psi\rangle_n - \alpha \left(\langle 0|_a \otimes I_n\right) U_\psi |0\rangle_{a+n} \|_2 \leq \epsilon. \tag{1.10}$$

Note that $(\langle 0|_a \otimes I_n)\, U_\psi |0\rangle_{a+n}$ corresponds to the exact vector encoded by $U_\psi$, specifically encoded in the first $2^n$ rows of the first column of $U_\psi$. The parameter $\alpha$ is a measure of the norm of the encoded vector, e.g., if $\epsilon = 0$ then $\|(\langle 0|_a \otimes I_n)\, U_\psi |0\rangle_{a+n}\|_2 = 1/\alpha$. One of the most essential components of working with matrix-vector arithmetic in quantum algorithms is tracking the norm of the encoded vectors throughout the algorithm, as the quantum complexity is usually inversely proportional to the norm of the encoded vector. Vector encodings give a methodical way to track encoded vector norms when implementing various matrix-arithmetic operations on the encoded vectors.

### Quantum Singular Value Transformation

I will now state the result enabling the quantum eigenvalue transform of Hermitian matrices by arbitrary complex polynomials of [26].

**Theorem 1.2.1** (Theorem 56 of [26])**.** *Given a degree$-k$ polynomial of the form $P(x) = \sum_{j=0}^{k} a_j x^j$ with $x \in [-1, 1]$ and $\forall j, a_j \in \mathbb{R}$ such that $\forall x, |P(x)| \leq 1/4$, a $(\alpha, a, \epsilon)$-block-encoding $U_H$ of a Hermitian matrix $H$, one can construct a quantum circuit $\tilde{U}$, implementing a $(1, a+2, 4k\sqrt{\epsilon/\alpha} + \delta)$-block-encoding of $P(H/\alpha)$. $\tilde{U}$ can be constructed with $O(ka)$ single and two-qubit gates, and a total of $k+1$ calls to a controlled $U_H$ and controlled $U_H^\dagger$ circuit. Moreover, the circuit description of $\tilde{U}$ can be computed with classical time complexity of $O(poly(k, \log(1/\delta))$.*

# 2

# The Limitations of Quantum Random Access Memory

This chapter (and the associated Appendix A) primarily presents Section IV (and the associated appendix entries), and the necessary prerequisite material, of the publication:

[1] Samuel Jaques and Arthur G Rattew. "Qram: A survey and critique". In: *arXiv preprint arXiv:2305.10310* (2023)

Some text is taken verbatim from the sections of the publication for which I was the original author. In summary, my primary contributions to this paper were the analysis of the limitations of active QRAM when used for quantum linear algebra applications. I also helped to establish the limitations of active and passive QRAM systems. For complete acknowledgments and funding disclaimers, please see the relevant sections of [1].

## Contents

**2.1 Introduction . . . . . 10**
**2.2 The Different Types of QRAM . . . . . 11**
2.2.1 QRAM Categories Depending on Data Type . . . . . 11
2.2.2 Circuit QRAM Derivation and Complexity Analysis . . 13
2.2.3 Passive and Active QRAM . . . . . 21
**2.3 Limitations of QRAM Applied to Linear Algebra and Machine Learning . . . . . 24**

2.3.1 Input Model and Matrix Functions . . . . . 25
2.3.2 Dequantization . . . . . 26
2.3.3 QRAM Versus Parallel Classical Computation . . . . . 29
2.3.4 Quantum Linear Algebra with Noisy QRAM . . . . . 34

## 2.1 Introduction

In many applications where we wish to use quantum processors to accelerate the underlying linear algebra of classical machine learning problems, we are interested in processing or analyzing some source of classical data which was produced in the real-world. This necessitates a storage mechanism to enable the processing of the data. For classical machine learning algorithms, these data will usually be stored in some sort of long-term storage (e.g., hard drives, solid state memory, etc.) and then moved into Random Access Memory (RAM) when it is actively being processed. Importantly, in classical algorithm analysis, memory access to a memory with $N$ elements is usually assumed to have $O(1)$ or $O(\log N)$ access cost. A quantum algorithm processing real-world data also needs a mechanism to store and access those data. Quantum Random Access Memory (QRAM) is the mechanism widely assumed to serve this purpose in the literature. The seminal bucket-brigade proposal of [27] introduced many of the ideas and techniques used in later QRAM proposals.

I will formally introduce QRAM in Section 2.2. It is widely assumed that QRAM has either an $O(1)$ or $O(\log N)$ access cost, matching the corresponding complexity of RAM access. However, this will turn out to be a problematic assumption which fails under scrutiny, leading to a number of substantial problems which must be addressed. In this chapter, I will outline some of the key issues with QRAM that we showed in [1], and then I will demonstrate how quantum speed-up in a broad range of linear algebra and machine learning tasks is actually lost when correctly accounting for the true cost of QRAM access.

## 2.2 The Different Types of QRAM

First, I will present four types of QRAM which depend on the nature of the accessing mechanism and on the nature of the data. I will next derive an implementation of QRAM in the circuit model with logarithmic circuit depth in the size of the memory. While it is widely assumed that such a circuit QRAM has a logarithmic depth, I'm not aware of any results explicitly proving a logarithmic depth circuit (although many will find it obvious). This is a new way to derive it, building up the circuit with layers of abstraction which clearly reveal the underlying routing behaviour, which assists with subsequent discussion. However, to emphasize, the novelty in Section 2.2.2 is primarily as an educational aid, and as a simple reference should one be needed for the exact circuit complexity of circuit QRAM. Finally, I will present two categories which each of these different QRAMs can be assigned to, active and passive, depending on details of the underlying implementation.

### 2.2.1 QRAM Categories Depending on Data Type

In [28] two categories are presented, which leads them to a natural definition for four types of QRAM:

- Is the data stored quantum or classical?
- Is the data accessed by a quantum or classical process?

Consquently, as given by [28], the four types of QRAM are: 1) Classical Access Classical Memory (CRACM), 2) Quantum Access Classical Memory (QRACM), 3) Classical Access Quantum Memory (CRAQM), and 4) Quantum Access Quantum Memory (QRAQM). Its worth briefly noting that CRACM is just classical memory (e.g., RAM), and that defining CRAQM requires some care, as obviously a classical process cannot directly read quantum data. We formally define each of these different types of QRAM in our paper. However, as I am primarily concerned with using quantum computers to accelerate classical algorithms operating with classical data, in this document I will use QRAM to mean QRACM, noting that this is the convention implicitly used in most of the quantum algorithms literature.

As such, I will now formally define QRAM:

**Definition 2.2.1** (Quantum Random Access Memory (QRAM))**.** *Let $n \in \mathbb{N}$, $d \in \mathbb{N}$, and define $N = 2^n$. Given a set of $N$ classical values, $\{x_i\}_N$, each represented with $d$ bits, We define the unitary $U$ to be a QRAM if it implements the mapping, $U|i\rangle_n|0\rangle_d = |i\rangle_n|x_i\rangle_d$.*

A key question is how the mapping $U|i\rangle_n|0\rangle_d = |i\rangle_n|x_i\rangle_d$ is actually implemented for arbitrary integer values of $x_i$. In some highly structured cases, e.g., where there exists an efficiently computable function (see Definition 1.2.4 for a formal definition), it can be possible to implement the mapping directly via quantum arithmetic (e.g., [29–34]) without the need for a large quantum memory. However, in general, there will not exist an efficiently computable function being able to map $N$ inputs to $D$ possible outputs (there are $D^N$ such functions). Thus, in general, $\Omega(N)$ memory is required to implement such a mapping. To see this, consider a paraphrased argument due to my co-author based on the original argument of Shannon [35]. Noting that NAND is universal for classical computation, assuming that either a NAND gate may be applied to any two wires or no operation will be applied, there are $O(2^{\text{poly}(n,m)})$ possible gate assignments in a single circuit layer of width $O(\text{poly}(n,m))$. Assuming the circuit depth is $O(\text{poly}(n,m))$ (and thus that this is an efficient circuit), then there are $O(2^{\text{poly}(n,m)})$ possible efficient circuits with $n$ bit inputs and $m$ bit outputs. Consequently, as there are $(2^m)^{2^n}$ possible such functions, there are exponentially more possible functions mapping $n$ bits to $m$ bits than there are efficient circuits acting on those bits. My coauthor makes a similar argument to show that $\Omega(N)$ gates are required in general to implement a QRAM with $N$ memory cells. This challenge is only exacerbated when the values in the QRAM need to be updated classically overtime (as a function efficiently implementing a specific function cannot necessarily be easily modified when values are updated). Finding cases where certain inputs can be prepared without necessitating large quantum memories is the focus of Chapter 3. Therefore, to reiterate, a QRAM which needs to assign values to $N$ possible addresses will

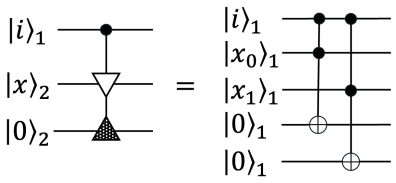


**Figure 2.1: Controlled register copy circuit example**. An example of the controlled register copy circuit, with a two qubit source and target register. The shaded triangle makes the target register pictorially explicit. This figure was prepared for a forthcoming manuscript currently in preparation, and I included it in my thesis to improve expositional clarity.

necessitate $\Omega(N)$ memory in general. This naturally leads to a discussion as to how such memory architectures can be designed to interface with classical data. In Section 2.2.2, I will derive a simple QRAM in the quantum circuit model which illustrates the key ideas. In essence, in one model, one can imagine each quantum memory register controlled by a classical register which sets the appropriate bit values at that address. The quantum circuit then handles routing the data to the appropriate address, yielding the output of the QRAM query.

### 2.2.2 Circuit QRAM Derivation and Complexity Analysis

I will now build up the tools necessary to compactly depict and analyse a QRAM circuit with logarithmic depth in the number of memory registers contained. To motivate how the circuit works, I will also present a naive QRAM circuit with depth linear in the number of constituent memory registers. These are standard results, and are presented to provide context and motivation for the following sections.

A key idea in all QRAM proposals, be it in the circuit model or otherwise, is that an input address signal needs to be efficiently routed to the appropriate memory register so that the data can be coupled to the correct state in an input address superposition [15]. In the following derivation, rather than routing the address signal, I use the address signal to route the memory data to the output register, which is equivalent but easier to depict with circuit diagrams.

First, I will begin by defining the following multi-register gate.

**Lemma 2.2.1** (Controlled Register Copy Circuit)**.** *We can define a controlled register copy circuit, which has a single control qubit, a d-qubit source register (containing a basis vector), and a d-qubit target register initialized as* $|0\rangle_d$*. The circuit copies the value of the basis vector in the source register into the target register with a total circuit depth of* $O(d)$*. It is easy to verify that this operation is a self-inverse.* [1]

An example of this circuit where the source and target registers have $d = 2$ is shown in Fig. 2.1. This circuit is easily implemented by simply copying each bit with a CX gate, conditioned on the control qubit (i.e., with a chain of Toffoli gates), yielding the $O(d)$ circuit depth mentioned. The fact that each bit must be conditioned on the control bit in sequence is why the circuit doesn't have $O(1)$ depth (as the CX gates would otherwise be parallelizable). This could actually be optimized by splitting the control signal through a binary cascade with $O(d)$ ancillas to have a circuit depth of $O(\log_2 d)$, but this is not a necessary optimization for the purpose of this exposition.

This gate definition is sufficient to provide a simple derivation of a naive circuit QRAM construction, which then motivates the depth-efficient QRAM circuit.

**Theorem 2.2.1** (Naive Circuit QRAM)**.** *Consider a QRAM architecture with n-bit address states (and thus* $N = 2^n$ *memory registers,* $\{|x_i\rangle_d\}_i$*), with each memory register containing d-bits of classical data. A simple QRAM circuit can be implemented by a sequence of N multi-controlled register copy circuits, with the* $i^{th}$ *only active when the address register has basis vector* $|i\rangle_n$*, and copying the contents of* $|x_i\rangle_d$ *to the output register. This circuit has* $\Omega(N)$ *depth.*

An example of such a naive QRAM circuit is shown in Fig. 2.2, where $n = 2$ (and thus there are 4 memory registers). The pattern of control bits selects, in order, the memory registers with addresses 00, 01, 10 and 11. This QRAM architecture is fundamentally limited by the fact that each register copy gate must be controlled on the same set of address bits, and output on the same output register. If the

[1] Obviously, since this only copies basis vectors, this does not violate the no-cloning theorem.

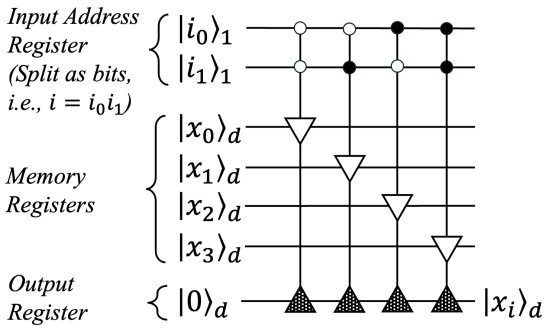


**Figure 2.2: Naive QRAM circuit example with 4 memory registers**. An example of a naive QRAM circuit consisting of 4 memory registers each with $d$-bits. This figure was prepared for a forthcoming manuscript currently in preparation, and I included it in my thesis to improve expositional clarity.

address registers where separate for each of the $N$ gates, and the output register was also separate, the gates could be parallelized with $O(1)$ depth. Consequently, to get a more efficient architecture, we will create $O(N)$ copies of the address bits through a binary branching process, and then we will use a set of routing ancillary qubits to efficiently move the addressed data to the lone output register, while enabling mass parallelization.

Before I present the efficient circuit QRAM architecture, I will define one last gate which implicitly handles much of the address signal splitting, enabling an abstract and simple representation of the circuit QRAM.

While the action of the following gate (called a Controlled Multi-Register Copy Circuit) may be a little obfuscated by the formal definition, its action is quite simple. If the control register is 1, it copies the basis vector in the $i^{th}$ source register into the $i^{th}$ target register. If the control register is 0, the gate acts as identity. Figure 2.3 clearly illustrates this gate, and defines it on an example with 4 target and source registers. The following formal result does not have to be understood to understand how the subsequent circuit QRAM architecture I present works. It is only necessary to give the analysis of the circuit complexity. To gain a high-level picture, only the previously stated action of this gate needs to be understood.

**Lemma 2.2.2** (Controlled Multi-Register Copy Circuit)**.** *Given a single-qubit control register, $N$ source registers each with $d$-qubits $\{|x_i\rangle_d\}_i$ (with each source*

*register a basis vector), and $N$ target registers in the $|0\rangle_d$ state, the controlled multi-register copy circuit ($\mathsf{C}$) is defined by its action in the standard basis as follows:*

$$\mathsf{C}|i\rangle_1 \left(|x_0\rangle_d \dots |x_{N-1}\rangle_d\right) \left(|0\rangle_d \dots |0\rangle_d\right) = \begin{cases} |i\rangle_1 \left(|x_0\rangle_d \dots |x_{N-1}\rangle_d\right) \left(|0\rangle_d \dots |0\rangle_d\right) & \text{if } i = 0 \\ |i\rangle_1 \left(|x_0\rangle_d \dots |x_{N-1}\rangle_d\right) \left(|x_0\rangle_d \dots |x_{N-1}\rangle\right) & \text{if } i = 1. \end{cases}$$

*Parens are included to visually distinguish the sets of source and target registers. This gate can be implemented with $O(N)$ total ancilla qubits, $O(dN)$ total operations, and $O(d + \log_2 N)$ circuit depth. Note, this gate can act on a subset of the registers in a given circuit. In this case, assuming the source and target registers are all numbered, the gate can be written as $\mathsf{C}^{i:j}_{k:l}$ to indicate that it copies source registers $i$ through $j$ into target registers $k$ through $l$ (then, $N = j - i + 1 = l - k + 1$). Moreover, it is easy to verify that this operation is a self-inverse.* [2]

*Proof.* This circuit can be implemented in three stages, (a), (b) and (c). To visualize the stages, consider Fig. 2.3.

(a) In stage (a), the input control signal is split into $N$ separate qubits, through a sequence of binary copy operation with $|0\rangle_1$ initialized ancillas. This is done through $\log_2 N$ steps, where in each step the number of qubits containing the control signal is doubled. Each qubit is doubled by applying two successive CNOT gates to it, each targetting a different fresh $|0\rangle_1$ ancilla qubit. Consequently, there are $2+4+...+N/2+N = N\sum_{i=0}^{\log_2 N-1} \frac{1}{2^i} = 2(N-1)$ ancillary registers (using the closed form for a geometric series). Moreover, there are two sequential operations performed in each step, yielding a circuit depth in this stage of $2\log_2 N$. Indeed, in box (a) of Fig. 2.3, you can see $4 = 2\log_2 4$ non-parallelizable CNOTs. Finally, it can be observed that each ancillary control qubit is the target of one and only one CX gate, and so the

[2] By optimizing the controlled register copy circuit of Lemma 2.2.1, the depth complexity of this gate could be improved to $O(\log_2 d + \log_2 N)$, while adding $O(d)$ ancillas, but this is not necessary for this discussion.

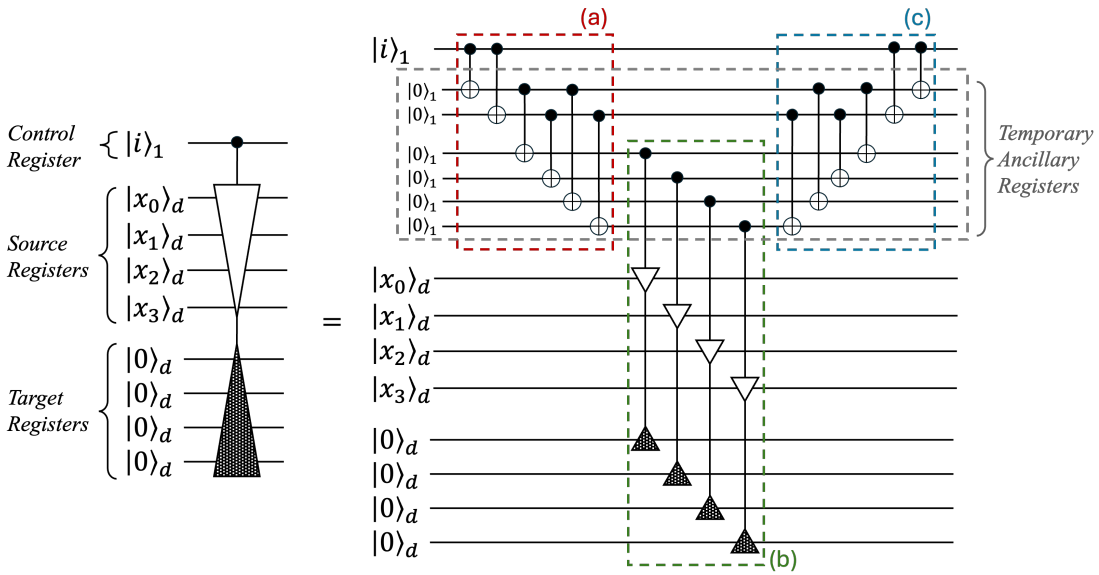


**Figure 2.3: Pictorial example of a controlled multi-register copy circuit**. An example of a naive QRAM circuit consisting of 4 memory registers each with $d$-bits. Again the shaded triangle makes the target registers explicit. This figure was prepared for a forthcoming manuscript currently in preparation, and I included it in my thesis to improve expositional clarity.

total count of operations is equally to the total count of ancilla qubits, i.e., is $O(N)$.

(b) As seen in Fig. 2.3 in stage (b) a sequence of controlled register copy circuits (from Lemma 2.2.1) is applied, such that each source register is copied into the corresponding target register, using a unique copy of the control signal. This allows all of the operations in this stage to be performed in the same layer (i.e., in parallel), giving a total circuit depth of the cost of one controlled copy circuit (i.e., a depth of exactly $d$ toffoli gates, so $O(d)$) and requiring a total of $O(dN)$ parallel operations.

(c) In stage (c), the ancillary registers are uncomputed, returning them to their original $|0\rangle_1$ states, allowing the ancillary registers to be removed after the gate is applied. This has the same complexity as stage (a) and thus does not change the asymptotic complexity of the gate.

Thus, the exact, non-asymptotic circuit depth in terms of the number of toffoli

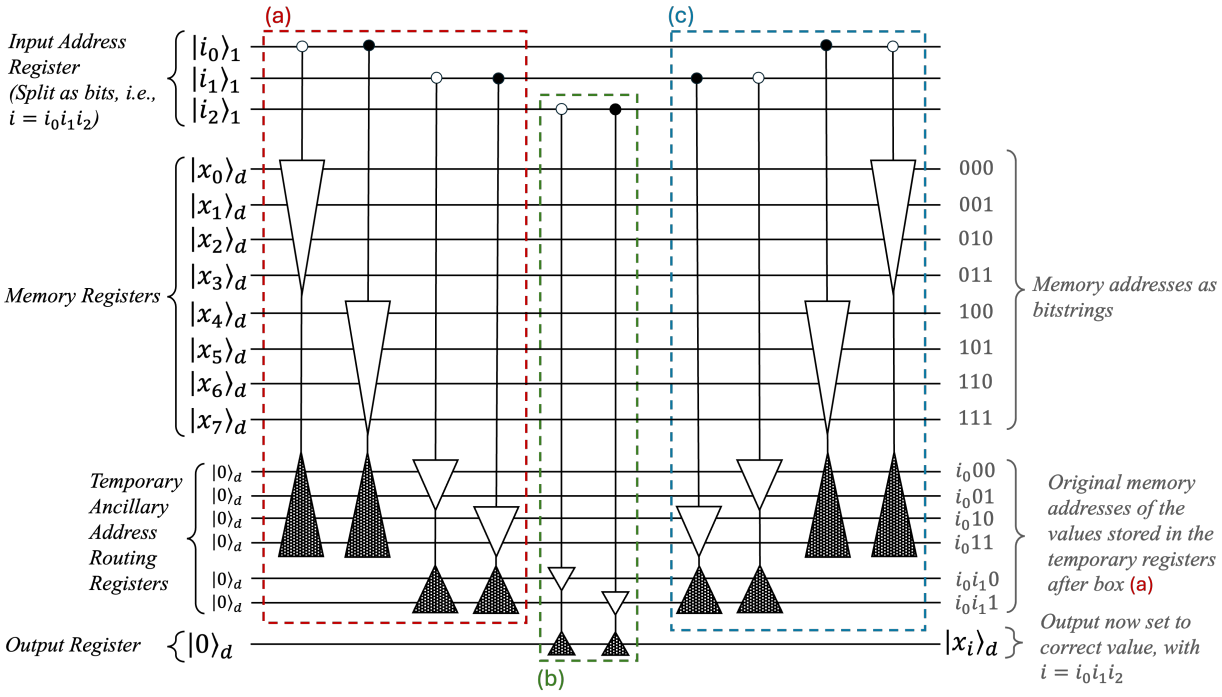


**Figure 2.4: Logarithmic depth circuit QRAM example with 8 memory registers.** The only gates used in this circuit are the controlled multi-register copy gates previously defined, whose action are to copy the set of registers indicated by the white triangles into the corresponding target registers indicated by the shaded triangles. Every gate in this circuit can be implemented exclusively with Toffoli gates, and so this is an entirely classical construction. This figure was prepared for a forthcoming manuscript currently in preparation, and I included it in my thesis to improve expositional clarity.

gates is $2\log_2 N + d + 2\log_2 N = 4\log_2 N + d$. Asymptotically, adding the costs from each stage, we see that the total number of ancilla qubits required are $O(N)$, the total circuit depth is $O(d + \log_2 N)$, and the total number of parallel operations performed are $O(dN)$. □

The machinery necessary to provide a concise and clear description of a depth-efficient QRAM circuit has now been developed. I will now provide an intuitive understanding for the efficient circuit QRAM architecture shown in Fig. 2.4, and will subsequently provide a formal proof. In the formal proof, I first prove the complexity of the QRAM circuit exactly as shown. I then give a simple modification to the circuit which reduces visual clarity, but removes the power of two from the logarithm in the depth complexity.

The circuit in Fig. 2.4 operates in three stages. In the first stage, the box labelled (a) in the figure, half of the data in the memory registers is copied to a set of $N/2$ ancillary registers, with the set copied depending on the first bit of the address register. For example, if the first bit in the address register is 1, then all the data with address $1i_1i_2\ldots i_{n-1}$ will be copied to the ancillary register with $N/2$ ancillary registers. This process is then repeated, conditioning on the next bit in the address register, and moving the data to the next set of appropriate ancillary registers (the number of which is halved each time a new bit is considered). In the second stage, the box labelled (b) in the figure, there are only two options for the correct output, and the final bit is used to determine which of the data in temporary storage is output. In the final stage, box (c) in the figure, the ancillary registers are all reset to their original starting state of $|0\rangle_d$. Essentially, a binary routing tree is implicitly happening where half the data is copied at each stage to satisfy the corresponding bit in the address register until the only data to be copied is the data that should be output for the given address query. Finally, observe that all of the gates in the circuit can be implemented by Toffoli gates (with some constant additional ancillas set to either 0 or 1 as needed).

**Theorem 2.2.2** (Log-Depth QRAM Circuit)**.** *Consider a QRAM architecture with $n$-bit address states (and thus $N = 2^n$ memory registers, $\{|x_i\rangle_d\}_i$), with each memory register containing $d$-bits of classical data. A circuit with an architecture based off the one shown in Fig. 2.4 implements a QRAM with circuit depth $O(d\log_2 N)$, with a total of $O(dN)$ parallel operations, and with $O(Nd + N\log_2 N)$ ancillary qubits.* [3]

*Proof.* First, I will prove the complexity of the QRAM circuit implied by Fig. 2.4. Then I will give a simple modification which improves the depth complexity. I will begin by proving the complexity of each of the three stages of the circuit, as they correspond in the figure:

[3] If the controlled copy circuit itself used a binary splitting of the control qubit, the factor of $d$ from its depth complexity could be reduced to $\log_2(d)$, and the overall circuit depth could be something like $O(\log_2 d \log_2 N)$, but this is beyond the scope of this illustration of circuit QRAM.

(a) There are $O(n)$ layers of controlled multi-register copy circuits (Lemma 2.2.2). The $i^{th}$ layer acts on a set of $N/2^{i+1}$ source and target registers, meaning that it uses $O(N/2^i)$ ancillary registers, has a circuit depth of $4\log_2(N/2^{i-1}) + d$, and performs a total of $O(dN/2^i)$ operations in parallel. Noting that the ancillary registers are cleared after each controlled multi-register copy circuit, the ancilla count is dominated by the number of ancillas in the QRAM circuit itself (rather than the internal ancilla in the controlled multi-register copy gates) and is thus bounded by $dN\log_{i=0}^{\log_2 N}\frac{1}{2^i} \in O(dN)$. The total number of operations is dominated by the case where $i = 0$, and so the total number of operations are given by $O(dN)$. Finally, the circuit depth is bounded by $\sum_{i=1}^{\log_2 N}(4\log_2(\frac{N}{2^i}) + d) \in O(d\log_2 N + \log_2^2 N)$.

(b) This is just a controlled copy gate (Fig. 2.1) acting on a $d$ bit target and source register, and thus has a depth of $O(d)$.

(c) This is just cleaning the ancilla qubits, and has the same complexity as (a), so it doesn't change the overall complexity.

Consequently, with the circuit as presented, the overall ancilla count and total number of parallel operations are $O(dN)$, and the circuit depth is $O(d\log_2 N + \log_2^2 N)$. However, lets examine where the $\log_2^2 N$ complexity term comes from. Since each of the controlled multi-register copy gates act on some of the qubits from the previous gate, they cannot naively be parallelized. However, the only overlapping operations are on the target and source registers of each controlled multi-register copy gate, and these operations have $O(d)$ complexity. As a result, if we simply handle all the address splitting operations in parallel at the start of the algorithm, we will reduce the depth complexity from $\log^2(N)$ to $\log N$, as I will now prove. The first step of the circuit would now be to create $2^{n-i-1}$ copies of the $i^{th}$ address bit, all in parallel. The circuit depth of this copying stage is thus bounded by the case where $i = 0$, would thus be given by $4\log_2(N/2) + d \in O(d + \log_2 N)$. Then, the controlled multi-register copy gates could be replaced with only the part in box (b) in Fig. 2.3 (i.e., removing the binary splitting stage and the ancilla clean-up

stage). Then, each controlled multi-qubit copy gate would thus add $O(d)$ circuit depth. Since there are $O(\log_2 N)$ of these gates, the circuit depth from box (a) in Fig. 2.4 would thus be $O(d \log_2 N)$. This would be the dominant term in terms of circuit depth, and so the overall circuit depth of this QRAM circuit would thus be given by $O(d \log_2 N)$. The total number of operations would remain unchanged (they've just been arranged in a more parallelizable manner now), and would thus be $O(dN)$. However, by parallelizing the splitting of the address bits, we can no longer reuse the ancilla qubits use for the controlled multi-register copy circuits. As a result, noting that the $i^{th}$ address bit requires $O(N/2^i) \in O(N)$ ancillas, and that there are $n = \log_2 N$ bits, the total ancilla count from splitting the address register would be $O(N \log_2 N)$. Adding in the ancillas from the QRAM circuit for routing addresses, the total ancilla count would be $O(Nd + Nlog_2 N)$. □

Throughout the proofs in this subsection, I've paid particular attention to tracking the total number of operations and not just the circuit depth (as is usually the case when deriving quantum algorithms). The reason for doing so will become clear in the next section.

### 2.2.3 Passive and Active QRAM

In this subsection, I condense key arguments made by my coauthor and myself relating to active and passive QRAM. This is important to motivate my subsequent contributions regarding the application of QRAM to quantum linear algebra problems.

A key idea in our paper is dividing all types of QRAM into two categories: active and passive. An active QRAM with $N$ memory cells has $\Omega(N)$ total (potentially parallelizable) cost per query. A passive QRAM has a total query cost of $o(N)$. Clearly, whether a given QRAM is active or passive depends on the underlying physical implementation of the memory.

As argued by my co-author, with regular RAM containing $N$ values, when a single address is provided, through a binary routing tree the cost to access the data at the specified address is $O(\log N)$; only the nodes along the path to the specified

address need be activated (with all nodes sitting in an idle state until receiving a signal to activate from a parent node). One might hope that a similar argument might also hold in the quantum case potentially enabling a QRAM to activate only the necessary routing nodes for a given input superposition, hopefully enabling an $o(N)$ access cost for certain input address superpositions (e.g., by only enacting certain gates in the routing network in a circuit QRAM implementation). This has a fundamental problem in that the controlling hardware would need to know which states are present in the input superposition to provide energy to the appropriate routing nodes, and it cannot learn this in general without collapsing the state. As a result, if a QRAM requires external intervention (depending on the input address superposition) at any node, then all nodes at that stage in the system must receive external intervention. For example, in the circuit QRAM presented in Section 2.2.2, the controlled copy operations routing the address to the memory nodes must all be applied in order to be able to handle any possible unknown input. This immediately implies that all circuit-based QRAMs are active (all the gates must be applied as though a uniform superposition of addresses were input), and that any QRAM using active error-correction on $\Omega(N)$ nodes must be active. To be absolutely clear, if the QRAM is implemented in the circuit model, then its circuit depth is $O(\log n)$, but it still has $\Omega(N)$ active gates which *must* operate in parallel. Consequently, the total (parallel) cost for this circuit QRAM is $\Omega(N)$. For an easy example illustrating why the special case of a circuit QRAM implemented by an error corrected circuit must be active, consider the following. If the circuit is error corrected, then each qubit has $O(1)$ classical processors that manage the error-correction, and so there are $\Omega(N)$ parallel resources required per QRAM query.

As a result, for a QRAM to be passive, it *cannot be error-corrected*[4]. Moreover, for any given input, a passive QRAM must be able to propagate the address signal through the QRAM system without expending energy at any given routing step, or requiring any sort of external intervention which depends on the specifics of the input address superposition. This means that even purpose built hardware

[4]There are some methods for passive error-correction, which are mentioned in the paper, but are not relevant to this discussion

implementing QRAM (outside of the circuit model) will be active if the underlying architecture requires external energy intervention to enact operations on each qubit (e.g., in the form of laser pulses on superconducting qubits). This greatly limits the types of possible implementations for a passive QRAM.

Essentially, for a QRAM system to be passive, each of the constituent qubits needs to be able to able to route arbitrary superpositions of input addresses to the corresponding memory registers *without requiring any energy to do so*, must be able to maintain the state of the memory registers without any energy input, and must do this while maintaining sufficiently high fidelity on all routing and coupling operations without any active error correction. While there are a handful of proposals which *may* eventually satisfy all these constraints (e.g., [27, 36–38]), even neglecting the formidable engineering challenges in their construction and *even assuming that such devices can be built*, there is still a substantial theoretical challenge that might preclude the utility of passive QRAM.

As mentioned, for a QRAM system to be passive it cannot be error-corrected. However, the quantum computer using the QRAM will have to be error-corrected to run any of the proposed fault-tolerant quantum linear algebra algorithms. As such, how can a noisy QRAM system be accessed by the logically encoded state in the error-corrected quantum processor? One naive answer would be to temporarily leave the protection of the error-corrected state in the main computational register, leaving the state as a physical quantum state, querying the QRAM, and then re-encoding the state. This has obvious fundamental problems. Another option would be to discover an error-correcting code which supports the entire QRAM operation transversally, which in essence, would mean that the QRAM could be applied to each of the physical qubits in some pattern and that the resulting operation would be equivalent to querying the QRAM from the logically encoded quantum state. If such an error-correcting code exists, and if a bucket-brigade-like [27] scheme (enjoying the error suppression benefits thereof [39]) could actually be implemented passively, then it is conceivable that there could be certain quantum linear algebra applications that could actually get real value using such a QRAM device. I briefly discuss this

possibility in Section 2.3.4. As shown in the circuit QRAM example, a QRAM can be implemented entirely using Toffoli gates, and so such an error-correcting code would need to support Toffoli transversally. Such an error correcting code may not exist (in a useful way), and my coauthor discusses this further in the paper.

In summary, it is dubious if a passive QRAM is physically realizable, and even if it is, it is unclear if it could actually be accessed in a useful way by an error-corrected quantum computer. As a result, the default assumption should be that QRAM will be active, and so algorithms need to be analysed through the lens of opportunity cost. This is what I will do in the next section (Section 2.3).

## 2.3 Limitations of QRAM Applied to Linear Algebra and Machine Learning

Given that I have now outlined the substantial challenges in constructing a passive QRAM, and that the only feasible QRAMs appear to be active, I will now analyse quantum algorithms for linear algebra and machine learning assuming that all QRAM assumptions are implemented by active QRAM systems.

Certain statistical and/or machine learning problems can be framed as linear algebra, where we transform some vectors by matrices. Quantum linear algebra aims to accelerate these transformations; however, if the input data are large databases, this requires QRAM. The following references provide an overview of some of the techniques used in, and applications of, quantum machine learning, in some cases discussing their assumptions regarding the necessity of QRAM [26, 40–42]. Moreover, the following references provide an overview of, and detail important techniques in, quantum linear algebra with some discussing the creation of the QRAM data structures relevant for linear algebra [15, 26, 43–47].

I now detail the input model and formally define the general linear algebra task considered. I then discuss the necessity of QRAM (i.e. for "unstructured" input data), and evaluate the opportunity cost of QRAM by giving algorithms for the classical co-processors to solve the defined linear algebra problem. I conclude that without passive QRAM, quantum linear algebra algorithms provide no asymptotic

advantage, unless at a scale large enough that the classical co-processors cannot share access to the same memory, but small enough that signal latency is negligible.

Ref [48] made these opportunity cost arguments for matrix inversion, [49] applied them to an approach to "Quantum PageRank", and [50] briefly note that these arguments generalize. This section expands and generalizes these ideas, incorporating questions of physical layout.

### 2.3.1 Input Model and Matrix Functions

Assume we are given some matrix $H \in \mathbb{C}^{N\times N}$ such that $H = H^\dagger$, via query access to the elements of $H$, and some vector $\boldsymbol{v} \in \mathbb{C}^N$ again via query access to its elements. The query access is provided via oracles which return the output associated with any given input. For example, the oracle for matrix $H$ may be given by

$$O_H|x\rangle|y\rangle|0\rangle = |x\rangle|y\rangle|H_{xy}\rangle. \tag{2.1}$$

In the case where $H$ is sparse (i.e. $H$ has at most $d$ non-zero elements in any given row or column), we also assume we are given another oracle specifying the locations of the non-zero elements. Moreover, I will focus on the case where the matrix under consideration is both square and Hermitian, noting that non-square and non-Hermitian matrices $\tilde{H}$ can be embedded into a Hermitian matrix $H := \begin{pmatrix} 0 & \tilde{H} \\ \tilde{H}^\dagger & 0 \end{pmatrix}$, as is commonly done in the literature [15]. In this embedding, the eigenvalues of $H$ are directly related to the $\pm$ singular values of $\tilde{H}$, and its eigenvectors are a simple function of the right and left singular vectors of $H$. Finally, define a function $f$ of a Hermitian matrix $H$ as a function of its eigenvalues, i.e. when $H|\lambda_i\rangle = \lambda_i|\lambda_i\rangle$, $f(H) := \sum_j f(\lambda_j)|\lambda_j\rangle\langle\lambda_j|$. This differs slightly from some definitions, which define the function on the singular values. I will focus on the general set of linear algebra problems which are equivalent to computing the vector $f(H)\boldsymbol{v}$. I further restrict $f$ to be a degree-$k$ polynomial, with small loss of generality since polynomials can approximate most functions of interest on the domain $[-1, 1]$. I give the formal problem statement in Problem 2.3.1.

**Problem 2.3.1** (Polynomial Eigenvalue Transform Problem)**.** *Given a polynomial function $f : [-1, 1] \mapsto \mathbb{C}$ of degree at most $k$, such that $\forall x, |f(x)| \leq 1$, an $N \times 1$ initial vector $\boldsymbol{v}$ and a $d$-sparse, $N \times N$, Hermitian matrix $H$, specified by an oracle $O_H$ (and an additional location-oracle in the case that $d \in o(N)$), such that $\|H\|_2 \leq 1$, compute $f(H)\boldsymbol{v}/\|f(H)\boldsymbol{v}\|_2$.*

Many quantum linear algebra tasks can be cast in the framework of Problem 2.3.1, such as matrix inversion, Gibbs sampling, power iteration, and Hamiltonian simulation [46]. In essence we are describing the Quantum Singular Value Transform (QSVT) framework [26], which can be seen as a unifying framework for nearly all quantum algorithms as summarized in [46]. The QSVT is defined on the singular values rather than the eigenvalues, but in the case of Hermitian matrices, the singular values are just the absolute eigenvalues, and so the problems are practically equivalent (with some additional tedium).

Of course, QRAM is not always necessary to offer query access to $H$ or $\boldsymbol{v}$. In some cases, there exist circuits with polylog($N$) depth which can implement the mappings, such as in [2, 29, 34, 51]. With entries that are not efficiently computable, an oracle to $H$ needs QRAM access to the $Nd$ non-zero entries of $H$. I assume a "wide" QRAM such as a bucket-brigade or fanout-and-swap, which needs $Nd$ quantum registers but provides $O(\log(Nd))$ circuit-depth per access. This analysis holds if the QRAM is implemented with fewer quantum registers but with longer access times, as that benefits the classical co-processors in comparison.

### 2.3.2 Dequantization

Tang [52] first introduced dequantization proving that the quantum algorithm for recommendation systems of Kerenidis and Prakash [44] does not have exponential quantum advantage over a randomized classical algorithm making a similar input assumption. The argument starts by noting that the $Nd$ non-zero elements of the input matrix must be stored somewhere (e.g., by preparing a QRAM gate or data structure), and this creates a $\Omega(dN)$ set-up cost. For example, the nodes of a bucket-brigade QRAM must be initialized with the appropriate data.

A similar amount of classical precomputation can create a data structure that allows efficient $\ell_2$-norm sampling from all columns of the input matrix. Even if data is added in an online setting to the quantum memory (amortizing the cost), with similar cost a classical data structure can also be maintained with online updates. Combined with a large body of classical randomized numerical linear algebra techniques (e.g. [53, 54]), one can obtain classical algorithms for Problem 2.3.1 with polylogarithmic dependence on the dimensions of the input.

Dequantization has some limitations. Most approaches require either the matrix to be low-rank [52, 55, 56], or for the matrix to be sparse and the function being applied to have at most a constant degree [56]. It also leaves room for some quantum advantage, with QSVT requiring $O(k)$ QRAM queries for a degree-$k$ polynomial, compared to an $O(k^9)$ cost for leading classical techniques [56]. In general, see Figure 1 of ref [57] to compare complexities of dequantization. New classical techniques could close these gaps, but currently there are many algorithms with at least a quartic speed-up over their dequantized counterparts, so if cheap QRAM existed, even early fault-tolerant devices might see an advantage [58].

Importantly, dequantization arguments allow cheap *access* to the QRAM, and only consider the opportunity cost of the quantum algorithm's input assumptions (e.g. the cost to construct the QRAM data structure). Instead I focus on the access cost. If the QRAM requires classical control or co-processing (e.g. laser pulses enacting the gates, or classical resources dedicated to error-correction), these polynomial quantum advantages over their dequantized counterparts vanish (and indeed the quantum algorithms would potentially even be exponentially slower than their dequantized counterparts) – such arguments even apply for some algorithms that cannot be dequantized.

**Dequantizing QRAM With Tensor Networks** It is also interesting to consider when the sampling data structures used in dequantization can be replaced with a tensor network.

In the following, I will primarily focus on the case where QRAM is used to construct a given quantum state (e.g., via a KP-tree [59]), i.e., where QRAM is used to justify a state-input assumption. The dequantized equivalent input assumption is called Sample Query (SQ) access [52]. For a given vector $\boldsymbol{x} \in \mathbb{R}^N$, $SQ(X)$ allows a sample $i$ to be drawn with probability $p_i = |x_i|^2 / \|\boldsymbol{x}\|_2^2$, and given an index $i$ also allows for the value $x_i$ to be computed efficiently. This can be readily extended to allow similar access for matrices, by creating an $SQ$ data structure for each row (and a separate $SQ$ structure for the norm of each row).

Tensor networks [60, 61] offer an efficient approach to approximate quantum states with low entanglement entropy by exploiting their intrinsic entanglement structure. In ideal cases, they can allow for drawing samples from a given state with complexity polylogarithmic in the number of qubits. If the tensor network also allows for query access to the amplitudes of the $N$-dimensional sampled state, then such a tensor network could be used to replace the $\Omega(N)$ sized $SQ$ dequantization data structure from [52] with a more compact representation. Loosely speaking, compact tensor network representations exist when the state is "close" to being in a product state, with maximally entangled states being the hardest to represent. As an example, tensor networks can be used to efficiently represent quantum states corresponding to discretized smooth 1-dimensional functions, since each additional qubit added increases the resulting entanglement entropy at an exponentially decreasing rate [62]. As noted above, these ideas could then be extended to similarly work with matrices. Combining tensor networks with dequantization approaches is an interesting area for future exploration, potentially allowing for improved randomized classical algorithms in practice.

Finally, considering a general QRAM oracle implementing the standard map, $|0\rangle|i\rangle \mapsto |x_i\rangle|i\rangle$, it would be interesting to explore for what kinds of data distributions $\{x_i\}_i$ tensor networks could be used to offer similar classical input assumptions, or potentially even to optimize quantum data-input procedures.

### 2.3.3 QRAM Versus Parallel Classical Computation

In this section I discuss the possibility of a general asymptotic quantum speed-up in Problem 2.3.1, by considering the opportunity cost of the classical control for the QRAM. I assume the QRAM has $O(Nd)$ logical qubits. With active error correction for each qubit, syndrome measurements and corrections will require substantial classical co-processing, with $O(1)$ classical processors per logical qubit. I assume each of these processors has constant-sized local memory with fast and efficient access. It seems unlikely that any *error-corrected* QRAM architecture could use *asymptotically* fewer classical resources. Even without error correction, only passive QRAM avoids costs proportional to $Nd$. Advances in quantum computing could drive the constant of proportionality to be smaller, but asymptotically the arguments in this section will still hold, short of a significant breakthrough in the manufacturing of passive QRAM (see Section 2.2.3 for a comparison of active vs. passive QRAM).

I then further consider some frequently neglected constraints on large scale parallel computing, such as signal latency, connectivity, and wire length, which gives three regimes of scale. The regime that considers total wire length but neglects the speed of light is probably the least realistic regime considered, and it is likely that the number of connections between processors/memory is more relevant than the total length of wires. Nevertheless, I include this regime as it is the only one we found which led to potential quantum advantage.

To start the comparison, I give a lower-bound on the number of QRAM accesses that any quantum algorithm will need to solve Problem 2.3.1, then show a classical algorithm that solves the problem with the same number of matrix-vector multiplications. I then compare parallel classical matrix-vector multiplication to a single QRAM access.

**Theorem 2.3.1** (Quantum Polynomial Eigenvalue Transform)**.** *Given a degree $k$ polynomial on the interval $x \in [-1, 1]$, with $f(x) = \sum_{j=0}^{k} a_j x^j$, a $d$-sparse Hermitian matrix $H \in \mathbb{C}^{N\times N}$ and an $\ell_2$-normalized vector $|\psi\rangle \in \mathbb{C}^N$, no general quantum algorithm can prepare the state $f(H)|\psi\rangle / \|f(H)|\psi\rangle\|_2$ with asymptotically fewer than $\Omega(k)$ QRAM queries to $H$, each query with cost $T$.*

| Scale | Assumptions | | Quantum advantage | |
|---|---|---|---|---|
| | Free wires | Instant Communication | Sparse Matrices | Dense Matrices |
| Small | Yes | Yes | None | None |
| Medium | No | Yes | $\tilde{O}((Nd)^{1/2})$ | None |
| Large | No | No | None | None |

**Table 2.1:** Comparing classical versus quantum algorithms for Problem 2.3.1 for $N \times N$ matrices, under different architectural assumptions with active QRAM. It may appear strange that the potential quantum speedup in the Medium-Scale regime increases as $d$ increases, and then suddenly vanishes when $d \in \Omega(N)$. We suspect that a classical algorithm for the medium-scale regime for sparse matrices with $d \gg 1$ could be improved. This table was made in collaboration with my co-author.

I delegate the formal details of this proof to Lemma A.0.1, and the result of [26] almost immediately implies this bound. The following concise derivation is included to enhance the clarity of presentation.

The main idea is that one can efficiently construct a block-encoding (see Definition 1.2.6) of a rank one projector from a uniform superposition to the marked state of an unstructured search problem (specified via an oracle) with a singular value equal to $\frac{1}{\sqrt{N}}$ (where $N$ is the dimension of the space). This can be done with $O(1)$ queries to the unstructured search oracle; however, immediately applying the block encoding to an initial state has a low probability of success. I thus embed this in a rank-2 Hermitian matrix and apply a polynomial approximation to the sign function on the eigenvalues of this Hermitian matrix. The degree of this polynomial is $\tilde{O}(\sqrt{N})$, and it gives a constant probability of success in measuring a solution to the unstructured search problem. Consequently, if this polynomial could be implemented with $o(\sqrt{N})$ queries, we could solve the unstructured search with less than $\Omega(\sqrt{N})$ complexity. Since unstructured search has an $\Omega(\sqrt{N})$ lower bound, this creates a lower bound of $\Omega(k)$ for any general quantum algorithm implementing an eigenvalue transform of a degree-$k$ polynomial.

**Theorem 2.3.2** (Classical Parallel Polynomial Eigenvalue Transform)**.** *Given a degree $k$ polynomial on the interval $x \in [-1, 1]$, with $f(x) = \sum_{j=0}^{k} a_j x^j$, a $d$-sparse Hermitian matrix $H \in \mathbb{C}^{N \times N}$, and an $\ell_2$-normalized vector $\boldsymbol{v} \in \mathbb{C}^N$, a parallel*

*classical algorithm can exactly compute the vector* $f(H)\boldsymbol{v}/\left\|f(H)\boldsymbol{v}\right\|_2$ *with* $O(k)$ *matrix-vector multiplications and* $k$ *vector-vector additions.*

*Proof.* By definition,

$$f(H) = \sum_{j=0}^{N-1} \sum_{l=0}^{k} a_l \lambda_j^l |\lambda_j\rangle\langle\lambda_j|. \tag{2.2}$$

Applying the completeness of the eigenvectors of $H$, we immediately get $f(H) = \sum_{j=0}^{k} a_j H^j$. This means we must simply compute $f(H)\boldsymbol{v} = \sum_{j=0}^{k} a_j H^j \boldsymbol{v}$. This can be done with $k$ matrix-vector multiplications by computing the powers $H^1\boldsymbol{v}$, $H^2\boldsymbol{v}$,...,$H^k\boldsymbol{v}$, and adding $a_j H^j \boldsymbol{v}$ to a running total vector after each multiplication. All the powers $H^j\boldsymbol{v}$ do not need to be stored, since each one can be deleted once added to the total and the the next power of $H$ has been computed. Once the unnormalised vector $f(H)\boldsymbol{v}$ has been constructed, it can normalize by computing the $\ell_2$-norm (which requires $O(N)$ parallelizable operations, and $O(\log N)$ joining steps), and then by diving each element in the output vector by this norm, which requires a further $O(N/P)$ steps. I assume that the cost of matrix-vector multiplications dominates the cost of this normalization. □

From now on I ignore the cost of vector-vector additions (which parallelize perfectly), since they are likely negligible compared to matrix-vector multiplications. Thus, this comparison comes down to to the expense of matrix-vector multiplication. In this computational model, the QRAM with $Nd$ registers implies a proportional number of classical processors, so I will now consider highly parallel matrix-vector multiplication at different scales. Table 2.1 summarizes our conclusions.

**Small-Scale Regime:** At this scale all latency and connectivity concerns are ignored. The QRAM access can then be done in $\Theta(\log(Nd))$ time (i.e., circuit depth). Assume the classical processors can access a shared memory of size $Nd$, also with $O(\log(Nd))$ access time. A standard parallel matrix multiplication algorithm (see Lemma A.0.2) allows the the $P = \Theta(dN)$ processors to multiply the matrix and vector in $\tilde{O}(1)$ time, matching the assumed time of the QRAM access. Together with

Theorem 2.3.1 and Theorem 2.3.2, the classical algorithms achieves $\tilde{O}(k)$ scaling, while the quantum algorithm achieves a $\tilde{\Omega}(k)$ lower-bound for the general linear algebra task, so *there can be no asymptotic quantum advantage* at this scale.

**Medium-Scale Regime:** In this regime, we assume that total wire length must be counted (e.g., as longer wires cost more money), while still assuming that the speed-of-light is instant. Given classical processor clock-speeds, this is probably the least realistic regime, but we consider it nevertheless. A significant consequence of this assumption is that we cannot immediately assume that our classical co-processors have read/write access to all the memory cells – instead we must assume that they only have access to their local memory, and we must count the total length of all the connections in the entire system. Interestingly, in this regime we obtain different results for the quantum speedup when $d \in \Omega(N)$ (dense matrix), and when $d \in o(N)$ (sparse matrix), and so I will break the proof into two cases. To facilitate comparison, I will first discuss the complexity of QRAM in this regime.

First, if $Nd$ registers (quantum or classical) are to be arranged in 2D space, the width of this collection must be $\Omega((Nd)^{1/2})$. However, as noted by my co-author, if the QRAM is well-designed it can have $O(Nd)$ total wire length (e.g., with an H-tree). Moreover, as the speed of light is assumed to be instant, the width of the device does not impact the query time, and so it will still have $O(\log(Nd))$ access time complexity.

Now I will consider the dense case, when $d \in \Omega(N)$. In Lemma A.0.4 I prove that a parallel matrix vector multiplication algorithm with *only* local connectivity and a total wire length of $\tilde{O}(dN)$ can perform matrix multiplication with a parallel time complexity of $\tilde{O}(1)$. As a result, Theorem 2.3.2 gives us a complexity for Problem 2.3.1 of $\tilde{O}(k)$, matching the quantum lower-bound for this problem, proving that for dense $H$, in this asymptotic regime, there can be no asymptotic quantum advantage in this general linear algebra task.

Now to consider the sparse case. In order to invoke Theorem 2.3.2, we must show the complexity of classical matrix-vector multiplication for a sparse matrix in this

regime. The fundamental challenge for the classical algorithm is in efficiently routing the sparse elements to the corresponding vector elements with the local connectivity constraints, without exceeding the total wire length of the 2D QRAM embedding. My co-author solved this by embedding the processors into a two-dimensional grid, which he notes can sort in time $O((Nd)^{1/2})$, with a total wire length of $O(Nd)$. Invoking Lemma A.0.3 then shows that in this setting matrix-vector multiplication has time complexity $\tilde{O}((Nd)^{1/2})$. Invoking Theorem 2.3.2 then shows that the quantum speedup in this regime is $\tilde{O}((Nd)^{1/2})$.

Strangely the quantum advantage increases with $d$ until vanishing when $d = \Omega(N)$. It seems likely that either one can give a better classical algorithm for the sparse case in general, or that there exists a classical algorithm whose performance improves as $d$ gets larger (using the additional processors to perform the routing more efficiently), eventually matching the dense case as $d \mapsto N$.

**Large-Scale Regime:** At sufficiently large scales the speed of light becomes an important bottleneck, as algorithms must wait for signals to propagate through the underlying computer. Given a classical processor architecture of $O(dN)$ processors, each with local memory, and a QRAM with $O(dN)$ memory cells, the classical and quantum systems need to be embedded in our three spatial dimensions. Here, a $2d$ grid embedding with local connections is assumed. One could use a different embedding, such as one of the ones described previously, but they all necessitate wire-lengths that end up incurring transmission time penalties that result in overall asymptotics no better than a simple grid embedding. In this model, the QRAM access cost of the quantum algorithm is $T = O(\sqrt{dN})$, implying a lower-bound for a quantum algorithm solving Problem 2.3.1 of $\Omega(k\sqrt{dN})$, as per Theorem 2.3.1. In the classical case, we used the reduction to sorting-network matrix multiplication, which has asymptotic cost equal to the complexity of the sort-time, i.e. of $\tilde{O}(\sqrt{(dN)})$. As a result, using Theorem 2.3.2, we get an overall classical bound for Problem 2.3.1

of $\tilde{O}(k\sqrt{dN})$. Thus, in the regime of large memory scales where the speed-of-light must be considered, there is no quantum advantage in the general linear algebra problem I have defined.

In conclusion, which of these regimes will govern a specific practical application comes down to the constants of that application. In all regimes apart from the medium-scale regime (where wires are not free, but communication across wires is instant) with sparse matrices, we showed that there can be no asymptotic quantum advantage in Problem 2.3.1. It is likely that there is actually no asymptotic advantage in this regime as well, should a more efficient matrix-vector multiplication algorithm exist given these constraints. However, if no such algorithm exists, then quantum speed up at this scale depends on whether the constant costs of quantum computing exceed the constant cost due to the time it takes EM waves to propagate signals through the wires in the classical parallel computing network.

### 2.3.4 Quantum Linear Algebra with Noisy QRAM

The opportunity cost arguments made in the last section apply for any active QRAM system, but are easiest to understand for error-corrected architectures (which require significant classical co-processing). In contrast, passive QRAM systems (potentially such as certain implementations of bucket-brigade) suffer no such opportunity cost, as once the QRAM query is initiated, the system requires no further energy input or intervention (although they will necessarily be noisy). As such, if sufficient progress can be made in the construction of a passive QRAM system (despite the challenges identified in Section 2.2.3), then for certain linear algebra algorithms that can tolerate some noise in their QRAM queries, it is possible that practically relevant quantum advantage could be realized.

For example, with the bucket brigade architecture, the overall probability of an error per query to a QRAM with $N$ memory cells scales as $\sim p\log(N)^2$, where $p$ is the physical probability of an error (see [39]). This immediately rules out an asymptotic advantage for any quantum algorithm asymptotically improving error-rate dependence (for arbitrarily small errors), but may be practically useful in

cases where either the overall algorithm is relatively insensitive to QRAM access errors, or if the physical error rate can be made to approach the necessary precision without error correction or external intervention.

In some linear algebra tasks, a small constant error per QRAM access *may* be fine. For instance, in some machine learning applications the underlying data is already noisy, so additional noise may be acceptable. In other cases, such as the training and deployment of large language models (LLMs), it has been demonstrated that models can still function with reasonable accuracy when the weights are quantized down to 16-bits [63], 8-bits [64], or *even down to* 2*-bits* [65]. Clearly a two-bit quantization introduces substantial noise in the evaluation of a model, and if LLMs can still function in this regime, it is possible they could be similarly resilient to noise in a QRAM query to their parameters.

Moreover, in reinforcement learning, additional noise could assist an agent to explore its environment, rather than just exploit its existing policy [66].

For matrix inversion with a structured matrix (i.e. a matrix not stored in QRAM) on an input vector constructed from QRAM, for well-conditioned matrices the output of the matrix inversion is relatively insensitive to small perturbations in the input vector. For matrices in QRAM, for some classes of matrices the inversion could be insensitive to perturbations in the structure of the matrix itself [67].

Further exploration into error-corrected quantum linear algebra algorithms using noisy QRAM systems would be a valuable addition to the literature. I do not rule out the possibility of practical quantum advantage in such cases. Of course, at some scale the uncorrected noise will overwhelm nearly any algorithm, but with sufficient noise resilience (in both e.g. the bucket-brigade QRAM and in the algorithm itself), at realistic scales such quantum algorithms could still be of significant commercial and scientific value.

# 3
# Quantum State Preparation

This chapter (and the associated Appendix B) presents the results of the following paper:

[2] Arthur G Rattew and Bálint Koczor. "Preparing arbitrary continuous functions in quantum registers with logarithmic complexity". In: *arXiv preprint arXiv:2205.00519* (2022)

This paper was a close collaboration with Bálint Koczor, and we contributed equally. Some text is taken verbatim from the paper. Most of the results in this chapter were developed through a highly collaborative process. When results are primarily due to my co-author, I explicitly label them as such. For complete acknowledgments, funding sources, and disclaimers, please see the relevant sections of [2].

## Contents

**3.1 Introduction . . . . . 38**
3.1.1 Prior Work . . . . . 40
**3.2 Algorithm Overview . . . . . 42**
3.2.1 Efficient Implementation of the Time Evolution . . . . . 45
3.2.2 Limitations . . . . . 47
**3.3 Numerical Demonstration . . . . . 49**
**3.4 Further Applications . . . . . 51**
**3.5 Discussion and Conclusion . . . . . 52**

Given the challenges associated with constructing a passive QRAM, as outlined in Chapter 2, one naturally wonders what classes of data can be efficiently loaded into quantum computers. In this chapter, we will consider discretized continuous functions. First, I present our contributions showing that the asymptotic properties of continuous functions allow them to be efficiently prepared. Then, I present our novel adiabatic state-preparation algorithm.

## 3.1 Introduction

With rapidly accelerating progress in hardware development, quantum computers are quickly becoming a realistic technology. New records are constantly being set, demonstrating that these machines are capable of significantly outperforming the best classical computers in certain settings [5–9]. As the technology is scaled up, it will ultimately enable powerful quantum algorithms capable of both polynomial and exponential speed-ups over their classical counterparts. However, many of these algorithms require efficient state preparation procedures to achieve their speedup, as is the case in: option pricing [68–70], machine learning [41, 71, 72], matrix inversion [15], quantum chemistry [73, 74], and quantum Monte Carlo based algorithms [75, 76]. As such, it remains a crucial open question: can we *efficiently* prepare quantum states given a set of prespecified amplitudes?

Unfortunately, a quantum circuit capable of producing an arbitrary quantum state necessitates exponential complexity in general [77], although recent works have shown that the exponential circuit depth cost can actually be made linear in exchange for utilizing an exponential number of ancillary qubits [51, 78–80]. As a result, in practice, it is essential to exploit specific properties of the state being produced.

In this work, we devise a novel quantum algorithm for efficiently preparing continuous (as well as some more general) functions in quantum registers, and we do so with rigorous theoretical guarantees. An additional important contribution of our work is the observation and proof that continuous functions have asymptotic properties that enable state preparation procedures to prepare such quantum states

efficiently (i.e. with constant query complexity). While we illustrate the core subroutine underpinning our technique in Fig. 3.1, our main result is stated as Theorem 3.2.1. Succinctly and informally stated, we prove that our approach has a constant query complexity (asymptotically in the number of qubits), and obtains a bounded error which can be made arbitrarily small. A remarkable feature of our approach, and a consequence of our observations regarding the asymptotics of continuous functions, is that the performance of our algorithm becomes independent of the qubit count as we increase the number of qubits, and that we can reach this asymptotic independence exponentially quickly in the number of qubits.

The constant cost of our algorithm depends inverse-polynomially on the *filling ratio* of the function – a quantity that we define in a subsequent section and illustrate in Fig. 3.2 as the ratio of blue area to the area of the total bounding box. While it is possible to construct pathological functions for which the filling ratio is arbitrarily small, for realistic functions encountered in practice this quantity is typically very reasonable (e.g. usually around $10^{-1}$ to $10^{-3}$). We additionally suggest ways that this constant cost can be made tractable even in pathological cases. Moreover, if future works improve the error-dependence of our simulation subroutine, the impact of the filling ratio will be even further negated.

I note that the filling ratio is actually a special asymptotic case of a quantity we term the *generalized filling ratio*. Given a function $f$ defined on the grid $\{x_1, ..., x_N\}$, the vector norm $\mathcal{N}^2 = \sum_{j=1}^{N} |f(x_j)|^2$, and the maximum of the function $M$, the generalized filling ratio is defined as $\tilde{\mathcal{F}} := \mathcal{N}/\sqrt{N}M$. The generalized filling ratio is a quantity frequently used to describe the complexity of existing state-preparation procedures in the literature, and in this work we establish that in the limit $N \mapsto \infty$, $\tilde{\mathcal{F}}$ tends to $\mathcal{F}$ (proven in Lemma B.6.2). A consequence of this observation is that our novel technique actually has an asymptotically *constant* query complexity when preparing discretized continuous functions in quantum registers (in the limit of large $N$). We note that this observation can also be applied to some of the existing techniques in the literature (which we will summarize shortly) enabling them to also efficiently prepare discretized continuous functions in quantum registers.

The implications of our work go significantly beyond just enabling the speedup of existing algorithms via efficient state preparation. Functions have been used for centuries to model physical, chemical, biological, and financial phenomena, and have played crucial roles in the industrial revolution and in modern engineering. As the use of analytical techniques for functions modelling realistic systems rapidly becomes intractable, numerical techniques often become the only available option [81]. As a result, computational tools accelerating such numerical analyses are of broad and vital importance. To this end, the techniques we develop in this paper not only allow for the preparation of broad classes of functions in quantum registers, but also enable a variety of tools such as for numerical integration of Lipschitz continuous functions, for the optimization of continuous functions, and even for sampling from probability density functions.

We begin by reviewing related work. In the next section we briefly summarise our problem statement and our main results. There, we informally state our rigorous error bounds which confirm that our approach is efficient, noting that we defer all technical details to the appendix. We then summarize the implementation of a key subroutine in our approach: the efficient time-evolution of rank-one dense matrices corresponding to projectors onto discretized continuous functions. We also discuss the limitations of our approach in a subsequent section. We then demonstrate the main algorithm in relevant applications, and support the practicality of the approach with large-scale numerical simulations confirming theoretical expectations. We finally discuss further applications and then conclude.

### 3.1.1 Prior Work

As early as in 2000 Grover proposed a general approach that can be used to load any efficiently computable number sequence as amplitudes of a quantum state [87]. The approach relies on amplitude amplification and achieves provably optimal complexity $O(\tilde{\mathcal{F}}^{-1})$ (which as we prove gives an asymptotic complexity of $O(\mathcal{F}^{-1})$). A series of follow-up works have reduced its absolute gate count in [88–90]. We summarise these techniques in Appendix B.5.1 and compare to the present approach.

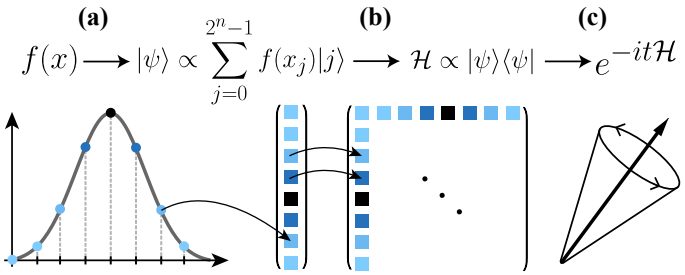


**Figure 3.1:** Key rank-1 Hamiltonian simulation subroutine. **(a)** Samples (coloured circles) of the continuous function $f(x_k)$ (grey smooth curve) are proportional to the amplitudes (coloured squares) of the quantum state $|\psi\rangle$ as per Eq. (3.1). **(b)** We embed this vector into a rank-1 matrix $\mathcal{H} \propto |\psi\rangle\langle\psi|$ whose matrix entries are given as $f(x_k)f^*(x_l)$ (colours are only illustrative, not exact) and are efficiently computed by a quantum oracle, e.g., via arithmetic operations. **(c)** We then efficiently simulate the time evolution $e^{-it\mathcal{H}}$ via low-rank [82] and 1-sparse [83–86] simulation techniques. This powerful time evolution resource allows us to prepare the ground state of $\mathcal{H}$ as our desired $|\psi\rangle$ using, e.g., adiabatic evolution, phase estimation and Hadamard tests. Note that each stage occurs coherently in superposition. This figure (and caption) was produced by my co-author, and is included for completeness.

While we introduce a fundamentally different approach to solve the continuous state preparation problem, a major contribution of our work is that it introduces an asymptotic analysis of discretised continuous functions and thus complexities can be stated in terms of properties (norms) of the continuous functions.

Furthermore, in 2002 Grover and Rudolph presented a procedure for preparing an efficiently integrable probability density (i.e., non-negative, $L^1$-normalised) function [91], which we briefly summarise in Appendix B.5.3. Under certain assumptions on the efficient integrability of the function this approach theoretically has a complexity polylogarithmic in the desired resolution, but exponentially scaling variants have also been suggested [62, 92, 93]. Nevertheless, in certain applications any variant of the procedure negates any potential quantum speedup due to its use of quantum integration [68, 94]. For example, quantum Monte Carlo based algorithms lose their speedup as pointed out by Herbert [94] and similarly in derivative pricing as observed in the work of Chakrabarti *et al.* [68]. In contrast, the set of techniques presented in this work strictly generalize the Grover-Rudolph-type approaches, as they are not limited to efficiently integrable non-negative probability density functions – they can be applied to any complex

valued continuous function and beyond. Moreover, our techniques are not reliant on quantum integration, and thus do not suffer from the limitations of the Grover-Rudolph based approaches just mentioned.

We also note that Holmes *et al.* recently presented an approximate procedure for preparing smooth analytic functions [62] via a piecewise polynomial approximation. However, it is not straightforward to assess the approximation error of the approach, its overall complexity has not been established, and it is asymptotically limited in its use of the piecewise polynomial approximation.

## 3.2 Algorithm Overview

While our approach is applicable to the general class of efficiently computable mappings $F^{(n)} : \{0,1\}^n \mapsto \mathbb{C}$ via Appendix B.3.2, in the following we restrict ourselves to the practically most important case of discretised continuous functions. In particular, given an arbitrary continuous function $f : [a,b] \mapsto \mathbb{C}$ on the closed interval $[a,b]$, we wish to prepare the $n$-qubit normalized quantum state

$$|\psi\rangle = \frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} f(x_j)|j\rangle, \tag{3.1}$$

where $\mathcal{N}$ is the normalization factor, $N = 2^n$, $|j\rangle$ is a standard basis vector with $j \in \{0,1,...,N-1\}$, and $x_j$ is the $j^{th}$ grid point in a uniform discretization of the interval $[a,b]$—while we later remark that non-uniform grid spacings may sometimes be advantageous. In general, preparing an arbitrary state in a Hilbert space of $n$ qubits requires exponential complexity as the dimension of the space grows exponentially in $n$ [51, 77]. However, the algorithm we are about to present demonstrates that the very general set of states of the form shown in Eq. (3.1) (i.e. those corresponding to discretized continuous functions) can be prepared efficiently.

In Appendix B, we present our results for the analogous, but special, case of efficiently integrable probability density functions as in the case of the Grover-Rudolph [91] approach. However, for brevity, here we focus only on the integration-free case where the state is prepared via samples of $f$ rather than through its

integration. Moreover, we note that the point-wise state preparation approach is strictly more general than integration-based approaches.

Below we present a rank-1 simulation procedure which is both a fundamental subroutine and a primary contribution of this paper. We note that this is an alternative to QSVT based approaches, and likely offers advantages in constants for fixed error rates. In summary, we establish how the time evolution $e^{-it\mathcal{H}}$ under the dense rank-one matrix $\mathcal{H} \propto |\psi\rangle\langle\psi|$ can be efficiently implemented. Simulating this time evolution is a powerful resource and can be combined with diverse tools from quantum information processing to prepare the quantum states $|\psi\rangle$ efficiently. Nevertheless, in the main text we focus on the conceptually most straightforward variant for preparing states of the form in Eq. (3.1), based on adiabatic evolution.

We first formally define the filling ratio of the function being prepared as it is intrinsic to the asymptotic complexity of our state preparation procedures. We then informally state our main result for the adiabatic procedure, and then we describe the algorithm.

As stated in Definition B.1.2, the filling ratio $\mathcal{F}$ of a function $f$ is defined as $\mathcal{F} := \|f\|_1/[(b-a)\|f\|_{max}]$ and has a value $0 < \mathcal{F} \leq 1$. Intuitively, it can be understood as the integral of the absolute function (i.e. $|f(x)|$) divided by the area of the box bounding the absolute function (i.e. the box with width $b-a$ and height $\max_{x\in[a,b]} |f(x)|$). The filling ratio is clearly pictured for some common functions in Figure 3.2, as the ratio under the curves, to the total area of the bounding box. Finally, we note that we formally define the notion of efficient computability in Section 3.2.1 as well as in Definition 1.2.4.

The following result is an asymptotic improvement over the version in our paper [2].

**Theorem 3.2.1** (informal version of Theorem B.3.1)**.** *For an efficiently computable continuous function $f$, our approach prepares the $n$-qubit state $|\psi\rangle = \frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} f(x_j)|j\rangle$ with $N = 2^n$ up to error $\epsilon$ with query complexity $O\left(\mathcal{F}^{-2}/\epsilon^2\right)$, where $\mathcal{F}$ is the filling ratio. The error $\epsilon$ is the deviation from an ideal unitary in terms of a spectral distance. The algorithm uses $O(n+d)$ ancillary qubits (where*

*d is the number of digits used in the discretization of f), and has a probability of failure bounded by* $O(\epsilon^2)$.

It is worth additionally noting that the success probability only enters our bound to simplify our proofs – in practice our approach effectively succeeds with probability 1. This is discussed comprehensively in the Appendix.

To enable the adiabatic state preparation procedure, we define a parameterized family of continuous functions $f_s := (1-s)f_0 + sf_1$, where $f_0 \propto 1$ (i.e. corresponds to an easy-to-prepare state) and $f_1$ corresponds to the state we wish to prepare (i.e. $f_1 := f$) with $0 \leq s \leq 1$. We then define the parameterized quantum state, $|\psi_s\rangle \propto \sum_{j=0}^{N-1} f_s(x_j)|j\rangle$. The procedure begins by creating the state $|\psi_0\rangle$, which by construction of $f_0$ is simply $|+\rangle^{\otimes n}$. We then slowly adiabatically morph this starting state to the final state $|\psi_1\rangle$ by evolving according to the rank-1 projector $\mathcal{H}(s) \propto |\psi_s\rangle\langle\psi_s|$. Here, one must pick the total evolution time $T$ so as to satisfy the adiabatic theorem and ensure that the state remains in the ground state of the morphing Hamiltonian [95]. As we prove in Appendix B.3, $T$ is constant bounded (and can therefore be selected utilizing that bound). Of course, we can't implement a continuous time evolution in practice, and so we discretize the temporal grid uniformly into $r$ time-steps. In the $j^{th}$ time-step, we evolve according to the time-independent Hamiltonian $\mathcal{H}(\frac{j}{r})$, yielding the overall evolution:

$$e^{-i\frac{T}{r}\mathcal{H}(\frac{r}{r})}e^{-i\frac{T}{r}\mathcal{H}(\frac{r-1}{r})}\cdots e^{-i\frac{T}{r}\mathcal{H}(\frac{1}{r})}. \tag{3.2}$$

As shown in Appendix B.3 this discretization has constant bounded error, which can be made arbitrarily small. Naturally, it is important to explain how each evolution $e^{-i\frac{T}{r}\mathcal{H}(\frac{j}{r})}$ is implemented, and we do so in Section 3.2.1.

However, we first explain the conceptual difference between "conventional" adiabatic quantum computation, and the approach we have just described. In the standard formulation of adiabatic quantum computation, one defines the interpolated Hamiltonian $\mathcal{H}(s) = (1-s)\mathcal{H}_0 + s\mathcal{H}_1$, where $\mathcal{H}_0$ has an easy to prepare ground-state, and $\mathcal{H}_1$ encodes the problem of interest [96]. However, in this definition the energy gap often vanishes (or becomes exponentially small), resulting in the total evolution

time $T$ becoming intractable. By defining our interpolated Hamiltonian as the rank-1 projector $\mathcal{H}(s) \propto |\psi_s\rangle\langle\psi_s|$, we circumvent this limitation. Indeed, in the appendix we prove that the spectral gap $g(s)$ is generally lower bounded by a constant, e.g., $g(s) \geq 1/2$. This ensures that our total evolution time required is independent of the problem size $n$.

### 3.2.1 Efficient Implementation of the Time Evolution

Before describing our time evolution procedure, we outline our oracle query model. Our procedure allows the preparation of a quantum state following an arbitrary continuous distribution, so long as the function is *efficiently computable*. Precisely, by efficiently computable, we mean that given an input $x$ represented with $n$ bits of precision, there exists a *classical* (and thus a quantum algorithm) returning $f(x)$ with $d$ bits of precision with complexity scaling as $\text{poly}(n, d)$. Equivalently, an efficiently computable function is any function with the oracle implementation $O_f$ efficiently performing the mapping $O_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle$. See Definition 1.2.4. Without a loss of generality, both the input and output of $f$ are given in a binary encoding. Intuitively, if there is an efficient classical program (e.g. in Python) evaluating $f(x)$, then in principle, the function is efficiently computable. Indeed optimised, efficient quantum-arithmetic procedures are already available [97]. Classically, to evaluate the function $f$ with $N$ distinct inputs would require $O(N)$ queries to the function – in the quantum setting, only $O(1)$ queries are required. This exponential advantage is intuitively suggestive of the potential power of the applications utilizing this procedure.

Furthermore, the time complexity of our approach is ultimately determined by the circuit depth. Given in most practical cases the function values can be computed via arithmetic operations, we expect that the oracle implementation requires less than a quadratic circuit depth (in terms of $n$ and the number of digits $d$ used in the discretization of $f$) [97].

Given access to this oracle, we utilise the approach of Rebentrost *et al.* [82] to efficiently simulate quantum dynamics under the rank-1 matrix $A(s)$ whose entries are given by $[A(s)]_{kl} := f_s(x_k)f_s^*(x_l)$. Note that the function values $f_s :=$

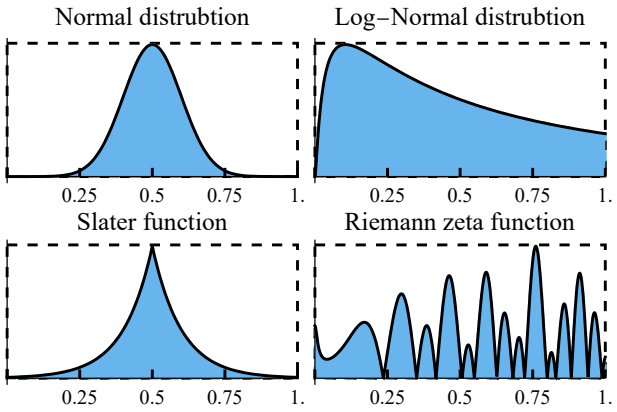


**Figure 3.2:** Applications of efficiently loading functions into quantum registers include probability distributions (normal and log-normal distributions shown) Slater-type functions in quantum chemistry [98] as well as the Riemann zeta function [99]. Our state-preparation error bounds depend on the filling ratio $\mathcal{F}$ as the absolute area under a function (light blue area) relative to the area of its bounding box (area in the dashed rectangle). This figure and caption were produced by my co-author, and are included for completeness.

$(1-s)f_0 + sf_1$ can be computed by an oracle, but our definition also permits storing $f_0(x_j)$ and $f_1(x_j)$ in QRAM [27] and computing linear combinations thereof using only addition and multiplication. Depending on the QRAM model, this could potentially make the cost of the oracle query $O(1)$, and thus the algorithm's circuit depth would be the same as its query complexity.

We proceed by encoding the matrix $A(s) \in \mathbb{C}^{N\times N}$ into a quadratically larger one-sparse matrix $S_A \in \mathbb{C}^{N^2\times N^2}$ – whereby a one-sparse matrix contains only a single non-zero entry in each row/column. This property allows us to utilise powerful one-sparse simulation techniques [83–86] thereby *exactly* simulating the time evolution $e^{-itS_A}$ which acts on the $n$-qubit main register as well as on an $n$-qubit ancillary register – and the simulation requires a further $d$-qubit, and two 1-qubit ancilla registers. After measuring and discarding the ancilla register we implement the mapping

$$U''(\Delta t, s)|\psi_0\rangle = e^{-i\Delta t\mathcal{H}(s)}|\psi_0\rangle + |\mathcal{E}\rangle$$

under the effective rank-1 Hamiltonian $\mathcal{H}(s) := A(s)/N$ up to a bounded error term $\|\mathcal{E}\| \in O(\Delta t^2)$. By setting the time step sufficiently small $\Delta t \ll 1$ we can simulate the evolution $U'(\Delta t, s) = e^{-i\Delta t\mathcal{H}(s)}$ up to an arbitrarily small error. This allows us to implement the piecewise constant adiabatic evolution for overall

time $T$ in $r$ iterations (timesteps) as per Eq. (3.2). This has the added benefit of also decreasing the adiabatic discretization error as we increase $r$. In the Appendix, we prove that applying this unitary evolution to the initial state $|+\rangle^{\otimes n}$ indeed maps onto our desired state $|\psi\rangle$ with a bounded error that only depends on properties of the encoded function.

### 3.2.2 Limitations

Our approach incurs two types of algorithmic error. First, we approximate an idealised continuous adiabatic evolution through a finite series of $r$ time-independent evolutions. Second, our Hamiltonian simulation also incurs algorithmic error, as previously mentioned. However, both kind of errors can be suppressed arbitrarily by increasing the number of iterations $r$ and thus decreasing the time each time-independent unitary is evolved for (i.e. by increasing the resolution in the temporal discretization).

While our approach is very general and is also applicable to non-continuous general functions (mappings), its main practical limitation is that its complexity depends on the filling ratio $\mathcal{F}$ of the function. In practice, we expect this filling ratio to be a *modest constant*. See Table 3.1 for filling ratios of common probability distributions and other functions. Nevertheless, we can easily construct artificial, pathological instances where $\mathcal{F}$ can be arbitrarily small.

For an example of a pathological instance that helps build intuition as to the limitation of $\mathcal{F}$, consider the unstructured search problem where we are given a function $F^{(n)} : \{0,1\}^n \mapsto \{0,1\}$ (where only one input $x = m$ corresponds to a non-zero output). Preparing the function $F^{(n)}$ in an $n$-qubit quantum register would reveal the non-zero value at $m$ upon measuring the state and thus would solve the unstructured search problem with high probability [13, 100]. While this is not a discretized (continuous) function, it is directly analogous to a box function of width $\approx 2^{-n}$ depending on the qubit count $n$. This results in a generalised filling ratio $\tilde{\mathcal{F}} \approx 2^{-n/2}$ that decreases exponentially as we detail in Appendix B.3.2 and thus our approach would require $\mathcal{O}(2^n)$ calls to the oracle $f(x)$ and is thus less

| Normal distribution | | | Log-normal distribution | | |
|---|---|---|---|---|---|
| $\mu$ | $\sigma$ | filling ratio $\mathcal{F}$ | $\mu$ | $\sigma$ | filling ratio $\mathcal{F}$ |
| 1/2 | 0.1 | $2.5\times10^{-1}$ | 0 | 0.5 | $5.5\times10^{-1}$ |
| 1/2 | 0.05 | $1.3\times10^{-1}$ | 0 | 1.0 | $7.6\times10^{-1}$ |
| 1/2 | 0.01 | $2.5\times10^{-2}$ | 0 | 1.5 | $6.1\times10^{-3}$ |
| **Slater function** | | | **Riemann zeta function** | | |
| – | $\alpha$ | filling ratio $\mathcal{F}$ | – | $\alpha$ | filling ratio $\mathcal{F}$ |
| | 5 | $3.7\times10^{-1}$ | | 10 | $6.1\times10^{-1}$ |
| | 10 | $2.0\times10^{-1}$ | | 20 | $5.0\times10^{-1}$ |
| | 20 | $1.0\times10^{-1}$ | | 100 | $3.1\times10^{-1}$ |

**Table 3.1:** Filling ratios (as illustrated in Fig. 3.2) for common probability distributions as a function of their parameters. All data was computed for a domain $0 \leq x \leq 1$ and the Slater function is $e^{-\alpha|x-0.5|}$. The filling ratio of the Riemann zeta function along the critical line $\zeta(1/2 + i\alpha x)$ is nearly independent from the scaling parameter $\alpha$ given its heavily oscillatory nature. This figure (and caption) were produced by my co-author and are included for completeness.

efficient than a Grover search. Note that such a pathological problem instance was only constructed because the function is dependent on the number of qubits it is being prepared on, resulting in the generalised filling ratio $\tilde{\mathcal{F}}$ not being an asymptotic constant in the limit of large $n$.

Finally, the impact of the filling ratio may in practice be circumvented. In one scenario, suppose the function being prepared has the majority of its density in an interval $[a', b'] \subset [a, b]$ that is relatively small compared to the domain. Then, we can simply prepare the function in $[a', b']$ (at some smaller qubit count) with a proportionally larger filling ratio. We can then add further qubits in the $|0\rangle$ state and shift the resulting function via quantum addition. If the function being prepared is non-zero on an exponentially small portion of the overall state space, and the location of the non-zero states are known, then this technique can yield an exponential improvement in the filling ratio. As a result, functions that are very narrow (i.e. very peaked), but have all of their density within a single small known region, including periodic narrow peaks (e.g. functions approaching a Dirac comb), can have their effective filling ratios increased significantly. We also note that $d$-sparse simulation techniques may also be useful for such very peaked functions if we can specify an oracle that efficiently computes the locations of non-zero function values [101].

Another technique one can employ to mitigate the constant cost of the filling ratio is to utilize a non-uniform grid. In such a grid, more samples of the function are taken in regions where the function is particularly narrow, thus giving a larger effective filling ratio. We leave such non-uniform grids as a topic for future exploration.

## 3.3 Numerical Demonstration

My co-author ran the numerics outlined in this section and produced Fig. 3.3. I have included this section for completeness, but have cut the corresponding appendix entry. Please see the paper [2] for more details.

We now demonstrate our approach in the preparation of two single-variate continuous functions which are practically relevant. First, we consider the probability density function of a log-normal distribution in Fig. 3.3 (a, above) which is relevant in modelling a wide range of natural phenomena as well as financial systems [68, 69, 102]. Second, we consider a Slater-type wave-function as a linear combination of primitive functions $\beta e^{-\alpha|x-x_0|}$ with constants $\alpha, \beta, x_0 \in \mathbb{R}$. As noted by my co-author, Gaussian and Fourier approximations to atomic orbitals are primarily used in quantum chemistry applications but despite their paramount practical utility they cannot accurately capture the crucial cusps at positions of atomic nuclei [98]. Indeed our approach allows the efficient preparation of *exact* Slater-type orbitals thereby increasing the accuracy of grid-based simulations [73, 98].

In Fig. 3.3 (a) we initially prepare a uniform superposition $|+\rangle^{\otimes n}$ in a register of $n$ qubits and then adiabatically morph this into our desired function. In the figure, the color gradient represents the smooth interpolation of $f_s$ throughout the adiabatic trajectory, with the lightest shade representing $f_0$ and the black line representing $f_1$. Taking a slice in the figure where only a certain color is kept (corresponding to a time $s$), would yield a plot of the function $f_s$.

In Fig. 3.3 (b) we explicitly simulate the dynamics under $e^{-iT/rS_A}$ that acts on overall $2n$ qubits thereby taking into account all sources of error (including error in binary encoding of the function values $f(x)$). Here, each line corresponds to a series of simulations evaluated with a fixed time $T$ and a fixed number of

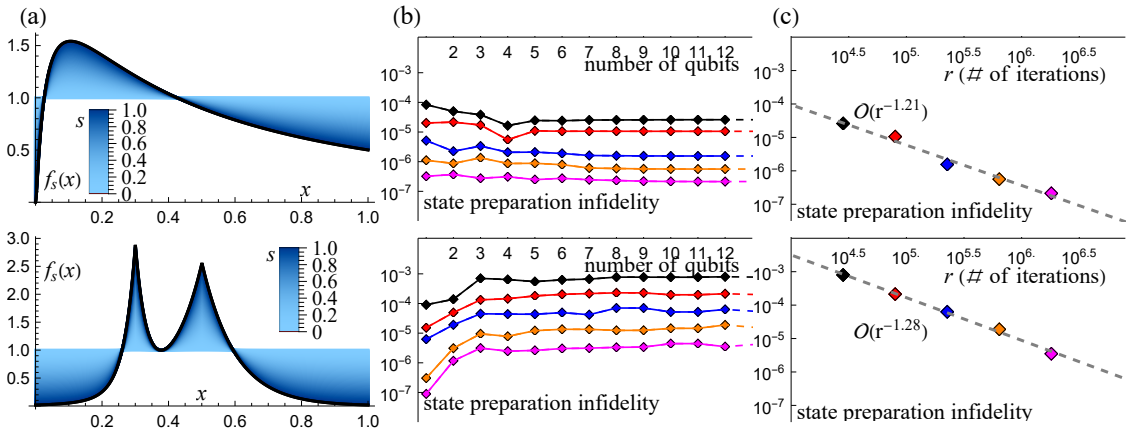


**Figure 3.3:** (a) Preparing a log-normal distribution (above) as relevant in financial systems [68, 69, 102] and (below) a Slater-type function used in quantum chemistry. We efficiently simulate $e^{-i\Delta t\mathcal{H}(s)}$ where $\mathcal{H}(s)$ encodes the continuous function $f_s$ as its ground state and we adiabatically evolve $f_s$ ($s$ is colour coded) from the constant function $f_0 = 1$ to the desired function $f_1$ (black solid line). (b) State preparation error $1 - \Phi$ as the infidelity $\Phi = |\langle\psi|\tilde{\psi}\rangle|^2$ at increasing numbers $n$ of qubits explicitly taking into account all sources of error by simulating dynamics of the $2n$-qubit *exact* 1-sparse simulation. Increasing $n$ (horizontal direction) the infidelity approaches a constant in exponential order – the maximum of the discretised function determines the error, which depends on the qubit count, hence the small/large errors at $N < 5$. Increasing the number of iterations $r$ (varying colours in vertical direction) allows us to better approximate the ideal adiabatic evolution and decreases the error. (c) The theoretically leading error $\mathcal{O}(1/\sqrt{r})$ of adiabatic evolution is dominated in the practically relevant region by the error of the low-rank simulation technique $\mathcal{O}(1/r)$ due to a larger absolute factor as confirmed by the empirical fits (dashed grey line). This figure and caption were produced by my coauthor, and are included for completeness.

iterations $r$ as we increase the number of qubits $n$. Our simulations nicely confirm our bounds in the following two distinct asymptotic limits.

First, as we increase the number of qubits $n$ the state-preparation error (solid lines) rapidly, in exponential order, approaches a constant value as expected from our asymptotic bounds – the asymptotic constant error (dashed lines are extrapolations) is then determined by the filling ratio $\mathcal{F}$ and by $T$ and $r$. Second, the series of lines from black to magenta in Fig. 3.3 (b) represent simulations with an increasing $r$ and confirm that as we increase the number of iterations $r$ the error decreases polynomially (as both the precision of the simulation step increases, and the discretization error of the adiabatic evolution decreases).

It is important to note that our error bounds in Theorem 3.2.1 are very pessimistic

as we have used an unrealistic, worst-case scenario argument in our proofs. In particular, while our bounds scale as $O(\sqrt{1/r})$, in practically relevant cases the error is expected to be dominated by the algorithmic error of the low-rank simulation approach – as it depends polynomially on the filling ratio. A simple argument shows that neglecting the adiabatic evolution error allows us to obtain the tighter error bound $\mathcal{O}(1/r)$. This is nicely confirmed in Fig. 3.3 (c) whereby we plot the asymptotic constant errors from Fig. 3.3 (b) as a function of the number of iterations $r$ and obtain an empirical scaling from approximately $\mathcal{O}(r^{-1.19})$ to $\mathcal{O}(r^{-1.3})$.

## 3.4 Further Applications

The techniques we have developed in this paper allow for a number of promising further applications. Some of these applications are related to the state preparation procedure, and some are consequences of the techniques we have developed. Of course, the adiabatic state preparation procedure that we present immediately enables applications that rely on continuous input states, such as those we have outlined in Section 3.1.

However, the state preparation procedure can actually be an application in its own-right. For instance, an important problem is to draw samples following a particular given continuous probability density function. Classically, techniques to do so, such as inverse transform sampling, generally require integrating the probability density function in question, and thus have complexity that scales with the resolution of the grid discretization [103]. Our state-preparation approach immediately enables such sampling, simply by preparing the square-root of the desired density function and then measuring the resulting state. Our approach has overall logarithmic complexity in doing so, and moreover produces *truly random samples* from the density function, rather than just pseudo-random samples.

Furthermore, given that our approach allows us to prepare a quantum state whose amplitudes are proportional to the samples $f(x_j)$, it follows that the measurement outcome of the bitstring $j \in \{0,1\}^n$ has a probability proportional to $|f(x_j)|^2$. As a result, we can directly use our approach to efficiently draw importance samples from

efficiently computable functions, and if the function has one dominant maximum then sampling can even find this global maximum with a high probability.

An alternative proposal for using our techniques for global optimization follows. Given a function $g : \mathbb{R} \mapsto \mathbb{C}$, preparing $f(x) = g(x)^k$ for some constant $k > 1$ allows us to find the *global maximum* of $g(x)$ with probability that improves exponentially in $k$ through direct sampling. However, the filling ratio of $f(x)$ also decreases exponentially in $k$, and so the complexity of preparing the state negates any benefit of using $k > 1$. On the other hand, if the algorithm's asymptotic dependence of $\mathcal{F}$ can be improved, then such an approach could be a promising global optimization procedure.

## 3.5 Discussion and Conclusion

The most important contribution of this paper is the asymptotic observation that for continuous functions the generalized filling ratio $\tilde{\mathcal{F}}$ converges to the filling ratio $\mathcal{F}$ – a constant independent of $n$. As a consequence, we prove that both our proposed state preparation technique, and some of the existing techniques in the literature, can be used to efficiently prepare quantum states corresponding *arbitrary* discretized continuous functions.

We also introduced a novel family of techniques that allow the efficient loading of any efficiently computable function onto a quantum register such that amplitudes of the state vector $|\psi\rangle$ are proportional to samples of the function. The primary novel algorithm is an adiabatic one utilizing rank$-1$ Hamiltonian simulation techniques. This is the first such adiabatic algorithm, but I note that (at least in its current form) it does not have better asymptotic performance than Grover black-box based approaches [87, 89]. It is conceivable that since our approach does not require amplitude amplification (this is implicitly handled by the adiabatic trajectory), that our approach could offer an improvement in constants over black-box based techniques (and we suggest this for subsequent exploration).

Moreover, our the state preparation technique is not just limited to continuous functions, and also applies to certain discontinuous functions. Application of our

state-preparation approach include fields ranging from the simulation of quantum physics and chemistry [73] to financial forecasting [69].

A central idea for the main algorithm presented this paper is that we can embed the desired quantum state $|\psi\rangle$ into a rank-1 Hamiltonian $\mathcal{H} \propto |\psi\rangle\langle\psi|$ whose ground state is exactly $|\psi\rangle$, and with all other orthogonal states having an eigenvalue of 0. We then utilized powerful low-rank [82] and 1-sparse [83–86] Hamiltonian simulation techniques from the literature, allowing us to efficiently simulate $\mathcal{H}$. Among other variants, we presented an adiabatic-evolution approach that enables quasi-deterministic preparation of the ground state of the rank-1 matrix $\mathcal{H}$. Central to the argument of this procedure's efficiency is that the spectral gap of such a rank-1 matrix is the same as its spectral norm, leading to a rare case where an adiabatic algorithm has a provably bounded spectral gap (which is essential to proving the efficiency of the adiabatic procedure). We rigorously proved that the query complexity of our adiabatic approach is a constant independent of the number of qubits. That is, we need only make a constant number of calls to the oracle computing the function amplitudes, regardless of the number of qubits the state is being prepared on. Moreover, we also prove that the state-preparation error and failure probability can be suppressed arbitrarily by increasing the temporal resolution (i.e. the number of steps in the adiabatic evolution $r$). Furthermore, we stress that the non-deterministic component of the algorithm is essentially an analytical convenience.

I would additionally like to emphasize that the presented ideas are ripe for future modification, optimization, and generalization. In particular, I suggest replacing the current rank-1 simulation procedure with QSVT based approaches, giving an exponential improvement in the error-dependence for the simulation component. This would change the dominant error source to adiabatic error, the optimization of which would require further investigation.

Our work almost immediately allows for the preparation of continuous multivariate functions, albeit with the main limitation that the filling ratio may

decrease exponentially in the dimension of the function being prepared, as we discuss in Appendix B.4.

The presented algorithms (and related ones in the literature) assume execution on fault-tolerant quantum computers. However, the quantum devices of the near-future are error prone [104] and there is limited work on preparing quantum states for NISQ devices. One possible approach utilizes quantum generative adversarial networks (qGAN) to tune the parameters in variational circuits to load a given distribution [105]. Some work on using qGANs to produce multivariate quantum states has also been conducted [106, 107]. Alternatively, in my prior work (not included in this thesis), we demonstrate how normal distributions may be efficiently produced in a manner resistant to most hardware errors [108]. The techniques presented in that work can potentially be directly applied to the work presented in this paper to discard state preparation attempts in which hardware errors likely occurred. Furthermore, combination with powerful error mitigation strategies may also be possible, should the states produced be used to compute the expectation values of observables [109]. However, to be useful with the algorithms presented in this work the infidelity of (uncorrected) hardware operations would likely need to be orders of magnitude lower than current state-of-the-art approaches (e.g. refs [110, 111]).

# 4

# Non-Linear Transformation of Quantum Amplitudes

This chapter (and the associated Appendix C) presents the results of the following publication, which I presented as a talk at Quantum Information Processing (QIP) 2024:

[3] Arthur G Rattew and Patrick Rebentrost. "Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications". In: *arXiv preprint arXiv:2309.09839* (2023)

Some text for which I was the original author is taken verbatim from the paper. This work was done in close collaboration with Patrick Rebentrost, who acted in a supervisory capacity. Unless stated otherwise, with input from Patrick, I developed all the proofs, derivations, and results. I identified the technique responsible for our exponential speedup in non-linear amplitude transformation, which is the foundation for nearly all the results in this chapter. For complete acknowledgments and funding disclaimers, please see the relevant sections of [3].

## Contents


**4.1 Introduction . . . . . 56**
4.1.1 Prior Work . . . . . 57
4.1.2 Overview of Our Work . . . . . 59

4.1.3 Preliminaries . . . . . 60
**4.2 New Algorithmic Subroutines . . . . . 61**
4.2.1 Diagonal Block-Encoding of State Amplitudes . . . . . 62
4.2.2 Importance-Weighted Amplitude Transformation . . . . 63
**4.3 Applications . . . . . 67**
4.3.1 Exponential Improvement in Applying tanh Activation Function . . . . . 67
4.3.2 End-to-End Efficiency in Applying Common Functions . 68
4.3.3 Quantum Maximum Finding in the Unitary Input Model 69
4.3.4 Generalizing Quantum State Preparation . . . . . 71
**4.4 Conclusion . . . . . 72**

## 4.1 Introduction

In Chapter 2, I outlined the limitations of QRAM, in particular when large amounts of classical data are required. In Chapter 3 I explored special cases where inputs to quantum algorithms could be prepared without the need for a large quantum memory. Having considered inputs, the next crucial ingredient towards enabling the coherent quantum implementation of neural networks are the activation functions. However, the applications of non-linear transformations are not just limited to machine learning applications.

The macroscopic world often behaves in non-linear ways, and non-linear functions appear in many problems in fields such as engineering [112, 113], optimization [114], machine learning [115, 116], and finance [41, 117]. Further illustrating their pervasiveness, the Kolmogorov-Arnold theorem [118] implies that all multivariate functions can be approximated by the summation and composition of univariate non-linear functions.

To reiterate, quantum algorithms find solutions to computational problems by manipulating the amplitudes of quantum states through unitary, and thus linear, transformations. Therefore, it is a fundamental and important question how to make quantum algorithms exhibit non-linear behaviour. Quantum signal processing (QSP) [45, 119] and quantum singular value transformation (QSVT) [26] are powerful frameworks encompassing nearly all of gate-based fault-tolerant quantum

computing [46]. They provide a machinery to develop quantum subroutines in a modular fashion, based on the block-encodings of arbitrary (not necessarily unitary) matrices in the subspaces of higher dimensional unitary operators. The power of QSP and QSVT originates from their ability to enact non-linear operations on these block-encoded matrices, such as transforming their eigenvalues and singular values by polynomial functions.

With an eye on applications for machine learning and optimization, it is interesting to consider how to apply non-linear transformations to the amplitudes of a given quantum state. This naturally leads one to wonder: what are the best ways of performing non-linear amplitude transformations (NLATs), and what are the relevant applications and their performance guarantees? To formalize this question, I now define an input model, and motivate the NLAT problem statement. I outline some of the prior work, and then present the main contributions of this work, comparing them to the prior literature. We assume we are given a state preparation unitary specified by a quantum circuit, and we wish to enact non-linear transformations on the state amplitudes, likely as a subroutine in another algorithm.

**Definition 4.1.1** (Non-Linear Amplitude Transformation (NLAT), Informal)**.** *Let $U$ be an $n$-qubit state preparation unitary specifying the $N = 2^n$ dimensional normalized quantum state $|\psi\rangle = U|0\rangle = \sum_{j=0}^{N-1} \psi_j |j\rangle$, $|\psi\rangle \in \mathbb{C}^N$. Let $f$ be a function $f : \mathbb{C} \mapsto \mathbb{C}$, and let $\epsilon \geq 0$. The non-linear amplitude transformation problem is to prepare a state $\epsilon$-close (in some measure of distance) to the state, $\frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} f(\psi_j)|j\rangle$ where $\mathcal{N}^2 := \sum_{j=0}^{N-1} |f(\psi_j)|^2$.*

### 4.1.1 Prior Work

Some prior work exists in the context of implementing NLAT. Mitarai *et al.* [120] demonstrated how to convert the amplitudes of a quantum state into a digitized register coupled to the state, enabling the amplitudes to be then be modified via coherent arithmetic, analogously to how the inverse is computed in the original quantum matrix inversion proposal [15]. Using a subspace error

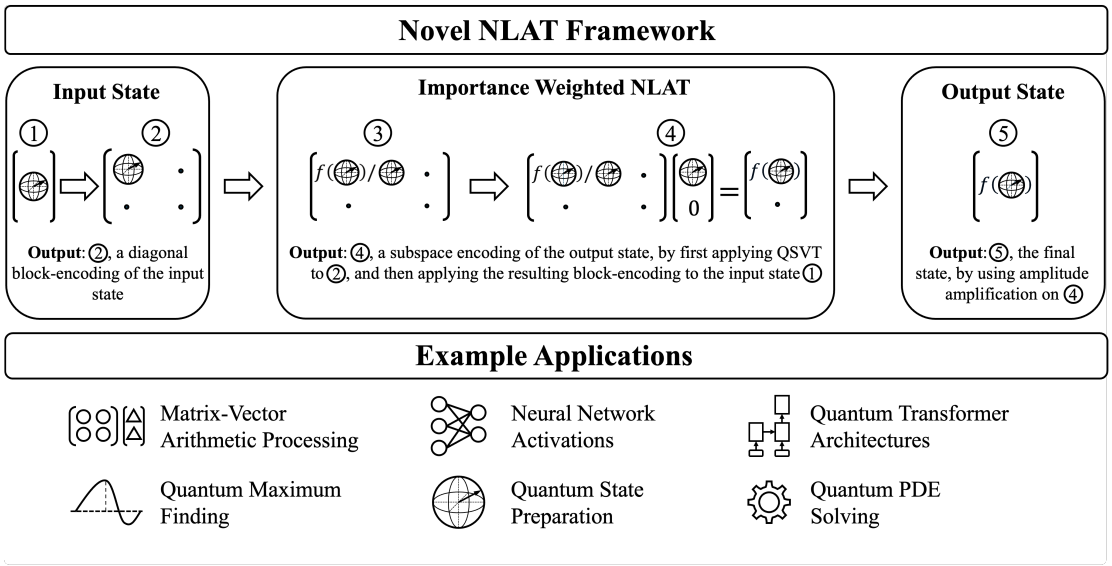


**Figure 4.1:** Illustration of our novel algorithm subroutines and some of their applications. In the top half of the figure, we show our new diagonal block-encodings (see Definition 1.2.6) of the amplitudes of a quantum state specified by a quantum circuit. We then use the block-encoding to give an algorithm for NLAT through importance-weighting. Applications of our techniques are enumerated in the second half of the figure. Our NLAT procedures require the function $f$ to be approximable by a degree-$k$ polynomial, and the overall complexity is $\Omega(k)$. For full technical details, see Section 4.2.2.

definition, and excluding other complexity parameters, their query complexity to the state-preparation unitary is $O(1/\epsilon)$, analogous to phase estimation.

Their clever construction led to the development of an algorithm which obtains an exponential improvement in the error-dependence for applying polynomial functions [121]. The construction of [121] leverages QSVT to avoid the error dependence associated with quantum-phase-estimation based approaches. Their results also utilize a subspace error definition. However, their technique has complexity exponential in the number of qubits for a broad class of important functions, including tanh and GELU, preventing its use as a subroutine in many applications (including for accelerating deep learning).

There has also been work on non-linear transformations in the variational circuit model [122]. Finally, I additionally note recent work on non-linear amplitude transformations built around the notion of a "weighted-state" [123].

### 4.1.2 Overview of Our Work

While QSVT and QSP allow for non-linear transformations on the eigenvalues of sub-blocks of a unitary operator, any resulting operator will obviously still be a linear operator. A question thus remains: how can a linear operator effect a non-linear transformation on a vector? As it turns out, the answer is to make the definition of the operator depend on the vector it is being applied to. For a simple example, consider a vector $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 & ... & x_n \end{pmatrix}^T$. Then, defining the linear operator $A = \text{diag}(\boldsymbol{x})$, $A\boldsymbol{x}$ enacts an element-wise non-linear transformation, yielding $\begin{pmatrix} x_1^2 & x_2^2 & ... & x_n^2 \end{pmatrix}^T$. As a result, as highlighted in Fig. 4.1, a crucial first step in our NLAT framework is the creation of a diagonal block-encoding of the amplitudes of a given quantum state. I then process this block-encoding, and apply it to the input state in close analogy to the preceding simple example.

The framework we present generalizes previous work, and our importance-weighted algorithm also gives an exponential speedup over the prior work in applying a broad set of important functions.

#### New Algorithmic Subroutines

I now explicitly highlight the two new main techniques that I illustrate in Fig. 4.1, presenting them in detail in Section 4.2:

1. *A diagonal block-encoding of state-amplitudes.* Given a state preparation unitary, in Section 4.2.1 I give a new diagonal block encoding of those state amplitudes (see Definition 1.2.6 for a definition of block encodings). This builds on the work of [120, 121], where they construct a similar operator which gives a block encoding of the amplitudes in a quantum state, but importantly is not diagonal. Diagonalizing this operator greatly simplifies analysis, and facilitates our subsequent results.

2. *An importance-weighted NLAT algorithm.* In Section 4.2.2 I give a novel algorithm for NLAT based off of importance-weighting. This yields an exponential speedup over the prior work for a broad class of functions

(including tanh and GELU), and is the enabling component which allows for many of the applications in this paper.

**Applications**

In Section 4.3, I then use these novel subroutines to prove new results for a number of applications, which I now outline:

1. *Obtaining an exponential improvement in applying* tanh. In Section 4.3.1 I prove that our work obtains an exponential speedup over the prior work in applying the function tanh to arbitrary quantum states.

2. *End-to-end complexity analysis for applying common functions.* In Section 4.3.2 I prove the efficient end-to-end complexity in applying a number of common functions to quantum states. All of these results are novel.

3. *Quantum maximum finding.* In Section 4.3.3 I use our importance-weighted subroutine to give a new maximum finding algorithm in the unitary input model. Our importance-weighted subroutine is essential to the efficiency of this result, and is not possible with the prior NLAT work.

4. *Generalizing quantum state preparation.* In Section 4.3.4 I demonstrate how the preparation of quantum states with amplitudes following a discretized continuous function is a special case of our NLAT framework.

Our results, as presented in the pre-print version, have found further applications in differential equation solvers [124], numerical quadrature [125], and in accelerating the transformer deep-learning architecture [126] with quantum computers [127].

### 4.1.3 Preliminaries

I will now briefly provide an overview of the mathematical preliminaries and notational conventions used. For a comprehensive list, see the notation chapter. For some function $f$ and a normal operator $H = \sum_j \lambda_j |\lambda_j\rangle\langle\lambda_j|$, I use the notation $f(H) := \sum_j f(\lambda_j)|\lambda_j\rangle\langle\lambda_j|$ to represent the transformation of the eigenvalues of $H$

by $f$. I will extensively make use of the framework of quantum block-encodings; see Definition 1.2.6 for a formal definition.

## 4.2 New Algorithmic Subroutines

I now derive our non-linear amplitude transformation subroutine, as illustrated in Fig. 4.1. Our framework accepts an $n$ qubit state preparation unitary $U$ which specifies the $n$ qubit state $|\psi\rangle$ through the mapping $|\psi\rangle := U|0\rangle$. I then construct a diagonal block-encoding of the amplitudes of this state, as shown in the box labelled "Input State" of Fig. 4.1. This builds on and simplifies prior work. I formally present our diagonal block-encoding in Section 4.2.1. Techniques from quantum signal processing/ quantum singular value transformation are then used to apply particular polynomial functions to this block-encoding, which is then applied to some initial state. When $f(0) = 0$, we can implement the transformation of the state amplitudes via importance-weighting, obtaining exponential improvements over the prior work (see Section 4.3.1). In essence, given a function $f(x)$, I instead work with functions $h(x)$ such that $xh(x) = f(x)$, as is illustrated in the box labelled "Importance Weighted NLAT" in the figure. As a consequence, I necessitate a number of technical results, ultimately yielding the substantial circuit and query complexity improvements over uniform-weighting. Finally, the output state is produced by applying amplitude amplification on the desired subspace, as shown in the box labelled "Output State".

It is worth noting that by modifying the function applied to the diagonal block encoding (in box "Importance Weighted NLAT" of Fig. 4.1) and the state the transformed block-encoding is applied to, our framework immediately gives a simplified version of the uniform-weighting procedure of [121] in our NLAT framework. In particular, this is done by directly applying $f(x)$ to the diagonal block-encoding, and applying the result to a uniform superposition. This allows for novel end-to-end complexity analysis (see Appendix C.9 for details).

This framework automatically gives rigorous error-bounds and complexity statements so long as the user can prove: (1) the asymptotic complexity of a

uniform approximation of the function of interest, (2) an upper-bound on the Lipschitz constant of the approximating polynomial (or a bound on the maximum of the function in the case of uniform-weighting), and (3) a lower-bound on the normalization factor of the transformed state.

### 4.2.1 Diagonal Block-Encoding of State Amplitudes

An essential intermediate result in our work, which may of be independent interest, yields a diagonal block-encoding of the amplitudes in a given quantum state.

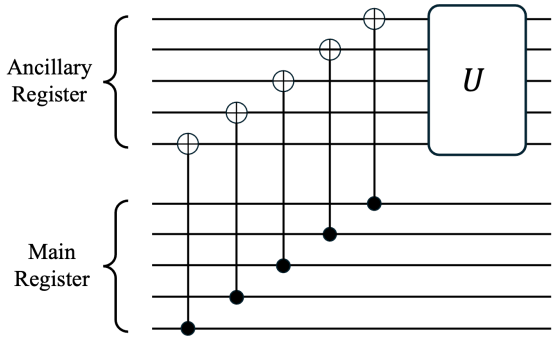


**Figure 4.2:** 5 qubit example of the circuit for the diagonal block-encoding of state amplitudes. Let $U$ be a state-preparation unitary yielding the state $U|0\rangle_n = \sum_{j=0}^{N-1} \psi_j |j\rangle_n$. The circuit consists of an $n$-qubit ancillary register, and an $n$-qubit main register. First, a chain of $n$ controlled-$X$ gates is applied with the control on the $i^{th}$ qubit of the main register targeting the $i^{th}$ qubit of the ancillary register. Then, $U$ is applied to the ancillary register.

**Theorem 4.2.1** (Diagonal block encoding of amplitudes)**.** *Given an $n$-qubit quantum state specified by a state-preparation-unitary $U$, such that $|\psi\rangle_n = U|0\rangle_n = \sum_{j=0}^{N-1} \psi_j |j\rangle_n$ (with $\psi_j \in \mathbb{C}$), then the cirucit shown in Fig. 4.2 yields a $(1, n, 0)$-block-encoding $U_A$ of the diagonal matrix $A = diag(\psi_0, ..., \psi_{N-1})$ with a sequence of $n$ controlled-$X$ gates and a single query to the $U$ circuit.*

*Proof.* This result builds on the techniques developed in [120, 121], offering a substantial compression in both circuit complexity and derivation simplicity. It is also a substantial improvement over the result presented in our original paper [3]. The matrix form of the chain of controlled-$X$ gates is given by $C := \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} |j \oplus$

$k\rangle\langle j| \otimes |k\rangle\langle k|$. Then, the whole block-encoding circuit is given by $U_A := C(U \otimes I_n)$. Noting that $j \oplus k = 0 \implies j = k$,

$$(\langle 0|_n \otimes I_n)C(U \otimes I_n)(|0\rangle_n \otimes I_n) \tag{4.1}$$

$$= \sum_{k=0}^{N-1} (\langle k|_n \otimes |k\rangle\langle k|)(|\psi\rangle_n \otimes I_n) = \sum_{k=0}^{N-1} \psi_k |k\rangle\langle k|. \tag{4.2}$$

Consequently, $U_A$ is a $(1, n, 0)$-block-encoding for $\mathrm{diag}(\psi_0, \ldots, \psi_{N-1})$. □

If one wishes to get a diagonal block-encoding only of either the real or imaginary parts of the state amplitudes, the result given in Lemma C.2.4 can be used instead.

Interestingly, since this diagonal block-encoding requires only a *single* use of the state-preparation unitary (not even a controlled-$U$), it can be repeatedly invoked on itself without an exponential increase in circuit complexity (unlike with Lemma C.2.4, or the prior work [120, 121]).

### 4.2.2 Importance-Weighted Amplitude Transformation

I now present our main result on importance-weighted amplitude transformation, as illustrated in Fig. 4.1. Note that "efficiently-approximable functions" are formally defined in Definition 1.2.3.

**Theorem 4.2.2** (Non-linear transformation of quantum amplitudes via importance-weighting)**.** *We are given an $n$-qubit circuit $U$ such that $U|0\rangle = \sum_{j=1}^{N} \psi_j |j\rangle$, with $N = 2^n$ and $\forall j, \psi_j \in \mathbb{R}$. Additionally, we are given a function $f(x)$ such that $f(0) = 0$ and $\gamma := \max_{x\in[-1,1]} |f(x)|$ and $\mathcal{N}^2 := \sum_{j=1}^{N} |f(\psi_j)|^2$. For $1 > \epsilon > 0$, let $f(x)$ be $\frac{\epsilon\mathcal{N}^2}{\gamma N}$-approximable by a polynomial $P_k(x)$, such that $P_k(0) = 0$, with degree $k_\epsilon := K(\epsilon\mathcal{N}^2/\gamma N)$, where $K$ is a function describing the degree of the polynomial required to get a uniform-approximation to $f$ with error at most $\frac{\epsilon\mathcal{N}^2}{\gamma N}$. Additionally, define $\tilde{\gamma} := \max_{x\in[-1,1]} |P_k(x)/x|$. Then, we can prepare a state $\epsilon$ close in $\ell_2$ norm-distance from $\frac{1}{\mathcal{N}} \sum_j f(\psi_j)|j\rangle$ with arbitrarily high probability of success with $O(k_\epsilon \tilde{\gamma}/\mathcal{N})$ query complexity to a controlled $U$ and $U^\dagger$ circuit, with overall circuit depth of $O(nk_\epsilon\tilde{\gamma}/\mathcal{N})$, with polylogarithmic classical computation, and with $O(n)$ ancillary qubits.*

I also prove a simpler result in Theorem C.3.1, where I assume the function being implemented is exactly a degree $k$ polynomial (i.e., $f(x) = \sum_{j=0}^{k} \alpha_j x^j$ for some set of coefficients $\{\alpha_j\}_j$). This simpler result is then used to prove the general result of Theorem 4.2.2, thereby handling most details arising from approximation errors. The full proof of Theorem 4.2.2 is provided in Appendix C.4.

I will now give an explanation of each of the three primary terms the complexity of Theorem 4.2.2, $k_\epsilon$, $\tilde{\gamma}$ and $\mathcal{N}$. I will then provide intuition for which functions can be implemented efficiently. Finally, I will provide two examples of functions for which this procedure is not generally efficient.

First, $\epsilon$ is the user-input specifying the desired $\ell_2$-norm error in the final transformed state, and so $k_\epsilon$ gives the degree of the polynomial required to satisfy this error bound. If the degree$-k_\epsilon$ polynomial approximation $P_{k_\epsilon}(x)$ satisfies the uniform-approximation error bound of $\max_{x\in[-1,1]} |P_{k_\epsilon}(x) - f(x)| \leq \frac{\epsilon\mathcal{N}^2}{\gamma N}$, then the $\ell_2$-norm error bound holds. As a simple example, consider the function $\sin(x)$. Later, I will show that we can get a polynomial approximation $P_k(x)$ to $\sin(x)$ satisfying $\max_{x\in[-1,1]} |\sin(x) - P_k(x)| \leq \epsilon_0$ with degree $k_\epsilon \in O(\log(1/\epsilon_0))$. Moreover, I also show that $\frac{\gamma N}{\epsilon\mathcal{N}^2} \in O(N/\epsilon)$. Since theorem requires a $\frac{\epsilon\mathcal{N}^2}{\gamma N}$-uniform-approximation to get an overall $\ell_2$-norm error of $\epsilon$, we can then set the degree of the polynomial to $k_\epsilon \in O(\log(\frac{\gamma N}{\epsilon\mathcal{N}^2}))$, and thus $k_\epsilon \in O(\log(N/\epsilon))$.

Second, intuitively, $\tilde{\gamma}$ is usually fairly similar to $\max_{x\in[-1,1]} |f(x)/x|$, which for a function satisfying $f(0) = 0$, is upper-bounded by the Lipschitz constant. However, because of pathological edge-cases this is not universally true, and so one must bound $\tilde{\gamma}$ for the specific polynomial selected. As a pathological case, consider a function with small Lipschitz constant, approximated by a rapidly oscillating function that stays within the uniform approximation error. Then, $\max_{x\in[-1,1]} |f(x)/x|$ will be small, but $\tilde{\gamma}$ could potentially be quite large. However, in practice, I have never encountered this problem for any of the polynomial approximations I have used.

Third, and often of the most impact to the complexity, $\mathcal{N}$ must lower-bounded. Intuitively, $\mathcal{N}$ is the norm of the transformed state. Therefore, if the function shrinks the state-norm a lot, the complexity will be poor. As a pathological example,

consider the function $f(x) = 0$, clearly, $\sum_j |f(\psi_j)|^2 = 0$, and so the procedure will have infinite complexity. To avoid norm-shrinking, two important properties of generally efficient functions are: (1) for $x \neq 0$, $f(x) \neq 0$, and (2) either the degree-0 or the degree-1 terms of the polynomial approximation are non-zero. It is possible that a function not satisfying these two properties can be efficient for specific input states, but as a general rule if one of these properties does not hold an input to the function can usually be constructed for which the complexity will be arbitrarily poor.

**Comparison to Classical Element-Wise Function Application** A quantum NLAT procedure is a subroutine for use in other algorithms (it is not an end-to-end algorithm in and of itself). Nevertheless, to help understand when a quantum algorithm might use an NLAT procedure to obtain a speedup, it can be helpful to consider the classical complexity of applying a function element-wise to a given vector. I will consider three possible approaches to perform this task. First, a direct and exact application. Second, a randomized application. Finally, applying a function to a tensor-network representation of a given vector.

First, for an arbitrary efficiently computable function (see Definition 1.2.4) the complexity of applying it to an $N$-dimensional vector is trivially $O(N)$ (neglecting the cost to evaluate $f$). No other properties need be assumed for the function (e.g., does not have to admit an efficient polynomial approximation). In contrast, for a function satisfying the properties outlined in Theorem 4.2.2 (namely, an efficient polynomial uniform approximation on a fixed interval, satisfying $f(0) = 0$, Lipschitz bounds on the approximating polynomial, and lower-bounds on "norm-shrinkage") our importance-weighted NLAT procedure has $O(\text{polylog}(N))$ complexity (only considering the dimension), at the cost of a dependence on the norm of the output state, and a linear dependence on the degree of the approximating polynomial. For many common functions I subsequently show that the output norm can be shown to be a constant. At a high-level, for vectors with high dimension, and "nice" polynomial approximation properties, implementing NLAT is generally much more efficient for high-dimensional vectors than a direct classical implementation when

only coherent downstream access is needed (rather than e.g., reading out all the resulting amplitudes, where the classical approach is exact).

Second, consider a speculative randomized classical implementation. Given sample and query access to a given vector $\boldsymbol{x}$, that is $SQ(\boldsymbol{x})$ (see [52] or Section 2.3.2), the dequantized equivalent of NLAT would be to obtain $SQ(f(\boldsymbol{x}))$ for some given function $f$ (assuming $f(0) = 0$, and $f$ has a bounded Lipschitz constant, similarly to Theorem 4.2.2). To the best of my knowledge, such a procedure has not yet been provided in the literature. Nevertheless, one possible approach could be to use rejection sampling. The key governing the complexity would be the acceptance probability, i.e., the probability that given a sample from $SQ(\boldsymbol{x})$ we accept it as a sample from $SQ(f(\boldsymbol{x}))$. Using the Lipschitz constant of $f$, and the importance-weighted assumption of $f(0) = 0$, I anticipate that this can then be bounded using analysis similar to that performed in Theorem 4.2.2. This is an interesting direction for future exploration, and would be worth considering in more detail, and may give quite an efficient procedure for obtaining $SQ(f(\boldsymbol{x}))$. This is beyond the scope of the present work, as the goal is to give an efficient quantum NLAT procedure.

Finally, consider a given vector $\boldsymbol{x}$ encoded as a tensor network with a bond dimension $r$. A polynomial function can be applied element-wise to $\boldsymbol{x}$ by taking repeated Hadamard products (to implement powers) and linear combinations. In general, summing vectors only increases the bond dimension linearly in the number of terms being summed, but a tensor network representing $\boldsymbol{x} \odot \boldsymbol{x}$ will usually have bond dimension $O(r^2)$. Consequently, the bond dimension (and thus complexity) of the tensor network will generally increase exponentially in the degree of the polynomial function being implemented. Thus, such approaches should either explore alternative methods of implementing element-wise function applications, or should consider approximating the result (e.g., by truncating the bond dimension).

## 4.3 Applications

### 4.3.1 Exponential Improvement in Applying tanh Activation Function

The central goal of this thesis is to show that quantum computers can be used to accelerate the underlying linear algebra of existing classical deep learning algorithms (as has been done for certain classical machine learning algorithms, e.g., in [26]). A natural way to do this is through the block-encoding framework, as it provides a straight-forward way for implementing sequences of matrix-vector operations. However, a missing ingredient to accelerate deep-learning has been the ability to coherently interleave activation functions after each matrix-vector multiplication. Clearly, in this model, quantum states implement the vectors, and the block-encoded matrices implement the network layers (e.g., a weight matrix for a feed-forward network). Consequently, efficiently applying a non-linear transformation to a quantum state enables quantum algorithms to utilize activation functions.

To clearly demonstrate the exponential improvement offered by our importance-weighting technique over uniform sampling, I give the end-to-end complexity statement for both algorithms applied to the important activation function $f(x) = \tanh(x)$.

**Theorem 4.3.1** (Exponential speedup in importance-weighted application of tanh activation function)**.** *Given an arbitrary n-qubit quantum state preparation unitary $U$ such that $U|0\rangle = |\psi\rangle = \sum_{j=1}^{N} \psi_j |j\rangle$ (with $\forall j, \psi_j \in \mathbb{R}$, and $N = 2^n$), and the function $f(x) = \tanh(x)$, the algorithm of [121], as proven in Theorem C.9.1, prepares a state $\epsilon$ close in $\ell_2$-norm distance (for small $\epsilon$) to $\frac{1}{\mathcal{N}} \sum_j f(\psi_j)|j\rangle$ (where $\mathcal{N}$ is the normalization factor) with arbitrarily high probability of success with worst-case $O(\sqrt{N}\log(N/\epsilon))$ query complexity to a controlled $U$ and $U^\dagger$ circuit, with overall circuit depth of $O(n\sqrt{N}\log(N/\epsilon))$, and $O(n)$ ancillary qubits. Our importance-weighted procedure prepares a state $\epsilon$ close in $\ell_2$-norm distance from $\frac{1}{\mathcal{N}} \sum_j f(\psi_j)|j\rangle$ with arbitrarily high probability of success with worst-case $O(\log(N/\epsilon))$*

*query complexity, with overall circuit depth of* $O(n\log(N/\epsilon))$*, and* $O(n)$ *ancillary qubits.*

This result immediately follows by applying Theorem 4.2.2 with a particular polynomial approximation to $\tanh(x)$, and I relegate the technical details of the proof to Appendix C.6.

### 4.3.2 End-to-End Efficiency in Applying Common Functions

I will now use the theorems and lemmas presented so far to easily derive the end-to-end complexity for applying a number of common functions to arbitrary quantum states. When $f(0) = 0$, I invoke Theorem 4.2.2. When $f(0) \neq 0$, I invoke Theorem C.9.2 (our diagonal variant of [121], extended with $\ell_2$-norm error guarantees, as derived in the appendix). These examples illustrate how when combined with our new error bound, our technique and the technique of [121] enable the transformation of real amplitudes by a rich set of functions *with overall efficient end-to-end complexity.* I include the full proof of this result in Appendix C.5.

**Theorem 4.3.2** (End-to-end complexity for several functions)**.** *We are given an arbitrary n-qubit quantum state preparation unitary* $U$ *such that* $U|0\rangle = |\psi\rangle = \sum_{j=1}^{N}\psi_j|j\rangle$ *with* $\forall j, \psi_j \in \mathbb{R}$ *and* $N = 2^n$*. Let* $\epsilon > 0$*, for the functions* $f(x)$ *given below there exists circuits to prepare a state* $|\phi\rangle$ *such that* $\||\phi\rangle - \frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle\|_2 \leq \epsilon$*, where* $\mathcal{N}$ *is the normalization factor. The circuits all require* $O(n)$ *ancillas, and their complexities are:*

| *Case* | *f(x)* | *Query complexity* |
|---|---|---|
| *i.)* | $e^x$ | $O\left(\log\left(1/\epsilon\right)\right)$ |
| *ii.)* | $\cos(x)$ | $O\left(\log\left(1/\epsilon\right)\right)$ |
| *iii.)* | $\frac{1}{1+e^{-2x}}$ | $O\left(\log\left(1/\epsilon\right)\right)$ |
| *iv.)* | $\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(x/\sigma)^2}$ | $O\left(\sigma\log\left(\sigma^2/\epsilon\right)\right)$ |
| *v.)* | $\sin(x)$ | $O\left(\log\left(N/\epsilon\right)\right)$ |

*The query complexity is to controlled* $U$ *and* $U^\dagger$ *circuits. For the table and the results, assume that inputs to the functions are* $x \in [-1, 1]$*. For Case iv.), we assume that* $\sigma^2 \geq 1/2$*. In all cases, the additional circuit depth is simply the query complexity multiplied by n, e.g., the depth of Case i.) is simply* $O(n\log(1/\epsilon))$*.*

### 4.3.3 Quantum Maximum Finding in the Unitary Input Model

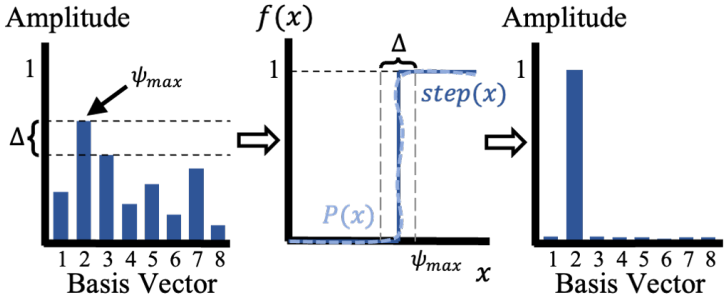


**Figure 4.3:** Illustration of maximum finding via non-linear amplitude transformation. Here, the polynomial approximation $P(x)$ to the shifted step function step$(x)$ is applied to each amplitude, yielding nearly all of the amplitude in the output in the basis vector with maximum amplitude.

I now derive a technique for maximum finding in the state preparation unitary input model. Variants of this technique will likely be of significant interest to domains ranging from machine learning (for instance, in using the maximum output of a multi-layer neural network for identifying class labels), to financial derivative pricing [68], to real-space molecular chemistry simulations [128]. This algorithm follows immediately by applying our importance-weighted amplitude transformation procedure. Given a state preparation unitary $U$, (as per Definition C.2.1) preparing the $\ell_2$-normalized quantum state $U|0\rangle = \sum_{j=0}^{N-1} \psi_j |j\rangle$, the objective is to identify the basis vector associated with the maximum amplitude, with constant probability of success. This problem format has some overlap with the state tomography results of [129, 130], but the examination of this overlap remains as future work.

I consider a simplified problem statement, assuming that the maximum amplitude ($\psi_{max}$) and the gap between the largest and second largest amplitudes ($\Delta$) are known *a priori*, and we wish to identify the basis vector associated with the maximum amplitude. Moreover, for simplicity, I also assume all the amplitudes are real and non-negative.

It is worth briefly noting that sampling the basis vector with maximum amplitude is an easier problem than sampling *and* concluding that the sampled basis vector

is the maximum. The former occurs with probability $1/|\psi_{\max}|^2$, while the latter will necessarily have a dependence on $\Delta$.

In essence, as illustrated in Fig. 4.3, this problem can be solved by applying a shifted step-function (heavyside function) to the amplitudes of the quantum state, such that all the amplitudes smaller than $\psi_{max} - \Delta/2$ are mapped to 0, and the amplitudes greater than $\psi_{max} - \Delta/2$ are mapped to 1. As a result, measuring this output state would immediately yield a sample from one of the basis states with maximum amplitude.

However, as invoking Theorem 4.2.2 requires obtaining an upper-bound on $\hat{\gamma}$ (i.e., upper-bounding the maximum value on the interval of consideration of the polynomial approximation divided by $x$), and there is no immediately obvious bound for the polynomial I use, I will take a slightly modified approach. Let $f(x)$ represent the heavyside function being utilized. Instead, I actually consider the desired function $xf(x)$. The analysis easily follows, and this can intuitively be thought of as obtaining a filter in the block-encoding where each entry in the block-encoding is 0, apart from the entry corresponding to the maximum which has value 1. Applying this filter to the input state then solves the problem, analogously to the simpler intuition I just described.

I now present our result solving this problem, leaving the technical details of the proof to Appendix C.7.

**Theorem 4.3.3** (Maximum finding in state preparation input model)**.** *Let $U$ be an $n$-qubit state preparation unitary $U|0\rangle = \sum_{j=1}^{N} \psi_j |x_j\rangle$ such that $\forall j : \psi_j \in \mathbb{R}_{\geq 0}$, and $\{x_j\}_j$ is some ordering on the set of standard basis vectors, where the subscript makes the mapping to the corresponding amplitude label explicit. Without loss of generality, assume that $\psi_1 > \psi_2 \geq ... \geq \psi_N$, and define the amplitude gap $\Delta := \psi_1 - \psi_2$. We are given the values $\psi_1$ and $\Delta$. Then, with arbitrarily high probability, and $\tilde{O}(\frac{1}{\psi_1 \Delta})$ query complexity to a controlled $U$ and $U^\dagger$ circuit, we can determine the index of the basis state with maximum probability amplitude (i.e., we learn which basis state the label $x_1$ corresponds to).*

### 4.3.4 Generalizing Quantum State Preparation

In this section, I demonstrate that previous ideas on quantum state preparation can be easily captured in our framework of non-linear amplitude transformations, matching previous asymptotic complexities. Given some function $f$ (e.g. an efficiently computable function as described in Definition 1.2.4), and a set of $N$ points $\{x_j\}_j$ at which $f$ is to be evaluated, the goal is to prepare the $n$ qubit quantum state (with $N = 2^n$) $|f\rangle_n := \frac{1}{\mathcal{N}} \sum_{j=1}^{N} f(x_j)|j\rangle_n$ where $\mathcal{N}^2 = \sum_{j=1}^{N} |f(x_j)|^2$. In this section, I assume that the function is scaled so that its maximum over its domain is at most 1.

There is a wide literature on this topic, which I will briefly summarize here, presenting more details in Chapter 3 and in Appendix C.8. To reiterate, the literature begins with Grover's work on black-box state synthesis [87], in which a state with amplitudes following an arbitrary function can be prepared with complexity $O(\sqrt{N}/\mathcal{N})$. In the case of a discretized continuous function, and a uniformly spaced grid of $N$ points on the interval $[x_1, x_N]$, [2] (i.e., Chapter 3) I proved that this converges to the inverse of an asymptotic constant we called the "filling ratio", $\mathcal{F}$. The rate of convergence was shown to be exponential in the number of qubits in the general case of Riemann integrable functions. The filling ratio was shown to only depend on properties of the function, and intuitively can be visualized by a bound measuring the percentage of a bounding box filled by the function. A number of results have improved the practical efficiency of black-box techniques (see e.g., [88, 131]). However, many of these techniques have substantial constant costs from their dependence on quantum arithmetic.

As a result, of most relevance to this chapter are the ideas in [132, 133], where quantum arithmetic free state preparation ideas are presented. These two works were initially developed independently, but interestingly, when viewed through the lens of NLAT with of our diagonal block-encoding, they can be seen as different expressions of the same underlying idea. In essence, both approaches can be viewed as implicitly creating a diagonal block-encoding of an initial state with linear amplitudes (i.e., with amplitudes $\propto f(x) = x$), applying the desired function to this block-encoding, and then applying the result to a uniform superposition. In this sense, starting with

an initial state with linear amplitudes, uniform-weighted NLAT directly reproduces both of these state-preparation algorithms. While the two algorithms differ in some technical details, they can both be easily understood through in this framework, with rigorous end-to-end complexity analysis using our results in Theorem C.9.2.

To maintain our end-to-end error bounds while avoiding quantum arithmetic, in Theorem 4.3.4 I re-derive the approach of [132] in our framework, avoiding the need to prepare a quantum state with amplitudes proportional to the linear function without using quantum arithmetic or approximate methods. I note that while our framework provides a conceptually simple way to re-derive their results, our implementation requires slightly more ancillary qubits than theirs does.

I now present the results of [132], with the proof given in Appendix C.8.

**Theorem 4.3.4** (State preparation through non-linear amplitude transformation [132, 133])**.** *Let $\epsilon > 0$. Let $f : [a, b] \mapsto \mathbb{R}$ be a Lipschitz function such that $\max_{x\in[-1,1]} |f(x)| \leq 1$. Let $\{x_j\}_j$ be a uniform grid such that $x_0 = a$ and $x_{N-1} = b$. Define $|f(\psi)\rangle_n := \frac{1}{\mathcal{N}} \sum_j f(x_j)|j\rangle_n$, where $\mathcal{N} := \sum_j |f(x_j)|^2$. Then, using Theorem C.9.2, a quantum state $|\phi\rangle_n$ satisfying $\||\phi\rangle_n - |f(\psi)\rangle_n\|_2 \leq \epsilon$ can be prepared with overall complexity $\tilde{O}(\mathcal{F}^{-1})$, where $\mathcal{F}^{-1}$ is the function-dependent asymptotic constant called the "filling ratio" as per Chapter 3.*

## 4.4 Conclusion

I have presented a new framework for applying non-linear functions to the amplitudes of quantum states. I give a diagonal block-encoding of the amplitudes of an input state specified by a state preparation unitary, simplifying analysis over previous approaches. I then give a new algorithm based off of importance weighting, which gives up to an exponential speedup in important machine learning applications. I then provide a sketch, which is the foundation of Chapter 5, as to how this technique can be used to accelerate deep-learning with quantum computers. Our importance-weighted algorithm also enables a new maximum finding application and result. Furthermore, we demonstrate how this framework can be used to

efficiently apply a wide range of functions to the amplitudes of quantum states. I conclude by re-deriving state-preparation algorithms in our framework, showing its general versatility and power.

For future work, one direction is to generalize our results to work with quantum states with complex amplitudes, potentially using the recent Ref. [134]. Additionally, for an $n$-qubit quantum state, it would be worth exploring ways to implement the diagonal block-encoding of state amplitudes with $O(1)$ ancilla qubits instead of the $O(n)$ currently required. Another direction is to investigate the quantum state exponentiation setting [71, 135] as opposed to the state preparation circuit input model. Moreover, application to problems related to non-linear ordinary or partial differential equations are interesting to consider and may be enabled by our subroutine. Furthermore, these techniques naturally lend themselves to further developing the theory of quantum machine learning, both in the parameterized quantum circuit model [136], and in the fault-tolerant linear algebra model [26] (which is the focus of Chapter 5. In this setting, it would be interesting to explore the complexity in applying a sequence of non-linear transformations, as even for efficiently implementable functions, it appears that the overall circuit depth would grow exponentially in the number of non-linear transformations (as the circuit implementing the preceding transformed state-preparation unitary becomes the state preparation unitary for the next transformation, and so on). There might be specific settings, e.g. $\ell_1$ or $\ell_2$ norm preserving transformations, where this is not the case. It may also be possible that interesting results can be obtained by making distributional assumptions. It would also be worthwhile expanding this work to allow the implementation of multivariate activation functions (such as softmax). Beyond the computational model considered in this work, certain quantum systems may offer non-linear functions at the hardware level, in contrast to the systematic polynomial approximation approach.

# 5

# Quantum Processors as Accelerators for Machine Learning

This chapter (and the associated Appendix D) presents results from the following paper, which I presented as a talk at the conference Quantum Techniques in Machine Learning (QTML) 2025, and was accepted at the International Conference on Learning Representations (ICLR) 2026:

[4] Arthur G. Rattew et al. *Accelerating Inference for Multilayer Neural Networks with Quantum Computers*. 2025. arXiv: 2510.07195 [quant-ph]. URL: https://arxiv.org/abs/2510.07195

Some text is taken verbatim from the sections of the publication for which I was the primary author. I developed all the theory and results presented in this chapter, with some supervision from Patrick Rebentrost. The specific architectures selected were designed in collaboration with Po-Wei Huang. For complete acknowledgments and funding disclaimers, please see the relevant sections of [4].

## Contents


**5.1 Introduction . . . . . 76**
**5.2 Quantum Matrix-Vector Arithmetic . . . . . 79**
5.2.1 New Operations on Vector Encodings . . . . . 80
5.2.2 Matrix Vector Squared Product . . . . . 82

5.2.3 QRAM-Free Quantum Encoding of 2D Multi-Filter Convolutions . . . . . 83
**5.3 Architectural Blocks . . . . . 83**
**5.4 Architectures . . . . . 86**
5.4.1 Key Results Under Differing Quantum Data Access Assumptions . . . . . 87
**5.5 Conclusion . . . . . 89**
**5.6 Future Work . . . . . 90**

## 5.1 Introduction

In this chapter, I put together the ingredients developed in Chapters 2 to 4, to achieve the primary goal of this thesis: using quantum computers to accelerate inference for classical multilayer neural networks.

Ushered in by advanced in classical computing, namely with GPUs, the last decade has seen widespread adoption of deep learning to solve a range of important problems [16–20]. Yet, the progress in classical hardware is not expected to continue indefinitely as we approach the physical limits of Moore's law [22]. This motivates an important question: if quantum mechanical effects are the obstruction preventing further scaling of classical hardware, can those same effects be exploited to provide a new kind of acceleration? Can quantum computers be the next stage in hardware acceleration?

Quantum machine learning research (see e.g., [137–139]) can broadly be divided into two categories: (1) architectures designed to run purely on quantum computers with no classical counterparts (often based on parameterized quantum circuits), and (2) quantum techniques to accelerate the underlying linear algebra of existing machine learning algorithms.

The first line of research began with the seminal variational quantum eigensolver (VQE) [140]. Following this line of investigation, a number of quantum machine learning algorithms have been proposed [141–146]. However, it was subsequently found that many of such algorithms face substantial challenges with their training landscapes, due to vanishing gradients and barren plateaus [147–150]. In cases

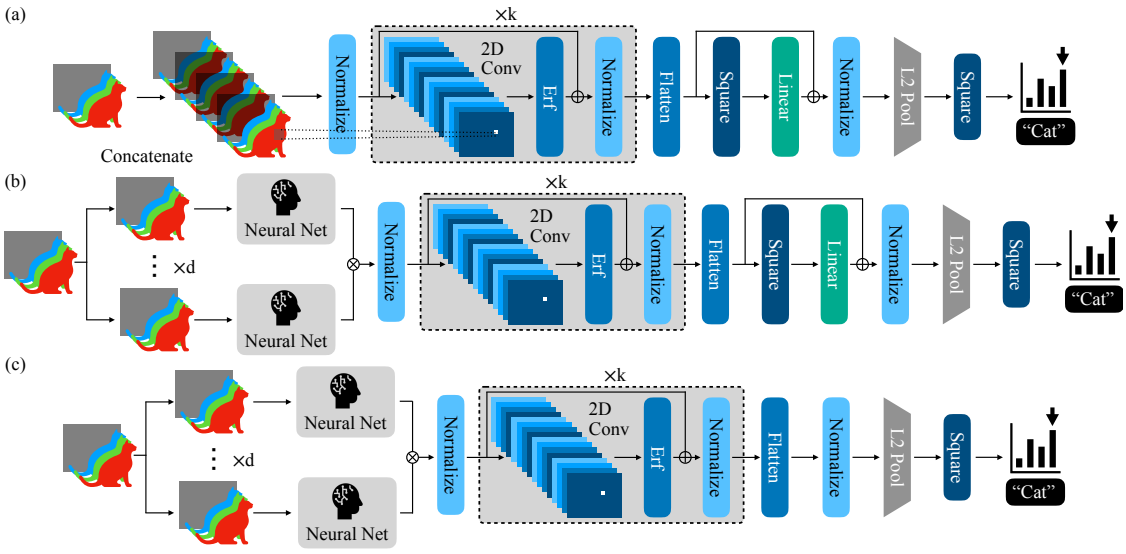


**Figure 5.1: Architecture for Convolutional Neural Networks.** This figure shows the architectures we consider with provable quantum complexity guarantees for inference under three regimes of quantum data access assumptions. (a) Depicts the architecture where both the inputs and network weights are provided in an efficient quantum data structure. (b) Only the network weights are provided in an efficient quantum data structure. (c) No input assumptions are made. In all architectures, the input is assumed to be a rank-3 tensor (e.g., images with 4 channels). Note that this figure and caption was produced by a co-author, and are included for completeness.

where these issues are not present, it appears that many of these algorithms can be instead simulated classically efficiently, removing quantum speedups [151–154].

The second line of research utilizes quantum linear algebra primitives (e.g., [26, 155, 156]) to accelerate classical machine learning models. For example, see refs [71, 82, 157–168]. Some work has also focused on accelerating classical neural network training and inference [42, 169, 170]; a comparison to the relevant prior work subsequently in this chapter.

**Main Contributions.** In this paper, I propose a method that can be used to accelerate inference for multilayer residual networks (ResNets) [18], and derive a number of key subroutines to be able to do so. The main contributions are listed as follows.

- In Section 5.2, I further develop the modular vector-encoding framework for quantum matrix-vector arithmetic that I first presented in Chapter Chapter 4.

- In Section 5.2.2, I derive a novel quantum algorithm for the multiplication of arbitrary full-rank and dense matrices with the element-wise square of a given vector, *without* incurring a rank-dependence.

- In Section 5.2.3, I provide a novel QRAM-free block-encoding for 2D multi-filter convolutions.

- In Section 5.4, to the best of my knowledge, I derive the *first coherent quantum implementations of multi-layer neural networks with non-linear activations.* I provide rigorous end-to-end complexity proofs for inference under three QRAM regimes:

  - **Regime 1 (inputs and weights provided via QRAM):** Assuming QRAM access to both inputs and weights, for a network with $k$ non-linear activations acting on $N$-dimensional inputs $\tilde{O}(\text{polylog}(N/\epsilon)^k)$ inference cost is shown. Moreover, I argue that existing techniques are insufficient to dequantize this result.

  - **Regime 2 (weights provided via QRAM):** When a cost linear in the dimension of the input must be paid (i.e., no QRAM for the input), but the network weights are stored in QRAM, I prove a quartic speedup over exact classical implementations for shallow architectures.

  - **Regime 3 (no QRAM):** In the absence of any QRAM, a quadratic speedup over an exact classical implementation is proven.

The relevant architectures in each regime can be seen in Fig. 5.1. I derive a number of techniques and algorithms which have broad utility in implementing machine learning architectures on quantum computers. However, my main focus is on accelerating inference for classification, with the formal problem statement given in Definition 5.1.1.

**Definition 5.1.1** (The Approximate Sampling-Based Classification Problem)**.** *Given a neural network represented by function $h : \mathbb{R}^D \mapsto \mathbb{R}^C$ (i.e. with $D$-dimensional inputs and $C$-dimensional outputs) which returns a probability distribution as its output (i.e., for any $\boldsymbol{x} \in \mathbb{R}^D$, $\boldsymbol{y} := h(\boldsymbol{x})$ is all non-negative, and $\|\boldsymbol{y}\|_1 = 1$), then the sampling-based classification problem is to return a sample from some probability vector $\hat{\boldsymbol{y}}$ such that $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2 \leq \epsilon$.*

**Comparison to Prior Work.** A substantial differentiating factor between this work and the prior literature is the coherent applications of multiple rounds of non-linearities. In prior work, [159–161] intermediate state tomography or partial measurements are used to enact the non-linear transformations, breaking coherence and limiting possible quantum performance (tomography necessarily gives a polynomial error-scaling and dimension-dependence). Moreover, in tomography based approaches where the activations are applied classically, the resulting vector then needs to be reloaded into the quantum computer, and even with an efficient QRAM implementation, this necessitates a linear dimension-dependence (each modified output element needs to be loaded into the QRAM). Additionally, I demonstrate that careful tracking on the bounds of the norm of a vector propagating through a network forward pass is essential for proving end-to-end complexities, and the developments made in the VE framework help enable this. To this end, I make the observation that residual skip connections that enable deep networks classically are fundamental to norm stability and preservation, enabling coherent multilayer architectures not present in prior work. In Table 5.1, a figure produced jointly with my co-authors, we provide a detailed comparison to some of the prior work.

## 5.2 Quantum Matrix-Vector Arithmetic

In this section, I further develop the tools necessary to perform quantum matrix-vector arithmetic. For an overview of block-encodings, please see Chapter 1. For an overview of vector-encodings (VEs), please see Chapter 1 and Chapter 4 (and in particular, Appendix C.10). **Novel contributions in this section:**

| | Architecture | Coherent Multi-Layer | Coherent Non-Linearity | QRAM-Free | Norm Preservation | Polylog $1/\epsilon$ | Polylog N |
|---|---|---|---|---|---|---|---|
| Ref [171]* | CNN Inspired PQC | ✗ | ✗ | ✓ | ✓ | N/A | N/A |
| Ref [159] | Feed-forward | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Ref [160] | CNN | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Ref [161] | Transformer | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Our work Regime 1 | Residual CNN | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| Our work Regime 2 | Bilinear Residual CNN | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ |
| Our work Regime 3 | Bilinear Residual CNN | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |

**Table 5.1: Comparison with prior work.** We briefly explain the meaning of each column. Coherent multi-layer refers to the construction of multi-layer architectures separated by non-linear activation functions without tomography. Coherent non-linearity refers to the implementation of non-linear transformations on the quantum computer without readout. Norm preservation refers to the preservation of vector norms throughout the network forward pass. Next, each quantum implementation of a classical architecture incurs some error over the exact classical implementation, and as such an entry ✓ in the polylog $1/\epsilon$ column indicates a $O(\text{polylog}(1/\epsilon))$ error-dependence, whilst a ✗ entry indicates a $O(\text{poly}(1/\epsilon))$ error-dependence. Finally, polylog $N$ refers to polylogarithmic complexity in the input dimension $N$. *__Note:__ the architecture presented in [171], is inspired by CNNs but is based on parameterized quantum circuits (PQC). As they do not aim to accelerate an existing classical architecture, it is not possible to provide an entry in the polylog $\epsilon$ column. Moreover, they do not provide complexities when considering classical input data, and so we do not give an entry in the column corresponding to polylog $N$. Note that this figure was produced jointly with my co-authors and is included for completeness.

In Section 5.2.1, I further develop the framework of vector-encodings, introducing straight-forward new quantum algorithms for vectors encoded as VEs, enabling vector sums, matrix-vector products, tensor products, and vector concatenations. In Section 5.2.2, I present a novel algorithm which applies an arbitrary full-rank and dense matrix to the element-wise square of a vector, without incurring a Frobenius norm dependence. Finally, in Section 5.2.3, I give a novel QRAM-free block-encoding for 2D multi-filter convolutions.

### 5.2.1 New Operations on Vector Encodings

To enable the subsequent results on architectural blocks, I will now develop primitive operations on vector-encodings. These results are straight-forward modifications

of existing techniques into the VE framework, but are necessary to allow easy tracking of the norm of encoded vectors, which is a crucial parameter dictating the complexity of quantum neural network accelerations.

**Lemma 5.2.1** (Vector Sum, Proof in Appendix D.2)**.** *Let $0 \leq \tau \leq 1$. We are given unitary circuits $U_\psi$ and $U_\phi$ which are $(\alpha, a, \epsilon_0)$ and $(\beta, b, \epsilon_1)$ VEs for $|\psi\rangle_n$ and $|\phi\rangle_n$, respectively. Define $c := \max(a, b)$, $|\Gamma\rangle_n := \frac{\tau}{\alpha}|\psi\rangle_n + \frac{(1-\tau)}{\beta}|\phi\rangle_n$, $\mathcal{N} := \||\Gamma\rangle_n\|_2$ and $|\overline{\Gamma}\rangle_n := |\Gamma\rangle_n/\mathcal{N}$. Then, using one controlled $U_\psi$ circuit, one controlled $U_\phi$ circuit, and two additional single-qubit gates, we can construct a unitary matrix $V$ such that $V$ is a $(\mathcal{N}^{-1}, c+1, (\frac{\epsilon_0}{\alpha} + \frac{\epsilon_1}{\beta})/\mathcal{N})$-VE for $|\overline{\Gamma}\rangle$.*

**Lemma 5.2.2** (Matrix-Vector Product, Proof in Appendix D.2)**.** *We are given an $(\alpha, a, \epsilon_0)$-block-encoding $U_A$ for the $n$-qubit operator $A$, and $U_\psi$ a $(\beta, b, \epsilon_1)$-VE for the $\ell_2$-normalized $n$-qubit quantum state $|\psi\rangle$. Let $\mathcal{N} := \|A|\psi\rangle_n\|_2$. $U_\psi$ has $T_\psi$ circuit complexity, and $U_A$ has $T_A$ circuit complexity. Then, we can obtain an $a + b + n$ qubit unitary $U$ with $O(T_\psi + T_A)$ circuit complexity such that $U$ is an $(\alpha\beta/\mathcal{N}, a + b, (\epsilon_0 + \alpha\epsilon_1)/\mathcal{N})$-VE for the quantum state state $A|\psi\rangle_n/\mathcal{N}$.*

**Lemma 5.2.3** (Tensor Product of Vector Encodings, Proof in Appendix D.2)**.** *Given $U_\psi$ an $(\alpha, a, \epsilon)$-VE for $|\psi\rangle_n$ with $O(T_\psi)$ circuit complexity, and $U_\phi$ an $(\beta, b, \delta)$-VE for $|\phi\rangle_m$ with $O(T_\phi)$ circuit complexity, then we can obtain the circuit $V$ which is an $(\alpha\beta, a + b, \epsilon + \delta + \epsilon\delta)$-VE for $|\psi\rangle_n \otimes |\phi\rangle_m$ with $O(\max(T_\psi, T_\phi) + \max(n, b))$ circuit depth.*

**Lemma 5.2.4** (Concatenation of Vector Encodings, Proof in Appendix D.2)**.** *Let $D = 2^d$, $N = 2^n$, and $0 \leq \epsilon < 1$. Assume that $d \leq n$. Suppose we are given a set of $D$ unitary circuits, $\{U_i\}_{i\in[d]}$ such that each $U_i$ is an $(\alpha_i, a, \epsilon)$-VE for the quantum state $|\psi_i\rangle_n$ with $O(T)$ circuit complexity.* [1] *Let $|\Psi\rangle_{d+n} = \sum_{j=0}^{D-1} |j\rangle_d|\psi_j\rangle/\alpha_j$, and let $\mathcal{N} := \||\Psi\rangle_{d+n}\|_2 = \sqrt{\sum_{j=0}^{D-1} \frac{1}{\alpha_j^2}}$. Then, we can obtain a $(D/\mathcal{N}, d + a, \epsilon)$ for $\frac{|\Psi\rangle_{d+n}}{\mathcal{N}}$ with $O(dDT)$ circuit complexity.*

[1] If $D$ is not a power of 2, padding can be added.

### 5.2.2 Matrix Vector Squared Product

I will now present the first key result of this section, showing how given a matrix $W$ (with $\|W\|_2 \leq 1$) and a vector encoding of $\boldsymbol{x}$, a vector encoding of $W(\boldsymbol{x})^2$ can be obtained. The key idea is to avoid obtaining a quantum block-encoding of the operator $W$ (which in general requires $W$ to be either low-rank, or sparse [26]). The product is then implemented by using importance-weighting to coherently combine the columns of $W$ weighted by the corresponding elements of the input vector, and then apply the result to a modified version of the input vector.

**Theorem 5.2.1** (Product of Arbitrary Matrix with a Vector Element-wise Squared, Informal)**.** *Let $N = 2^n$. We are given a matrix $W \in \mathbb{C}^{N\times N}$, provided via a pre-processed efficient quantum accessible data structure (see Definition D.2.2). Additionally, we are given the unitary $U_\psi$ with circuit complexity $O(T_\psi)$, a $(\alpha, a, \epsilon)$-VE for the quantum state $|\psi\rangle_n$. Define the function $g : \mathbb{C} \mapsto \mathbb{R}$ as $g(x) = |x|^2$, and $\mathcal{N} := \|Wg(|\psi\rangle_n)\|_2$. Then we can construct the unitary $U_f$ which is a $(\frac{\alpha^2}{\mathcal{N}}, 2a+2n+3, \frac{2\alpha\epsilon}{\mathcal{N}})$-VE for $Wg(|\psi\rangle_n)/\mathcal{N}$, and has $O(T_\psi + n^2)$ circuit depth.*[2]

This result is stated formally and proven as Theorem D.2.1 in the Appendix. To use this to prepare the quantum state $Wg(|\psi\rangle_n)/\mathcal{N}$, the vector normalization result (Lemma D.2.7) can be directly applied to the output VE yielded by Theorem 5.2.1, preparing the state with $\tilde{O}(\alpha^2(T_\psi + n^2)/\mathcal{N})$ circuit complexity. This is the first such result *without a Frobenius norm dependence* on $A$.

I will now informally sketch the proof of this procedure. First, define the columns of $W$ as $W = \begin{pmatrix}\boldsymbol{w}_0 & \dots & \boldsymbol{w}_{N-1}\end{pmatrix}$. Define the normalized version as $|w_j\rangle = \boldsymbol{w}_j / \|\boldsymbol{w}_j\|_2$, and define $a_j := \|\boldsymbol{w}_j\|_2$. We assume access to three objects. (1) A block-encoding of $A := \mathrm{diag}(a_0, \dots, a_{N_1})$. (2) An oracle implementing $U_W|0\rangle|j\rangle = |w_j\rangle|j\rangle$. (3) A vector-encoding for $|\psi\rangle = \sum_j \psi_j|j\rangle$. Then, by using our vector-encoding circuit, we can get an encoding of $|\phi\rangle := \sum_j \psi_j|j\rangle|w_j\rangle = \begin{pmatrix}\psi_0\langle w_0| & \dots & \psi_{N-1}\langle w_1|\end{pmatrix}^\dagger$. Then, using our block-encoding of $A$, we can efficiently get a block-encoding of

[2]For simplicity, here we are assuming that the parameter $d$ (as defined in Theorem D.2.1) is set to $n$.

$\begin{pmatrix} a_0\psi_0 I_n & \dots & a_{N_1}\psi_{N-1} I_n \\ & \mathbf{0} & \end{pmatrix}$ (where $I_n$ is a $2^n$ dimensional identity matrix, and only the first $N$ rows are non-zero). We can then use the product of matrix-encoding with vector encoding result to take the product of $\begin{pmatrix} a_0\psi_0 I_n & \dots & a_{N_1}\psi_{N-1} I_n \\ & \mathbf{0} & \end{pmatrix}$ with $|\phi\rangle$ yielding the desired vector-encoding.

### 5.2.3 QRAM-Free Quantum Encoding of 2D Multi-Filter Convolutions

While the matrix-form of a $2D$ convolution has been given many times before in the literature, to the best of our knowledge the following is the first result giving a block-encoding of a QRAM-free 2D multi-filter convolution. I also stress that the following result can be highly optimized, especially if QRAM is used. Such optimizations are left to future work. The full proof is provided in Appendix D.2.2.

**Lemma 5.2.5** (QRAM-Free Block-Encoding of 2D Convolution With Filters)**.** *Let $M = 2^m$, let $n = 2m$, let $N = 2^n$, and let $D = 2^d$. Define the matrix form of the 2D multi-filter convolution operation, $\mathcal{C} \in \mathbb{R}^{CM^2 \times CM^2}$, as per Lemma D.2.15. Here, $C$ represents the number of input and output channels, and $D$ represents the dimension of the kernel over rows and columns (i.e., the kernel is a rank$-4$ tensor containing $C$, $C \times D \times D$ filters). Then, after performing some one-time classical pre-computation, we can obtain a $(1, 3+8D+2\log(CD), 0)$- block-encoding of $\frac{\mathcal{C}}{2\|\mathcal{C}\|_2}$ with $O(m^2C^3D^4\log(C)\log(D))$ circuit depth.*

While the degrees on the number of channels and the filter size $D$ seem large, the filter size is usually quite small in practice (e.g., often 3). Moreover, there are straight-forward optimizations of this result which can substantially reduce the degrees on both $C$ and $D$.

## 5.3 Architectural Blocks

In this section I will derive two key architectural blocks, a residual block, and a multi-layer residual block, which allow the subsequent complexity claims. I present an additional architectural block building on these in Appendix D.3, but do not

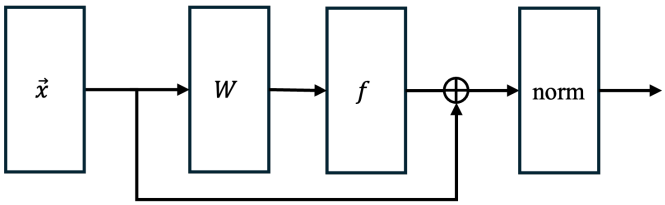


**Figure 5.2: Generic Residual Architectural Block.** This diagram illustrates the structure of a typical residual block used in deep neural networks. The input vector $\boldsymbol{x}$ is transformed through a sequence of operations: a learnable linear transformation $W$, a non-linear activation function $f$, and a residual (skip) connection that adds the original input to the transformed signal. The output is then passed through a normalization layer (`norm`).

include it in the main text as it is not essential for understanding the key complexity details of such quantum implementations.

**Lemma 5.3.1** (General Skip Norm Block)**.** *Let $\epsilon_1 \in (0, 1]$. Let $\kappa \in [1, 2]$. Consider the architecture shown in Fig. 5.2. Let $N = 2^n$. We are given the unitary $U_\psi$ a $(1, a, \epsilon_0)$-VE for $|\psi\rangle_n$ with circuit complexity $O(T_1)$, and are given the unitary $U_W$ a $(1, b, 0)$-block-encoding for the $n$-qubit operator $W/\kappa$ with circuit complexity $O(T_2)$ such that $\|W\|_2 \leq 1$. Define $f(x) := \mathrm{erf}(4x/5)$, $|\psi_f\rangle_n := |\psi\rangle_n + f(W|\psi\rangle_n)$, and $\mathcal{N} := \||\psi_f\rangle_n\|_2$. Then, we can obtain a $(1, 2(a + b) + n + 9,712(\epsilon_0 + \epsilon_1))$-VE for $|\psi_f\rangle_n/\mathcal{N}$ with circuit complexity $O(\log(\frac{\sqrt{N}}{\epsilon_1}) \log(\frac{1}{\epsilon_1})(a + b + n + T_1 + T_2))$.*[3]

I provide the rigorous proof of this result in Appendix D.3, but it essentially follows from using the preceding results on matrix-vector multiplication, vector sums, and the results on layer normalization and applications of the error-function (which are provided are provided in Appendix D.2, and are slight improvements on the results presented in Chapter 4). The key insight enabling this proof is that in a residual block such as the one described, the forward norm of the vector is efficiently lower-bounded prior to every normalization layer. Without such skip connection, and the techniques developed for working with vector-encodings (which enable effective tracking of the norm of a vector propagating through a network), the norm at the end of such a block could be arbitrarily small, leading to complexities

[3]We implicitly assume that $\|W|\psi\rangle_n\| > 0$, which is a reasonable assumption for any input which comes from the same distribution as the training data.

which could be on the order of $\approx N^k$ (or even unbounded) for $k$-layer architectures – completely intractable even for constant depth networks. As a consequence, I am able to prove the following result for multi-layer residual blocks.

**Lemma 5.3.2** (Sequence of $k$ Residual Blocks)**.** *Let $N = 2^n$. Suppose we are given a unitary $U_\psi$ with circuit complexity $O(T_1)$ such that it is a $(1, a, 0)$-VE for $|\psi\rangle_n$. Let $k$ be an asymptotic constant. Suppose we have a sequence of $k$ residual blocks (as per Lemma 5.3.1), with weight layers implemented by $k$ unitaries $\{U_{W_i}\}_i$ such that $U_{W_i}$ (with circuit complexity $O(T_2)$) is a $(1, b, 0)$-block-encoding for the $n$-qubit operator $W_i/2$, and for all $i$, $\|W_i\|_2 \leq 1$. Then, we can prepare a $(1, 2^k(a + 2b + n + 9), \epsilon)$-VE for the output of the $k$ residual blocks with $O(\log(\sqrt{N}/\epsilon)^{2k}(a + 2b + n + T_1 + T_2))$ circuit depth.*

I prove this result in Appendix D.3, and it follows by repeatedly invoking Lemma 5.3.1 with its output as the next input. It appears that the complexity of this result as a function of the number of layers $k$ is a fundamental limitation of any quantum algorithm. As described in greater detail in Appendix D.3, for a unitary matrix (a linear operator) to enact a non-linear transformation on a vector, its definition must in general be input-dependent. Consequently, unless Lemma 5.3.1 can be implemented with only a single copy of its input, it seems unlikely that this complexity can be avoided. This suggests that quantum computers are best suited for accelerating the wide and shallow regime, which is a popular regime for classical inference accelerators (since wide networks can be parallelized on classical hardware, but depth cannot be parallelized). Classically, with the aim of accelerating both inference and training, there are a range of techniques for compressing neural networks [172]. Moreover, classically, deep neural networks are much harder to accelerate than their shallow and wide counterparts (you can parallelize matrix-multiplications, but not consecutive layers). Consequently, there are a number of classical architectures striving for shallow networks (e.g. [173]) which can serve as sources of inspiration for designing architectures best suited for quantum acceleration. This is discussed in greater detail in Appendix D.3.

## 5.4 Architectures

I will now use the architectural blocks derived thus far to prove the quantum complexity in inference for the architectures shown in Fig. 5.1 (a), which is then used to prove the complexity of the architecture in panel (b), and a corollary is used to prove the complexity of the architecture in panel (c).

The results in all three data-access regimes all follow from the following general result.

**Theorem 5.4.1** (General Multilayer Convolutional Network with Skip Connections)**.** *Let $M = 2^m$, $N = 2^n = M^2$. Consider the neural network architecture shown in Fig. 5.1 (a). Let the input $X$ be a rank$-3$ tensor of dimension $4 \times M \times M$ (with an R, G, B and null channel, where the null channel has all $0$s). Assume that $\|vec(X)\|_2 = 1$, and that we have access to a unitary $U_X$ that is a $(1, 0, 0)$-VE for the input in column-major layout $|X\rangle_{2+2m} = \sum_{i=0}^{4} \sum_{j=0}^{M-1} \sum_{k=0}^{M-1} X_{i,k,j} |i\rangle_2 |j\rangle_m |k\rangle_m$. Assume that $U_X$ has $O(T_X)$ circuit complexity. As shown in the figure, we have a sequence of $k$ residual convolutional layers, where each convolutional layer has $16$ input channels, $16$ output channels (i.e., $16$ filters) with filter width and height $3$. I.e., each convolutional layer has $16 \times 16 \times 3 \times 3 = 2304$ parameters. Assume that there is $0$ padding so the input and outputs always have the same dimension, and that there is a stride of $1$. Suppose each convolutional layer has been regularized, so that its spectral norm is at most 1. Let $W$ represent the $N \times N$ full-rank linear layer applied in the final output block of the network, and assume that $\|W\|_2 \leq 1$. Let $C$ represent the number of output classes, and assume that $C = 2^c$ (padding can be added otherwise). Let the overall network be represented by the function $h : \mathbb{R}^{4 \times M \times M} \mapsto \mathbb{R}^C$. Let $\boldsymbol{y} = h(X)$ (and note that $\|\boldsymbol{y}\|_1 = 1$, and $\boldsymbol{y} \in \mathbb{R}^C$). Then, with $O(\log(\sqrt{N}/\epsilon)^{2k+1}(T_X + n^2))$ total circuit depth, and with $O(2^k n)$ ancillary qubits, we can draw a sample from an $\ell_1$-normalized $C$-dimensional vector $\tilde{\boldsymbol{y}}$ such that $\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|_2 \leq \epsilon$.*

*Proof.* We have a 4 channel input and we want to map this to a 16 channel input (by concatenating $|X\rangle_{2+2m}$ vector with itself 4 times). Let $|X\rangle_{4+2m} :=$

$\frac{1}{\sqrt{4}}\left(\langle X|_{2+2m} \;\; \langle X|_{2+2m} \;\; \langle X|_{2+2m} \;\; \langle X|_{2+2m}\right)^T$. We can invoke Lemma 5.2.4 with $U_X$ four times, obtaining a $(1,0,0)$-VE for $|X\rangle_{4+2m}$ with $O(T_X)$ circuit complexity. Using Lemma 5.2.5, for each of the $i = 0, ..., k-1$ convolutions, we can obtain a $(1, 27, 0)$-block-encoding for $\mathcal{C}_i/2\left\|\mathcal{C}_i\right\|_2$ (the matrix form of the corresponding convolution) with $O(m^2)$ circuit depth. Consequently, we can invoke Lemma 5.3.2 to obtain $U_{\text{conv}}$ a $(1, 2^k(63+n), \epsilon)$-VE for the $\ell_2$-normalized output of the sequence of $k$ residual blocks. Moreover, $U_{\text{conv}}$ has $O(\log(\sqrt{N}/\epsilon)^{2k}(n + T_X + m^2))$ circuit depth. Then, we can invoke Lemma D.3.2 with $U_{\text{conv}}$ to draw a sample from some probability vector $\tilde{\boldsymbol{y}} \in \mathbb{R}^C$ such that $\left\|\tilde{\boldsymbol{y}} - \boldsymbol{y}\right\|_2 \leq \epsilon$ with $O(\log(\sqrt{N}/\epsilon)^{2k+1}(T_X + n^2))$ total circuit depth and with $O(2^k n)$ ancilla qubits. □

### 5.4.1 Key Results Under Differing Quantum Data Access Assumptions

As discussed in Chapter 2, the feasibility of quantum random access memory is widely debated. Following Chapter 2 and Appendix D.4 (based on [1]), a QRAM device can be broadly split into two categories: *active* and *passive*. Given a QRAM device with $N$ memory qubits (or constant-sized registers), a QRAM is passive if it requires $o(N)$ *total cost* per query, and is active if it requires $\Omega(N)$ total cost. The distinction of total cost vs time per query (be it circuit depth or otherwise) is essential. A QRAM can be easily implemented as a fault-tolerant circuit with $O(\text{polylog}(N))$ depth, but such an implementation necessitates $\Omega(N)$ parallel gates. This represents an opportunity cost, and indeed in Chapter 2 when considering this opportunity cost, most asymptotic quantum speed-ups in linear algebra tasks disappear. Consequently, a useful QRAM device should be passive (or practically passive, as defined in Appendix D.4). Yet, this necessarily implies that such a passive QRAM device cannot be fully error-corrected, introducing substantial challenges in its realization. However, recent work [174] provides a promising path forward, offering hope for the limitations raised in [1]. Because of the possibility of a passive QRAM, but the formidable challenges in obtaining one, it is worthwhile to consider quantum acceleration under three regimes of QRAM feasibility.

**Regime 1: Input and Network Use QRAM**

This regime is realistic if passive QRAM can be realized, and if it can be efficiently updated (making it feasible to use QRAM to store online inputs). Consequently, in this regime I assume that both the input to the network and the network weights are available via QRAM.

The primary purpose of the architecture I presented in Regime 1 is to show that quantum computers can implement multi-layer neural networks based on real architectures coherently, with reasonable input assumptions, and with cost polylogarithmic in the dimension of the network. As per the main-text, in this regime I assume that the matrix weights (in particular for the final full-rank linear layer) and vectorized input are provided via QRAM. The architecture for this regime is shown in Fig. 5.1 (a). Let the dimension of the vectorized input be $O(N)$. Since the input is provided via QRAM, $T_X$ as defined in Theorem 5.4.1 is $T_X \in O(\text{polylog}(N))$ (see, Appendix D.4.2). **Thus, for a constant number of layers $k$, the cost to perform inference (in accordance with Definition 5.1.1) becomes** $O(\text{polylog}(\sqrt{N}/\epsilon)^k)$. Please see Appendix D.5.1 for a detailed discussion outlining important application areas where such input assumptions are practical (namely, where the input can be constructed in an amortized fashion online). Moreover, I also discuss considerations relating the receptive field of such architectures, and argue that existing techniques are insufficient to dequantize this result.

**Regime 2: Network Stored in QRAM, Input Loaded Without QRAM**

This QRAM regime is realistic if a QRAM can be constructed with pre-computation, but cannot be efficiently updated. In this case, the QRAM data structure for the weights of the architecture can be pre-constructed, but any online input cannot be.

The architecture in this regime is shown in Fig. 5.1 (b). The architecture contains $d$ paths of purely classical neural networks, which each operate on $O(N)$ dimensional (vectorized) inputs. These classical architectures are assumed to have $\tilde{O}(N)$ time complexity in terms of the input. These separate paths are then normalized, converted to quantum states, and then the Kronecker product of the result is taken.

The result is fed into exactly the same architecture as in Regime 1. This architecture is inspired by bilinear neural networks [175]. Consequently, to determine the cost of this architecture, I again invoke Theorem 5.4.1. Here, we need to pay an $\tilde{O}(N)$ cost to load each of the input paths in as a quantum state (via brute-force [77]),$T_X \in O(N)$. Consequently, I obtain an overall algorithmic complexity of $O(N \log(\frac{N^{d/2}}{\epsilon})^{2k})$, which for constant $k$ and $d$, simplifies to $\tilde{O}(N \log(1/\epsilon)^{2k})$. When $d = 2$, the dimension after the tensor product is $N^2$. Consequently, the final linear layer contains a matrix multiplication of an $N^2 \times N^2$ matrix with an $N^2$ dimensional vector, which takes $\Omega(N^4)$ time. **Consequently, for a constant $k$, this architecture produces a quartic speedup for the inference problem defined in Definition 5.1.1 over exact classical computation**. When $d = 1$, the speedup due to the final layer is instead quadratic. This speedup can be increased by setting $d$ to larger values.

**Regime 3: No QRAM**

In this regime, I assume that passive QRAM is impossible, and thus do not utilize QRAM.

This architecture is identical to the one presented in Regime 2, only dropping the final full-rank linear block. In Appendix D.5.3 I show that the architecture in Fig. 5.1 (c) can perform inference with a total $O(N \log(1/\epsilon)^{2k})$ circuit complexity. Since the dimension of the vector acted on by the 2D convolution is $O(N^2)$ (when d=2), the classical cost to compute this is $\Omega(N^2)$: showing **a quadratic speedup over an exact classical implementation**. The speedup can be made asymptotically larger by increasing $d$. We have a more detailed discussion of this regime in Appendix D.5.3.

## 5.5 Conclusion

In summary, I provide a number of novel theoretical contributions. I further develop the VE framework for quantum vector encodings. I derive a novel quantum algorithm for the multiplication of an arbitrary dense and full-rank matrix with the element-wise square of a given vector, the first such result which does not incur a Frobenius norm (and thus rank) complexity dependence. We provide a

novel QRAM-free block-encoding of multi-filter 2D convolutions. I then prove the first end-to-end complexity guarantees for the coherent quantum acceleration of multi-layer neural network inference, under three QRAM regimes. In the first regime, I give complexity which is polylogarithmic in both the dimension of the input, and the number of parameters in the network. In the second, I show a quartic speedup over exact classical computation. In the third, I show a quadratic speedup.

## 5.6 Future Work

Progress towards achieving a practically passive QRAM is important for realizing the speedups in the first two regimes. Additionally, future work can explore classical, quantum inspired, algorithms using the norm-preservation ideas developed. Most importantly, I wonder if it is possible to coherently enact sequences of non-linear transformations without an exponentially increasing circuit depth (and with polylogarithmic error-dependence), thereby allowing very deep multi-layer architectures to be quantized, but I suspect that this is provably impossible (at least in general). Furthermore, it is conceivable that an approach enacting the non-linear transformations coherently with techniques based on QPE [176] might be able to enact a sequence of non-linearities without exponentially increasing circuit depth (albeit at the cost of an exponentially worse and exponentially decaying error-dependency). Combining such approaches may let quantum computers coherently accelerate architectures with depths of e.g., up to 25. Alternatively, one could implement a set of layers coherently, followed by tomography on a lower-dimensional state to reset the exponential depth cost, while still gaining some of the benefits of implementing multiple layers coherently. This would in effect be a hybrid approach between this work and the prior work. This could be useful in e.g., UNet based architectures, where meaningful low-dimensional paths are present. Finally, it would be interesting to investigate how the techniques developed in this chapter might be combined with theoretical machine learning ideas, such as those found in quantum PAC learning and boosting [177, 178]. For instance, if the training data distribution is produced by a neural network implementable with the techniques presented in

Regime 3, then this potentially provides a method to avoid relying on QRAM to prepare the amplitude encoded training distribution in quantum PAC. A setting where it is assumed that a quantum circuit yields a quantum state encoding the data distribution is examined in [179], and since the techniques presented in this chapter constructively yield circuits, such a connection may be possible.

# Appendices

# A

# The Limitations of Quantum Random Access Memory

**Lemma A.0.1** (Optimality of Quantum Eigenvalue Transform)**.** *Any quantum algorithm implementing a degree-k polynomial of a matrix H (with the polynomial defined on the eigenvalues) requires $\Omega(k)$ queries to $O_H$, in general.*

*Proof.* I prove this lower-bound by a reduction to unstructured search, in the QSVT framework (as per [26]), from which the bound immediately follows. Note that the result of [26] almost immediately implies this bound, but I present the following derivation in a format to enhance the clarity of the preceding discussion. For a formal definition of a block-encoding, please see Section 1.2.2. To reiterate, as per [26], an $n+a$-qubit unitary matrix $U_A$ is called an $(\alpha, a, \epsilon)$-block encoding of the $n$-qubit matrix $A$ if $\|A - \alpha(\langle 0|^{\otimes a} \otimes I_n)U_A(|0\rangle^{\otimes a} \otimes I_n)\|_2 \leq \epsilon$.

Given an unstructured search function $f(x) = 0$ for all $x \neq m$, and $f(m) = 1$ (with m being unique), specified by an $O_f$, oracle defined as $O_f|x\rangle = (-1)^{f(x)\oplus 1}|x\rangle$ (where we negate the sign for ease of analysis), one can construct a $(1, 9, 0)$-block-encoding $U_A$ of $A := \frac{1}{\sqrt{N}}|m\rangle\langle +^n|$, where $|+^n\rangle = H^{\otimes n}|0\rangle$. I give a very inefficient block encoding (in terms of the number of ancillas) for ease of exposition. First, construct a $(1, 4, 0)$-block-encoding of $H^{\otimes n}|0\rangle\langle 0|H^{\otimes n}$. This can first be done by constructing a $(1, 2, 0)$-block-encoding of $|0\rangle\langle 0| = \frac{1}{2}(I_n + (2|0\rangle\langle 0| - I_n))$ (which can

be done by applying the standard sum of block encoding lemma [26]) of an $n$ qubit identity matrix with the standard $2|0\rangle\langle 0| - I_n$ unitary from Grover search. The product of block encodings lemma can then be used with $H^{\otimes n}$ and $|0\rangle\langle 0|$, and again with $H^{\otimes n}$ to get a $(1, 4, 0)$-block encoding of $|+^n\rangle\langle +^n|$. Using the product of block encoding with $I_1 \otimes O_f$ and $|+^n\rangle\langle +^n|$ then gives a $(1, 5, 0)$-block encoding of $O_f|+^n\rangle\langle +^n|$. Then, using the sum of block encodings with $|+^n\rangle\langle +^n|$ and $O_f|+^n\rangle\langle +^n|$ yields a $(1, 9, 0)$-block-encoding of $\frac{1}{2}((O_f + I)|+^n\rangle\langle +^n|) = \frac{1}{\sqrt{N}}|m\rangle\langle +^n|$, as desired.

The polynomial approximation of the sign function, as per [26], can then be applied to the preceding block encoding unitary $U_A$, giving a constant probability of success of measuring the marked state. They give the degree of this (odd degree) polynomial as $k = O(\log(1/\epsilon)/\delta)$, where $\epsilon$ is an upper-bound on the maximum deviation of the polynomial approximation to the sign function in the $x \in [-2, 2]\backslash(-\delta, \delta)$. Setting $\delta = \frac{1}{\sqrt{N}}$, the sign function maps the non-zero singular value to 1, while leaving the zero-valued singular values unchanged. As a result, the ancilla register can be measured in the $|0\rangle$ state with $\propto 1 - \epsilon$ probability, yielding the result of the unstructured search problem. If it were possible to implement this degree $\tilde{O}(1/\delta)$ degree-polynomial with fewer than $\Omega(1/\delta)$ queries to $U_A$, the unstructured search problem could be solved in general with fewer than $\Omega(\sqrt{N})$ queries to the unstructured search oracle, violating well-known lower-bounds for unstructured search. As a consequence, any general algorithm implementing a SVT of a degree-$k$ polynomial of some matrix $H$ must use at least $\Omega(k)$ queries to that matrix.

A polynomial implementing the sign function applied to the eigenvalues of a block encoding of $\tilde{A} := \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$ would also solve unstructured search in the same way (noting that the non-zero eigenvalues of $\tilde{A}$ correspond to $\pm\frac{1}{\sqrt{N}}$ and have associated eigenvectors $\frac{1}{\sqrt{2}}(|0\rangle_1|m\rangle \pm |1\rangle_1|+^n\rangle)$. Thus, applying $\text{Sign}(\tilde{A})$ leaves the 0 eigenvalues unchanged, and maps the $\pm\frac{1}{\sqrt{N}}$ eigenvalues to $\pm 1$ (within $\epsilon$ distance). Consequently, $\text{Sign}(\tilde{A}) \approx \begin{pmatrix} 0 & |m\rangle\langle +^n| \\ |+^n\rangle\langle m| & 0 \end{pmatrix}$. Applying $\text{Sign}(\tilde{A})$ to an initial state $|1\rangle|+^n\rangle$ then gives the solution to the unstructured search problem, and so the same bound holds for Problem 2.3.1. □

**Lemma A.0.2.** *Given $P$ classical processors sharing a common memory of access time $O(1)$, a $d$-sparse matrix $A \in \mathbb{C}^{N\times N}$ and a vector $\boldsymbol{v} \in \mathbb{C}^N$, we can compute $A\boldsymbol{v}$ with $\tilde{O}(Nd/P)$ time complexity.*

*Proof.* This is a standard result; see [180]. Assume $A$ is represented as $N$ lists of elements $(j, A_{ij})$, where $A_{ij}$ are the non-zero components in row $i$, and $\boldsymbol{v}$ as an $N$-element array. With fewer than $N$ processors, each processor will iterate through a row of $A$. For an entry $(j, A_{ij})$, it will look up the $j$th element of $\boldsymbol{v}$, multiply it by $A_{ij}$ and add that to a running total. This produces one element of $A\boldsymbol{v}$. Each element can be computed independently, so this parallelizes perfectly. With more than $N$ processors, rows will have multiple processors. The processors for row $i$ can divide the entries of $A$ in that row and compute sub-totals. To compute the full sum, they add their sub-totals together in a tree structure; this tree has depth $O(\log(P/N))$. □

The following lemma was proven by my co-author, following the techniques of [181] using sorting to build matrix multiplication..

**Lemma A.0.3.** *[Matrix Multiplication via Sorting Network [1]] Given $P$ parallel processors forming a sorting network that can sort in time $\mathsf{S}$, multiplying an $N$-dimensional vector $\boldsymbol{v}$ by a $d$-sparse $N \times N$ matrix $A$ takes time $O(\mathsf{S}+Nd/P+\lg(d))$.*

**Lemma A.0.4** (Dense Matrix-Vector Multiplication with Local Memory)**.** *A set of $O(N^2\log(N))$ processors arranged in a $3D$ grid and with nearly local connectivity, leading to a total wire-length of $\tilde{O}(N^2)$, can implement a matrix-vector multiplication with an $N \times N$ matrix $A$ and an $N$-dimensional vector $\boldsymbol{v}$ with $O(\log(N))$ complexity.*

*Proof.* Consider a grid of $N \times N$ processors, each with local memory, and *no* connections to their nearest neighbors. Assume that each element in $A$ is assigned to a corresponding processor in the grid, i.e. the processor in the $i^{th}$ row and $j^{th}$ column stores element $A_{ij}$. If the initial vector $\boldsymbol{v}$ is stored in a set of $N$ data cells, using a stack of $\log(N)$ local processors, of dimension $2 \times N, 4 \times N, ..., N \times N$ we can recursively spread the elements in the vector to construct an $N \times N$ matrix

$V$ such that $V_{ij} = \boldsymbol{v}_j$. The first layer in this stack has 2 wires per layer, each of length $N/2$, for a total wire-length of $O(N^2)$. The final layer in this stack, the one that connects element $V_{ij}$ to element $A_{ij}$ in the main $N \times N$ grid, has $O(N^2)$ wires, each of length $O(1)$ for a total wire length of $O(N^2)$. The middle layers in the stack similarly have a total wire length of $O(N^2)$ (summing progressively more shorter wires). Thus, the main $N \times N$ processor grid can store $A_{ij}\boldsymbol{v}_j$ with a total of $O(N^2 \log N)$ ancillary local processors with a total wire length of $O(N^2 \log N)$. We can then repeat this process of spreading out the elements of $\boldsymbol{v}$ in reverse, adding another stack of $\log N$ processors of size $N \times N, N \times N/2, ..., N \times 2, N \times 1$. We then add elements in adjacent cells, sending them to the cell in the layer above, building up the sum of products of the appropriate matrix and vector element. This produces the output vector $A\boldsymbol{v}$, where the $i^{th}$ row stores $\sum_{j=1}^{N} A_{ij}\boldsymbol{v}_j$, with a total wire-length of $O(N^2 \log N)$, and with a total of $O(N^2 \log N)$ processors. □

# B

# Quantum State Preparation

The structure of this appendix is as follows. We first introduce basic definitions and notation in Appendix B.1. In Appendix B.2 we review relevant Hamiltonian simulation techniques of crucial importance which form the basis of our state-preparation procedure. These include 1-sparse in Appendix B.2.1 and low-rank in Appendix B.2.2 simulation techniques as well as adiabatic simulation in Appendix B.2.4.

We detail our adiabatic state-preparation approach in Appendix B.3; First we state a compact form of our main result in Theorem B.3.1 and then describe details of the procedure in Appendix B.3.1. We then systematically derive our upper bounds, first for an arbitrary finite qubit count in Appendix B.3.2 and then we prove existence of constant asymptotic bounds in Appendix B.3.2 for continuous functions. We then outline generalizations to arbitrary functions (mappings) in Appendix B.3.2. Finally, we discuss generalizations to continuous multi-variate functions. The last section Appendix B.6 contains further technical details.

## B.1 Preliminaries and Definitions

### B.1.1 Continuous and Efficiently Computable Functions

While our approach naturally applies to arbitrary functions as we detail in Appendix B.3.2, we mostly focus on the pivotal case of continuous functions given

these contain most practically important cases and they naturally admit convenient asymptotic properties as we show later.

In this work we refer to a continuous function $f : [a, b] \mapsto \mathbb{C}$ on the bounded interval $a \leq x \leq b$ as $f(x)$ whose general definition is $f \in \mathcal{C}$ as $\lim_{c\to x} f(c) = f(x)$. Recall that due to the boundedness theorem all such functions are bounded as $|f(x)| \leq \|f\|_{max}$. While we keep our results as general as possible and consider arbitrary continuous functions (unless stated otherwise), we note that so-called Lipschitz continuous functions cover nearly all instances one can encounter in practice, and we detail in Appendix B.6 that these Lipschitz continuous functions additionally satisfy powerful exponential scaling properties.

In the rest of this work we will refer to continuous functions and their discretization interchangeably. A number of important definitions will now be provided.

The ability to efficiently integrate a function makes state preparation much easier, for instance by allowing the normalization factor of the discretized function to be computed with high precision (for large $n$) thereby allowing the function oracle to be rescaled.

**Definition B.1.1** (efficiently integrable functions)**.** *We define the non-negative function $g : [a, b] \mapsto \mathbb{R}_+$ such that $g(x_j)$ gives the density in the grid interval between state $|x_j\rangle$ and $|x_{j+1}\rangle$,*

$$g(x_j) = \left[\int_{x_j}^{x_j+\Delta_N} f(x)\,\mathrm{d}x\right]^{1/2}, \tag{B.1}$$

*where $f : [a, b] \mapsto \mathbb{R}_+$ is a non-negative function and we assume that the resulting integral function $g(x_j)$ is efficiently computable as in Definition 1.2.4. Here, $\Delta_N \equiv \frac{b-a}{N}$. As such, for efficiently integrable functions we can efficiently enforce the normalization $\sum_j |g(x_j)|^2 = \|f\|_1 = 1$ which we will assume is always the case.*

**Definition B.1.2** (filling ratio)**.** *Given the usual definition for $L^p$ function norms of a Riemann integrable, bounded function $f$, $\|f\|_p = \left[\int_a^b |f(x)|^p\,\mathrm{d}x\right]^{1/p}$, we define the filling ratio of $f$,*

$$\mathcal{F} := \frac{\|f\|_1}{(b-a)\|f\|_{max}}.$$

*Note that for continuous functions over a closed interval* $\|f\|_\infty =: \|f\|_{max}$ *represents the absolute largest value of the function.*

The filling ratio $0 < \mathcal{F} \leq 1$ quantifies the absolute area under the function relative to a rectangle of the same maximal height. It is maximized for a uniform function, and minimized by a delta-like functions.

**Definition B.1.3** (generalized filling ratio)**.** *Given a function* $f : \{0,1\}^n \mapsto \mathbb{C}$*, its maximum* $M = \max_{x\in\{0,1\}^n} |f(x)|$*, and the dimension of the space it acts on* $N = 2^n$*, we define the generalized filling ratio*

$$\tilde{\mathcal{F}} := \frac{\mathcal{N}}{\sqrt{N}M}. \tag{B.2}$$

A proof that the generalized filling ratio is an asymptotic constant is provided in Lemma B.6.2.

### B.1.2 Encoding Functions into Quantum States

We are generally concerned with the preparation of quantum states following continuous functions. We now introduce some of the notation used throughout the remainder of this paper. First, we assume that we have an $n$ qubit system, with $N = 2^n$ computational basis states. Second, as explained above we consider continuous functions $f(x)$ on some real interval $x \in [a, b]$ and we discretise the interval into $N$ equally spaced samples with $a = x_0$ is the leftmost value in the interval, and $x_{N-1}$ is the rightmost value via $b = x_N$. The step size is thus given by $\Delta_N = (x_N - x_0)/N = (b - a)/N$. Note that the binary integers $j \in \{0,1\}^n$ index the standard standard basis states $\{|j\rangle\}_j$ of our quantum computer and we can equivalently refer to these in terms of the x values $\{|x_j\rangle\}_j$ given the relation $x_j = x_0 + j\Delta_N$ admits the inverse mapping $j = (x_j - x_0)\Delta_N^{-1}$.

Suppose we have an $n$ qubit quantum system, prepared in the state $|\psi\rangle$. We define two kinds of encodings as

$$\textbf{pointwise:}\quad |\psi\rangle = (\mathcal{N}_N)^{-1} \sum_{j\in\{0,1\}^n} f(x_j)|j\rangle, \qquad \textbf{integral:}\quad |\psi\rangle = \sum_{j\in\{0,1\}^n} g(x_j)|j\rangle, \tag{B.3}$$

where the integral encoding is identical to that in the Grover-Rudolph approach detailed in Appendix B.6. Here we assume that $f(x)$ is an efficiently computable function from Definition 1.2.4 and $\mathcal{N}_N := \sum_{j\in\{0,1\}^n} |f(x_j)|^2$ is a normalization constant which we cannot necessarily compute efficiently. Instead, $g(x_j)$ are piecewise integrals of an efficiently integrable function from Definition B.1.1 and, as such, we can efficiently enforce normalization.

While the two encodings may have very different behaviours for finite $N$ we show in Property B.6.2 that for large $N$ asyomptotically the integral encoding becomes identical to the corresponding pointwise encoding of a non-negative function.

Note that later we will also make use of a binary encoding of the function values $f(x)$, which we we discuss in detail in Sec. B.2.1.

## B.2 Hamiltonian Simulation Techniques

### B.2.1 1-Sparse Hamiltonian Simulation

I now summarize the approach described in refs. [83–86] for efficiently simulating the time evolution under 1-sparse Hamiltonians. For the case of 1-sparse Hamiltonians, this is substantially more efficient than general QSVT based approaches.

**Lemma B.2.1.** *Suppose we are given a one-sparse Hermitian Hamiltonian $H$ acting on $n$ qubits. $H$ is specified by the oracles $O_f$ and $O_H$, defined such that*

$$O_f|x\rangle_n|0\rangle_n = |x\rangle_n|f(x)\rangle_n, \tag{B.4}$$

$$O_H|x\rangle_n|y\rangle_n|0\rangle_d = |x\rangle_n|y\rangle_n|H_{x,y}\rangle_d, \tag{B.5}$$

*where the function $f(x)$ gives the column of the only non-zero matrix element of $A$ in row $x$, $H_{x,y}$ corresponds to the element in the $x^{th}$ row and $y^{th}$ column of $H$, and we use $d$-bits of resolution to encode the matrix elements of $H$. Then, an initial state $|\psi_0\rangle$ can be simulated by $e^{-iHt}$ for an arbitrary $t$ with 0 error using $O(1)$ oracle calls (and circuit depth equal to the circuit depth of the oracle implementations).*

*Proof.* As I include this proof simply to help the reader build intuition, I only consider the case where the matrix elements of $H$ are real, and defer the reader to the existing literature to see how this technique is adapted for the general case of a Hermitian $H$ [83–86](noting that the overall asymptotic complexity remains the same) [1]. First, observe that for a one-sparse $H$ satisfying $H = H^T$, $H|x\rangle \propto |f(x)\rangle$ as $H$ maps $|x\rangle$ to the only non-zero row, which must be the same as the non-zero column in row $x$ since $H$ is symmetric. Moreover, it must also be the case that $f(f(x)) = x$, otherwise $H$ would not be one-sparse. As a result, $H$ can be thought of as an adjacency matrix describing a graph with $2^n$ vertices, such that the graph consists of a set of connected components with each component containing either one or two vertices. Therefore, $H$ acts on $1 \times 1$ and $2 \times 2$ invariant subspaces. Consequently, simulation of $e^{-iHt}$ is equivalent to simply performing simulations in each of these $1 \times 1$ and $2 \times 2$ subspaces. Thus, the simulation circuit needs the logic to identify if a given basis vector is in a $1 \times 1$ or a $2 \times 2$ block, and then needs to perform the corresponding simulation.

First, we write our initial quantum state in the general form $|\psi_0\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle$. We then initialize our quantum state coupled to an $n$-qubit ancillary register, a single qubit ancillary register, a $d$-qubit ancillary register, and a final 1-qubit ancillary register. Note that a more efficient implementation of this procedure exists, but I use the following for expository purposes. Our joint quantum state is thus given by,

$$|\psi_0\rangle_n|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle_n|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1. \tag{B.6}$$

We begin the simulation procedure by applying the $O_f$ oracle to the first and second registers,

$$|\psi_0\rangle_n|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 \xrightarrow{O_f} \sum_{x=0}^{2^n-1} \alpha_x |x\rangle_n|f(x)\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1. \tag{B.7}$$

We now define the comparison operator, CMP, which acts on the first three registers with the mapping $\text{CMP}|x\rangle_n|y\rangle_n|0\rangle_1 = |x\rangle_n|y\rangle_n|x > y\rangle_1$ (where $x > y$ is 1 if true,

[1] Nathan Weibe has an excellent video lecture covering 1-sparse Hamiltonian simulation in [link], which is where I first learned this 1-sparse simulation technique. I adopt much of the notation used there.

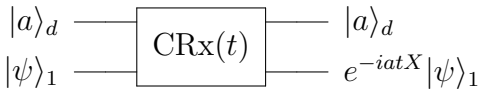


**Figure B.1:** Quantum circuit representation of the CRx($t$) gate.

and 0 if false). Applying CMP yields,

$$\xrightarrow{\text{CMP}} \sum_{x=0}^{2^n-1} \begin{cases} \alpha_x|x\rangle_n|f(x)\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 & \text{if } x \leq f(x) \\ \alpha_x|x\rangle_n|f(x)\rangle_n|1\rangle_1|0\rangle_d|0\rangle_1 & \text{if } x > f(x). \end{cases} \tag{B.8}$$

We now define the controlled-swap operation, CSWAP, acting on the first three registers such that $\text{CSWAP}|x\rangle_n|y\rangle_n|0\rangle_1 = |x\rangle_n|y\rangle_n|0\rangle_1$ and $\text{CSWAP}|x\rangle_n|y\rangle_n|1\rangle_1 = |y\rangle_n|x\rangle_n|1\rangle_1$. Applying CSWAP yields,

$$\xrightarrow{\text{CSWAP}} \sum_{x=0}^{2^n-1} \begin{cases} \alpha_x|x\rangle_n|f(x)\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 & \text{if } x \leq f(x) \\ \alpha_x|f(x)\rangle_n|x\rangle_n|1\rangle_1|0\rangle_d|0\rangle_1 & \text{if } x > f(x). \end{cases} \tag{B.9}$$

We now apply $O_H$, acting on the first, second, and fourth registers.

$$\xrightarrow{O_H} \sum_{x=0}^{2^n-1} \begin{cases} \alpha_x|x\rangle_n|f(x)\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|0\rangle_1 & \text{if } x \leq f(x) \\ \alpha_x|f(x)\rangle_n|x\rangle_n|1\rangle_1|H_{f(x),x}\rangle_d|0\rangle_1 & \text{if } x > f(x). \end{cases} \tag{B.10}$$

We must now take a brief interlude, and define two operations: CRx($t$) and CRz($t$) such that the gates act on a $d$-qubit control register and a 1-qubit target register, as shown in Figure B.1. In particular, we define the mapping when the input state is some basis vector $|a\rangle_d$, and clearly this definition holds in general by linearity. First, allowing the binary expansion $a = a_1a_2...a_d$ we note,

$$e^{-iatX} = e^{-i(a_1 2^{d-1} + a_2 2^{d-2} + ... + a_d 2^{d-d})tX} = e^{-ia_1 2^{d-1} tX} e^{-ia_2 2^{d-2} tX} ... e^{-ia_d 2^0 tX}. \tag{B.11}$$

Thus, we can clearly implement this gate by a sequence of $d$ single-qubit controlled Rx gates, with $\text{Rx}(t) = \cos(\frac{t}{2})I - i\sin(\frac{t}{2})X$, where the $j^{th}$ gate is controlled on bit $a_j$ and the time is set to $t2^{d-j}$. Thus, the cost of this operation scales as $O(d)$ in terms of circuit depth and required parameterized gates. The same argument applies to creating the CRz($t$) gate (although it can trivially be created from the CRx($t$) gate by applying a Hadamard before and after the gate on the target qubit).

We now must determine whether a given state is in a $1 \times 1$ or $2 \times 2$ invariant subspace, to do this we use the final ancillary qubit (note that this qubit is not

necessary, and this operation can be done in place, however we include it for notational clarity). Define a new operator, EQ, which acts on the first two registers and the fifth register, with the mapping $\text{EQ}|x\rangle_n|y\rangle_n|0\rangle_1 = |x\rangle_n|y\rangle_n|x=y\rangle$, where $x=y$ is 1 if true, and 0 otherwise. Then,

$$\xrightarrow{\text{EQ}} \sum_{x=0}^{2^n-1} \begin{cases} \alpha_x|x\rangle_n|x\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|1\rangle_1 & \text{if } x = f(x) \\ \alpha_x|x\rangle_n|f(x)\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|0\rangle_1 & \text{if } x < f(x) \\ \alpha_x|f(x)\rangle_n|x\rangle_n|1\rangle_1|H_{f(x),x}\rangle_d|0\rangle_1 & \text{if } x > f(x). \end{cases} \tag{B.12}$$

Since the first two registers are only equal if $f(x) = x$, they are only equal if $|x\rangle$ is in a $1 \times 1$ invariant subspace, and as such the correct evolution prescribes a rotation by CRz. If the first two registers are not equal, then we must be in a $2 \times 2$ invariant subspace, and so the correct evolution follows a rotation by CRx. Noting that both the CRx and CRz gates use the fourth register as their "pass-through" input, and the third register as their target input, we can then apply CRx gate conditioned on the $|0\rangle_1$ state of the fifth register followed by a CRz gate conditioned on the $|1\rangle_1$ state of the fifth register (we refer to these controlled-controlled rotation gates as CCRx and CCRz). Note that this is equivalent to only performing an uncontrolled CRx gate, and then applying Hadamard gates on the third register before and after the CRx gate conditioned on the fifth register, however the preceding description yields a clearer analysis. We then obtain,

$$\xrightarrow{\text{CCRx(t) CCRz(t)}} \sum_{x=0}^{2^n-1} \begin{cases} \text{CRz}(t)\alpha_x|x\rangle_n|x\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|1\rangle_1 & \text{if } x = f(x) \\ \text{CRx}(t)\alpha_x|x\rangle_n|f(x)\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|0\rangle_1 & \text{if } x < f(x) \\ \text{CRx}(t)\alpha_x|f(x)\rangle_n|x\rangle_n|1\rangle_1|H_{f(x),x}\rangle_d|0\rangle_1 & \text{if } x > f(x). \end{cases} \tag{B.13}$$

$$= \sum_{x=f(x)} \text{CRz}(t)\alpha_x|x\rangle_n|x\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|1\rangle_1 \tag{B.14}$$

$$+ \sum_{x<f(x)} \text{CRx}(t)\left(|x\rangle_n|f(x)\rangle(\alpha_x|0\rangle_1 + \alpha_{f(x)}|1\rangle_1)|H_{x,f(x)}\rangle_d|0\rangle_1\right) \tag{B.15}$$

$$= \sum_{x=f(x)} e^{-iH_{x,x}t}\alpha_x|x\rangle_n|x\rangle_n|0\rangle_1|H_{x,f(x)}\rangle_d|1\rangle_1 \tag{B.16}$$

$$+ \sum_{x<f(x)} |x\rangle_n|f(x)\rangle\left(e^{-iH_{x,f(x)}Xt}(\alpha_x|0\rangle_1 + \alpha_{f(x)}|1\rangle_1)\right)|H_{x,f(x)}\rangle_d|0\rangle_1 \tag{B.17}$$

where the second last equation follows from the fact that if $x < f(x)$, then $|x\rangle_n|f(x)\rangle_n$ is equal to $|f(x)\rangle_n|x\rangle_n$ when $f(x) > x$, and the notation $\sum_{x=f(x)}$ means

sum over all $x$ such that $x = f(x)$, and similarly $\sum_{x<f(x)}$ means sum over all $x$ such that $x < f(x)$. Finally, we uncompute all the ancillary operations (EQ, $O_H$, CSWAP, CMP, $O_f$), obtaining the state

$$\sum_{x=f(x)} e^{-iH_{x,x}t}\alpha_x|x\rangle_n|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 + \sum_{x\neq f(x)} \alpha_x \left(e^{-iHt}|x\rangle_n\right)|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 \quad \text{(B.18)}$$

$$= \sum_{x=0}^{2^n-1} \alpha_x \left(e^{-iHt}|x\rangle_n\right)|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1 = \left(e^{-iHt}|\psi_0\rangle_n\right)|0\rangle_n|0\rangle_1|0\rangle_d|0\rangle_1. \quad \text{(B.19)}$$

where this simply follows from the observation that $e^{-iH_{x,x}t}$ correctly describes the evolution of a $1 \times 1$ invariant subspace, while $e^{-iH_{x,f(x)}Xt}$ correctly describes the evolution of a $2 \times 2$ invariant subspace, and then we simply apply linearity of $e^{-iHt}$.

It is worth briefly mentioning that if the matrix elements of $H$ can be exactly encoded in $d$ bits, then this procedure incurs 0 error, and if they cannot, then the error vanishes exponentially in $d$ and so is effectively negligible. □

Note that if the elements of $H$ are efficiently computable, then a quantum circuit implementing $O_H$ can clearly be implemented efficiently (i.e. with polynomial bounded complexity). Similarly, if the location of the non-zero element in any given row can also be computed efficiently, then the overall one-sparse simulation circuit can also be implemented efficiently.

### B.2.2 Low-Rank Hamiltonian Simulation

As low-rank Hamiltonian simulation is an essential subroutine in our algorithm, I now present the results of Rebentrost *et al.* and summarize their proof [82].

**Statement B.2.1.** *The procedure presented in [82] allows the dynamics under an arbitrary low-rank dense matrix $A \in \mathbb{C}^{n\times n}$ to be simulated efficiently if the matrix satisfies the two conditions: $\|A\|_{max} = \Theta(1)$ and $\|A\|_2 = \Theta(N)$ as we increase the dimension $N$. Moreover, we suppose we have oracular access to the elements of $A$. Then, for time $\Delta t$, we can simulate the dynamics under $e^{-i\Delta tA}$ using 4 oracle queries, with trace-norm error bounded by $2\|A\|_{max}^2\Delta t^2$.*

*Proof.* We now summarize the results of [82]. First, we begin by embedding the elements of $A$ in a matrix $S_A \in \mathbb{C}^{N^2 \times N^2}$ as follows,

$$S_A = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} A_{jk} |k\rangle\langle j| \otimes |j\rangle\langle k|. \tag{B.20}$$

Clearly, $S_A$ is one-sparse. Moreover, element $x, y$ of $S_A$, $[S_A]_{xy}$ can be computed easily and is given by $A_{jk}$, where $x = jN + k$ and $y = jN + k$. Thus, $O_{S_A}$ is clearly efficiently implementable, given access to the elements of $A$. Moreover, given the structure of $S_A$ it is also straightforward to implement $O_f$, and as a result we can perform the simulation $e^{-iS_A \Delta t}$ using Lemma B.2.1 with a total of 4 oracle calls, and with 0 error (assuming no discretization error in the matrix elements of $A$). We now assume that the state we wish to evolve has a density operator given by $\sigma$, and we have a separate $n$-qubit register in state $\rho = |\phi\rangle\langle\phi|$ such that $|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$ (i.e. a uniform superposition). The exact evolution for time $\Delta t$ under $S_A$ of state $\rho \otimes \sigma$ generates the following dynamics under the partial trace,

$$\mathrm{tr}_1[e^{-i\Delta t S_A} \rho \otimes \sigma e^{i\Delta t S_A}] = e^{-i\Delta t A/N} \sigma e^{i\Delta t A/N} + \epsilon_0. \tag{B.21}$$

The leading error in $\Delta t$ in terms of a trace distance was established in [82] as

$$\epsilon_0 \leq 2[\|A\|_{max} \, \Delta t]^2, \tag{B.22}$$

which depends on the absolute largest entry of the matrix $\|A\|_{max}$. We will use this approach to simulate the dynamcis under our rank-1 matrix as $A/N \propto |\psi\rangle\langle\psi|$. □

### B.2.3 Tighter Bounds on 1-Sparse Simulation Error

Here we tighten the bounds of ref. [82] for the specific case of rank-1 projector matrices $A$, assuming that a measurement is performed on the ancillary register after the Hamiltonian simulation procedure.

**Lemma B.2.2** (Error of low-rank simulation)**.** *Given a function $f : \mathbb{R} \mapsto \mathbb{C}$, define the unnormalized state $|\phi\rangle := \sum_{j=1}^{N} f(x_j)|j\rangle$ (with the uniform grid $x_j \in [a, b]$). We then define our rank-one matrix $A := |\phi\rangle\langle\phi|$. Define $\mathcal{N} := \sqrt{\sum_j |f(x_j)|^2}$ (i.e. $\mathcal{N}$ is the $\ell_2$ norm of the discretized function vector). We apply the low-rank*

*simulation from Statement B.2.1 for time $\Delta t$. The quantum state after measuring the ancilla in the $|+^n\rangle$ state and discarding the ancilla is $|\psi'\rangle \propto e^{-i\Delta t A/N}|\psi\rangle + |\xi\rangle$ with $\|\xi\| \leq \frac{3}{2}\Delta t^2 \frac{\mathcal{N}^2}{N}\|A\|_{max}$ and with bounded probability* $\text{Prob} \geq 1 - \Delta t^2 \frac{\mathcal{N}^2}{N}\|A\|_{max}$ *expressing only leading terms in $\Delta t$. Given the above expression holds for any input state $|\psi\rangle$ it follows that the algorithmic error $\epsilon_0(\Delta t) := \|U(\Delta t) - U(\Delta t)'\|$ of the approach is bounded as*

$$\epsilon_0(\Delta t) \leq \tfrac{3}{2}\Delta t^2 \tfrac{\mathcal{N}^2}{N}\|A\|_{max}$$

*where $U$ is the ideal, piecewise constant unitary $e^{-i\Delta t A/N}$ and $U'$ represents the mapping of our procedure (1-sparse evolution then Hadamard measurement).*

*Proof.* We use a rank-2 tensor to represent the amplitudes of our two-register quantum system. In particular, let $|\Psi\rangle := \sum_{j,k} \Psi_{jk}|j\rangle|k\rangle$. Then,

$$S_A|\Psi\rangle = \left(\sum_{jk} A_{jk}|k\rangle\langle j| \otimes |j\rangle\langle k|\right)|\Psi\rangle = \sum_{j,k} A_{kj}\Psi_{kj}|j\rangle|k\rangle, \tag{B.23}$$

i.e. each amplitude is updated as $\Psi^{(1)}_{jk} = A_{kj}\Psi_{kj}$. We can then use the notation $|\Psi^{(k)}\rangle := S_A^k|\Psi\rangle$. Assign $|\Psi\rangle := |+^n\rangle|\psi\rangle$ for some initial state $|\psi\rangle$ that we wish to evolve. We begin by explicitly computing the joint state of the ancilla and main registers $|\Psi(t)\rangle := e^{-itS_A}|\Psi\rangle$ up to order $t^2$ (neglecting discretization error in the entries of $S_A$ which can be suppressed exponentially in the number of digitization bits used),

$$e^{-itS_A}|\Psi\rangle = (I - itS_A)|\Psi\rangle + \mathcal{O}(t^2) = |\Psi\rangle - it\sum_{j,k} A_{kj}\Psi_{kj}|j\rangle|k\rangle + \mathcal{O}(t^2), \tag{B.24}$$

where we have used the action of $S_A$ on the initial joint state as per Eq. (B.23). Now, given some efficiently computable function $f$, we assume that the matrix $A$ is defined as $A = |\phi\rangle\langle\phi|$ for the unnormalized state $|\phi\rangle = \sum_{j=1}^N f(x_j)|j\rangle$. Allowing the notation $\phi_j := f(x_j)$, we can then write the elements of the matrix $A$ as $A_{jk} = \phi_j^*\phi_k$. Similarly, we observe that $|\Psi\rangle := |+^n\rangle|\psi\rangle$ implies that $\Psi_{jk} = \frac{1}{\sqrt{N}}\psi_k$. We can then further simplify Eq. (B.24) as

$$e^{-itS_A}|\Psi\rangle = |+^n\rangle|\psi\rangle - \frac{it}{\sqrt{N}}\sum_{j,k}\phi_k^*\phi_j\psi_j|j\rangle|k\rangle + \mathcal{O}(t^2) \tag{B.25}$$

$$= |+^n\rangle|\psi\rangle - \frac{it}{\sqrt{N}}\left(\sum_j \phi_j\psi_j|j\rangle\right)\left(\sum_k \phi_k^*|k\rangle\right) + \mathcal{O}(t^2). \tag{B.26}$$

We now compute the state obtained after projecting the ancilla register onto the all plus state, i.e. by applying the projector $\mathcal{P} := |+^n\rangle\langle+^n| \otimes \mathrm{I}$. Allowing $\chi := \sum_j \phi_j \psi_j$, simple algebra shows,

$$\mathcal{P} e^{-itS_A} |\Psi\rangle = |+^n\rangle \left( |\psi\rangle - \frac{it}{N} \chi \sum_k \phi_k^* |k\rangle \right) + \mathcal{O}(t^2). \tag{B.27}$$

We will now show that this is equal to the exact evolution applied to the single register state $|\psi\rangle$ up to a second order error term. First,

$$e^{-iAt/N}|\psi\rangle = \left(I - \frac{it}{N}A\right)|\psi\rangle + \mathcal{O}(t^2) = |\psi\rangle - \frac{it}{N}\sum_{j,k} \psi_j A_{kj} |k\rangle + \mathcal{O}(t^2), \tag{B.28}$$

where we used $A|\psi\rangle = \sum_{j,k} \psi_j A_{kj} |k\rangle$. Again using $A_{jk} = \phi_j^* \phi_k$, we obtain

$$e^{-iAt/N}|\psi\rangle = |\psi\rangle - \frac{it}{N}\sum_{j,k} \phi_k^* \phi_j \psi_j |k\rangle + \mathcal{O}(t^2) = |\psi\rangle - \frac{it}{N}\chi \sum_k \phi_k^* |k\rangle + \mathcal{O}(t^2). \tag{B.29}$$

Consequently, we see that the state of the quantum system in Eq. (B.27) (i.e. the state obtained after simulating $e^{-itS_A}$ and applying a protective measurement to the ancillary register) is exactly equal to the evolution implied by $e^{-iAt/N}$ up to a second order error term in $t$.

We now must bound the norm of the error term obtained by using this approximation, and we must also bound the probability of success (i.e. the probability of measuring the all plus state in our ancillary measurement).

Now, we compute the probability $p$ of measuring the all plus state in the ancillary measurement. This probability can be given by the norm of the projected vector, i.e.

$$p := \langle \Psi | e^{iS_A t} \mathcal{P} e^{-iS_A t} | \Psi \rangle, \tag{B.30}$$

noting that $\mathcal{P}^\dagger \mathcal{P} = \mathcal{P}$. Taking a second order series expansion and performing simple algebra yields,

$$p = \langle \Psi | \left( I + itS_A - \frac{t^2}{2} S_A^2 \right) \mathcal{P} \left( I - itS_A - \frac{t^2}{2} S_A^2 \right) |\Psi\rangle + \mathcal{O}(t^3) \tag{B.31}$$

$$= 1 + t^2 \left( \langle \Psi | S_A \mathcal{P} S_A | \Psi \rangle - \langle \Psi | S_A^2 | \Psi \rangle \right) + \mathcal{O}(t^3). \tag{B.32}$$

It is straightforward to show that $\langle\Psi|S_A\mathcal{P}S_A|\Psi\rangle \geq 0$, so we can obtain the lower-bound on the probability of success as,

$$p \geq 1 - t^2\langle\Psi|S_A^2|\Psi\rangle + \mathcal{O}(t^3) \tag{B.33}$$

$$= 1 - t^2\langle\Psi|\sum_{j,k} A_{jk}A_{kj}\Psi_{jk}|j\rangle|k + \mathcal{O}(t^3)\rangle \tag{B.34}$$

$$= 1 - t^2\sum_{j,k}\psi_k^*\psi_k\phi_k^*\phi_k\phi_j^*\phi_j + \mathcal{O}(t^3) \tag{B.35}$$

$$= 1 - \frac{t^2\mathcal{N}^2}{N}\sum_k |\phi_k|^2|\psi_k|^2 + \mathcal{O}(t^3) \tag{B.36}$$

$$\geq 1 - \frac{t^2\mathcal{N}^2}{N}\|A\|_{max} + \mathcal{O}(t^3), \tag{B.37}$$

where the second equality follows from the fact that $A_{jk} = \phi_j^*\phi_k$ and $\Psi_{jk} = \frac{1}{\sqrt{N}}\psi_k$. Thus, expressing only leading-order terms in $t$, we get the lower-bound on the probability of success (measuring the all plus ancillary state) as

$$p \geq 1 - \frac{t^2\mathcal{N}^2}{N}\|A\|_{max}. \tag{B.38}$$

Finally, we want to bound the magnitude of the error term obtained when using our one-sparse Hamiltonian simulation embedding technique combined with a protective measurement. Allow $p$ to represent the probability of measuring the $|+^n\rangle$ state in the ancillary register, when performing the projective measurement $\mathcal{P}$ on the state $e^{-iS_At}|+^n\rangle|\psi\rangle$. Then, we wish to bound the norm,

$$\||\epsilon\rangle\|_2 := \left\|\left(\frac{1}{\sqrt{p}}\mathcal{P}e^{-iS_At} - I\otimes e^{-iAt/N}\right)|+^n\rangle|\psi\rangle\right\|_2 \tag{B.39}$$

$$\leq \left\|\frac{\mathcal{P}}{\sqrt{p}}(I - itS_A - \frac{t^2}{2}S_A^2)|+^n\rangle|\psi\rangle - |+^n\rangle\left(I - \frac{it}{N}A - \frac{t^2}{2N^2}A^2\right)|\psi\rangle\right\|_2 + \mathcal{O}(t^3), \tag{B.40}$$

where the inequality follows from simple algebra and the application of the triangle inequality to all terms higher than second order in $t$. As we previously showed, $p \geq 1 - \frac{t^2\mathcal{N}^2}{N}\|A\|_{max} + \mathcal{O}(t^3)$. As a result, a simple series expansion shows that $\frac{1}{\sqrt{p}} \leq 1 + \frac{t^2\mathcal{N}^2}{2N}\|A\|_{max} + \mathcal{O}(t^4)$. Thus, the following quantity must be maximized by

some $Q$ satisfying $0 \leq Q \leq \frac{t^2\mathcal{N}^2}{N}\|A\|_{max} + \mathcal{O}(t^4)$.

$$\begin{aligned}
\||\epsilon\rangle\|_2 &\leq \left\|(1+Q)\,\mathcal{P}(I - itS_A - \frac{t^2}{2}S_A^2)|+^n\rangle|\psi\rangle - |+^n\rangle\left(I - \frac{it}{N}A - \frac{t^2}{2N^2}A^2\right)|\psi\rangle\right\|_2 \\
&\quad + \mathcal{O}(t^3) \\
&\leq \left\|\frac{t^2}{2}\mathcal{P}S_A^2|+^n\rangle|\psi\rangle - \frac{t^2}{2N^2}|+^n\rangle A^2|\psi\rangle\right\|_2 + Q\left\|\mathcal{P}(I - itS_A - \frac{t^2}{2}S_A^2)|+^n\rangle|\psi\rangle\right\|_2 \\
&\quad + \mathcal{O}(t^3) \\
&\leq \left\|\frac{t^2}{2}\mathcal{P}S_A^2|+^n\rangle|\psi\rangle - \frac{t^2}{2N^2}|+^n\rangle A^2|\psi\rangle\right\|_2 + \frac{t^2\mathcal{N}^2}{2N}\|A\|_{max}\,\|\mathcal{P}|+^n\rangle|\psi\rangle\|_2 \\
&\quad + \mathcal{O}(t^3),
\end{aligned}$$

where the second inequality follows from the observation that $\mathcal{P}(I - itS_A - \frac{t^2}{2}S_A^2)|+^n\rangle|\psi\rangle$ is exactly equal to $|+^n\rangle\left(I - \frac{it}{N}A - \frac{t^2}{2N^2}A^2\right)|\psi\rangle$ up to (and excluding) the second order in $t$, as we showed above, and the final equality follows from simply observing that the quantity is maximized when $Q$ takes the value of its upper-bound.

Observing that $\|\mathcal{P}|+^n\rangle|\psi\rangle\|_2 \leq 1$ and that $A^2 = \mathcal{N}^2 A$ and thus that $\|A^2|\psi\rangle\|_2 \leq \mathcal{N}^2\|A\|_2 = \mathcal{N}^4$, we obtain

$$\||\epsilon\rangle\|_2 \leq \frac{t^2}{2}\left\|\mathcal{P}S_A^2|+^n\rangle|\psi\rangle\right\|_2 + \frac{t^2\mathcal{N}^4}{2N^2} + \frac{t^2\mathcal{N}^2}{2N}\|A\|_{max} + \mathcal{O}(t^3). \tag{B.41}$$

We will now bound $\|\mathcal{P}S_A^2|+^n\rangle|\psi\rangle\|_2$ by bounding its square. Using $A_{jk} = \phi_j^*\phi_k$ and $\Psi_{jk} = \frac{1}{\sqrt{N}}\psi_k$, we get

$$\left\|\mathcal{P}S_A^2|\Psi\rangle\right\|_2^2 = \left\|\frac{1}{\sqrt{N}}\sum_{j,k}A_{jk}A_{kj}\Psi_{jk}|+^n\rangle|k\rangle\right\|_2^2 \tag{B.42}$$

$$\leq \frac{\mathcal{N}^4}{N^2}\sum_k \phi_k^*\phi_k\phi_k^*\phi_k\psi_k^*\psi_k \tag{B.43}$$

$$\leq \frac{\mathcal{N}^4}{N^2}\|A\|_{max}^2\sum_k|\psi_k|^2 = \frac{\mathcal{N}^4}{N^2}\|A\|_{max}^2, \tag{B.44}$$

since $\phi_k^*\phi_k \leq \|A\|_{max}$. Consequently, $\|\mathcal{P}S_A^2|+^n\rangle|\psi\rangle\|_2 \leq \frac{\mathcal{N}^2}{N}\|A\|_{max}$. Substituting this into Eq. (B.41), we obtain

$$\||\epsilon\rangle\|_2 \leq t^2\frac{\mathcal{N}^2}{N}\|A\|_{max} + \frac{t^2\mathcal{N}^4}{2N^2} + \mathcal{O}(t^3). \tag{B.45}$$

Finally, we exploit the fact that $\mathcal{N}^2/N \leq \|A\|_{max}$ and set $t = \Delta t$ to obtain,

$$\||\epsilon\rangle\|_2 \leq \tfrac{3}{2}\Delta t^2 \tfrac{\mathcal{N}^2}{N}\|A\|_{max} + \mathcal{O}(t^3). \tag{B.46}$$

□

**Lemma B.2.3** (Total success probability)**.** *We use the low-rank approach to simulate an evolution for overall time $T$ by applying $r$ piece-wise constant evolutions with time $T/r$. The probability that throughout $r$ consecutive iterations we always measure the all plus state is lower bounded by*

$$\text{prob}_{tot} \geq 1 - T^2 \tfrac{\mathcal{N}^2}{N}\|A\|_{max}/r + \mathcal{O}(T^4/r^2),$$

*and we discard all other measurement outcomes for the sake of analytical simplicity.*

*Proof.* We can compute the overall probability for $r$ iterations using Eq. (B.38) and substituting that $t \equiv \Delta t = T/r$ as

$$\text{Prob}^r \geq (1 - T^2 \tfrac{\mathcal{N}^2}{N}\|A\|_{max}/r^2)^r + \mathcal{O}(T^3/r^3) = 1 - T^2 \tfrac{\mathcal{N}^2}{N}\|A\|_{max}/r + \mathcal{O}(T^4/r^2).$$

□

Note that the above is a loose bound – in practice one does not need to discard the non-+outcomes as they lead to nearly the same performance; but in Lemma B.2.2 we only took into account and bounded the $|+^n\rangle$ outcome fidelities, and bounding other measurement outcomes is beyond the scope of the present work. We note that the probabilistic aspect of the algorithm is simply an analytical convenience, rather than an intrinsic property of the approach.

### B.2.4 Adiabatic Computation

I now provide a brief outline of adiabatic quantum computing. Given an initial Hamiltonian $\mathcal{H}(0) = \mathcal{H}_0$ whose ground state is easy to prepare, and a final Hamiltonian $\mathcal{H}(1) = \mathcal{H}_1$ whose ground state we would like to prepare, we define the time-dependent Hamiltonian $\mathcal{H}(s)$, with $0 \leq s \leq 1$. If we evolve the state according to $\mathcal{H}(s)$ "sufficiently slowly", the adiabatic theorem guarantees that the state will

remain in the instantaneous ground state of $\mathcal{H}(s)$ throughout the evolution, and so at the end of the procedure we will have obtained the ground state of $\mathcal{H}_1$. For an insightful discussion of adiabatic quantum computation, see the paper by van Dam *et al.* [95], or the comprehensive review by Albash *et al.* [182]. As introduced by Farhi *et al.* [96], $\mathcal{H}(s)$ is often defined as an interpolation between $\mathcal{H}_0$ and $\mathcal{H}_1$ in the form $\mathcal{H}(s) = (1-s)\mathcal{H}_0 + s\mathcal{H}_1$, however, this generally need not be the case. Precisely, we perform the evolution obtained by solving the Shrödinger equation,

$$\frac{d}{ds}|\psi(s)\rangle = -i\mathcal{H}(s)|\psi(s)\rangle, \tag{B.47}$$

where we select $|\psi(s)\rangle$ such that $|\psi(0)\rangle$ is the ground state of $\mathcal{H}_0$. Then, our objective is to obtain a circuit performing the evolution under the time-dependent Hamiltonian $\mathcal{H}(s)$. Of note, the adiabtic evolution is considered sufficiently slow if throughout the evolution the adiabatic schedule $\tau(s)$ satisfies,

$$\frac{\|\frac{d}{ds}\mathcal{H}(s)\|_2}{g^2(s)} \ll \tau(s), \tag{B.48}$$

where $g(s)$ is the spectral gap of $\mathcal{H}(s)$ (i.e. the difference in the two lowest energy levels), and $\|\frac{d}{ds}\mathcal{H}(s)\|_2$ is the spectral norm of the derivative of our time-dependent Hamiltonian. The spectral norm of a matrix $M$ is defined as $\max_{||x||_2=1} ||Mx||_2$. In the simplest implementation it is sufficient to use a constant schedule throughout, defined as

$$\max_{s\in[0,1]} \frac{\|\frac{d}{ds}\mathcal{H}(s)\|_2}{g_{min}^2} \ll \tau \tag{B.49}$$

where $g_{min}$ is the minimum spectral gap of $\mathcal{H}(s)$ for any time $s$. Then, for the constant schedule, the delay factor is given by $T = \int_0^1 \tau ds = \tau$ and so we have the bound $T = O\left(\max_{s\in[0,1]} \frac{\|\frac{d}{ds}\mathcal{H}(s)\|_2}{g_{min}^2}\right)$. As described by van Dam *et al.*, when using a constant schedule, a discretized unitary transformation induced by $\mathcal{H}(t)$ may be written as,

$$U'(T) = \exp\left[-i\frac{T}{r}\mathcal{H}(\frac{r}{r})\right] \cdot ... \cdot \exp\left[-i\frac{T}{r}\mathcal{H}(\frac{1}{r})\right], \tag{B.50}$$

That is, we approximate the overall transformation by a sequence of $r$ time-independent evolutions, each applied for an equal amount of time $\frac{T}{r}$. An important

fact we utilize in bounding the error from such a discretization comes from Lemma 1 presented by van Dam *et al.* [95]. I summarize their lemma and describe its proof for completeness.

**Lemma B.2.4.** *Suppose you have two time-dependent Hamiltonians, $\mathcal{H}(t)$ and $\mathcal{H}'(t)$ such that they induce the unitaries $U(T)$ and $U'(T)$ respectively. If the spectral norm of the difference of the Hamiltonians is bounded by at most $\delta$ for all times $t$, i.e. $\|\mathcal{H}(t) - \mathcal{H}'(t)\|_2 \leq \delta$, then $\|U(T) - U'(T)\|_2 \leq \sqrt{2T\delta}$.*

*Proof.* Define the states $|\psi'(t)\rangle$ and $|\psi(t)\rangle$, such that $|\psi'(0)\rangle = |\psi(0)\rangle$ which are respectively obtained by evolving the starting state according to $\mathcal{H}'(t)$ and $\mathcal{H}(t)$. We then start by computing the operator distance via the (maximal) vector norm $\|(U(T) - U'(T))\,|\psi(0)\rangle\|_2 = \sqrt{\langle\psi(0)|\,(U(T) - U'(T))^\dagger\,(U(T) - U'(T))\,|\psi(0)\rangle}$, we have

$$\|U(T) - U'(T)\|_2 \leq \sqrt{2 - 2\mathrm{Re}\langle\psi'(T)|\psi(T)\rangle}.$$

Here we consider how the inner product of the two states change with respect to time (and note the parametrisation is smooth in our case), obtaining

$$\frac{d}{dt}\langle\psi'(t)|\psi(t)\rangle = -i\langle\psi'(t)|\,(\mathcal{H}(t) - \mathcal{H}'(t))\,|\psi(t)\rangle. \tag{B.51}$$

We can then integrate both sides with respect to time from 0 to $T$ and lower bound its absolute value as

$$\begin{aligned}|\langle\psi'(T)|\psi(T)\rangle| &= |1 - i\int_0^T \langle\psi'(t)|\,(\mathcal{H}(t) - \mathcal{H}'(t))\,|\psi(t)\rangle\,\mathrm{d}t| \\ &\geq 1 - |\int_0^T \langle\psi'(t)|\,(\mathcal{H}(t) - \mathcal{H}'(t))\,|\psi(t)\rangle\,\mathrm{d}t|\end{aligned}$$

Of course, by assumption the spectral norm of the difference of the two Hamiltonians is bounded by $\delta$, and thus $|\int_0^T \langle\psi'(t)|\,(\mathcal{H}(t) - \mathcal{H}'(t))\,|\psi(t)\rangle\,\mathrm{d}t| \leq \delta T$. Substituting this bound back we obtain

$$|\langle\psi'(T)|\psi(T)\rangle| \geq 1 - \delta T, \tag{B.52}$$

and we thus immediately obtain the bound $\|U(T) - U'(T)\|_2 \leq \sqrt{2\delta T}$. □

For completeness, I now present a result due to my co-author showing that bounds on the distance between induced unitaries implies a bound on the distance of the generating Hermitian matrices. For the proof, please see our paper [3]. Note that this result only holds for small $t$.

**Corollary B.2.1.** *Suppose we have the ideal unitary evolution $U' := e^{-itH'}$ for fixed time $t$ generated by the time-independent Hamiltonian $H'$ and we have an approximation to this unitary as $U''$ with the error bound $\|U' - U''\|_2 \leq \kappa t^2$ for some $\kappa \geq 0$. The error bound on the unitaries implies that any effective Hamiltonian $H''$ that generates the approximate unitary $U'' := e^{-itH''}$ satisfies $\|H' - H''\|_2 \leq \kappa t + O(t^2)$.*

I now present a simple bound.

**Lemma B.2.5.** *Suppose you have Hamiltonians $\mathcal{H}(t)$, $\mathcal{H}'(t)$ and $\mathcal{H}''(t)$, which induce the unitary transformations $U(T)$, $U'(T)$ and $U''(T)$ respectively, and that $\|\mathcal{H}(t) - \mathcal{H}'(t)\|_2 \leq \delta_0$ and $\|\mathcal{H}'(t) - \mathcal{H}''(t)\|_2 \leq \delta_1$. Then, $\|U(T) - U''(T)\|_2 \leq \sqrt{2T(\delta_0 + \delta_1)}$.*

*Proof.* First, by the triangle inequality,

$$\|\mathcal{H}(t) - \mathcal{H}''(t)\|_2 = \|\mathcal{H}(t) - \mathcal{H}'(t) + \mathcal{H}'(t) - \mathcal{H}''(t)\|_2 \tag{B.53}$$

$$\leq \|\mathcal{H}(t) - \mathcal{H}'(t)\|_2 + \|\mathcal{H}'(t) - \mathcal{H}''(t)\|_2 \tag{B.54}$$

$$\leq \delta_0 + \delta_1. \tag{B.55}$$

Applying Lemma B.2.4 immediately then gives the result $\|U(T) - U''(T)\|_2 \leq \sqrt{2T(\delta_0 + \delta_1)}$. □

## B.3 General Distribution Loading Through Adiabatic Evolution: Direct State Preparation on $n$ Qubits

We now present the main result in the setting of the generalized filling ratio, and then we present two corollaries for the asymptotic cases when a pointwise and integral function encoding is used.

**Theorem B.3.1.** *Asymptotically in $N$, for an efficiently computable continuous function $f : [a, b] \mapsto \mathbb{C}$, the direct state preparation algorithm prepares the $n$ qubit state (with $N = 2^n$), $|\psi\rangle = \frac{1}{\mathcal{N}(1)} \sum_{j=0}^{N-1} f(x_j)|j\rangle$, up to error $\epsilon$ with query complexity bounded as*

$$O\left(\frac{\tilde{\mathcal{F}}^{-2}}{\epsilon^2}\right),$$

*where $\tilde{\mathcal{F}}$ is the filling ratio from Definition B.1.3. The error $\epsilon$ is the deviation from an ideal unitary in terms of a spectral distance. Moreover, the algorithm uses $O(n + d)$ ancillary qubits (where $d$ is the number of digits used in the discretization of $f$), and has a probability of failure bounded by*

$$O\left(\epsilon^2\right).$$

*The probability of failure decreases as we decrease the target error $\epsilon$.*

*Proof.* First, we take the function given as input, $f$ (or $g$ in the integral encoding case), and estimate its $\ell_2$ norm, $\mathcal{N}(1)^2 = \sum_{j=0}^{N-1} |f(x_j)|^2$. This can be done with complexity $O(\hat{\mathcal{F}}^{-1}/\epsilon)$ with a quantum algorithm (such as the Grover amplitude state preparation procedure described in the appendix, combined with amplitude estimation). We then define the re-normalized function, $\hat{f}(x) := f(x)\sqrt{N}/\mathcal{N}(1)$. We note that we do not propagate the error in learning the value of $\mathcal{N}(1)$ through our following analysis. Throughout all the following proofs, we assume that $A$ is defined in terms of $\hat{f}(x)$. From Lemma B.3.5 we know that the algorithm has query complexity $O(r)$, and uses $O(n + d)$ ancillary qubits. Allowing the notation $M_f = \max_j(f(x_j))$, we have $M_{\hat{f}}^2 = \|A(1)\|_{max}$. From Lemma B.3.6, we have that $\epsilon \in O\left(\sqrt{\|A(1)\|_{max}/r}\right)$. Of course,

$$M_{\hat{f}} = \max_j\left(\hat{f}(x_j)\right) = \max_j\left(f(x_j)\sqrt{N}/\mathcal{N}(1)\right) = M_f\sqrt{N}/\mathcal{N}(1) = \tilde{\mathcal{F}}^{-1}. \quad \text{(B.56)}$$

Consequently, $\epsilon \in O\left(\tilde{\mathcal{F}}^{-2}\right)$ for the pointwise encoding of the amplitudes. It immediately follows that to obtain an error $\epsilon$ we can set $r$ to the upper-bound in $r \in O(\tilde{\mathcal{F}}^{-2}/\epsilon^2)$, yielding the query bound. Finally, from Lemma B.2.3 we have that the probability of failure is bounded by $O(\tilde{\mathcal{F}}^{-2}/r)$, which immediately gives

the probability failure bound of $O(\epsilon^2)$ when substituting our above upper bound for $r$ as a *lower bound*, i.e., we use $r$ larger than the upper bound to guarantee a worst-case error $\epsilon$.

**Corollary B.3.1.** *[Asymptotic Pointwise Encoding] Consider the pointwise encoding case of Theorem B.3.1, in the limit as $N \mapsto \infty$. Then, the complexity of the procedure is given by*

$$O\left(\frac{\mathcal{F}^{-2}}{\epsilon^2}\right),$$

*with a probability of failure bounded by $O(\epsilon^2)$.*

*Proof.* The proof follows immediately from Theorem B.3.1, using the following additional observation:

$$\lim_{N\to\infty} \tilde{\mathcal{F}}^2 = \lim_{N\to\infty} \frac{\mathcal{N}(1)^2}{N M_f^2} = \lim_{N\to\infty} \frac{\sum_k |f(x_k)|^2 \Delta_N/(b-a)}{\max_k |f(x_k)|^2} = \mathcal{F}^2$$

where we used Lemma B.6.3. □

**Corollary B.3.2.** *[Asymptotic Integral Encoding] Again consider Theorem B.3.1 in the limit as $N \mapsto \infty$, this time using $g$ instead of $f$, with $g$ representing an integral encoding of the function as per Definition B.1.1. Then, the complexity of the procedure is given by*

$$O\left(\frac{\mathcal{F}^{-1}}{\epsilon^2}\right),$$

*with a probability of failure bounded by $O(\epsilon^2)$.*

*Proof.* We use Eq. (B.133) to find the maximum of the function

$$\lim_{N\to\infty} \|A\|_{max} = \lim_{N\to\infty} N \max_k g^2(x_k) = (b-a)\|f\|_{max}$$

and so finally

$$\lim_{N\to\infty} M_g/\sqrt{\|f\|_1} = \mathcal{F}^{-1/2}.$$

Consequently, $\epsilon \in O(\mathcal{F}^{-1})$. The rest of the proof follows immediately from Theorem B.3.1. □

□

Furthermore, we note that while the above asymptotic limits exist for any continuous functions, in the specific case of Lipschitz continuous functions (which cover nearly all scenarios in practice) these limits are approached rapidly in exponential order in $n$. For commonly encountered non-pathological functions, the asymptotic limit is usually reached with $n < 30$.

**Remark B.3.1.** *In general, while not necessary, QRAM can provide an asymptotic improvement in the circuit depth by alleviating the need for most of the quantum arithmetic involved in implementing the oracle (e.g. by storing $f_0$ and $f_1$ for all values in the grid, and then using basic addition and multiplication to compute $f_s$ as needed). Moreover, in certain QRAM architectures, (namely where each quantum register in the architecture has its own controlling classical mini-cpu and batch instructions can be issued to all the classical controllers in parallel) the cost of the oracle access can be made $O(1)$, thus rendering the algorithm's circuit complexity the same as its query complexity.*

### B.3.1 The Procedure

In the standard model of adiabatic quantum computation originally proposed by Farhi *et al.* in 2000 [96], one defines a time-dependent interpolated Hamiltonian $H(s) = (1 - s)H_0 + sH_1$ where $H_0$ is some initial Hamiltonian with an easy to prepare ground state, and $H_1$ is a final Hamiltonian whose ground state we wish to prepare. One then evolves the initial state according to $H(s)$ to obtain the final ground state of the problem. However, a challenge with this approach is that the spectral gap can become exponentially small in general (thus necessitating exponential evolution time).

However, in some cases this problem can be circumvented by viewing the paradigm of adiabatic quantum computation more generally – we need not limit ourselves to using a time-dependent Hamiltonian of the form $H(s) = (1-s)H_0+sH_1$. In our case, we begin at $n$ qubits (i.e. use a resolution of $N = 2^n$) and define a

time-dependent *rank-one* Hamiltonian. That is, rather than interpolating between two separate Hamiltonians, we define our Hamiltonian as proportional to a projector onto a discretized time-dependent quantum state the encodes a parametrised function. Precisely, we define

$$\mathcal{H}(s) := -A(s)/N \quad \begin{array}{r}\text{pointwise: } A(s) \text{ encodes } f_s \text{ as } f_s := (1-s)f_0 + sf_1. \\ \text{integral: } A(s) \text{ encodes } g_s \text{ as } g_s := (1-s)g_0 + sg_1.\end{array} \tag{B.57}$$

Here $A(s)$ is a rank-one matrix with entries $[A(s)]_{kl} := f_s(x_k)f_s^*(x_l)$ ($[A(s)]_{kl} := Ng_s(x_k)g_s(x_l)$). Furthermore, $f_0(x)$ and $g_0(x)$ are trivial functions, such as the uniform distribution, that we can easily prepare as our initial state while $f_1(x)$ and $g_1(x)$ are our final desired functions. This definition of $\mathcal{H}(s)$ has multiple nice properties enabling general analytic performance guarantees, which we now derive. As a matter of notation, we define the normalization constant $\mathcal{N}(s)$ as

$$\mathcal{N}(s) := \sqrt{\sum_{j=0}^{N-1} |f_s(x_j)|^2}, \tag{B.58}$$

for the interpolated function $f_s$ (and similarly for the integration case $g_s$) as per Eq. (B.57). $\mathcal{N}(1)$ is the main quantity we are concerned with in our proofs, and we derive all of our general bounds in terms of it (noting that as per Remark B.3.2 it grows as $O(\sqrt{N})$).

In the case of the pointwise state preparation approach, while it is not a limitation of the presented algorithm, it is important to keep in mind that we define our problem such that we aim to represent a continuous function over a finite grid – which is only meaningful if the grid is fine enough to not miss features of the function. While we prove asymptotic properties for an increasing resolution $N \to \infty$, the approach might fail in ill-defined cases when $N$ is too small. In particular, if the function $f_1$ consists of only narrow peaks of width $\epsilon$ then we need at least $n \in O[\log_2(\epsilon^{-1})]$ qubits to resolve these peaks and for qubit counts below this threshold our discretisation may in a worst-case scenario correspond to the null-vector (i.e., the function value is 0 at all grid points). A particular

example could be the function $f(x) = \sin(2^{20}\pi x)$ which requires at least 21 qubits of resolution in the range $0 \leq x \leq 1$ using our standard grid. It is important to stress that this is not a limitation of the algorithm being presented – when preparing a function according to point-wise samples no algorithm can handle such ill-defined cases. However, in practice this is unlikely to ever be a problem, because of the exponential growth of the grid in the number of qubits. For example, at 128 qubits, $N$ is sufficiently large to resolve a function defined on a kilometer interval with grid points at the Planck scale.

We now summarize the algorithm, along with the main proofs.

- We begin by preparing the state $|\psi_0\rangle = \frac{1}{\mathcal{N}(0)} \sum_{j=0}^{N-1} f_0(x_j)|j\rangle$, which by definition of $f_0$ is easy to prepare.

- We then implement adiabatic evolution for a total time $T$ according to the time-dependent rank-one Hamiltonian $\mathcal{H}(s)$ as defined in Lemma B.3.1.

- We prove that $T$ is constant bounded in Lemma B.3.2 (to see that this is indeed a constant bound, consider Remark B.3.2).

- The adiabatic evolution is implemented by the sequence of time-independent unitary transformations given by $e^{-i\frac{T}{r}\mathcal{H}(\frac{r}{r})} e^{-i\frac{T}{r}\mathcal{H}(\frac{r-1}{r})} \cdots e^{-i\frac{T}{r}\mathcal{H}(\frac{1}{r})}$.

- We prove a bound on the error incurred by this discretization of the adiabatic evolution in Lemma B.3.3.

- In Lemmas B.2.2 and B.3.4 we demonstrate how each of the time-independent terms $e^{-i\frac{T}{r}\mathcal{H}(\frac{j}{r})}$ may be implemented using the low-rank Hamiltonian simulation procedure, and bound the error and probability of failure incurred in the simulation.

- In Lemma B.2.3, we bound the cumulative probability of failure from implementing the low-rank Hamiltonian simulation technique $r$ times in a row (in $e^{-i\frac{T}{r}\mathcal{H}(\frac{r}{r})} e^{-i\frac{T}{r}\mathcal{H}(\frac{r-1}{r})} \cdots e^{-i\frac{T}{r}\mathcal{H}(\frac{1}{r})}$).

- In Lemma B.3.6 we bound the total error incurred in the procedure.

- While we present these bounds for an arbitrary, finite qubit count $n$ in Appendix B.3.2 without assuming continuity of the functions, we establish asymptotic limits in Appendix B.3.2 using properties of continuous functions.

- Finally, in Appendix B.3.2 we establish that our approach is efficient not only for continuous but also for arbitrary functions (mappings) that satisfy certain properties.

### B.3.2 Proofs for Finite Qubit Count $n$

As we noted above we are primarily concerned with the normalization factor from Eq. (B.58) when proving performance properties of the adiabatic approach. We note that we can efficiently estimate this normalization constant. I note that the easiest way to do this is to simply perform Grover black-box state preparation, and then perform amplitude estimation to obtain an estimate for the normalization factor. If classical pre-computation is allowed, then the normalization factor can just be computed to high-precision classically for univariate continuous functions.

**Remark B.3.2** (Efficiently computable normalization)**.** *As demonstrated in Appendix D of our paper [2] (cut from this thesis as it is primarily a contribution of my co-author), it is possible to efficiently compute the normalization factor for an arbitrary $f_1$, and thus to re-scale $f_1$ such that $\mathcal{N}(1) = \sqrt{N}$.*

The above property will ensure that our following upper bounds on the performance of adiabatic evolution are simplified and tight. In the following subsection we first prove that the adiabatic evolution needs to run only for a constant bounded time $T$ by bounding the delay factor. The main results are illustrated in Fig. B.2 (left). In in the second subsection we prove bounds on the algorithmic errors incurred by our procedure. The algorithmic error of our adiabatic evolution is illustrated in Fig. B.2 (right).

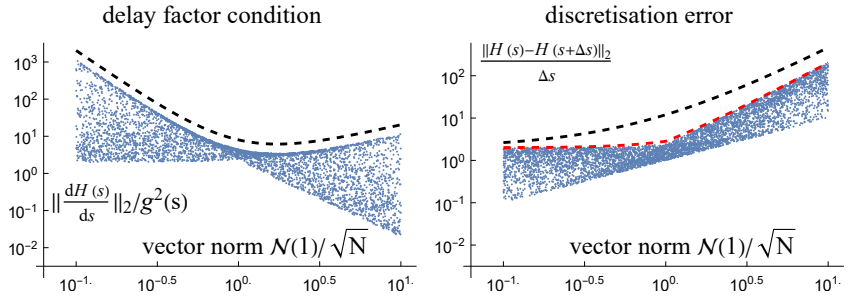


**Figure B.2:** Numerically verifying bounds that guarantee the efficient performance of adiabatic evolution from Appendix B.3.2. (left) The energy gap $g(s)$ of our rank-1 Hamiltonian $H(s) = -A(s)/N$ from Eq. (B.57) is determined by the normalization $\mathcal{N}(s)$ from Eq. (B.58) and is bounded via Lemma B.3.1. The matrix norm of the derivative $\|\frac{d}{ds}\mathcal{H}(s)\|_2/g^2(s)$ relative to this spectral gap determines the overall evolution time $T$ required for adiabatic evolution. Upper bound (dashed black line) from Lemma B.3.2. (right) The matrix distance $\|H(s) - H'(s)\|_2 \propto \Delta s$ determines the error when approximating a continuous adiabatic evolution via a piecewise constant evolution. We note that with similar proof techniques we could derive tighter asymptotic (in $\Delta s$, not presented in this work) upper bounds (dashed red line) but we prefer the upper bound (dashed black line) in Lemma B.3.3 due to its more compact expression. Given *all bounds* in Appendix B.3.2 are valid to arbitrary rank-1 matrices $A$ (i.e., we need only invoke continuity of functions when computing asymptotic limits in Appendix B.3.2), we simulated 5000 unitary Haar-random discretisation vectors $v_1$ at $n = 10$ qubits for uniformly randomly selected $0 \leq s \leq 1$ and logarithmically uniformly randomly selected vector norms $\mathcal{N}$. Please note that this figure (and caption) was produced by my co-author, and is included for completeness.

## Proofs on boundedness of the spectral gap

**Lemma B.3.1** (Spectral gap bound)**.** *We choose the phase of our trivial initial function properly via the efficient verification protocol in Appendix E of our paper [2] (cut from this thesis as it was primarily a contribution of my co-author) as one of the four possibilities* $f_0(x) = (\pm 1 \pm i)/\sqrt{2}$*. Given a parameterised rank-1 encoding* $A(s)$ *from Eq.* (B.57)*, the spectral gap of our Hamiltonian* $\mathcal{H}(s) = -A(s)/N =: |v_s\rangle\langle v_s|$ *is determined by the vector norm* $g(s) = |||v_s\rangle||_2^2$ *and is bounded as*

$$g(s) \geq \frac{\mathcal{N}(1)^2/N}{\mathcal{N}(1)^2/N + 1}.$$

*Proof.* First, observe that since $\mathcal{H}(s)$ is rank-one, its spectral gap is the same as its spectral norm, and thus

$$g(s) = \|H(s)\|_2 = \left\|\frac{A(s)}{N}\right\|_2 = \||v_s\rangle\langle v_s|\|_2 = \||v_s\rangle\|_2^2. \tag{B.59}$$

Moreover, the definition for $\mathcal{H}(s)$ in Equation B.57 is equivalent to the following,

$$|v_s\rangle = (1-s)|v_0\rangle + s|v_1\rangle, \tag{B.60}$$

where $|v_0\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_0(x_j)|j\rangle$ and $|v_1\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_1(x_j)|j\rangle$ (and thus $|v_0\rangle$, $|v_1\rangle$ and $|v_s\rangle$ are not normalized). In the integration case, simply replace each $f(x)$ with the appropriate integral of $f$ (as per Equation B.3). Then, using the Equation B.60,

$$g(s) = \||v_s\rangle\|_2^2 = (1-s)^2\langle v_0|v_0\rangle + s^2\langle v_1|v_1\rangle + 2s(1-s)\mathrm{Re}\langle v_0|v_1\rangle. \tag{B.61}$$

Noting that $\langle v_0|v_0\rangle = \frac{1}{N}\mathcal{N}(0)^2$, $\langle v_1|v_1\rangle = \frac{1}{N}\mathcal{N}(1)^2$, and that by selecting the appropriate phase and sign in $f_0(x) = (\pm 1 \pm i)/\sqrt{2}$ we ensure that $2\mathrm{Re}\langle v_0|v_1\rangle \geq 0$, we get

$$g(s) \geq (1-s)^2\frac{\mathcal{N}(0)^2}{N} + s^2\frac{\mathcal{N}(1)^2}{N}. \tag{B.62}$$

Noting that it is trivial to select an $f_0$ (or $g_0$) such that $\frac{\mathcal{N}(0)}{\sqrt{N}} = 1$, the above simplifies to

$$g(s) \geq (1-s)^2 + s^2\frac{\mathcal{N}(1)^2}{N}. \tag{B.63}$$

Minimizing the expression with respect to $s$, we find the minimum at $s = \frac{1}{\mathcal{N}(1)^2/N+1}$. Substituting this value for $s$, we then obtain the bound

$$g(s) \geq \frac{\mathcal{N}(1)^2}{\mathcal{N}(1)^2 + N}. \tag{B.64}$$

□

As an aside, $2\mathrm{Re}\langle v_0|v_1\rangle \geq 0$ only holds if we have properly chosen the phase of our initial trivial function $f_0(x) = (\pm 1 \pm i)/\sqrt{2}$. If we select the phase incorrectly, it is possible that $\mathrm{Re}\sum_j f_0(x_j)f_1(x_j) < 0$, and a negative term can decrease the spectral gap in Eq. (B.61) below our lower bound (which assumes a non-negative $\mathrm{Re}\sum_j f_0(x_j)f_1(x_j)$). Indeed for real functions we need only consider the sign $f_0(x) = \pm 1$ while for the integral encoding we trivially set $f_0(x) = 1$. Note that we can efficiently decide the sign (and phase) of the trivial function given we need only test the four possibilities for a phase and we can efficiently verify the result

of our state-preparation procedure via the destructive interference circuit given by my co-author in Appendix E of our paper [2] (which has been cut from this thesis, as it was primarily his contribution).

**Corollary B.3.3.** *By Lemma B.3.1, if we re-scale $f_1$ as per Remark B.3.2, the bound on the spectral gap of the Hamiltonian may be written as,*

$$g(s) \geq \frac{1}{2},$$

*which is indeed a constant.*

**Lemma B.3.2** (Delay factor bound)**.** *Given the Hamiltonian $\mathcal{H}(s)$ as defined in Lemma B.3.1, we have the bound*

$$\frac{\left\|\frac{d}{ds}\mathcal{H}(s)\right\|_2}{g(s)^2} \leq 2\frac{\left(\mathcal{N}(1)^2/N+1\right)^2}{\left(\mathcal{N}(1)/\sqrt{N}\right)^3}. \tag{B.65}$$

*This implies that the total adiabatic evolution time required for the procedure is asymptotically given by $T \in O\left(\frac{\left(\mathcal{N}(1)^2/N+1\right)^2}{\left(\mathcal{N}(1)/\sqrt{N}\right)^3}\right)$.*

*Proof.* We begin by providing an upper-bound for $\left\|\frac{d}{ds}\mathcal{H}(s)\right\|_2$.

$$\left\|\frac{d}{ds}\mathcal{H}(s)\right\|_2 = \left\|\left(\frac{d}{ds}|v_s\rangle\right)\langle v_s| + |v_s\rangle\left(\frac{d}{ds}\langle v_s|\right)\right\|_2 = \left\|\left(|v_1\rangle - |v_0\rangle\right)\langle v_s| + |v_s\rangle\left(|v_1\rangle - |v_0\rangle\right)\right\|_2. \tag{B.66}$$

For ease of notation, allow $|v'\rangle = |v_1\rangle - |v_0\rangle$. Using the triangle inequality,

$$\||v'\rangle\langle v_s| + |v_s\rangle\langle v'|\|_2 \leq \||v'\rangle\langle v_s|\|_2 + \||v_s\rangle\langle v'|\|_2 = 2\,\||v'\rangle\langle v_s|\|_2. \tag{B.67}$$

Suppose we have two normalized vectors $|a'\rangle = \frac{|a\rangle}{\sqrt{\langle a|a\rangle}}$ and $|b'\rangle = \frac{|b\rangle}{\sqrt{\langle b|b\rangle}}$. Then,

$$\||a\rangle\langle b|\|_2 = \sqrt{\langle a|a\rangle}\sqrt{\langle b|b\rangle}\||a'\rangle\langle b'|\|_2 \leq \||a\rangle\|_2\||b\rangle\|_2, \tag{B.68}$$

where the inequality follows from $\||a'\rangle\langle b'|\|_2 \leq 1$ for any normalized vectors $|a'\rangle$ and $|b'\rangle$. Applying this to Equation B.67 yields,

$$\left\|\frac{d}{ds}\mathcal{H}(s)\right\|_2 \leq 2\||v'\rangle\|_2\||v_s\rangle\|_2. \tag{B.69}$$

Of course,

$$\||v'\rangle\|_2^2 = \langle v'|v'\rangle = \langle v_0|v_0\rangle + \langle v_1|v_1\rangle - 2\mathrm{Re}\langle v_0|v_1\rangle \leq \||v_0\rangle\|_2^2 + \||v_1\rangle\|_2^2, \tag{B.70}$$

where the inequality follows from the fact that $\mathrm{Re}\langle v_0|v_1\rangle \geq 0$ given we have properly chose the phase of the trivial function. Thus,

$$\||v'\rangle\|_2 \leq \sqrt{\frac{\mathcal{N}(0)^2}{N} + \frac{\mathcal{N}(1)^2}{N}} = \sqrt{1 + \frac{\mathcal{N}(1)^2}{N}}, \tag{B.71}$$

where we again used the fact that its trivial to select an $f_0$ (or $g_0$) such that $\mathcal{N}(0) = \sqrt{N}$. Equation B.69 then becomes,

$$\left\| \frac{d}{ds}\mathcal{H}(s) \right\|_2 \leq 2\sqrt{1 + \frac{\mathcal{N}(1)^2}{N}} \||v_s\rangle\|_2. \tag{B.72}$$

Noting that $g(s) = \||v_s\rangle\|_2^2$, we can then write

$$\frac{\left\| \frac{d}{ds}\mathcal{H}(s) \right\|_2}{g(s)^2} \leq \frac{2\sqrt{1 + \frac{\mathcal{N}(1)^2}{N}}}{\||v_s\rangle\|_2^3}. \tag{B.73}$$

From Lemma B.3.1, we directly obtain,

$$\||v_s\rangle\|_2^3 \geq \frac{\mathcal{N}(1)^3}{(\mathcal{N}(1)^2 + N)^{3/2}}. \tag{B.74}$$

Thus,

$$\frac{\left\| \frac{d}{ds}\mathcal{H}(s) \right\|_2}{g(s)^2} \leq 2\frac{\sqrt{N + \mathcal{N}(1)^2}}{\sqrt{N}} \frac{(\mathcal{N}(1)^2 + N)^{3/2}}{\mathcal{N}(1)^3} = 2\frac{(\mathcal{N}(1)^2 + N)^2}{\sqrt{N}\mathcal{N}(1)^3}. \tag{B.75}$$

As discussed in Section B.2.4, a sufficiently slow Hamiltonian evolution schedule is given by

$$\tau(s) \gg \frac{\left\| \frac{d}{ds}\mathcal{H}(s) \right\|_2}{g(s)^2}. \tag{B.76}$$

Thus, $T = \int_0^1 \tau(s)ds$, and so asymptotically we also have the bound $T \in O\left(2\frac{\left(\mathcal{N}(1)^2+N\right)^2}{\sqrt{N}\mathcal{N}(1)^3}\right)$. □

**Corollary B.3.4.** *By Lemma B.3.2, if we re-scale $f_1$ as per Remark B.3.2, the bound on the delay factor of $\mathcal{H}(s)$ is given by,*

$$\frac{\left\| \frac{d}{ds}\mathcal{H}(s) \right\|_2}{g(s)^2} \leq 8.$$

*Thus, the total adiabatic evolution time required by the procedure is asymptotically constant, i.e. $T \in O(1)$.*

**Time Evolution Error Bounds**

**Lemma B.3.3** (Adiabatic discretization error bound)**.** *Given the parameterized Hamiltonian* $\mathcal{H}(s)$ *from Eq.* (B.57) *and its discretised approximation* $\mathcal{H}'(s) = \mathcal{H}(\lceil sr \rceil / r)$ *with a resolution* $r \in \mathbb{N}$*, the distance between the approximate and exact Hamiltonian is bounded as (in terms of the spectral norm)*

$$\|\mathcal{H}(s) - \mathcal{H}'(s)\|_2 \leq \delta_0 \equiv \frac{2}{r}\left(1 + 3\frac{\mathcal{N}(1)}{\sqrt{N}} + 2\frac{\mathcal{N}(1)^2}{N}\right).$$

*This may be equivalently written as,*

$$\|\mathcal{H}(s) - \mathcal{H}'(s)\|_2 \leq \frac{2}{r}\left(1 + 3\left\|A(1)/N\right\|_2^{1/2} + 2\left\|A(1)/N\right\|_2\right).$$

*Proof.* First, observe that the deviation between the exact and approximate Hamiltonians is maximized when $s = \frac{j+1}{r}$ (for $j \in \mathbb{Z}$ and $1 \leq j \leq r$), as $\mathcal{H}'(\frac{j+1}{r}) = \mathcal{H}(\frac{j}{r})$. Thus,

$$\|\mathcal{H}'(s) - \mathcal{H}(s)\|_2 \leq \left\|\mathcal{H}\left(\frac{j}{r}\right) - \mathcal{H}\left(\frac{j+1}{r}\right)\right\|_2. \tag{B.77}$$

For notational brevity, allow $X_0 = |v_0\rangle\langle v_0|$, $X_1 = |v_0\rangle\langle v_1| + |v_1\rangle\langle v_0|$, and $X_2 = |v_1\rangle\langle v_1|$. Then,

$$\mathcal{H}(s) = |v_s\rangle\langle v_s| = (1-s)^2 X_0 + s(1-s)X_1 + s^2 X_2. \tag{B.78}$$

Allowing $s = \frac{j+1}{r}$ and $\gamma = \frac{j}{r}$, simple computer algebra shows

$$\|\mathcal{H}(\gamma) - \mathcal{H}(s)\|_2 = \left\|\left(\frac{1+2j-2r}{r^2}\right)(X_0 - X_1 + X_2) + \frac{1}{r}(X_1 + 2X_2)\right\|_2. \tag{B.79}$$

Applying the triangle inequality, and the fact that $|\frac{1+2j-2r}{r^2}| \leq \frac{2}{r} - \frac{1}{r^2}$ (since $r \geq 1$), we get

$$\|\mathcal{H}'(s) - \mathcal{H}(s)\|_2 \leq \frac{1}{r}\left(2\|X_0\|_2 + 3\|X_1\|_2 + 4\|X_2\|_2\right). \tag{B.80}$$

Noting that $\|X_0\|_2 = 1$ , $\|X_1\|_2 \leq 2\frac{\mathcal{N}(1)}{\sqrt{N}}$, $\|X_2\|_2 = \frac{\mathcal{N}(1)^2}{N}$, we obtain

$$\|\mathcal{H}'(s) - \mathcal{H}(s)\|_2 \leq \frac{2}{r}\left(1 + 3\frac{\mathcal{N}(1)}{\sqrt{N}} + 2\frac{\mathcal{N}(1)^2}{N}\right). \tag{B.81}$$

Finally, using $\|A(1)/N\|_2 = \mathcal{N}(1)^2/N$, we obtain

$$\|\mathcal{H}'(s) - \mathcal{H}(s)\|_2 \leq \frac{2}{r}\left(1 + 3\|A(1)/N\|_2^{1/2} + 2\|A(1)/N\|_2\right). \tag{B.82}$$

Since all of the terms quadratic in $r$ are negative, we can then directly drop them to obtain the simplified expression for the upper-bound presented in the lemma. □

**Corollary B.3.5.** *By Lemma B.3.3, if we re-scale $f_1$ as per Remark B.3.2, the bound on the maximum deviation of the discretized and exact Hamiltonian is given by,*

$$\|\mathcal{H}(s) - \mathcal{H}'(s)\|_2 \leq \frac{12}{r}.$$

So far, we have shown that the adiabatic evolution need only run for a constant period of simulation time $T$ (with the constant factor depending on properties of the function), and that the discretization of the adiabatic evolution may be performed with arbitrarily small error. We now show that the simulation of each $e^{-\mathcal{H}(s)\frac{T}{r}}$ term may also be performed efficiently (and with bounded error) and thus that the overall algorithm is efficient.

**Lemma B.3.4** (Piecewise simulation error)**.** *For a fixed $s$, the algorithmic simulation error from using Hamiltonian simulation technique summarized in Lemma B.2.2 to simulate $\mathcal{H}(s)$ for a time $\frac{T}{r}$ is bounded by,*

$$\epsilon_0(s) \in O\left(\frac{1}{r^2}\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2}\frac{\left(\|A(1)/N\|_2 + 1\right)^5}{\|A(1)/N\|_2^c}\right),$$

*where $c = 2$ if $\|A(1)/N\|_2 \geq 1$ and $c = 3$ if $\|A(1)/N\|_2 < 1$.*

*Proof.* Using the low-rank Hamiltonian simulation technique of ref [82], which we improve for the case of rank-one Hamiltonians in Lemma B.2.2, the simulation of a rank-one projector $A$ for a time $\Delta t$ at normalized time $s$ has error bounded by

$$\epsilon_0(s) \leq \frac{3}{2}\Delta t^2 \frac{\mathcal{N}(s)^2}{N}\|A(s)\|_{max}. \tag{B.83}$$

In particular, implementing the evolution on a starting state $|\psi\rangle$ yields an error bounded by, $|\psi'\rangle \propto e^{-i\Delta t A(s)/N}|\psi\rangle + |\xi\rangle$ with $\||\xi\rangle\|_2 \leq \epsilon_0(s)$. Moreover, this approach

assumes that measurements are performed on the ancillary register and that only $|+^n\rangle$ measurement outcomes are accepted. We handle the discussion of the success probability in a separate lemma.

In the direct-state preparation algorithm, we must evolve for a total time $T$, which is constant bounded, as shown in Lemma B.3.2. We do so in $r$ evenly spaced intervals, resulting in $\Delta t = \frac{T}{r}$. As a result,

$$\epsilon_0(s) \leq \tfrac{3}{2}\tfrac{\mathcal{N}(s)^2}{N}\|A(s)\|_{max}\left(\frac{T}{r}\right)^2. \tag{B.84}$$

Using Lemma B.3.2, we have an upper-bound on the delay factor, and thus an *asymptotic* upper-bound on $T$. To allow treatment as a strict upper-bound, we introduce the constant $k$, such that $T \leq 2k\frac{\left(\mathcal{N}(1)^2/N+1\right)^2}{(\mathcal{N}(1)/\sqrt{N})^3}$ (in practice, we find that $k \approx 100$). Using this value, we get

$$\epsilon_0(s) \leq 6\tfrac{\mathcal{N}(s)^2}{N}\|A(s)\|_{max}\frac{k^2}{r^2}\frac{\left(\mathcal{N}(1)^2/N+1\right)^4}{(\mathcal{N}(1)^2/N)^3}. \tag{B.85}$$

Now, we upper-bound $\frac{\mathcal{N}(s)^2}{N}$ in terms of $\frac{\mathcal{N}(1)^2}{N}$,

$$\begin{aligned}
\|A(s)/N\|_2 &\leq (1-s)^2\langle v_0|v_0\rangle + s^2\langle v_1|v_1\rangle + 2s(1-s)\mathrm{Re}\langle v_0|v_1\rangle && \text{(B.86)}\\
&\leq (1-s)^2 + s^2\frac{\mathcal{N}(1)^2}{N} + 2s(1-s)\frac{\mathcal{N}(1)}{\sqrt{N}} && \text{(B.87)}\\
&= \left((1-s) + s\frac{\mathcal{N}(1)}{\sqrt{N}}\right)^2 && \text{(B.88)}\\
&\leq \max\left(1, \frac{\mathcal{N}(1)^2}{N}\right) && \text{(B.89)}\\
&\leq 1 + \frac{\mathcal{N}(1)^2}{N}, && \text{(B.90)}
\end{aligned}$$

where the second inequality follows from the application of the Cauchy-Schwarz inequality on $\langle v_0|v_1\rangle$. Additionally, it is trivial to see that $\|A(s)\|_{max} \leq 1+\|A(1)\|_{max}$ (since $\|A(0)\|_{max} = 1$ by construction). Applying these two upper-bounds, we get

$$\epsilon_0(s) \leq 6\frac{k^2}{r^2}\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2}\frac{\left(\|A(1)/N\|_2+1\right)^5}{\|A(1)/N\|_2^2} + 6\frac{k^2}{r^2}\frac{\left(\|A(1)/N\|_2+1\right)^5}{\|A(1)/N\|_2^3}, \tag{B.91}$$

where we additionally used the fact that $\|A(1)/N\|_2 = \mathcal{N}(1)^2/N$. Finally, we observe that $\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2} \geq 1$, yielding

$$\epsilon_0(s) \leq 12\frac{k^2}{r^2}\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2}\frac{\left(\|A(1)/N\|_2+1\right)^5}{\|A(1)/N\|_2^c}, \tag{B.92}$$

where $c = 2$ if $\|A(1)/N\|_2 \geq 1$ and $c = 3$ if $\|A(1)/N\|_2 < 1$. Thus, asymptotically our error is bounded by $\epsilon_0(s) \in O\left(\frac{1}{r^2}\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2}\frac{\left(\|A(1)/N\|_2+1\right)^5}{\|A(1)/N\|_2^c}\right)$. □

**Lemma B.3.5.** *The query complexity of simulating $e^{-i\mathcal{H}(s)\Delta t}$ as per Lemmas B.2.2 and B.3.4 is simply 4, i.e. is bounded by $O(1)$, while the total number of qubits required are $2n + d + 2$, and thus the algorithm requires $O(n + d)$ ancillary qubits. Therefore, the total state preparation algorithm has query complexity $O(r)$.*

*Proof.* As discussed at length, the simulation of $e^{-i\mathcal{H}(s)\Delta t}$ is implemented using the low-rank Hamiltonian simulation technique of ref [82], where they embed the $N \times N$ matrix $A_s \equiv \mathcal{H}(s)$ into the one-sparse $N^2 \times N^2$ matrix $S_{A_s}$. The one-sparse matrix can then be simulated as described in Lemma B.2.1 using a total of 4 calls to the oracles $O_f$ and $O_H$, and with a total of $2n + d + 2$ qubits (i.e. $n + d + 2$ ancillary qubits). As the algorithm consists of performing $r$ consecutive such Hamiltonian simulations, the total query complexity is given by $O(r)$. □

**Remark B.3.3.** *In the following we assume the function $f_1$ is rescaled such that $\mathcal{N}(1) = \sqrt{N}$. The normalization factor $\mathcal{N}(1)$ can be approximated efficiently as per Remark B.3.2 and we assume the estimation error is negligible.*

**Lemma B.3.6.** *The total error of the direct state preparation procedure, as measured by the deviation of the exact unitary evolution $U(T)$ from the worst-case unitary evolution we implement $U''(T)$, is given by*

$$\|U(T) - U''(T)\|_2 \in O\left(\sqrt{\frac{\|A(1)\|_{max}}{r}}\right), \tag{B.93}$$

*assuming that $A(1)$ corresponds to a function satisfying $\mathcal{N}(1)/\sqrt{N} = 1$.*

*Proof.* First, we allow $\mathcal{H}(s)$ to represent the exact time-dependent Hamiltonian we wish to simulate, $\mathcal{H}'(s)$ the discretized Hamiltonian as per Lemma B.3.3, and $\mathcal{H}''(s)$ the discretized Hamiltonian which induces the unitary with algorithmic error obtained via the low-rank simulation technique as per Lemmas B.2.2 and B.3.4. $\mathcal{H}(s)$ and $\mathcal{H}'(s)$ have both already been clearly defined, however, we have never explicitly considered $\mathcal{H}''(s)$ – instead we have directly treated with the unitary

evolution it induces and through this unitary mapping we can define $\mathcal{H}''(s)$ as its generator.

As shown in Lemma B.2.2 the operator deviation is bounded by,

$$\|U'(\Delta t) - U''(\Delta t)\|_2 \leq 3/2 \frac{\mathcal{N}^2(s)}{N} \|A(s)\|_{max} \Delta t^2. \tag{B.94}$$

First, we bound the $s$-dependent quantities at the extremes where $s = 0, 1$. Observing that $\mathcal{N}^2(s)/N = \langle v_s | v_s \rangle$, we get

$$\mathcal{N}^2(s)/N = (1-s)^2 \langle v_0 | v_0 \rangle + s^2 \langle v_1 | v_1 \rangle + 2s(1-s) \mathrm{Re}(\langle v_0 | v_1 \rangle). \tag{B.95}$$

Of course, $\mathrm{Re}(\langle v_0 | v_1 \rangle) \leq \langle v_0 | v_0 \rangle + \langle v_1 | v_1 \rangle$, so we can upper-bound the preceding expression with,

$$\mathcal{N}^2(s)/N = 2 \left( \frac{\mathcal{N}^2(0)}{N} + \frac{\mathcal{N}^2(1)}{N} \right) \leq 4, \tag{B.96}$$

where the final inequality follows from the fact that $\frac{\mathcal{N}^2(0)}{N} = 1$ by construction, and that $\frac{\mathcal{N}^2(1)}{N} = 1$ by assumption. Moreover, we can follow a similar argument to upper-bound $\|A(s)\|_{max}$. By construction, $A(s) = N |v_s\rangle\langle v_s|$. Consequently,

$$\|A(s)\|_{max} \leq N \left[ (1-s)^2 \| |v_0\rangle \|_{max}^2 + s^2 \| |v_1\rangle \|_{max}^2 + 2s(1-s) \| |v_0\rangle\langle v_1| \|_{max}^2 \right] \tag{B.97}$$

$$\leq N \left[ \frac{1}{N} \|A(0)\|_{max} + \frac{1}{N} \|A(1)\|_{max} + \| |v_0\rangle\langle v_1| \|_{max}^2 \right]. \tag{B.98}$$

Using the fact that $\| |v_0\rangle\langle v_1| \|_{max}^2 \leq \| |v_0\rangle \|_{max}^2 + \| |v_1\rangle \|_{max}^2$, we can then obtain the upper-bound,

$$\|A(s)\|_{max} \leq 2 \left[ \|A(0)\|_{max} + \|A(1)\|_{max} \right]. \tag{B.99}$$

Of course, $\|A(0)\|_{max} = 1$, yielding

$$\|A(s)\|_{max} \leq 2 \left( 1 + \|A(1)\|_{max} \right). \tag{B.100}$$

Moreover, $\frac{\mathcal{N}^2(1)}{N} = 1$ implies that $\|A(1)\|_{max} \geq 1$, and consequently we get the final bound,

$$\|A(s)\|_{max} \leq 2 \|A(1)\|_{max} \tag{B.101}$$

Thus, we get the expression

$$\|U'(\Delta t) - U''(\Delta t)\|_2 \leq 24\|A(1)\|_{max}\Delta t^2. \tag{B.102}$$

Using Corollary B.2.1 by observing that $\kappa = 24\|A(1)\|_{max}$ and that $\Delta t = \frac{T}{r}$, we get

$$\|\mathcal{H}'(s) - \mathcal{H}''(s)\|_2 \leq 24\|A(1)\|_{max}\frac{T}{r}. \tag{B.103}$$

Of course, by Corollary B.3.4, $T \leq 8k$, where $k$ is some constant ensuring that the delay factor bound rigorously upper-bounds the total adiabatic evolution time. As a consequence,

$$\|\mathcal{H}'(s) - \mathcal{H}''(s)\|_2 \leq \frac{192k}{r}\|A(1)\|_{max}. \tag{B.104}$$

Moreover, from Lemma B.3.3,

$$\|\mathcal{H}(s) - \mathcal{H}'(s)\|_2 \leq \frac{2}{r}\left(1 + 3\frac{\mathcal{N}(1)}{\sqrt{N}} + 2\frac{\mathcal{N}(1)^2}{N}\right) = \delta_0. \tag{B.105}$$

Due to rescaling as per Remark B.3.3 via Lemma B.3.3 we have

$$\|\mathcal{H}(s) - \mathcal{H}'(s)\|_2 \leq \frac{12}{r}.$$

As a result, Lemma B.2.5 gives the bound

$$\|\mathcal{H}(s) - \mathcal{H}''(s)\|_2 \leq \delta_0 + \delta_1. \tag{B.106}$$

Thus, Lemma B.2.4 gives,

$$\|U(T) - U''(T)\|_2 \leq \sqrt{2T(\delta_0 + \delta_1)}. \tag{B.107}$$

Again using the fact that $T \leq 8k$, and that without loss of generality we can assume $k \geq 1$, and observing once more that $\frac{\mathcal{N}^2(1)}{N} = 1$ implies that $\|A(1)\|_{max} \geq 1$, we obtain

$$\|U(T) - U''(T)\|_2 \leq \sqrt{16k\left(\frac{12}{r} + \frac{192k}{r}\|A(1)\|_{max}\right)} \tag{B.108}$$

$$\leq \sqrt{16k\left(\frac{12k\|A(1)\|_{max}}{r} + \frac{192k}{r}\|A(1)\|_{max}\right)} \tag{B.109}$$

$$= \sqrt{\frac{3264k^2\|A(1)\|_{max}}{r}} \tag{B.110}$$

Thus, in asymptotic terms, we find

$$\|U(T) - U''(T)\|_2 \in O\left(\sqrt{\frac{\|A(1)\|_{max}}{r}}\right) \tag{B.111}$$

□

**Asymptotic Limits**

Above we have derived our bounds in terms of two quantities as the matrix norms $\|A(1)/N\|_2 = \mathcal{N}(1)^2/N$ and $\|A(1)\|_{max}$. We prove in Appendix B.6 that both these quantities naturally admit asymptotic limits for $N \to \infty$ for any continuous function. Furthermore, we also prove that Lipschitz continuous functions—which cover nearly all instances in practice—approach these limits in exponential order in the number of qubits $n$. We apply these results and establish asymptotic limits of our error bounds.

**Remark B.3.4** (Growth rate of normalization)**.** *For any fixed continuous function $f_1$ we obtain the scaling for our pointswise encoding $\mathcal{N}(1) \in O(\sqrt{N})$ in the asymptotic limit of big $N$. More generally we can state $\mathcal{N}(1)/\sqrt{N} \in \Theta(1)$, which immediately follows from the fact that the normalization constant asymptotically approaches the Riemann integral $\mathcal{N}(1)^2/N \propto \|f_1\|_2^2$ as per Lemma B.6.1, Lemma B.6.3 and Lemma B.6.4.*

Intuitively, each $\epsilon$ neighbourhood around any $f_1(x)$ is asymptotically constant and so the summation for $\mathcal{N}(1)$ is a sum of $O(N)$ constants and therefore taking the square root results in a quantity which clearly grows as $O(\sqrt{N})$. As a consequence of this asymptotic limit, we are guaranteed that the total adiabatic evolution time from Lemma B.3.2 is $T \in \Theta(1)$, which property we exploit in the following.

**Corollary B.3.6** (Asymptotic simulation error)**.** *By Lemma B.3.4, if we re-scale $f_1$ as per Remark B.3.3, the algorithmic simulation error from simulating $\mathcal{H}(s)$ for time $T/r$ using the low-rank Hamiltonian simulation technique as per Statement B.2.1 is bounded by $\epsilon_0(s) \in O(\|A(1)\|_{max}/r^2)$. In the limit of big $N$, this may equivalently be written in terms of the filling ratio $\mathcal{F}$ from Definition B.1.2 as*

$$\epsilon_0(s) \in O\left(\frac{\mathcal{F}^p}{r^2}\right)$$

*where $p = -2$ in the point-wise sampling case, and $p = -1$ in the integration case via Lemma B.6.3 and Lemma B.6.4.*

**Lemma B.3.7** (Total success probability)**.** *The success probability in Lemma B.2.3 admits the asymptotic limit*

$$\text{prob}_{tot} \geq 1 - O(\mathcal{F}^p/r),$$

*where $p = -2$ in the point-wise sampling case, and $p = -1$ in the integration case via Lemma B.6.3 and Lemma B.6.4.*

*Proof.* We obtain the asymptotic bounds by substituting limits from Lemma B.6.3 and Lemma B.6.4 into our bounds in Lemma B.2.3. □

**Corollary B.3.7** (Total simulation error)**.** *By Lemma B.3.6 and Remark B.3.3, the total error of the direct state preparation procedure, as measured by the deviation of the exact unitary evolution $U(T)$ from the worst-case unitary evolution we implement $U''(T)$, is given by,*

$$\|U(T) - U''(T)\|_2 \in O\left(\sqrt{\frac{\mathcal{F}^p}{r}}\right), \tag{B.112}$$

*in the limit of big $N$ via Lemma B.6.3 and Lemma B.6.4 where $p = -2$ in the point-wise sampling case, and $p = -1$ in the integration case.*

*Proof.* Taking the limit of the bound in Lemma B.3.6 for big $N$, assuming a normalized $f_1$ (although this assumption is not necessary), $T \in O(1)$, $\delta_0 \in \frac{1}{r}$, $\epsilon_0(s) \in O(\frac{\mathcal{F}^p}{r^2})$, we get the asymptotic upper bound

$$\|U(T) - U''(T)\|_2 \in O\left(\sqrt{\frac{1 + \mathcal{F}^p}{r}}\right). \tag{B.113}$$

Noting that $\mathcal{F}^p \geq 1$, $1 + \mathcal{F}^p \leq 2\mathcal{F}^p$, and so our asymptotic statement simplifies to,

$$\|U(T) - U''(T)\|_2 \in O\left(\sqrt{\frac{\mathcal{F}^p}{r}}\right). \tag{B.114}$$

It is worth explicitly noting that this asymptotic bound also holds when $f_1$ is not normalized as in Remark B.3.3. If $f_1$ is not normalized, the ratio $\mathcal{N}(1)^2/N$ is

still asymptotically $O(1)$, however it may be a large constant, depending on how "unnormalized" $f_1$ is. As a result, this will be off by at most a constant factor, and so the asymptotic bound remains unchanged. □

**Remark B.3.5** (When adiabatic error is negligible)**.** *In practical applications (especially for small filling ratios $\mathcal{F}$) the total error is not dominated by the error of the discretised adiabatic evolution but rather by the low-rank simulation from Lemma B.3.4 for which we incur an error at each small timestep $\Delta t$ asymptotically via Corollary B.3.6 as $O(\mathcal{F}^p/r^2)$. Thus neglecting the adiabatic simulation error and applying the triangle inequality $r$ times consecutively (given $\epsilon_0(s)$ is a bound on the unitary operator distance) we obtain the bound*

$$\|U(T) - U''(T)\|_2 \lessapprox O\left(\frac{\mathcal{F}^p}{r}\right),$$

*where $p = -2$ in the point-wise sampling case, and $p = -1$ in the integration case. This scaling is tighter in $r$ but looser in the filling ratio $\mathcal{F}$ than in Corollary B.3.7.*

### Generalization to Arbitrary Functions

So far we have considered continuous functions $f_s(x)$ and discretised them to obtain our desired normalised quantum state from Eq. (B.3). We proved in Appendix B.3.2 that for any finite qubit count our approach allows us to prepare our final desired state with complexity that depends only on the two matrix norms $\|A(1)/N\|_2 = \mathcal{N}(1)^2/N$ and $\|A(1)\|_{max}$, in fact on the ratio of these two. We did not actually require the function to be continuous. In fact we invoked continuity because it guarantees that asymptotically in $N$ the matrix norm $\|A(1)\|_{max} \to \max_x |f(x)|^2$ via the boundedness theorem and that $\|A/N\|_2 \to (b-a)\|f\|_2^2$ due to Riemann integrability (as we prove in Lemma B.6.3). For this reason we do not actually need to necessary assume continuity of the function.

In fact our proofs in Appendix B.3.2 are applicable to the case of *any discrete mapping* as $F^{(n)} : \{0,1\}^n \mapsto \mathbb{C}$ in the sense that we can efficiently prepare *any* $n$-qubit quantum state of the form

$$|\psi\rangle_n = \mathcal{N}^{-1} \sum_{j \in \{0,1\}^n} F^{(n)}(j)|j\rangle_n, \tag{B.115}$$

given the following conditions are satisfied.

**Theorem B.3.2.** *Given an arbitrary efficiently computable mapping* $F^{(n)} : \{0,1\}^n \mapsto \mathbb{C}$ *that depends on the qubit count* $n$ *with absolute maximum value* $\|F^{(n)}\|_{max} = \max_{k\in\{0,1\}^n} |F^{(n)}(k)|$ *and vector norm* $\mathcal{N}^2 = \sum_{k\in\{0,1\}^n} |F^{(n)}(k)|^2$ *we can prepare the quantum state in Eq.* (B.115) *efficiently using our adiabatic approach given that* $\|F^{(n)}\|_{max} \in O(1)$ *and that* $\mathcal{N}/\sqrt{N} \in \Theta(1)$*. In direct analogy with Theorem B.3.1 the query complexity then depends on the generalised filling ratio* $r \in O(\tilde{\mathcal{F}}^{-2}/\epsilon^2)$*.*

*Proof.* We define the pointwise encoded matrix $[A(s)]_{kl} := F_s(k)F_s^*(l)$ following Eq. (B.57) with the parametric mapping $F_s := (1-s)F_0 + sF_1$ where $F_0$ is the same trivial function with a properly chosen phase as in Appendix B.3.2 and $F_1$ is our desired mapping $F_1 := F^{(n)}$ where we will drop the dependence on $n$ for ease of notation. We proved in Lemma B.3.2 that the total adiabatic evolution time is bounded as $T \in O\left(\frac{\left(\mathcal{N}(1)^2/N+1\right)^2}{\left(\mathcal{N}(1)/\sqrt{N}\right)^3}\right)$ and in Lemma B.3.4 that the simulation error is bounded as

$$\epsilon_0(s) \leq O\left(\frac{1}{r^2}\frac{\|A(1)\|_{max}}{\|A(1)/N\|_2}\frac{\left(\|A(1)/N\|_2+1\right)^5}{\|A(1)/N\|_2^c}\right), \tag{B.116}$$

for any finite qubit count, where $c = 2$ if $\|A(1)/N\|_2 \geq 1$ and $c = 3$ if $\|A(1)/N\|_2 < 1$. As such, given the matrix norms $\|A(1)/N\|_2 = \mathcal{N}^2(1)/N$ and $\|A(1)\|_{max} = \|F_1\|_{max}$ are asymptotically in $n$ both lower and upper bounded as $\mathcal{N}(1)/\sqrt{N} \in \Theta(1)$ and upper bounded $\|F_1\|_{max} \in O(1)$ we can guarantee a bounded total evolution time $T \in O(1)$ and that the simulation error is bounded as $\epsilon_0 \in O(1/r^2)$. Indeed in the special case when $F(j) = f(x_j)$, $F(j)$ is just a discretisation of a continuous function, and the norms are convergent for $n \to \infty$ as $\|A(1)\|_{max} \to \max_x |f(x)|^2$ is approached via the boundedness theorem and that $\mathcal{N}(1)/\sqrt{N} \to \sqrt{(b-a)}\|f\|_2$ due to Riemann integrability as we prove in Lemma B.6.3. In contrast, our more general asymptotic conditions do not require convergence and it suffices that there exist constants such that the $n$-dependent norms $\mathcal{N}(1)/\sqrt{N}$ and $\|F_1\|_{max}$ are asymptotically bounded. □

We will now provide two examples.

**Example:** Consider a mapping $F^{(n)} : \{0,1\}^n \mapsto [-1,1]$ that takes a binary integer $j \in \{0,1\}^n$ and uses it as a seed to efficiently compute a sample $F^{(n)}(j) = x_j$ from a random distribution $X$ such that the mean is $\mu(X) = 0$. By definition this mapping is bounded $\|F^{(n)}\|_{max} \leq 1$ and we can see that the norm converges to the asymptotic constant

$$\lim_{n\to\infty} \mathcal{N}^2/N = \lim_{n\to\infty} \frac{1}{N} \sum_{k\in\{0,1\}^n} x_k^2 = \mathrm{Var}[X].$$

As such, we can efficiently prepare an exponentially large list of $N$ random numbers of zero mean with complexity determined by the variance via the generalised filling ratio $\tilde{\mathcal{F}} = \mathrm{Var}[X]$.

**Counterexample:** Consider the unstructured search problem, where we have a function $F^{(n)} : \{0,1\}^n \mapsto \{0,1\}$ such that there is only one marked input $m$ for which $F^{(n)}(m) = 1$ and for all other inputs $F^{(n)}(j) = 0$ with $j \neq m$. By definition this mapping is bounded $\|F^{(n)}\|_{max} = 1$ and we can easily evaluate the normalization as

$$\mathcal{N}^2/N = \frac{1}{N} \sum_{k\in\{0,1\}^n} |F^{(n)}(k)|^2 = \frac{1}{N}.$$

Indeed we find the generalised filling ratio $\tilde{\mathcal{F}} = 2^{-n/2}$ and thus the query complexity of preparing this mapping grows exponentially as $O(2^n)$. Consequently, our approach does not violate unstructured search lower-bounds.

## B.4 Generalization to Multivariate Functions

Many applications actually require multivariate input states, such as in quantum chemistry [73], quantum field theory [183], and derivative pricing with multiple underlying assets [69]. However, there is only limited work exploring the preparation of such input states. In 2008, Kitaev and Webb presented an algorithm for preparing multivariate normal distributions [184], which was expanded upon by a 2021 paper conducting resource estimates and optimizations [185]. Finally, some work in preparing chemistry-specific multivariate input states has also been explored [74].

Recall that our algorithm uses an oracle that efficiently computes the function values in the sense of the mapping $O_f|k\rangle|0\rangle = |k\rangle|f(x_k)\rangle$ for a discretised $x$ value with $k \in \{0,1\}^n$. We can straightforwardly extend the present approach to multivariate functions which we illustrate on the example of a 2-variate function $f(x,y)$. As such, we aim to prepare a quantum state of $2n$ qubits such that its amplitudes are proportional to samples of the function. It is straightforward to show that this is equivalent to using our single-variate approach on $2n$ qubits such that the oracle implements the mapping $O_f|k\rangle|0\rangle = |k\rangle|f(x_{k_M}, y_{k_L})\rangle$ for a discretised $x$ value with $k \in \{0,1\}^{2n}$ and $k_M$ are the $n$ most significant bits of $k$ while $k_L$ are the $n$ least significant bits of $k$. With this definition of an oracle, our approach then prepares the quantum state

$$|\psi\rangle = (\mathcal{N}_N)^{-1} \sum_{k_M \in \{0,1\}^n} \sum_{k_L \in \{0,1\}^n} f(x_{k_M}, y_{k_L})|k_M\rangle|k_L\rangle.$$

The preparation of general $d$-dimensional (i.e. $d$-variate) functions then clearly follows by using our approach on $dn$ qubits and implementing the oracle $O_f|k\rangle|0\rangle = |k\rangle|f(x_1(k), x_2(k), \ldots x_d(k))\rangle$ with $k \in \{0,1\}^{dn}$ and, e.g, $x_1(k)$ is computed from the most significant $n$ bits of $k$.

While above we have related the preparation of multivariate distributions to our single-variate approach, it is important to note that the main limitation of our technique is that our state-preparation error bounds depend on the filling ratio of the function $f(x)$. It is easy to see that the filling ratio might decrease exponentially with an increasing number of variables, e.g., in case of separable functions $f_1(x_1)f_2(x_2)\cdots f_d(x_d)$ the filling ratios are products of the individual filling ratios. Nevertheless, such separable functions can be prepared by $d$ applications of our approach onto $d$ registers each preparing a single-variate function, e.g., $f_1(x)$. The joint quantum state of the $d$ registers is then naturally of a tensor product form. In typical applications it may already be a significant advantage to be able to prepare low-dimensional functions to high resolution in separate registers such that the joint quantum state is a separable high-dimensional function.

An example can be found in preparing 3-dimensional molecular orbitals in real-space quantum chemistry simulations [73]. The initial product state is only an approximation and the quantum computer is then used to entangle the registers anyway, e.g., by performing a simulation between interactive particles that are initially in a separable state.

## B.5 Further State-Preparation Techniques

### B.5.1 Summary of Prior Work on State Preparation

As early as in 2000 Grover proposed a black-box function-loading approach for loading an arbitrary, efficiently computable sequence $\boldsymbol{\alpha} = (\alpha_1, \alpha_2 \ldots, \alpha_N)$ as amplitudes of a quantum state [87]. Here the vector is assumed to be normalised such that $0 \leq \alpha_k < 1$ and thus we can define it as $\alpha_k := f(x_k)/M$ via the maximum $M$ of the function over any input. The approach relies on a $\mathfrak{rot}$ oracle that prepares an ancilla qubit in the superposition

$$\mathfrak{rot}|\xi\rangle|0\rangle := |\xi\rangle(\sin\theta|0\rangle + \cos\theta|1\rangle),$$

where $|\xi\rangle$ is a binary representation of one of the amplitudes $\alpha_k$ while $\theta$ is a high-precision approximation to $\arcsin(\xi/2^n)$ – we discuss details of the implementation below. Grover then prescribed $O(\sqrt{N}/\|\alpha\|_2)$ rounds of amplitude amplification which ultimately determines the complexity as the circuit depth of the approach – which is provably optimal. This complexity is determined by our generalised filling ratio $\frac{\mathcal{N}}{\sqrt{N}M} = \tilde{\mathcal{F}}$ from Definition B.1.3. Rejection sampling [186] is a closely related techniques and similarly prepares an ancilla qubit with the $\mathfrak{rot}$ oracle.

Realising that the $\mathfrak{rot}$ oracle in Grover's state preparation approach might require a large number of quantum gates to implement, ref. [88] introduced a more 'gate frugal' approach by replacing the $\mathfrak{rot}$ oracle with a $\mathfrak{comp}$ oracle. This reduced the absolute number of Toffoli gates by two orders of magnitudes in realistic examples.

These black-box techniques were recently further improved in refs. [89, 90]. For example, ref. [89] proposes the use of more efficient fixed-point amplitude amplification protocols and reports explicit complexities for specific discretised

functions, such as a Gaussian function. As such, ref. [89] concluded that if the standard deviation of the Gaussian distribution satisfies certain growth conditions then the corresponding state preparation only requires polynomial-depth circuits.

Indeed, as the black-box state preparation complexity scales as $O(\sqrt{N}/\|\alpha\|_2)$, for any fixed continuous function, this complexity asymptotically approaches the constant filling ratio $\mathcal{F}^{-1}$ for large $N$. As such, generally the class of black box function loading techniques could be used for preparing arbitrary continuous functions with a complexity $O(\mathcal{F}^{-1})$. While our results suggest that our approach is less efficient as its complexity is $O(\mathcal{F}^{-2})$, we expect our bounds are generally quite loose and could be tightened. In our numerical simulations we actually observed a significantly improved scaling compared to our bounds.

Finally, our work is also fundamentally different different than other techniques as it is based on an adiabatic, low-rank simulation approach – and our asymptotic analysis for continuous functions is a novel aspect.

### B.5.2 Possible Implementation of Grover's State Preparation

Here I present a modified and simplified version of the state preparation approach presented by Grover in 2000 [87]. First, I will outline the procedure, and then will explain how it can be implemented. In this approach, we assume access to an oracle implementation $O_f|j\rangle|0\rangle = |j\rangle|f(x_j)\rangle$ of the function $f : \mathbb{R} \mapsto \mathbb{C}$, and our objective is to prepare the quantum state $\frac{1}{\mathcal{N}}\sum_j f(x_j)|j\rangle$ (for normalization constant $\mathcal{N}$). We further assume that $M$ is the absolute maximum of $f$, such that $|f(x_j)|/M \leq 1$ for all inputs $x_j$. Then, this procedure works by first preparing the quantum state,

$$\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle\left(\frac{f(x_j)}{M}|0\rangle + \sqrt{1-\frac{f(x_j)^2}{M^2}}|1\rangle\right), \tag{B.117}$$

and then measuring the ancillary qubit, repeating the process until the $|0\rangle$ state is measured. Clearly, if we measure the ancillary register in the $|0\rangle$ state, we will immediately obtain the state $\frac{1}{\mathcal{N}}\sum_{j=1}^{N} f(x_j)|j\rangle$ as desired. One can boost the probability of success by using amplitude amplification, but we will now show

that for this procedure to be efficient (when $f$ is Lipschitz continuous), amplitude amplification is not necessary in the asymptotic limit of $n$. In particular, the probability of measuring the $|0\rangle$ ancillary state is given by $\frac{1}{NM^2}\sum_{j=1}^{N}|f(x_j)|^2 = \frac{\mathcal{N}^2}{NM^2}$. As we have discussed at length in this manuscript, for continuous discretized continuous functions $\mathcal{N} \in \Theta(\sqrt{N})$, and so (ignoring constant factors) the probability of success is given by $\Theta(1)$ (of course, the constant factor makes the procedure non-deterministic, and intuitively the probability of success depends on the filling ratio). Precisely, the probability of success is exactly the square generalised filling ratio $\frac{\mathcal{N}^2}{NM^2} = \tilde{\mathcal{F}}^2$ from Definition B.1.3. This probability of success is improved quadratically through the use of amplitude amplification in Grover's approach.

We now outline a possible implementation. We begin by introducing an $n$ qubit main register, a $d$ qubit ancillary register, as well as the single qubit ancillary register used above. We initialize the main register in a uniform superposition, and place all other registers in the zero state, yielding,

$$\frac{1}{\sqrt{N}}\sum_{j=1}|j\rangle_n|0\rangle_d|0\rangle_1. \tag{B.118}$$

Now, we define a new oracle, $O_g$, defined such that $O_g|x_j\rangle_n|0\rangle_d = |x_j\rangle_n|\arccos(f(x_j)/M)\rangle_d$. We apply $O_g \otimes I$ to the preceding quantum state, yielding

$$\frac{1}{\sqrt{N}}\sum_{j=1}|j\rangle_n|\arccos(f(x_j)/M)\rangle_d|0\rangle_1. \tag{B.119}$$

Now consider the $CR_Y(t)$ circuit similar to the $CR_X(t)$ circuit defined in Fig. B.1, performing the mapping $|a\rangle_d|\psi\rangle_1 = |a\rangle_d e^{-iatY}|\psi\rangle_1$ (for the standard single-qubit NOT gate $Y$). We then apply $I \otimes CR_Y(t=0)$ to our quantum state, yielding

$$\frac{1}{\sqrt{N}}\sum_{j=1}|j\rangle_n|\arccos f(x_j)\rangle_d\left(\frac{f(x_j)}{M}|0\rangle_1 + \sqrt{1-\frac{f(x_j)^2}{M^2}}|1\rangle_1\right). \tag{B.120}$$

We then simply uncompute the contents of the middle register (by applying $O_g^{-1}\otimes I$), discard the resulting unentangled $d$-qubit register, yielding the desired state in Eq. (B.117). The overall complexity of this procedure is thus the complexity of implementing the $O_f$ oracle twice, and the complexity of implementing the $CR_Y$ gate (which is $O(d)$).

### B.5.3 Possible Implementation of Grover-Rudolph State Preparation

I will now outline Grover-Rudolph state preparation [91]. Their objective is to prepare the $n$ qubit state,

$$|\psi\rangle_n = \sum_{j=0}^{2^n-1} g(x_j)|j\rangle_n, \tag{B.121}$$

where $g(x_j)$ is the (normalized) density of the given function $f$ in the $j^{th}$ grid interval. In essence, given the correct $k$ qubit state (with $1 \leq k < n$), they construct the $k+1$ qubit state, iterating this process until the desired $n$ qubit state is obtained. They then define the function,

$$h^{(k)}(j) = \frac{\int_{x_j^{(k)}}^{x_j^{(k)}+\frac{1}{2}\Delta_K} f(x)dx}{\int_{x_j^{(k)}}^{x_{j+1}^{(k)}} f(x)dx}, \tag{B.122}$$

where $K = 2^k$ and $\Delta_K$ is the step-size in a $k$ qubit grid. That is, $h^{(k)}(j)$ represents the conditional probability of $x$ being on the left half of grid interval $j$ on a $k$-qubit grid, given that $x$ lies in the $j^{th}$ interval. Clearly, if $f$ can be integrated efficiently, then $h^{(k)}(j)$ can be computed efficiently and so can $\theta_j^{(k)} \equiv \arccos\left(h^{(k)}(j)\right)$. An ancillary $d$ qubit register is then added, as well as a least-significant 1 qubit register. The following state is then obtained by evaluating the oracle performing the mapping $|j\rangle_k|0\rangle_d \mapsto |j\rangle_k|\theta_j^{(k)}\rangle_d$,

$$\sum_{j=0}^{2^k-1} g(x_j^{(k)})|j\rangle_k|\theta_j^{(k)}\rangle_d|0\rangle_1. \tag{B.123}$$

It is important to note that this step requires integration to be performed in superposition. As observed in the Grover paper, any log-concave distribution can be efficiently integrated using Monte Carlo integration, and can thus be integrated efficiently on a quantum computer. Define a CRy gate (with a definition analogous to the CRx gate shown in Figure B.1) such that it acts on the last two registers and performs the mapping $|a\rangle_d|0\rangle_1 \mapsto |a\rangle_d\left(\cos a|0\rangle_1 + \sin a|1\rangle_1\right)$. Note that such a

gate is derived in Section B.2 (in the general setting, in this case $t = 1$). The CRy gate is then applied to the last two registers in Equation B.123, yielding the state

$$\sum_{j=0}^{2^k-1} g(x_j^{(k)})|j\rangle_k|\theta_j^{(k)}\rangle_d \left(\cos\theta_j^{(k)}|0\rangle_1 + \sin\theta_j^{(k)}|1\rangle_1\right) \tag{B.124}$$

$$= \sum_{j=0}^{2^k-1} g(x_j^{(k)})|j\rangle_k|\theta_j^{(k)}\rangle_d \left(h^{(k)}(j)|0\rangle_1 + (1-h^{(k)}(j))|1\rangle_1\right). \tag{B.125}$$

We then uncompute the $d$ qubit ancillary register (by applying the inverse of the integration circuit – noting that if the integration is classically stochastic, it must be seeded) and discard it, yielding the state

$$\sum_{j=0}^{2^k-1} g(x_j^{(k)})|j\rangle_k \left(h^{(k)}(j)|0\rangle_1 + (1-h^{(k)}(j))|1\rangle_1\right) \tag{B.126}$$

$$= \sum_{j=0}^{2^{k+1}-1} g(x_j^{(k+1)})|j\rangle_{k+1} = |\psi\rangle_{k+1}. \tag{B.127}$$

The procedure is then repeated until the desired $n$ qubit state is obtained.

## B.6 Asymptotic Analysis

In this section, I outline the key results form Appendix H of our paper, where we prove a number of results relating to the asymptotic properties of functions. The key insight that the generalized filling ratio is an asymptotic constant was a product of close collaboration. In Lemma B.6.2 I provide a simplified statement of this fact.

First, I state a result from our paper due to my co-author (and omit the proof).

**Lemma B.6.1** (Riemann sums)**.** *Given an arbitrary continuous function $f(x) : [a, b] \mapsto \mathbb{C}$ on the interval $a \leq x \leq b$, the finite Riemann sum approximates the integral as*

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{k=0}^{N-1} f(x_k)\Delta_N + \epsilon$$

*up to an asymptotically vanishing error $\lim_{N\to\infty}\epsilon = 0$. For functions of bounded variation the error term additionally satisfies $\epsilon \in \mathcal{O}(N^{-1})$, see refs. [187, 188] which also applies to Lipschitz continuous functions given they are of bounded variation.*

Using this result, I will now show a simplified proof, which is essentially just a corollary of the above result, showing that the generalized filling ratio is an asymptotic constant. Note that if one is not concerned about rigorously proving the asymptotic scaling of the error (and is happy to just take a limit), this result can be directly obtained from the definition of a Riemann integral, and the definition of a Lipschitz continuous function.

**Lemma B.6.2.** *Suppose we are given a Lipschitz continuous function $f : [a, b] \mapsto \mathbb{R}$. Define a discretized grid with $N$ points in the interval $[a, b]$, call it $\{x_0, ..., x_{N-1}\}$ with spacing $\Delta_N := (b-a)/N$. Define the normalization factor as $\mathcal{N}^2 := \sum_{k=0}^{N-1} f(x_k)^2$, and the generalized filling ratio (as per Definition B.1.3) as $\hat{\mathcal{F}} = \frac{\mathcal{N}}{\sqrt{N}M}$. Then,*

$$\hat{\mathcal{F}}^2 = \frac{\|f\|_2^2}{(b-a)\|f\|_\infty^2} + O(N^{-1}). \tag{B.128}$$

*Thus, for a Lipschitz function, $\hat{\mathcal{F}}$ is an asymptotic constant in the limit $N \to \infty$.*

*Proof.* Let the maximum of $f$ over the discrete input domain be $M = \max_{k\in[N]} |f(x_k)|$. From Lemma B.6.1, we can write

$$\|f\|_2^2 = \int_a^b f(x)^2 dx = \sum_{k=0}^{N-1} f(x_k)^2 \Delta_N + O(N^{-1}) \tag{B.129}$$

$$= \frac{(b-a)}{N}\mathcal{N}^2 + O(N^{-1}) = (b-a)M^2\hat{\mathcal{F}}^2 + O(N^{-1}). \tag{B.130}$$

Let the Lipschitz constant of $f$ be $L$. For any $x \in [x_k, x_{k+1})$, $|f(x_k) - f(x)| \leq L|x_k - x| \leq L\Delta_N = L(b-a)/N$. Suppose $x_* := \arg\max_{x\in[a,b]} |f(x)|$ (and note that $\|f\|_\infty = f(x_*)$). Then, there exists some grid point $x_j$ (within distance $\Delta_N$) such that $|f(x_j) - f(x_*)| \leq L(b-a)/N$, which implies that $\|f\|_\infty - \frac{L(b-a)}{N} \leq f(x_j) \leq \|f\|_\infty + \frac{L(b-a)}{N}$. Moreover, since $f(x_j) \leq M \leq f(x_*)$, we have $\|f\|_\infty - \frac{L(b-a)}{N} \leq M \leq \|f\|_\infty + \frac{L(b-a)}{N}$. Consequently, $M = \|f\|_\infty + O(N^{-1})$. Thus, we get

$$\hat{\mathcal{F}}^2 = \frac{\|f\|_2^2}{(b-a)\|f\|_\infty^2} + O(N^{-1}). \tag{B.131}$$

□

I omit the proofs of the following results as they are straight-forward corollaries of the above idea. Please see Appendix $H$ of the paper [2] for the full details.

**Property B.6.1** (pointwise encoded amplitudes)**.** *Given an arbitrary continuous function, the pointwise encoded amplitudes satisfy the general bound* $N|\psi_k|^2 \leq (b-a)\|f\|_{max}^2/\|f\|_2^2 + \epsilon$*. The error term asymptotically vanishes* $\lim_{N\to\infty}\epsilon = 0$ *and for Lipschitz continuous functions it additionally satisfies* $\epsilon \in \mathcal{O}(N^{-1})$

**Property B.6.2.** *Given a Lipschitz continuous, non-negative function* $f$*, the quantum state* $|\psi\rangle$ *in Eq. B.3 obtained via the integral encoding* $g(x)$ *becomes identical to the pointwise encoding of the corresponding function* $\sqrt{f(x)}$ *for large* $N$*.*

**Property B.6.3** (integral encoded amplitudes)**.** *Given a Lipschitz continuous function, the integral encoded amplitudes satisfy the general bound* $N|\psi_k|^2 \leq (b-a)\|f\|_{max} + \epsilon$*. The error term asymptotically vanishes* $\lim_{N\to\infty}\epsilon = 0$ *and for Lipschitz continuous function it additionally satisfies* $\epsilon \in \mathcal{O}(N^{-1})$*.*

The following result is due to my co-author, and I include it for completeness while omitting the proof.

**Property B.6.4** (pointwise overlap with uniform states)**.** *For an arbitrary continuous function* $f(x) : [a,b] \mapsto \mathbb{R}$ *on the interval* $a \leq x \leq b$ *we construct the non-negative function* $\tilde{f}(x) := [f(x) \pm |f(x)|]/2$ *which has overlap at least* $1/2$ *for a properly chosen sign with our function* $f$ *and we encode its function values into a quantum state via our pointwise encoding from Eq.* (B.3)*. For any such encoding we can establish the asymptotically bounded fidelity with the easy-to-prepare state* $|+\rangle^{\otimes n}$ *as* $|\langle\psi|(|+\rangle^{\otimes n})|^2 \geq \mathcal{F}^2/(b-a)^2 + \epsilon$ *in terms of our filling ratio* $\mathcal{F}$*. The error term asymptotically vanishes* $\lim_{N\to\infty}\epsilon = 0$ *and for Lipschitz continuous function it additionally satisfies* $\epsilon \in \mathcal{O}(N^{-1})$*.*

### B.6.1 Asymptotic Properties of the Rank-One Matrix $A$

In this section, we utilize our observations about the filling ratio to prove facts about the rank$-1$ matrix used in our adiabtic algorithm.

**Lemma B.6.3** (pointwise encoding)**.** *Given our pointwise encoding from Eq.* (B.3) *of an arbitrary continuous function* $f$ *and the corresponding matrix entries* $A_{kl} :=$

*$f(x_k)f^*(x_l)$ form Eq.* (B.57)*, the matrix norm is constant bounded* $\|A\|_{max} \leq \|f\|^2_{max}$*. The only non-zero eigenvalue of* $A/N$ *is given by the spectral norm as* $\|A/N\|_2 = \mathcal{N}_N^2/N = (b-a)\|f\|_2^2 + \epsilon$*. Furthermore, the ratio of matrix norms is generally upper bounded as* $\|A\|_{max}/\|A/N\|_2 \leq \mathcal{F}^{-2} + \epsilon$ *by our filling ratio from Defition B.1.2. The error term asymptotically vanishes* $\lim_{N\to\infty}\epsilon = 0$ *and for Lipschitz continuous functions it additionally satisfies* $\epsilon \in \mathcal{O}(N^{-1})$*.*

*Proof.* Define the rank-1 matrix $A_{kl} := f(x_k)f^*(x_l)$. We can bound the largest entry in the matrix as

$$\|A\|_{max} = \max_{kl}|A_{kl}| = \max_{kl}|f(x_k)f^*(x_l)| = \max_k |f(x_k)|^2 \leq \|f\|^2_{max}.$$

For Lipschitz continuous functions, the error scaling is $\|A\|_{max} = \|f\|^2_{max} + \mathcal{O}(N^{-1})$. We now bound the matrix norm using that $\|A\|_2^2 = \|A\|_{HS}^2$ for our rank-1 matrix $A$ as

$$\|A\|_2^2 = \|A\|_{HS}^2 = \sum_{k,l}|f(x_k)f(x_l)^*|^2 = [\sum_k |f(x_k)|^2]^2.$$

We can approximate this expression via the Riemann sum from Lemma B.6.1 as

$$\|A/N\|_2 = \sum_k |f(x_k)|^2/N = \sum_k |f(x_k)|^2 \frac{\Delta_N}{b-a} = \frac{\|f\|_2^2}{b-a} + \epsilon, \tag{B.132}$$

where the error term asymptotically vanishes $\lim_{N\to\infty}\epsilon = 0$ and for Lipschitz continuous function it additionally satisfies $\epsilon \in \mathcal{O}(N^{-1})$. Finally,

$$\begin{aligned}\frac{\|A\|_{max}}{\|A/N\|_2} &= \frac{\max_k |f(x_k)|^2}{\sum_k |f(x_k)|^2 \Delta_N/(b-a)} \leq \frac{(b-a)\|f\|^2_{max}}{\|f\|_2^2} + \epsilon \leq \frac{(b-a)^2\|f\|^2_{max}}{\|f\|_1^2} + \epsilon \\ &= \mathcal{F}^{-2} + \epsilon,\end{aligned}$$

where we have used the general function-norm inequality $\|f\|_2^2(b-a) \geq \|f\|_1^2$, which follows from Hoelders inequality $\|1 \times f^p\|_1 \leq \|1\|_{q/(q-p)}\|f^p\|_{q/p}$ upon substituting $p = 1$ and $q = 2$. We note that the application of this function-norm inequality makes our bound looser, however, the resulting loose bound is more intuitive and easier to interpret as illustrated in Fig. 3.2. We have also substituted back our filling ratio $\mathcal{F}$ from Definition B.1.2. □

**Lemma B.6.4** (integral encoding)**.** *Given an arbitrary continuous efficiently integrable function $f$ in Definition B.1.1 and our corresponding rank-1 matrix $A_{kl} := Ng(x_k)g^*(x_l)$ from Eq.* (B.57)*, the matrix satisfies $\|A/N\|_2 = \|f\|_1$ and its matrix entries are asymptotically constant bounded as $\|A\|_{max} = (b-a)\|f\|_{max} + \epsilon$. Furthermore, the ratio of the matrix norms is generally upper bounded as $\|A\|_{max}/\|A/N\|_2 \leq \mathcal{F}^{-1} + \epsilon$ by our filling ratio from Definition B.1.2. The error term asymptotically vanishes $\lim_{N\to\infty} \epsilon = 0$ and for Lipschitz continuous function it additionally satisfies $\epsilon \in \mathcal{O}(N^{-1})$.*

*Proof.* We first define the rank-1 matrix with non-negative entries as

$$A_{kl} := Ng(x_k)g(x_l).$$

We can bound the largest entry in $A$ as

$$\|A\|_{max} = \max_{kl} |A_{kl}| = N \max_k g^2(x_k) = N \max_k f(x_k)\Delta_N \tag{B.133}$$

$$= (b-a)\|f\|_{max} + \epsilon, \tag{B.134}$$

where we use Property B.6.2 which shows that $g(x_k) = \sqrt{f(x_k)\Delta_N} + \epsilon$, and note that $\Delta_N = (b-a)/N$. The error term asymptotically vanishes $\lim_{N\to\infty} \epsilon = 0$ and for Lipschitz continuous function it additionally satisfies $\epsilon \in \mathcal{O}(N^{-1})$ via Lemma B.6.1.

Given the matrix $A$ is rank-1 by definition, it only has 1 non-zero eigenvalue which is equivalent to the norm $\|A\|_2$ which we can compute via the (equivalent) Hilbert-Schmidt norm as

$$\|A\|_{HS}^2 = N^2 \sum_{k,l} [g(x_k)g(x_l)]^2 = [N \sum_k g^2(x_k)]^2.$$

As such, we exactly obtain the largest eigenvalue of the matrix as

$$\|A/N\|_2 = \sum_k g^2(x_k) = \sum_k \int_{x_k}^{x_k+\Delta_N} f(x)\,\mathrm{d}x = \int_a^b f(x)\,\mathrm{d}x = \|f\|_1, \tag{B.135}$$

where in the last equation we have used that $f$ is non-negative and therefore its integral is equivalent to is $L^1$ norm. Finally, we establish the ratio of matrix norms as

$$\frac{\|A\|_{max}}{\|A/N\|_2} = \frac{N \max_k g^2(x_k)}{\sum_k g^2(x_k)} \leq \frac{(b-a)\|f\|_{max}}{\|f\|_1} + \epsilon = \mathcal{F}^{-1} + \epsilon,$$

where we substituted our definition of the filling ratio $\mathcal{F}$ from Definition B.1.2. □

# C

# Non-Linear Transformation of Quantum Amplitudes

Understanding the contents of this section is not crucial to understanding the main novel contributions in this chapter, and simply contains the rigorous proofs of the results already presented.

## C.1 Preliminaries

Following [26], for a general operator $M$ with singular value decomposition $M = U\Sigma V^\dagger$, we use the notation $f^{(SV)}(M) := Uf(\Sigma)V^\dagger$ to represent the transformation of the singular values if $f$ is an odd function. If $f$ is an even function, we instead use the notation $f^{(SV)}(M) := Vf(\Sigma)V^\dagger$.

*Computational model–* To reiterate, I use the standard quantum circuit model of quantum computation [189]. An application of a quantum gate is equivalent to performing an elementary operation. The query complexity of a classical/quantum algorithm with some input is the maximum number of queries the algorithm makes on the input. Time complexity is measured in circuit depth, in terms of single and two qubit gates. Classical time complexity is measured in the number of sequential fundamental computation steps. We assume that all single and two qubit gates are implemented exactly.

I now present a simple result relating error-bounds on normalized states to error-bounds on un-normalized states.

**Lemma C.1.1.** *Given two normalized quantum states* $|\tilde{a}\rangle = \frac{1}{\mathcal{N}_a}|a\rangle$, *and* $|\tilde{b}\rangle = \frac{1}{\mathcal{N}_b}|b\rangle$ *such that* $|\mathcal{N}_a - \mathcal{N}_b| \leq \epsilon_0$, *and that* $\||a\rangle - |b\rangle\|_2 \leq \epsilon_1$, *then* $\left\||\tilde{a}\rangle - |\tilde{b}\rangle\right\|_2 \leq (\epsilon_0 + \epsilon_1)/\max\{\mathcal{N}_a, \mathcal{N}_b\}$.

*Proof.* A simple calculation obtains

$$\left\||\tilde{a}\rangle - |\tilde{b}\rangle\right\|_2 = \left\|\frac{|a\rangle}{\mathcal{N}_a} - \frac{|b\rangle}{\mathcal{N}_b}\right\|_2 = \frac{1}{\mathcal{N}_a\mathcal{N}_b}\left\|(\mathcal{N}_b - \mathcal{N}_a)|a\rangle + \mathcal{N}_a(|a\rangle - |b\rangle)\right\|_2.$$

Applying triangle inequality, and noting that $\mathcal{N}_a = \||a\rangle\|_2$,

$$\left\||\tilde{a}\rangle - |\tilde{b}\rangle\right\|_2 \leq \frac{1}{\mathcal{N}_b}\left(|\mathcal{N}_b - \mathcal{N}_a| + \||a\rangle - |b\rangle\|_2\right) \leq \frac{\epsilon_0 + \epsilon_1}{\mathcal{N}_b}.$$

□

## C.2 Diagonal Block-Encoding of Quantum State Amplitudes

In Definition C.2.1, I begin by specifying the assumed input model, a state preparation unitary as specified by a quantum circuit. When applied to the $|0\rangle$ state, this quantum circuit uniquely specifies a quantum state. I then present some further preliminary definitions and results, culminating in the derivation of our diagonal block-encoding of the state amplitudes, a variant of the block-encoding first presented in [120, 121]. I generalize this input model in Appendix C.10

**Definition C.2.1** (Input model of this work, State preparation unitary)**.** *Let* $n \in \mathbb{Z}_{>0}$. *We are given an* $n$*-qubit quantum circuit termed the "state-preparation circuit". This circuit is described by a unitary matrix denoted by* $U \in \mathbb{C}^{N\times N}$, *where* $N = 2^n$. *The action of* $U$ *on the all-zero state defines the following state,*

$$U|0\rangle_n = \sum_{j=0}^{N-1} \psi_j|j\rangle =: |\psi\rangle, \tag{C.1}$$

*where* $\psi_j \in \mathbb{C}$ *with* $\sum_{j=0}^{N-1}|\psi_j|^2 = 1$. *We call a "real state preparation circuit" a quantum circuit that has the property that* $\forall j, \psi_j \in \mathbb{R}$.

I define a few elementary gates in the following. I use the total working space of $2n+1$ qubits, in anticipation of the needs of the algorithm. The definitions make precise on which qubits the gates are acting on.

**Definition C.2.2.** *We define the following rotation gate, acting on* $2n+1$ *qubits:*

$$R := (I_{n+1} - 2|0\rangle\langle 0|_{n+1}) \otimes I_n. \tag{C.2}$$

This gate can be implemented by an $n$-controlled $XZX$ gate targeting the first qubit, controlled on the next $n$-qubits (controlled on the $|0\rangle_n$ state), and with an identity acting on the final $n$ qubits. The $n+1$ qubit controlled gate can then be decomposed into $O(n)$ two-qubit gates via the decomposition of [190], giving a total circuit depth for this operator of $O(n)$.

**Definition C.2.3.** *We define the following* $2n+1$ *qubit gates,*

$$\hat{Z} := I_n \otimes Z \otimes I_n, \quad \hat{H} := I_n \otimes H \otimes I_n, \quad \hat{S} := I_n \otimes S \otimes I_n, \tag{C.3}$$

*where* $Z$ *is the standard* 1*-qubit Pauli-Z gates,* $H$ *is the* 1*-qubit Hadamard gate, and* $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$.

These can all clearly be implemented with a single circuit layer.

**Definition C.2.4.** *Given an* $n$*-qubit unitary* $U$*, we define the* $2n+1$ *qubit operator* $U_C$ *as follows*

$$U_C := (U \otimes |0\rangle\langle 0|_1 + I_n \otimes |1\rangle\langle 1|_1) \otimes I_n. \tag{C.4}$$

This gate can be implemented with one query to a controlled $U$ gate, with an $X$ gate applied before and after the control (on the control qubit).

**Definition C.2.5.** *We define a* $2n+1$ *qubit "controlled-copy" circuit* $C$*, which implements the mapping of the* $|0\rangle_n|0\rangle_1|k\rangle_n$ *state to itself, and the* $|0\rangle_n|1\rangle_1|k\rangle_n$ *state to* $|k\rangle_n|1\rangle_1|k\rangle_n$*. The matrix form of the gate may be written as,*

$$C := I_n \otimes |0\rangle\langle 0|_1 \otimes I_n + \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} |j \oplus k\rangle\langle j|_n \otimes |1\rangle\langle 1|_1 \otimes |k\rangle\langle k|_n. \tag{C.5}$$

This gate can be implemented by $n$ Toffoli gates controlled on the $|1\rangle_1$ state of the middle qubit, and with the other control of the $i^{th}$ Toffoli gate acting on the $|1\rangle_1$ state of the $(n+1+i)^{th}$ qubit, and with the target being the $i^{th}$ qubit. This can clearly be implemented with $O(n)$ two-qubit gates and $O(n)$ circuit depth.

I continue by defining some gate sequences and proving their action. I define the $W_1$ operator slightly differently than in [120, 121], but note that the action on the desired subspace is unchanged. The flag $p$ determines if a relative phase is applied or not.

**Lemma C.2.1** ($W$ operator from [120, 121])**.** *For $p \in \{0,1\}$, define the $2n+1$ qubit operator $W_p$ by the circuit,*

$$W_p := \hat{H}\hat{S}^p CU_C \hat{H}. \tag{C.6}$$

*When operating on the states $|0\rangle_n|0\rangle_1|k\rangle_n$, where $|k\rangle_n$ is an $n$-qubit standard basis vector, it prepares the family of $2^n$ states $\{|\Phi_k^p\rangle\}_k$ given by*

$$|\Phi_k^p\rangle := W_p|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{2}\left((|\psi\rangle_n + i^p|k\rangle_n)|0\rangle_1 + (|\psi\rangle_n - i^p|k\rangle_n)|1\rangle_1\right)|k\rangle_n. \tag{C.7}$$

*The circuit $W_p$ uses $O(n)$ circuit depth with 1 query to a controlled-$U$ gate.*

*Proof.* First, with $\hat{H}$ as per Definition C.2.3, $\hat{H}|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{\sqrt{2}}\left(|0\rangle_n|0\rangle_1 + |0\rangle_n|1\rangle_1\right)|k\rangle_n$. Then, with $U_C$ as per Definition C.2.4, since $U|0\rangle_n = |\psi\rangle_n$, $U_C\hat{H}|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{\sqrt{2}}\left(|\psi\rangle_n|0\rangle_1 + |0\rangle_n|1\rangle_1\right)|k\rangle_n$. Applying $C$ as per Definition C.2.5 then yields,

$$CU_C\hat{H}|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{\sqrt{2}}\left(|\psi\rangle_n|0\rangle_1 + |k\rangle_n|1\rangle_1\right)|k\rangle_n. \tag{C.8}$$

As $\hat{S}^0 = I_{2n+1}$, applying $\hat{S}^p$ (as per Definition C.2.3) then yields,

$$\hat{S}^p CU_C\hat{H}|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{\sqrt{2}}\left(|\psi\rangle_n|0\rangle_1 + i^p|k\rangle_n|1\rangle_1\right)|k\rangle_n. \tag{C.9}$$

Finally, applying $\hat{H}$ again yields,

$$W_p|0\rangle_n|0\rangle_1|k\rangle_n = \frac{1}{2}\left((|\psi\rangle_n + i^p|k\rangle_n)|0\rangle_1 + (|\psi\rangle_n - i^p|k\rangle_n)|1\rangle_1\right)|k\rangle_n. \tag{C.10}$$

Clearly, this transformation has been implemented with $O(n)$ circuit depth, and one query to a controlled-$U$ gate. □

**Lemma C.2.2.** *Let $p \in \{0,1\}$. Let $G_p := W_p R W_p^\dagger \hat{Z}$ as per [121]. Then, $|\Phi_k^p\rangle$ is an eigenstate of $-\frac{1}{2}(G_p + G_p^\dagger)$, with eigenvalue $Re(\psi_k)$, if $p = 0$, and $Im(\psi_k)$, if $p = 1$.*

*Proof.* This fact was first identified and proven in its current form in [121], and in a similar form in [120]. First, since $\hat{Z} = \hat{Z}^\dagger$, $R = R^\dagger$, we have that $G_p^\dagger = \hat{Z} W_p R W_p^\dagger$. In addition, with $R = (I_{n+1} - 2|0\rangle\langle 0|_{n+1}) \otimes I_n$, we have

$$G_p^\dagger |\Phi_k^p\rangle = \hat{Z} W_p R W_p^\dagger |\Phi_k^p\rangle = \hat{Z} W_p R |0\rangle_n |0\rangle_1 |k\rangle_n = -\hat{Z} W_p |0\rangle_n |0\rangle_1 |k\rangle_n = -\hat{Z} |\Phi_k^p\rangle. \tag{C.11}$$

As a result, we have

$$(G_p + G_p^\dagger)|\Phi_k^p\rangle = W_p R W_p^\dagger \hat{Z} |\Phi_k^p\rangle - \hat{Z}|\Phi_k^p\rangle = (W_p R W_p^\dagger - I_{2n+1})\hat{Z}|\Phi_k^p\rangle. \tag{C.12}$$

Using the definition of $R$, we obtain

$$W_p R W_p^\dagger = W_p W_p^\dagger - 2W_p(|0\rangle\langle 0|_{n+1} \otimes I_n) W_p^\dagger = I_{2n+1} - 2W_p(|0\rangle\langle 0|_{n+1} \otimes I_n) W_p^\dagger. \tag{C.13}$$

Thus,

$$-\frac{1}{2}(G_p + G_p^\dagger)|\Phi_k^p\rangle = W_p(|0\rangle\langle 0|_{n+1} \otimes I_n) W_p^\dagger \hat{Z} |\Phi_k^p\rangle. \tag{C.14}$$

We will now examine $W_p^\dagger \hat{Z} |\Phi_k^p\rangle$, noting that

$$\hat{Z}|\Phi_k^p\rangle = \frac{1}{2}\left((|\psi\rangle_n + i^p|k\rangle_n)|0\rangle_1 - (|\psi\rangle_n - i^p|k\rangle_n)|1\rangle_1\right)|k\rangle_n. \tag{C.15}$$

Since $W_p = \hat{H}\hat{S}^p C U_C \hat{H}$, and thus $W_p^\dagger = \hat{H} U_C^\dagger C^\dagger (\hat{S}^p)^\dagger \hat{H}$. Applying $W_p^\dagger$ one part at a time, we first obtain

$$\begin{aligned}
&(\hat{S}^p)^\dagger \hat{H} \hat{Z} |\Phi_k^p\rangle && \text{(C.16)}\\
&= (\hat{S}^p)^\dagger \frac{1}{2}\left((|\psi\rangle_n + i^p|k\rangle_n)|+\rangle_1 - (|\psi\rangle_n - i^p|k\rangle_n)|-\rangle_1\right)|k\rangle_n && \text{(C.17)}\\
&= \frac{1}{\sqrt{2}}\left(i^p|k\rangle_n|0\rangle_1 + (-i)^p|\psi\rangle_n|1\rangle_1\right)|k\rangle_n && \text{(C.18)}\\
&= \frac{i^p}{\sqrt{2}}\left(|k\rangle_n|0\rangle_1 + (-1)^p|\psi\rangle_n|1\rangle_1\right)|k\rangle_n, && \text{(C.19)}
\end{aligned}$$

Applying $C^\dagger = I_n \otimes |0\rangle\langle 0|_1 \otimes I_n + \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} |j\rangle\langle j \oplus k|_n \otimes |1\rangle\langle 1|_1 \otimes |k\rangle\langle k|_n$, then yields

$$C^\dagger (\hat{S}^p)^\dagger \hat{H} \hat{Z} |\Phi_k^1\rangle = \frac{i^p}{\sqrt{2}}\left(|k\rangle_n|0\rangle_1 + (-1)^p \sum_{j=0}^{N-1} \psi_{j\oplus k}|j\rangle_n|1\rangle_1\right)|k\rangle_n. \tag{C.20}$$

Applying $(U^\dagger \otimes |0\rangle\langle 0|_1 \otimes I_n + I_n \otimes |1\rangle\langle 1|_1 \otimes I_n)$ then gives,

$$(U^\dagger \otimes |0\rangle\langle 0|_1 \otimes I_n + I_n \otimes |1\rangle\langle 1|_1 \otimes I_n)C^\dagger \hat{H}\hat{Z}|\Phi_k^p\rangle \tag{C.21}$$

$$= \frac{i^p}{\sqrt{2}}\left(U^\dagger|k\rangle_n|0\rangle_1 + (-1)^p \sum_{j=0}^{N-1} \psi_{j\oplus k}|j\rangle_n|1\rangle_1\right)|k\rangle_n. \tag{C.22}$$

Finally, applying $\hat{H}$ gives,

$$W_p^\dagger \hat{Z}|\Phi_k^p\rangle = \frac{i^p}{\sqrt{2}}\left(U^\dagger|k\rangle_n|+\rangle_1 + (-1)^p \sum_{j=0}^{N-1} \psi_{j\oplus k}|j\rangle_n|-\rangle_1\right)|k\rangle_n. \tag{C.23}$$

Then, $(|0\rangle\langle 0|_{n+1} \otimes I_n)W_p^\dagger \hat{Z}|\Phi_k^p\rangle$ becomes:

$$(|0\rangle\langle 0|_{n+1} \otimes I_n)W_p^\dagger \hat{Z}|\Phi_k^p\rangle \tag{C.24}$$

$$= \frac{i^p}{\sqrt{2}}(|0\rangle\langle 0|_{n+1} \otimes I_n)\left(U^\dagger|k\rangle_n|+\rangle_1 + (-1)^p \sum_{j=0}^{N-1} \psi_{j\oplus k}|j\rangle_n|-\rangle_1\right)|k\rangle_n \tag{C.25}$$

$$= \frac{i^p}{2}\left(|0\rangle_n\langle\psi|k\rangle|0\rangle + (-1)^p\psi_k|0\rangle_n|0\rangle_1\right)|k\rangle_n \tag{C.26}$$

$$= \frac{i^p}{2}(\psi_k^* + (-1)^p\psi_k)|0\rangle_n|0\rangle_1|k\rangle_n. \tag{C.27}$$

As a result, from Eq. (C.14), we obtain

$$-\frac{1}{2}(G_p + G_p^\dagger)|\Phi_k^p\rangle = W_p(|0\rangle\langle 0|_{n+1} \otimes I_n)W_p^\dagger \hat{Z}|\Phi_k^p\rangle \tag{C.28}$$

$$= \frac{i^p}{2}(\psi_k^* + (-1)^p\psi_k)|\Phi_k^p\rangle \tag{C.29}$$

$$= \begin{cases} \mathrm{Re}(\psi_k)|\Phi_k^p\rangle \text{ if } p = 0. \\ \mathrm{Im}(\psi_k)|\Phi_k^p\rangle \text{ if } p = 1. \end{cases}. \tag{C.30}$$

□

**Lemma C.2.3.** *Let $p = \{0, 1\}$. The operator $-\frac{1}{2}W_p^\dagger(G_p + G_p^\dagger)W_p$ has eigenvectors $|0\rangle_n|0\rangle_1|k\rangle_n$ with associated eigenvalue $Re(\psi_k)$ if $p = 0$ and $Im(\psi_k)$ if $p = 1$.*

*Proof.* This simply follows from $W_p|0\rangle_n|0\rangle_1|k\rangle_n = |\Phi_k^p\rangle$, $W_p^\dagger|\Phi_k^p\rangle = |0\rangle_n|0\rangle_1|k\rangle_n$, and Lemma C.2.2. □

**Lemma C.2.4.** *Let $p = \{0, 1\}$. The unitary*

$$U^{(p)} := (XZX \otimes I_{2n+1})(H \otimes W_p^\dagger)(|0\rangle\langle 0|_1 \otimes G_p + |1\rangle\langle 1|_1 \otimes G_p^\dagger)(H \otimes W_p). \tag{C.31}$$

*is a* $(1, n+2, 0)$*-block-encoding of* $A^{(p)} := diag(Re((-i)^p\psi_1), ..., Re((-i)^p\psi_N))$*, which is a shorthand for* $diag(Re(\psi_1), ..., Re(\psi_N))$ *if* $p = 0$ *and* $diag(Im(\psi_1), ..., Im(\psi_N))$ *if* $p = 1$*. The circuit* $U^{(p)}$ *may be implemented with* $O(n)$ *circuit depth, with a total of* 6 *queries to a controlled-U circuit.*

*Proof.* We aim to show that $\left\|A^{(p)} - (\langle 0|^{\otimes n+2} \otimes I_n)U^{(p)}(|0\rangle^{\otimes n+2} \otimes I_n)\right\|_2 = 0$. First, we begin by simplifying the expression of $U^{(p)}$,

$$U^{(p)} = (XZX|+\rangle\langle+|_1) \otimes (W_p^\dagger G_p W_p) + (XZX|-\rangle\langle-|_1) \otimes (W_p^\dagger G_p^\dagger W_p). \tag{C.32}$$

As a result,

$$(\langle 0|^{\otimes n+2} \otimes I_n)U^{(p)}(|0\rangle^{\otimes n+2} \otimes I_n) \tag{C.33}$$

$$= -(\langle 0|_1\langle 0|^{\otimes n+1} \otimes I_n)\left((|-\rangle\langle+|_1) \otimes (W_p^\dagger G_p W_p) + |+\rangle\langle-|_1) \otimes (W_p^\dagger G_p^\dagger W_p)\right) \cdot (|0\rangle_1|0\rangle^{\otimes n+1} \otimes I_n) \tag{C.34}$$

$$= -\frac{1}{2}(\langle 0|^{\otimes n+1} \otimes I_n)W_p^\dagger(G_p + G_p^\dagger)W_p(|0\rangle^{\otimes n+1} \otimes I_n). \tag{C.35}$$

Consider the $\langle j|_n \cdot |k\rangle_n$ matrix element of this matrix, $-\frac{1}{2}\langle 0|_n\langle 0|_1\langle j|_n W_p^\dagger(G_p + G_p^\dagger)W_p|0\rangle_n|0\rangle_1|k\rangle_n$. From Lemma C.2.3, we know that $W_p^\dagger(G_p+G_p^\dagger)W_p|0\rangle_n|0\rangle_1|k\rangle_n = -2\mathrm{Re}((-i)^p\psi_k)|0\rangle_n|0\rangle_1|k\rangle_n$, and so the assertion has been proven.

Clearly, the circuit cost of $U^{(p)}$ is dominated by $W_p$, $G_p$ and their inverses. As per Lemma C.2.1, each $W_p$ and $W_p^\dagger$ can be implemented with one query to the controlled-$U$ gate (and its inverse), and $O(n)$ additional depth in terms of single and two qubit gates. The $G_p$ operator can be implemented with a call to $W_p$ and $W_p^\dagger$ (adding two additional queries to a controlled-$U$ gate, and $O(n)$ additional circuit depth), a $\hat{Z}$ gate (with $O(1)$ depth) and the $R$ gate which has $O(n)$ depth as per Definition C.2.2, for a total of one query to the controlled-$U$ gate (and one query to its inverse) and $O(n)$ depth. Thus, $U^{(p)}$ can be implemented with $O(n)$ circuit depth, 3 queries to a controlled-$U$ gate and 3 queries to an inverse controlled-$U$ gate. □

**Lemma C.2.5** (Polynomial transformation of real/imaginary part of amplitudes)**.** *Let* $\delta > 0$*. Given a* $(1, n+2, 0)$*-block-encoding* $U^{(p)}$ *of*

$$A^{(p)} = diag(Re((-i)^p\psi_0), ..., Re((-i)^p\psi_{N-1}))$$

*(with $U|0\rangle_n = \sum_j \psi_j |j\rangle_n$), and a degree-k polynomial $P(x) = \sum_{j=0}^{k} a_j x^j$ (such that $\forall j, a_j \in \mathbb{C}$, and $\forall x, |P(x)| \leq 1/4$) we can obtain a $(1, n+4, \delta)$-block-encoding $\tilde{U}_P^{(p)}$ of $P(A^{(p)})$, which can be constructed with $O(kn)$ circuit depth, and a total of $k$ calls to the controlled $U^{(p)}$ and $(U^{(p)})^\dagger$ circuits. The circuit of $\tilde{U}_P^{(p)}$ can be computed with classical time complexity of $O(poly(k, \log(1/\delta))$.*

*Proof.* The resulting block encoding is very similar to the construction in [121], only it uses our diagonal block encoding (as opposed to the block encoding in [121] which is not diagonal in the standard basis). The proof follows trivially by applying Theorem 1.2.1 to the block encoding of $A^{(p)}$ constructed in Lemma C.2.4, noting that $A^{(p)}$ is Hermitian. □

## C.3 Importance-Weighted Amplitude Transform

In this section, I derive the technique responsible for our exponential improvement over prior work. In essence, rather than applying the desired polynomial to a block-encoding of the state amplitudes and then applying that to the uniform superposition (as is done in [121]), I instead consider a different polynomial transformation and then apply that to the state $U|0\rangle$. Intuitively, I define importance-weighted amplitude transformation as follows. Given a quantum state $|\psi\rangle = U|0\rangle = \sum_j \psi_j |j\rangle$ specified by the state preparation unitary $U$, we want to map each amplitude $\psi_j$ to an amplitude proportional to $f(\psi_j)$. Consider a function $f(x)$ such that $f(0) = 0$, and a function $h(x)$ such that $f(x) = xh(x)$. In importance weighted amplitude transformation, we create a diagonal block-encoding of $\text{diag}(h(\psi_1) \dots h(\psi_N))$, and then apply this to the state $|\psi\rangle$, so that each amplitude $\psi_j \mapsto h(\psi_j)\psi_j = f(\psi_j)$ thereby enacting the transformation. As mentioned, our importance-weighted transformation requires additional results and analysis, leading to our obtained improvements over the prior work.

As a result, I first consider a technical lemma about the Lipschitz constant of such functions. The following is a simplification of the corresponding result which appeared in the original version of the paper ([3]) this chapter is based on.

**Lemma C.3.1** (Bounding max of $|f(x)/x|$ by Lipschitz Constant of $f$ )**.** *Let $f(x)$ be a real-valued univariate function such that $f(0) = 0$ with Lipschitz constant $L$ in the interval $x \in [-1, 1]$, (i.e. $L := \max_{x\in[-1,1]} |\frac{d}{dx}P(x)|$). Let $h(x)$ be a function such that $f(x) = xh(x)$. Then, $\max_{x\in[-1,1]} |h(x)| \leq L$.*

*Proof.* By definition of a Lipschitz continuous real function, for all $a, b$ such that $a \neq b$, $|f(a) - f(b)|/|a - b| \leq L$. Then, $\max_{x\in[-1,0)\cup(0,1]} |f(x)/x| \leq L$. Thus, noting that $\lim_{x\mapsto 0} |f(x)/x| \leq L$, $\max_{x\in[-1,1]} |h(x)| = \max_{x\in[-1,1]} |f(x)/x| \leq L$. □

**Theorem C.3.1** (Polynomial transformation of real amplitudes via importance weighting)**.** *Let $\epsilon \in (0, 1]$. Assume we are given a state preparation unitary $U|0\rangle_n = |\psi\rangle_n = \sum_{j=1}^{2^n} \psi_j |j\rangle_n$, and $|\psi\rangle \in \mathbb{R}^{2^n}$ with $\||\psi\rangle\|_2 = 1$. We are given also a degree $k$ polynomial function $P(x) = \sum_{j=1}^{k} a_j x^j$ ($\forall j \in [k+1], a_j \in \mathbb{C}$) with Lipschitz constant $L$ (in the interval $x \in [-1, 1]$). Define $\eta := \max_{x\in[-1,1]} |P(x)/x|$, and note that Lemma C.3.1 implies that $\eta \leq L$. Define $\mathcal{N}^2 := \sum_{j=1}^{2^n} |P(\psi_j)|^2$. Then, we can construct an $\ell_2$-normalized quantum state $|\phi\rangle$ such that $\left\||\phi\rangle - \frac{1}{\mathcal{N}} \sum_{j=1}^{2^n} P(\psi_j)|j\rangle\right\|_2 \leq \epsilon$ with arbitrarily high probability of success with a total of $O(k\eta/\mathcal{N})$ queries to a controlled-$U$ circuit and a controlled-$U^\dagger$ circuit, with $O(kn\eta/\mathcal{N})$ additional circuit depth, and with a total of $O(poly(k, \log(\eta^2 N/\epsilon^2\mathcal{N}^2)))$ classical computation. This requires $O(n)$ ancillary qubits.*

*Proof.* Let $A := \text{diag}(\psi_0, ..., \psi_{N-1})$. The degree $k-1$ polynomial function $h(x)$ such that $P(x) = h(x)x$ is given by $h(x) := \sum_{j=1}^{k} a_j x^{j-1}$. Let $\eta := \max_{x\in[-1,1]} |h(x)|$. Then, define another function, $g(x) := h(x)/4\eta$. With the block-encoding of $A$, we invoke Lemma C.2.5 to construct a $(1, n + 4, \delta_0)$-block-encoding of $g(A)$, called $\tilde{U}_g$, with $O(kn)$ circuit depth, using a total of $k-1$ calls to a controlled $U_{\text{Re}(A)}$ and $U^\dagger_{\text{Re}(A)}$ circuit, and with $O(poly(k, \log(1/\delta_0)))$ classical time complexity. There is a function $\tilde{g}(x)$ for which $\tilde{U}_g$ is a $(1, n + 4, 0)$-block-encoding of $\tilde{g}(A)$ and $\|g(A) - \tilde{g}(A)\|_2 \leq \delta_0$. When we apply $\tilde{U}_g(|0\rangle_{n+4} \otimes U|0\rangle_n)$ and measure the $|0\rangle_{n+4}$ ancillary state, we obtain a state proportional to $\tilde{g}(A)|\psi\rangle = \sum_{j=1}^{2^n} \phi_j |j\rangle$, where we define the amplitudes of the unnormalized resulting state as $\{\phi_j\}_j$. We then define the normalization factor of this state as $\mathcal{N}_2 := \|\tilde{g}(A)|\psi\rangle\|_2$.

Observing that $g(A)|\psi\rangle = \sum_j g(\psi_j)\psi_j|j\rangle = \frac{1}{4\eta}\sum_j P(\psi_j)|j\rangle$, we additionally define the unnormalized exact state as $|P(\psi)\rangle := g(A)|\psi\rangle$, with the corresponding normalization factor of $\mathcal{N}_1 := \||P(\psi)\rangle\|_2$. Thus, the exact normalized state we wish to prepare is $\frac{1}{\mathcal{N}_1}|P(\psi)\rangle$, and the normalized state we are able to prepare is $\frac{1}{\mathcal{N}_2}\tilde{g}(A)|\psi\rangle$. We will now bound the $\ell_2$-norm error of the produced state, and determine the probability of success of the procedure.

First, we wish to determine the algorithmic complexity required to obtain an overall error $\epsilon$, defined as $\left\|\frac{1}{\mathcal{N}_1}|P(\psi)\rangle - \frac{1}{\mathcal{N}_2}\tilde{g}(A)|\psi\rangle\right\|_2 \leq \epsilon$. To do so, we must bound two quantities, $|\mathcal{N}_1 - \mathcal{N}_2|$ and $\||P(\psi)\rangle - \tilde{g}(A)|\psi\rangle\|_2$, so that Lemma C.1.1 may be applied. Using the definitions, we obtain

$$\||P(\psi)\rangle - \tilde{g}(A)|\psi\rangle\|_2 = \|g(A)|\psi\rangle - \tilde{g}(A)|\psi\rangle\|_2 \leq \|g(A) - \tilde{g}(A)\|_2 \leq \delta_0. \quad \text{(C.36)}$$

Now, we bound $|\mathcal{N}_1 - \mathcal{N}_2|$. For any non-negative real numbers $a, b$, we have $|\sqrt{a} - \sqrt{b}| \leq \sqrt{|a-b|}$. Consequently, $|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{|\mathcal{N}_1^2 - \mathcal{N}_2^2|}$. By definition, $g(x)x = P(x)/4\eta =: f(x)$, and so

$$|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{\left|\||P(\psi)\rangle\|_2^2 - \|\tilde{g}(A)|\psi\rangle\|_2^2\right|} = \sqrt{\left|\sum_j (|f(\psi_j)|^2 - |\phi_j|^2)\right|} \quad \text{(C.37)}$$

$$\leq \sqrt{\max_j ||f(\psi_j)| - |\phi_j|| \sum_j (|f(\psi_j)| + |\phi_j|)}. \quad \text{(C.38)}$$

Since $||a| - |b|| \leq |a - b|$, we have that $\max_j ||f(\psi_j)| - |\phi_j|| \leq \max_j |f(\psi_j) - \phi_j|$. Moreover, $\sum_j |f(\psi_j) - \phi_j|^2 = \||P(\psi)\rangle - \tilde{g}(A)|\psi\rangle\|_2^2 \leq \delta_0^2$ (as per Eq. (C.36)) which implies that $\max_j |f(\psi_j) - \phi_j| \leq \delta_0$. Of course, $\|\tilde{g}(A)|\psi\rangle\|_2^2 \leq 1$ and so $\sum_j |\phi_j|^2 \leq 1$ which implies that $\forall j, |\phi_j| \leq 1$. Similarly, by definition, $f(x) = \frac{P(x)}{4\eta}$, and since $|P(x)| \leq \eta$, $\forall j, f(\psi_j) \leq 1/4$, and so $\sum_{j=1}^N |f(\psi_j) + \phi_j| \leq 2N$. As a result,

$$|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{2N\delta_0}. \quad \text{(C.39)}$$

Combining Eq. (C.36), Eq. (C.39) and Lemma C.1.1, we then obtain the following bound on the total $\ell_2$-norm error,

$$\left\|\frac{1}{\mathcal{N}_1}|P(\psi)\rangle - \frac{1}{\mathcal{N}_2}\tilde{g}(A)|\psi\rangle\right\|_2 \leq \frac{\sqrt{2N\delta_0} + \delta_0}{\mathcal{N}_1} \leq \frac{3\sqrt{N\delta_0}}{\mathcal{N}_1}. \quad \text{(C.40)}$$

We now analyze the probability of success of the procedure, thus determining the complexity in terms of the number of amplitude amplification steps required to succeed with arbitrarily high probability. Thus, we must lower-bound $\mathcal{N}_2^2 \equiv \|\tilde{g}(A)|\psi\rangle\|_2^2$. We just showed that $|\mathcal{N}_1^2 - \mathcal{N}_2^2| \leq 2N\delta_0$, hence $\mathcal{N}_2^2 \geq \mathcal{N}_1^2 - 2N\delta_0$. Thus, requiring $\delta_0$ to be sufficiently small so as to ensure that $\mathcal{N}_1^2 > 2N\delta_0$, the complexity to boost to a constant probability of success is given by $O(\frac{1}{\sqrt{\mathcal{N}_1^2 - 2N\delta_0}})$.

To obtain an overall error of at most $\epsilon$, we set $\delta_0 = \epsilon^2\mathcal{N}_1^2/9N$. This indeed satisfies our requirement of $\mathcal{N}_1^2 > 2N\delta_0$ (for $\epsilon \leq 1$). As a result, the number of AA steps required can be simplified to $O\left(1/\mathcal{N}_1\sqrt{1-\frac{2}{9}\epsilon^2}\right) \subseteq O\left(1/\mathcal{N}_1\right)$, using $\epsilon \leq 1$. Since $\mathcal{N}_1 = \frac{1}{4\eta}\mathcal{N}$, this bound becomes $O(\eta/\mathcal{N})$. Similarly, $\delta_0 = \epsilon^2\mathcal{N}_1^2/9N = \epsilon^2\mathcal{N}^2/144\eta^2 N$. As a result, with constant probability of success, our procedure produces a normalized quantum state $\epsilon$ close (in $\ell_2$-norm) to the state $\frac{1}{\mathcal{N}}\sum_j P(\psi_j)|j\rangle$ using a total of $O(k\eta/\mathcal{N})$ queries to a controlled $U$ circuit and controlled $U^\dagger$ circuit, with $O(kn\eta/\mathcal{N})$ circuit depth, and with a total of $O(poly(k, \log(\eta^2 N/\epsilon^2\mathcal{N}^2)))$ classical computation.

□

The error bound assumes that the only error-source is the classical computation of the rotation angles in QSVT – in practice there would likely be an additional logarithmic error from the implementation of all gates on real hardware, and so there would be an additional multiplicative factor in the complexity of approximately $\log(1/\mathcal{N}_1)$ in order to sufficiently suppress this error source and ensure numerical stability.

Note that in many cases where $k$ is small, it can actually be significantly advantageous to just implement the desired polynomial via products and linear combinations of block encodings of the $A$ matrix, as opposed to using QSVT, as the polynomial can be implemented *exactly* in this manner (and for arbitrary complex initial states), albeit at the cost of using more ancillary qubits.

In Appendix C.9, I rederive the analogous result from [121] with our diagonal block-encoding, and extend it with our $\ell_2$ norm guarantee.

## C.4 Non-Linear Transformations of State Amplitudes via Polynomial Approximations

In this section, I extend our previous result (Theorem C.3.1) to arbitrary functions which can be approximated by polynomials. To this end, I begin by proving some simple results about functions and their polynomial approximations, and conclude with the main result of the section, Theorem 4.2.2.

**Lemma C.4.1.** *We are given a Lipschitz continuous real function $P(x)$ with Lipschitz constant $L$ on the interval $x \in [-1, 1]$, and the guarantee that $P(0) = 0$. We are additionally given an arbitrary $\ell_2$ normalized quantum state $|\psi\rangle = \sum_{j=1}^{N} \psi_j |\psi\rangle$. Define the normalization factor $\mathcal{N}^2 := \sum_{j=1}^{N} |P(\psi_j)|^2$. Then, $\mathcal{N} \leq L$.*

*Proof.* First, by Definition 1.2.2, and $P(0) = 0$, for all $x \in [-1, 0) \cup (0, 1]$, $|P(x)| = |P(x) - P(0)| \leq L|x| \leq L$. Then, for any $j$, $|P(\psi_j)|^2 \leq |x|^2 L^2$. Thus, since $\sum_{j=1}^{N} |\psi_j|^2 = 1$,

$$\mathcal{N}^2 = \sum_{j=1}^{N} |P(\psi_j)|^2 \leq \sum_{j=1}^{N} |\psi_j|^2 L^2 = L^2. \tag{C.41}$$

The result $\mathcal{N} \leq L$ immediately follows, for the case where $x \neq 0$. Since the input state is $\ell_2$-normalized, there must be non-zero amplitudes, and these non-zero amplitudes must have a sum of squares equal to one. Thus, this result holds in full generality, and we have shown that $\mathcal{N} \leq L$. □

Consider the definition of an efficiently-approximable function, as per Definition 1.2.3. When it is clear from context, I will drop the subscript indicating the degree of the polynomial, i.e., sometimes I will write $P_k(x)$ as $P(x)$, and $e_k(x)$ as $e(x)$. I use the notation $e_k(x) := f(x) - P_k(x)$ for the error term. Note that the subscript on the error term does not correspond to the degree of the error term, rather it makes the dependence on the degree of approximating polynomial explicit. I now derive a couple of simple technical lemmas regarding some properties of such functions and their corresponding normalizations.

**Lemma C.4.2** (Composition of uniform approximations)**.** *Let $f(x)$ be an $L_f$-Lipschitz, $(\epsilon_0, k_0)$-approximable function with approximation polynomial $P(x)$. Let $g(x)$ be an $(\epsilon_1, k_1)$-approximable function with approximation polynomial $Q(x)$. Then $f(g(x))$ is an $(\epsilon_0 + L_f\epsilon_1, k_0k_1)$-approximable function. I.e., $|f(g(x)) - P(Q(x))| \leq \epsilon_0 + L_f\epsilon_1$.*

*Proof.* This follows simply from Definition 1.2.2, and Definition 1.2.3. □

**Lemma C.4.3.** *Let $f(x)$ be $\epsilon_0$-approximable as per Definition 1.2.3 with $P(x)$ the approximation polynomial and $\gamma := \max_{x\in[-1,1]} |f(x)|$. Require that $\epsilon_0 \leq \gamma$. In addition, define the vector $|\psi\rangle := \sum_j \psi_j|j\rangle$, such that $|\psi\rangle \in \mathbb{C}^N$, and $\||\psi\rangle\|_2 = 1$. Define $\mathcal{N}^2 := \sum_{j=1}^N |f(\psi_j)|^2$ and $\mathcal{N}_1^2 := \sum_{j=1}^N |P(\psi_j)|^2$. Then,*

*i.)* $\max_{x\in[-1,1]} |P(x)| \leq 2\gamma$.

*ii.)* $|\mathcal{N} - \mathcal{N}_1| \leq 3\gamma\epsilon_0 N/\mathcal{N}$.

*If $\epsilon_0 \leq \frac{\mathcal{N}^2}{6\gamma N}$ (noting that $\frac{\mathcal{N}^2}{6\gamma N} < \gamma$),*

*iii.)* $\mathcal{N}_1 \geq \frac{1}{2}\mathcal{N}$.

*Proof.* For proving i.), by Definition 1.2.3, $|P(x)| = |P(x) - f(x) + f(x)| \leq \epsilon_0 + |f(x)| \leq 2\gamma$, for all $x \in [-1, 1]$. For proving ii.), we obtain from simple algebra that $|\mathcal{N} - \mathcal{N}_1| = \left|\frac{\mathcal{N}^2 - \mathcal{N}_1^2}{\mathcal{N}+\mathcal{N}_1}\right| \leq \frac{1}{\mathcal{N}} |\mathcal{N}^2 - \mathcal{N}_1^2|$. By definition, we have $|\mathcal{N}^2 - \mathcal{N}_1^2| \leq \sum_{j=1}^N |(f(\psi_j) - P(\psi_j))(f(\psi_j) + P(\psi_j))|$. By i.), we have that $|P(\psi_j)| \leq 2\gamma$, and hence, $|f(\psi_j)+P(\psi_j)| \leq 3\gamma$. Since $f(x)$ is $\epsilon_0$-approximable, $|f(\psi_j)-P(\psi_j)| \leq \epsilon_0$, and so we obtain $|\mathcal{N} - \mathcal{N}_1| \leq 3\gamma\epsilon_0 N/\mathcal{N}$. For proving iii.) $\mathcal{N}_1^2 = \mathcal{N}^2 - \sum_j e(\psi_j)(P(\psi_j) + f(\psi_j)) \geq \mathcal{N}^2 - 3\epsilon_0\gamma N$. If $\epsilon_0 \leq \mathcal{N}^2/6\gamma N$, then $\mathcal{N}_1^2 \geq \mathcal{N}^2/2$. □

**Lemma C.4.4.** *Let $f(x)$ be $\epsilon_0$-approximable as per Definition 1.2.3 with $P(x)$ the approximation polynomial and $\gamma := \max_{x\in[-1,1]} |f(x)|$. Define the vector $|\psi\rangle = \sum_j \psi_j|j\rangle$ such that $\||\psi\rangle\|_2 = 1$ and $|\psi\rangle \in \mathbb{C}^N$. Let $\mathcal{N}^2 := \sum_{j=1}^N |f(\psi_j)|^2$ and $\mathcal{N}_1^2 := \sum_{j=1}^N |P(\psi_j)|^2$. Define $\epsilon \in (0, 1]$ and $\Delta_k := \left\|\frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle - \frac{1}{\mathcal{N}_1}\sum_j P_k(\psi_j)|j\rangle\right\|_2$. Then, $\epsilon_0 \leq \epsilon\mathcal{N}^2/8\gamma N \implies \Delta_k \leq \epsilon$.*

*Proof.* Let $E := \mathcal{N}_1 - \mathcal{N}$. Then, pulling out a common $\frac{1}{\mathcal{N}_1\mathcal{N}}$ term and simplifying by using $\mathcal{N}_1 = E + \mathcal{N}$ in the resulting numerator term, we get $\Delta \leq (\epsilon_0\sqrt{N} + E)/\mathcal{N}_1$, where we also used $\mathcal{N} = \left\|\sum_j f(\psi_j)|j\rangle\right\|_2$.

From Lemma C.4.3 ii.), we know that $|E| \leq 3\gamma\epsilon_0 N/\mathcal{N}$. From Lemma C.4.3 iii.), we know that $\mathcal{N}_1 \geq \mathcal{N}/2$, if $\epsilon_0 \leq \mathcal{N}^2/6\gamma N$. Noting that $\gamma\sqrt{N}/\mathcal{N} \geq 1$ implies that $\gamma\epsilon_0 N/\mathcal{N} = (\epsilon_0\sqrt{N})(\gamma\sqrt{N}/\mathcal{N}) \geq \epsilon_0\sqrt{N}$, we then get,

$$\Delta_k \leq \frac{2}{\mathcal{N}}\left(\epsilon_0\sqrt{N} + \frac{3\gamma\epsilon_0 N}{\mathcal{N}}\right) \leq \frac{8\epsilon_0\gamma N}{\mathcal{N}^2}. \tag{C.42}$$

Thus, to get an overall error at most $\epsilon > 0$, we require that $\frac{8\epsilon_0\gamma N}{\mathcal{N}^2} \leq \epsilon$ and so we set $\epsilon_0 \leq \epsilon\mathcal{N}^2/8\gamma N$. Since our proof added the constraint $\epsilon_0 \leq \mathcal{N}^2/6\gamma N$ (and technically also $\epsilon_0 \leq \gamma$, but since $\mathcal{N}^2/6\gamma N \leq \gamma$, this automatically satisfied), $\epsilon_0 = \mathcal{N}^2\epsilon/8\gamma N \leq \mathcal{N}^2/6\gamma N$ and so our proof holds without further constraint if $\epsilon \leq 4/3$. □

I will now prove Theorem 4.2.2.

*Proof of Theorem 4.2.2.* We will bound the $\ell_2$-norm error from the polynomial approximation, and then the $\ell_2$-norm error from the algorithmic error, combining the two via triangle inequality. Define $\mathcal{N}_1^2 := \sum_{j=1}^N |P_k(\psi_j)|^2$. By Lemma C.4.4, if $\epsilon/2 \leq \min(1, 1/\gamma)$, then

$$\left\|\frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle - \frac{1}{\mathcal{N}_1}\sum_j P_k(\psi_j)|j\rangle\right\|_2 \leq \epsilon/2. \tag{C.43}$$

Where $\max_{x\in[-1,1]}|f(x) - P_k(x)| \leq \epsilon_0$, Lemma C.4.4 says $\epsilon/2 = 8\epsilon_0\gamma N/\mathcal{N}^2$, consequently $\epsilon_0 = \epsilon\mathcal{N}^2/16\gamma N \leq \mathcal{N}^2/6\gamma N$, and so we can apply Lemma C.4.3. This gives $\mathcal{N}_1 \geq \mathcal{N}/2$.

We will invoke Theorem C.3.1 on $P_k(x)$, thereby obtaining the approximation to the desired transformation. Thus, we can prepare the state $\frac{1}{\mathcal{N}_1}\sum_j P_k(\psi_j)|j\rangle$ up to error $\epsilon/2$ by invoking Theorem C.3.1 with $P_k(x)$. This has a query complexity to a controlled $U$ and controlled $U^\dagger$ circuit of $O(\tilde{\gamma}h(\gamma N/\epsilon\mathcal{N}^2)/\mathcal{N})$. The additional circuit complexity is thus $O(n\tilde{\gamma}h(\gamma N/\epsilon\mathcal{N}^2)/\mathcal{N})$, requiring $O(\text{poly}(\log(\gamma N/\epsilon\mathcal{N}^2), \log(\tilde{\gamma}^2 N/\epsilon^2\mathcal{N}^2)))$ classical computation, and $O(n)$ ancilla qubits. To increase the simplicity of the

reported result, we assume that $\epsilon$ is sufficiently small such that $\log(\gamma N/\epsilon\mathcal{N}^2) \leq \log(\tilde{\gamma}^2 N/\epsilon^2\mathcal{N}^2)$, allowing the classical computation complexity to be simplified to $O(\text{poly}\log(\tilde{\gamma}^2 N/\epsilon^2\mathcal{N}^2))$.

□

When $K(\epsilon) = \log(1/\epsilon)$, the query and circuit complexities simplify accordingly.

## C.5 Application: End-to-End Complexities for Applying Various Functions to Arbitrary Quantum States

**Lemma C.5.1** (Absolute error in polynomial approximation to tanh [121])**.** *Let* $f(x) = \tanh(x)$*. Define*

$$P_k(x) := \sum_{n=1}^{k} \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!} x^{2n-1}, \tag{C.44}$$

*where* $B_n$ *is the* $n^{th}$ *Bernoulli number. Then, in the interval* $x \in [-1, 1]$*, when* $k \geq 2$*,*

$$|f(x) - P_k(x)| \leq 9\left(\frac{2}{\pi}\right)^k. \tag{C.45}$$

*Thus, there exists a* $k \in O(\log(1/\epsilon))$ *such that the error is at most* $\epsilon$*.*

*Proof.* The proof follows directly from [121], with (very) minor corrections. First, define $\alpha_n := \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}$. Note that in the interval $x \in [-\pi/2, \pi/2]$, $\tanh(x) = \sum_{n=1}^{\infty} \alpha_n x^{2n-1}$. Define $e_k(x) := f(x) - P_k(x)$. Assuming $x \in [-1, 1]$, wish to bound,

$$|e_k(x)| = \left|\sum_{n=k+1}^{\infty} \alpha_n x^{2n-1}\right| \leq \sum_{n=k+1}^{\infty} |\alpha_n|. \tag{C.46}$$

Using the fact that $|B_{2n}| \leq 5\sqrt{\pi n}(\frac{n}{\pi e})^{2n}$ as per [191] (as noted in [121]),

$$|\alpha_n| \leq \left|\frac{2^{4n}B_{2n}}{(2n)!}\right| \leq \frac{5\sqrt{\pi n}}{(2n)!}\left(\frac{4n}{\pi e}\right)^{2n}. \tag{C.47}$$

If we wish to use the polynomial approximation $P_1(x)$, we are just implementing an identity transformation, and so the minimum polynomial we might use is $P_2(x)$.

As a result, without a loss of generality, we can assume that the $n$ in the error term $\sum_n^\infty \alpha x^{2n-1}$ is at least 3. Noting that $\sqrt{\pi n} \leq (\pi/2)^n$ for all $n \geq 3$, and that $n! \geq (n/e)^n$,

$$|\alpha_n| \leq 5\sqrt{\pi n}\left(\frac{2}{\pi}\right)^{2n} \leq 5\left(\frac{2}{\pi}\right)^n. \tag{C.48}$$

This then gives the bound

$$|e_k(x)| \leq 5\sum_{n=k+1}^{\infty}\left(\frac{2}{\pi}\right)^n \leq 9\left(\frac{2}{\pi}\right)^k. \tag{C.49}$$

□

I will now prove Theorem 4.3.2.

*Proof of Theorem 4.3.2.* i.) Using the standard series representation of $f(x) = e^x$, for $x \in [-1, 1]$, $f(x) = \sum_{n=0}^\infty \frac{x^n}{n!}$. We have for the error term that $|f(x) - P_k(x)| = \left|\sum_{n=k+1}^\infty \frac{x^n}{n!}\right| \leq \sum_{n=k+1}^\infty \frac{1}{n!} \leq \sum_{n=k+1}^\infty 2^{-n} = 2^{-k}$, using the fact that for all $n \geq 3$, $1/n! \leq (1/2)^n$. Moreover, $\forall x \in [-1, 1]$, $1/e \leq e^x \leq e$. As a result, $\mathcal{N}^2 = \sum_j f(\psi_j)^2 \geq N/e^2$, and so $\mathcal{N} \geq \sqrt{N}/e$. Consequently, $O(\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$ is $O(\log(1/\epsilon))$, giving a query complexity of $O(\log(1/\epsilon))$ and an associated circuit depth of $O(n\log(1/\epsilon))$ through Theorem C.9.2.

ii.) Using the standard series representation of $f(x) = \cos(x)$, for $x \in [-1, 1]$, $f(x) = \sum_{n=0}^\infty (-1)^n \frac{x^{2n}}{(2n)!}$. The error term is $|f(x) - P_k(x)| = \left|\sum_{n=k+1}^\infty (-1)^n \frac{x^{2n}}{(2n)!}\right| \leq \sum_{n=k+1}^\infty \frac{1}{(2n)!} \leq \sum_{n=k+1}^\infty \left(\frac{1}{2}\right)^n = 2^{-k}$, using the fact that for $n \geq 2$, $(2n)! \geq 2^n$. Thus, $|e_k(x)| \leq \epsilon_0$ with overall $k \in O(\log(1/\epsilon_0))$. Of course, $\min_{x\in[-1,1]} |\cos(x)| = \cos(1) > 1/2$, and so $\mathcal{N}^2 = \sum_j \cos(\psi_j)^2 \geq N/4$ which implies that $\mathcal{N} \geq \sqrt{N}/2$. Consequently, $O(\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$ is $O(\log(1/\epsilon))$, giving a query complexity of $O(\log(1/\epsilon))$ and an associated circuit depth of $O(n\log(1/\epsilon))$ through Theorem C.9.2.

iii.) We use the common identity for the logistic function of $f(x) = \frac{1}{2} + \frac{1}{2}\tanh(x)$, Consequently, to obtain a series expansion for $f(x)$, we invoke Lemma C.5.1, again adopting the notation $\alpha_n := \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}$, yielding $f(x) = \frac{1}{2}\left(1 + \sum_{n=1}^\infty \alpha_n x^{2n-1}\right)$. Consequently, letting $P_k(x) := (1/2)(1 + \sum_{n=1}^k \alpha_n x^{2n-1})$, and defining $e_k(x) := f(x) - P_k(x)$, we obtain $|e_k(x)| = \left|\frac{1}{2}\sum_{n=k+1}^\infty \alpha_n x^{2n-1}\right| \leq 5\left(\frac{2}{\pi}\right)^k$, where the bound

follows directly from Lemma C.5.1, and again we require that $k \geq 2$. Moreover, $|f(x)| \geq 1/(1+e^2) > 1/10$, thus $\mathcal{N} = \sqrt{\sum_j f(\psi_j)^2} \geq \sqrt{N}/10$. Consequently, $O(\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$ is $O(\log(1/\epsilon))$, giving a query complexity of $O(\log(1/\epsilon))$ and an associated circuit depth of $O(n\log(1/\epsilon))$ through Theorem C.9.2.

iv.) Using the fact that $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, defining $\alpha := -1/2\sigma^2$ and $\beta := \frac{1}{\sigma\sqrt{2\pi}}$, we readily obtain $f(x) = \beta e^{\alpha x^2} = \beta \sum_{n=0}^{\infty} \frac{\alpha^n x^{2n}}{n!}$.A Defining the first $k+1$ terms of this series as $P_k(x)$, and defining $e_k(x) := f(x) - P_k(x)$, when $x \in [-1,1]$, for all $k \geq 3$,

$$|e_k(x)| = \beta \left| \sum_{n=k+1}^{\infty} \frac{\alpha^n x^{2n}}{n!} \right| \leq \frac{1}{\pi} \sum_{n=k+1}^{\infty} \frac{|\alpha|^n}{n!} \leq \frac{1}{\pi} \sum_{n=k+1}^{\infty} \frac{1}{n!} \leq \frac{1}{\pi} \sum_{n=k+1}^{\infty} (1/2)^n = \frac{2^{-k}}{\pi}, \tag{C.50}$$

following from assumption, since $\sigma^2 \geq 1/2$, and so $|\alpha| \leq 1$. Moreover, since $\forall x \in [-1,1], |f(x)| \geq 1/\sigma\sqrt{2\pi}$ (since $\sigma^2 \geq 1/2$), $\mathcal{N} = \sqrt{\sum_j f(\psi_j)^2} \geq \sqrt{N}/\sigma\sqrt{2\pi}$. Consequently, $O(\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$ is $O(\sigma\log(\sigma^2/\epsilon))$, giving a query complexity of $O(\sigma\log(\sigma^2/\epsilon))$ and an associated circuit depth of $O(n\sigma\log(\sigma^2/\epsilon))$ through Theorem C.9.2.

v.) First, we use the series representation for $f(x) = \sin(x)$ in the interval $[-1,1]$ of $f(x) = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$. The error term is $|e_k(x)| = |f(x) - P_k(x)| = \left|\sum_{n=k+1}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}\right| \leq \sum_{n=k+1}^{\infty}\left|\frac{1}{(2n+1)!}\right| \leq \sum_{n=k+1}^{\infty}\frac{1}{2^n} = 2^{-k}$, using the fact that for $n \geq 4$, $n! \geq 2^n$. We now prove that $\tilde{\gamma} := \max_{x\in[-1,1]}|P_k'(x)| \leq 2$. Noting that $P_k'(x) = \sum_{n=0}^{k}\frac{(-1)^n x^{2n}}{(2n)!}$, we can observe that this is the degree $k$ truncation of the polynomial representation of $\cos(x)$, allowing us to upper-bound this value with Lemma C.4.3, giving $\tilde{\gamma} \leq 2$. Obviously, $\max_{x\in[-1,1]}|f(x)| \leq 1$. Additionally, $\lim_{x\to 0}|\sin(x)|/|x| = 1$, and for all $x \in [-1,1]$, $|\sin(x)|/|x| \geq 3/4$, implying that $\mathcal{N}^2 = \sum_j |\sin(\psi_j)|^2 \geq \frac{9}{16}$. As a result, $O(\tilde{\gamma}\log(\gamma N/\epsilon\mathcal{N}^2)/\mathcal{N})$ is $O(\log(N/\epsilon))$, giving a query complexity of $O(\log(N/\epsilon))$ and an associated circuit depth of $O(n\log(N/\epsilon))$ through Theorem 4.2.2. □

## C.6 Application: Exponential Improvement in Applying Tanh Activation Function in Machine Learning

*Proof of Theorem 4.3.1.* Lemma C.5.1 implies that $f(x)$ is an efficiently approximable function as per Definition 1.2.3. Defining $\alpha_n$ as per Lemma C.5.1, we can write $f(x) = \sum_{n=1}^{\infty} \alpha_n x^{2n-1}$, $P_k(x) = \sum_{n=1}^{k} \alpha_n x^{2n-1}$ and $e_k(x) = \sum_{n=k+1}^{\infty} \alpha_n x^{2n-1}$.

We begin by proving three facts about $f(x)$: (1) $\mathcal{N} \geq 3/4$, (2) $|f(x)| \leq 1$, and (3) $\tilde{\gamma} \leq 2$ (for $k \geq 15$, with $\tilde{\gamma}$ defined in accordance with the notation in Theorem 4.2.2). First, noting that $|f(x)|/|x| \geq 3/4$ (with the minimum at $x = \pm 1$) implying that $|f(x)| \geq 3|x|/4$, we have $\mathcal{N}^2 = \sum_j |f(\psi_j)|^2 \geq (9/16) \sum_j |\psi_j|^2 = 9/16$. Thus, $\mathcal{N} \geq 3/4$. Second, trivially $|f(x)| \leq 1$. Finally, we prove that $\tilde{\gamma} \leq 2$. By Lemma C.3.1, $\tilde{\gamma} := \max_{x\in[-1,1]} |P_k(x)/x| \leq \max_{x\in[-1,1]} |\frac{d}{dx} P_k(x)|$. To bound $\tilde{\gamma}$, we will first compute the Lipschitz constant of $f(x) = \tanh(x)$ in the interval $x \in [-1, 1]$, $L$:

$$L = \max_{x\in[-1,1]} \left| \frac{d}{dx} \tanh(x) \right| = \max_{x\in[-1,1]} \left| 1 - \tanh^2(x) \right| = 1. \tag{C.51}$$

Then,

$$\left| \frac{d}{dx} P_k(x) \right| \leq \left| \frac{d}{dx} f(x) \right| + \left| \frac{d}{dx} e_k(x) \right| \leq 1 + \sum_{n=k+1}^{\infty} (2n-1) \left( \frac{2}{\pi} \right)^n. \tag{C.52}$$

Observing that $\left(\frac{\pi}{2}\right)^{n/2} \geq (2n-1)$ for all $n \geq 16$,

$$\tilde{\gamma} \leq 1 + \sum_{n=k+1}^{\infty} \left( \frac{2}{\pi} \right)^{n/2} \leq 1 + 2 \left( \frac{2}{\pi} \right)^k \leq 2, \tag{C.53}$$

imposing the restriction that $k \geq 15$.

We now branch the proof into two cases, one using the importance weighting technique, and the other using the $\ell_2$-norm version of [121]. In both algorithms, we aim to produce an $\ell_2$-normalized quantum state $|\phi\rangle$ such that $\left\| \frac{1}{\mathcal{N}} \sum_j f(\psi_j)|j\rangle - |\phi\rangle \right\|_2 \leq \epsilon$ for some $\epsilon \geq 0$. Noting that for a fixed function the classical computation can be precomputed and so we neglect the classical computation cost in our complexity statement. To obtain our result with the importance-weighted technique, we invoke

Theorem 4.2.2 finding a query complexity (to a controlled $U$ and $U^\dagger$ circuit) of $O(\log(N/\epsilon))$, and a circuit complexity of $O(n\log(N/\epsilon))$. We obtain the result with the previous technique by invoking Theorem C.9.2, finding a query complexity of $O(\sqrt{N}\log(N/\epsilon))$ and a circuit complexity of $O(n\sqrt{N}\log(N/\epsilon))$. Thus, an exponential improvement in the end-to-end complexity of applying the $\tanh(x)$ function to an arbitrary quantum state over the previous technique has been achieved. □

## C.7 Application: Maximum Finding

I require two results from [192]: the error function approximation to the sign function, and the polynomial approximation to the shifted error function.

**Lemma C.7.1** (Shifted and scaled error function approximation to the shifted Heaviside function [192])**.** *Let $\delta > 0$, $m > 0$, $\tau \in [-1,1]$, and $\epsilon \in (0,1]$. Define the shifted Heaviside function in terms of the sign function, i.e.,* $\text{hvi}_\tau(x) := \frac{1}{2}(\text{sgn}(x-\tau)+1)$*. Define the shifted and scaled error function as* $\text{erfs}_{m,\tau}(x) = \frac{1}{2}(\text{erf}(m(x-\tau))+1)$*. When* $m = \frac{\sqrt{2}}{\delta}\log^{1/2}(\frac{2}{\pi\epsilon^2})$*,*

$$\max_{x\in[-1,\tau-\delta/2]\cup[\tau+\delta/2,1]} |\,\text{erfs}_{m,\tau}(x) - \text{hvi}_\tau(x)| \leq \epsilon, \tag{C.54}$$

*Proof.* This is another way of stating Lemma 10 of [192]. □

**Lemma C.7.2** (Polynomial approximation to the shifted and scaled error function [192])**.** *Let $\tau \in [-1,1]$, $m > 0$ and $\epsilon \in (0, O(1)]$. Then, we can efficiently construct a degree-$k$ polynomial, $P_{erf,m,\tau,k}(x)$, satisfying*

$$\max_{x\in[-1,1]} |P_{erfs,m,\tau,k}(x) - \text{erfs}_{m,\tau}(x)| \leq \epsilon \tag{C.55}$$

*with $k \in O(m\log(1/\epsilon))$. Moreover, let $y = (x-\tau)$, then*

$$P_{erfs,m,\tau,k}(x) := \frac{1}{2} + \frac{2me^{-2m^2}}{\sqrt{\pi}}\left(I_0(2m^2)y + \sum_{j=1}^{(k-1)/2} I_j(2m^2)(-1)^j\left(\frac{T_{2j+1}(y)}{2j+1} - \frac{T_{2j-1}(y)}{2j-1}\right)\right) \tag{C.56}$$

*where $I_j(\cdot)$ is a modified Bessel function of the first kind, and $T_n(\cdot)$ is a Chebyshev polynomial of the first kind.*

*Proof.* Follows immediately from Corollary 5 of [192]. □

I will now prove Theorem 4.3.3.

*Proof of Theorem 4.3.3.* Let $m > 0$, $\epsilon > 0$, $\epsilon_0 > 0$ and $\epsilon_1 > 0$. Set the parameters $\delta$ and $\tau$ in Lemma C.7.1 as $\delta = \Delta$, and $\tau := \psi_1 - \frac{\Delta}{2}$. Lemma C.7.2 proves that the function $\mathrm{erfs}_{m,\tau}(x)$ is $\epsilon_0$-uniformly approximated by the polynomial $P_{m,k}(x) := P_{erfs,m,\tau,k}(x)$ of degree $k \in O(m \log(1/\epsilon_0))$. Define the function $f_{m,\tau}(x) := x\,\mathrm{erfs}_{m,\tau}(x)$. The function $f_{m,\tau}(x)$ is $\epsilon_0$-uniformly approximated by the polynomial $xP_{m,k}(x)$, and has $\gamma := \max_{x\in[-1,1]} |f(x)| \leq 1$. Finally, $\tilde{\gamma} = \max_{x\in[-1,1]} |xP_{m,k}(x)/x| = \max_{x\in[-1,1]} |P_{m,k}(x)| \leq 2$ (for $\epsilon_0 \leq 1$). Define $\mathcal{N}^2 := \sum_j f_{m,\tau}(\psi_j)^2$. Setting $m = \frac{\sqrt{2}}{\Delta} \log^{1/2}(\frac{2}{\pi\epsilon_1^2})$, Lemma C.7.1 ensures that $\mathrm{erfs}_{m,\tau}(\psi_1) \geq 1 - \epsilon_1$, and that $\forall j \neq 1 : \mathrm{erfs}_{m,\tau}(\psi_j) \leq \epsilon_1$. Consequently, $\mathcal{N}^2 \geq f_{m,\psi_1}(\psi_1)^2 \geq (1-\epsilon_1)^2\psi_1^2 \implies \mathcal{N} \geq (1-\epsilon_1)\psi_1$. Assuming that $\epsilon_1 \leq 1/2$, $\mathcal{N} \geq \psi_1/2$. Finally, define the $\ell_2$-normalized state $|f_{m,\tau}(\psi)\rangle := \frac{1}{\mathcal{N}} \sum_j f_{m,\tau}(\psi_j)|x_j\rangle$.

We want to determine the query and circuit complexities required to obtain the $\ell_2$-normalized state $|\phi\rangle = \sum_j \phi_j|x_j\rangle$ such that $\||\phi\rangle - |x_1\rangle\|_2 \leq \epsilon$, so that we can then use it to bound the complexity of identifying the basis vector with maximum amplitude, with high probability of success.

First, we will prove that by setting $m = \frac{\sqrt{2}}{\Delta} \log^{1/2}(\frac{16\sqrt{N}}{\pi\epsilon^2})$, we get the guarantee that $\||f_{m,\tau}(\psi)\rangle - |x_1\rangle\|_2 \leq \epsilon/2$. Using $\mathcal{N}^2 \leq \psi_1^2(\mathrm{erfs}_{m,\tau}(\psi_1)^2 + \epsilon_1^2 N)$, and that for non-negative numbers $a, b$, $\sqrt{a^2+b^2} \leq a+b$, we have $\mathcal{N} \leq \psi_1(\mathrm{erfs}_{m,\tau}(\psi_1) + \epsilon_1\sqrt{N})$, and so

$$\||f_{m,\tau}(\psi)\rangle - |x_1\rangle\|_2^2 = 2\left(1 - \frac{\psi_1\,\mathrm{erfs}_{m,\tau}(\psi_1)}{\mathcal{N}}\right) \leq 2\left(\frac{\sqrt{N}\epsilon_1}{\mathrm{erfs}_{m,\tau}(\psi_1) + \sqrt{N}\epsilon_1}\right). \quad \text{(C.57)}$$

Since $\mathrm{erfs}_{m,\tau}(\psi_1) \geq 1 - \epsilon_1$, requiring $\epsilon_1 \leq 1/2\sqrt{N}$, gives $\mathrm{erfs}_{m,\tau}(\psi_1) + \sqrt{N}\epsilon_1 \geq 1/2$. Then, $\||f_{m,\tau}(\psi)\rangle - |x_1\rangle\|_2^2 \leq 4\epsilon_1\sqrt{N}$. Setting $\epsilon_1 \leq \epsilon^2/8\sqrt{N}$ ensures that $\||f_{m,\tau}(\psi)\rangle - |x_1\rangle\|_2 \leq \epsilon/2$. Thus, the error bound holds when $m = \frac{\sqrt{2}}{\Delta} \log^{1/2}(\frac{16\sqrt{N}}{\pi\epsilon^2})$.

We will now bound the complexity of preparing the state $|\phi\rangle$ such that $\||\phi\rangle - |f_{m,\tau}(\psi)\rangle\|_2 \leq \epsilon/2$. We construct the state $|\phi\rangle$ by invoking Theorem 4.2.2 transforming $|\psi\rangle$ by $f_{m,\tau}(x)$. Noting that $\tilde{\gamma} \leq 2$, $\mathcal{N} \geq 1/\psi_1$, and that the degree of the approximation

polynomial is given asymptotically by $O(m\log(1/\epsilon_0))$. Noting that Theorem 4.2.2 gives the preceding $\epsilon/2$ error bound when $\epsilon_0 \leq \frac{\epsilon\mathcal{N}^2}{2\gamma N}$, the degree of the approximating polynomial is thus $k \in O(\log(N/\epsilon\psi_1^2)\log^{1/2}(\sqrt{N}/\epsilon^2)/\Delta) = \tilde{O}(1/\Delta)$. This then gives the guarantee that $\||\phi\rangle - |f_{m,\tau}(\psi)\rangle\|_2 \leq \epsilon/2$ with query complexity $\tilde{O}(\psi_1^{-1}\Delta^{-1})$ to controlled $U$ and $U^\dagger$ circuits, with $\tilde{O}(n\psi_1^{-1}\Delta^{-1})$ additional circuit depth.

Thus, we have prepared a state $|\phi\rangle$ satisfying $\||\phi\rangle - |x_1\rangle\|_2 \leq \epsilon$. Since $\||\phi\rangle - |x_1\rangle\|_2 \leq \epsilon$ implies $2(1 - \langle x_1|\phi\rangle) \leq \epsilon^2$, it is easy to show that the probability of successfully measuring $|x_1\rangle$, i.e., $\phi_1^2$, is lower-bounded as $\phi_1^2 \geq (1 - \frac{\epsilon^2}{2})^2$. Setting, e.g., $\epsilon = 0.1$ ensures that the procedure succeeds with probability at least 99/100. □

## C.8 Application: Continuous State Preparation

There are a number of other techniques in the state preparation literature that were not mentioned in the main text of this chapter (with some mentioned instead in Chapter 3), with different approaches that in some cases exploit additional structure, e.g., via matrix product states (MPS) [62, 133], for efficiently integrable function [91], and a number of techniques both in the variational setting (e.g., [105, 193, 194]), and the in the distribution-specific setting (e.g., [108, 184]).

I will now provide a theorem stating the complexity of state preparation in the non-linear amplitude transformation framework, giving a simple sketch for the proof.

I now re-derive [132] in our framework, presenting the proof of Theorem 4.3.4.

*Proof.* Define the $n$-qubit state $|\psi_0\rangle_n := \frac{1}{\mathcal{N}_0}\sum_{j=0}^{N-1}\sin(j/N)|j\rangle_n$ where $N = 2^n$ and $\mathcal{N}_0^2 := \sum_{j=0}^{N-1}\sin(j/N)^2$. There exists a unitary $U_{\psi_0}$ which is an $(1,1,0)$-VE for $|\psi_0\rangle$, where $\alpha = \sqrt{N}/\mathcal{N}_0$. Note that a unitary with the action

$$U_{\psi_0}|0\rangle_{1+n} = \frac{1}{\sqrt{N}}\left(\sum_{j=0}^{N}\left(\sin(j/N)|0\rangle_1 + \cos(j/N)|1\rangle_1\right)|j\rangle_n\right)$$

is essentially a special case of the standard "rotate" gate (e.g., as used implicitly in [15]) only with Hadamards first applied. As such, it can be implemented by first applying $(I_1 \otimes H^{\otimes n})$ (with $H$ the standard Hadamard gate), followed by a sequence of $n$ controlled $e^{-iY\theta_j}$ gates (with $Y$ the standard Pauli-$Y$ gate) with the target qubit

the ancilla, and the control of the $j^{th}$ gate set to the $j^{th}$ most-significant qubit of the main register (and with angle $2^{-(j+1)}$, counting from 1 to $n$), finally followed by a Pauli-$X$ gate on the ancilla qubit. We can then rewrite the action of this gate as $U_{\psi_0}|0\rangle_{1+n} = (\frac{\mathcal{N}_0}{\sqrt{N}})|0\rangle_1|\psi_0\rangle + \sqrt{1-(\frac{\mathcal{N}_0}{\sqrt{N}})^2}|\perp\rangle_{1+n}$ (for some orthogonal state $|\perp\rangle_{1+n}$, such that $\langle\perp|_{1+n}|0\rangle_1|\psi\rangle_n = 0$). As such, by Lemma C.10.1, we immediately find that $U_{\psi_0}$ is an $(\sqrt{N}/\mathcal{N}_0, 1, 0)$-VE for $|\psi_0\rangle_n$. Using observations first made in [2] about the asymptotic properties of continuous functions, it is clear that $\lim_{N\mapsto\infty}\sqrt{N}/\mathcal{N}_0 < 2$. Moreover, one can readily show that in the finite case, $\sqrt{N}/\mathcal{N}_0 \leq 4$.

On the interval $x \in [-0.85, 0.85]$, as per Lemma 70 of [26], $\arcsin(x)$ is $\epsilon_1$-approximable by a polynomial $Q_l(x)$ with degree $l \in O(\log(1/\epsilon_1))$. By assumption, the input function $f$ is $\epsilon_0$-approximable by a polynomial $P_k(x)$ with degree $O(polylog(1/\epsilon_0))$ on the interval $[a, b]$. Consequently, Lemma C.4.2 says that the function $f((b-a)\arcsin(x)+a)$ is $\epsilon_0 + L\epsilon_1$ approximable by the polynomial $\tilde{P}_m := P_k((b-a)Q_l(x)+a)$ (where $m := kl$). Noting that $L$ is a constant, we can exponentially suppress $\epsilon_1$, meaning that effectively $m \in O(polylog(1/\epsilon_0))$.

As a result, we can invoke Theorems C.9.2 and C.10.1 with $\tilde{P}_m$ and our $(O(1), 1, 0)$-VE, obtaining the state $|\phi\rangle_n$ with total circuit complexity $\tilde{O}(\mathcal{F}^{-1})$, noting that $\sqrt{N}/\mathcal{N} = \mathcal{F}^{-1}$. □

## C.9 Comparison to [121]

In Theorem C.9.1, I derive an analogous guarantee to that in [121], with the same asymptotic properties, only using our modified diagonal block encoding framework. I also extend their result to $\ell_2$-norm error bound, which was not included in their work. The proof mirrors that in Theorem C.3.1. I subsequently derive a new result in Theorem C.9.2 (building on Theorem C.9.1) with end-to-end complexity guarantees in the case where a non-polynomial function (with an efficient uniform approximation by a polynomial) is to be applied. The proof mirrors that in Theorem 4.2.2.

**Theorem C.9.1** (Modification of non-linear transformation of real amplitudes from [121] with $\ell_2$-error bounds)**.** *Given a state preparation unitary* $U|0\rangle_n = |\psi\rangle_n =$

*$\sum_j \psi_j|j\rangle_n$ (such that $\forall j, \psi_j \in \mathbb{R}$), and a degree $k$ polynomial function, $P(x) = \sum_{j=0}^{k} a_j x^j$, such that $\gamma := \max_{x\in[-1,1]} |P(x)|$, and $\forall j \in [k+1], a_j \in \mathbb{C}$, define $\mathcal{N}^2 := \sum_j |P(\psi_j)|^2$. Then, the technique in [121] constructs an $\ell_2$-normalized quantum state $|\phi\rangle$ such that $\left\||\phi\rangle - \frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle\right\|_2 \leq \epsilon$ with arbitrarily high probability of success with a total of $O(k\gamma\sqrt{N}/\mathcal{N})$ queries to a controlled-$U$ circuit and a controlled-$U^\dagger$ circuit, with $O(kn\gamma\sqrt{N}/\mathcal{N})$ additional circuit depth, and with a total of $O(poly(k, \log(N\gamma^2/\epsilon^2\mathcal{N}^2)))$ classical computation. This requires $O(n)$ ancillary qubits.*

*Proof.* Let $f(x) := P(x)/4\gamma$, i.e. $\forall x, |f(x)| \leq 1/4$. Next, we invoke Lemma C.2.5 to construct a $(1, n+4, \delta_0)$-block-encoding of $f(A)$, called $U_f$, with $O(kn)$ circuit depth, using a total of $k$ calls to a controlled $U_{\mathrm{Re}(A)}$ and $U^\dagger_{\mathrm{Re}(A)}$ circuit, and with $O(poly(k, \log(1/\delta_0)))$ classical time complexity. Then, in direct analogy to [120, 121], we apply $U_f$ to the state $|0\rangle_{n+4}|+^n\rangle$ (where $|+^n\rangle := \frac{1}{\sqrt{N}}\sum_j |j\rangle$). Since our block-encoding has error $\delta_0$, we actually implement the transformation $\tilde{f}(A)$, where $\tilde{f}(A)$ is some linear operator satisfying $\left\|\tilde{f}(A) - f(A)\right\|_2 \leq \delta_0$. Define $\mathcal{N}_1 := \|f(A)|+^n\rangle\|_2$ and $\mathcal{N}_2 := \left\|\tilde{f}(A)|+^n\rangle\right\|_2$. Then, noting that $\frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle = \frac{1}{\mathcal{N}_1} f(A)|+^n\rangle$, we wish to bound the total error $\epsilon$, in $\left\|\frac{1}{\mathcal{N}_1} f(A)|+^n\rangle - \frac{1}{\mathcal{N}_2}\tilde{f}(A)|+^n\rangle\right\|_2 \leq \epsilon$, and we also wish to bound the probability of success.

Again, to bound the error, we must bound two quantities, $|\mathcal{N}_1 - \mathcal{N}_2|$ and $\left\|f(A)|+^n\rangle - \tilde{f}(A)|+^n\rangle\right\|_2$, so that Lemma C.1.1 may be applied. The bound of,

$$\left\|f(A)|+^n\rangle - \tilde{f}(A)|+^n\rangle\right\|_2 \leq \delta_0 \tag{C.58}$$

follows trivially from the fact that for any operator $M$ and for any vector $\boldsymbol{z}$, $\|M\boldsymbol{z}\|_2 \leq \|M\|_2 \|\boldsymbol{z}\|_2$, and the fact that $\left\|\tilde{f}(A) - f(A)\right\|_2 \leq \delta_0$.

Define the notation $\tilde{f}(A)|+^n\rangle = \frac{1}{\sqrt{N}}\sum_j \phi_j|j\rangle$. For any non-negative real numbers $a, b$, we have $|\sqrt{a} - \sqrt{b}| \leq \sqrt{|a-b|}$. Consequently, $|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{|\mathcal{N}_1^2 - \mathcal{N}_2^2|}$. Thus,

$$|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{\left|\frac{1}{N}\sum_j |f(\psi_j)|^2 - |\phi_j|^2\right|} \leq \sqrt{\frac{1}{N}\max_j ||f(\psi_j)| - |\phi_j|| \sum_j (|f(\psi_j)| + |\phi_j|)}. \tag{C.59}$$

Since $||a|-|b|| \leq |a-b|$, $\max_j ||f(\psi_j)| - |\phi_j|| \leq \max_j |f(\psi_j) - \phi_j|$. From Eq. (C.58) we have that $\sum_j |f(\psi_j) - \phi_j|^2 \leq \delta_0^2 N$, implying that $\forall j, |f(\psi_j) - \phi_j| \leq \delta_0\sqrt{N}$. As a result,

$$|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{\frac{\delta_0}{\sqrt{N}} \sum_j (|f(\psi_j)| + |\phi_j|)}. \tag{C.60}$$

Observing that $\left\| f(A)|+^n\rangle + \tilde{f}(A)|+^n\rangle \right\|_2^2 \leq 4$, $\forall j, |f(\psi_j) + \phi_j| \leq 2\sqrt{N}$. Thus,

$$|\mathcal{N}_1 - \mathcal{N}_2| \leq \sqrt{2\delta_0 N}. \tag{C.61}$$

Combining Eq. (C.58), Eq. (C.61) and Lemma C.1.1, we then obtain the following bound on the total $\ell_2$-norm error,

$$\left\| \frac{1}{\mathcal{N}_1} f(A)|+^n\rangle - \frac{1}{\mathcal{N}_2} \tilde{f}(A)|+^n\rangle \right\|_2 \leq \frac{\sqrt{2\delta_0 N} + \delta_0}{\mathcal{N}_1} \leq \frac{16 N \gamma \sqrt{\delta_0}}{\mathcal{N}}, \tag{C.62}$$

using the fact that $\mathcal{N}_1 = \mathcal{N}/4\gamma\sqrt{N}$. We must now lower-bound the probability of success, and thus upper-bound the number of amplitude amplification steps required to obtain an arbitrarily high probability of success. The probability of success of the procedure is given by $\left\| \tilde{f}(A)|+^n\rangle \right\|_2^2 = \mathcal{N}_2^2$. From $|\mathcal{N}_1^2 - \mathcal{N}_2^2| \leq 2\delta_0 N$, follows that $\mathcal{N}_2^2 \geq \mathcal{N}_1^2 - 2\delta_0 N$. Imposing the restriction that $\mathcal{N}_1^2 \geq 2\delta_0 N$, which as we show later does not result in a loss of generality, this gives a number of AA steps required of $O(1/\sqrt{\mathcal{N}_1^2 - 2\delta_0 N})$. To obtain an overall error of at most $\epsilon$, set $\delta_0 = \epsilon^2 \mathcal{N}^2 / 256 \gamma^2 N^2$, satisfying the requirement that $\mathcal{N}^2/N \geq 32\delta_0 N$, since $1 \geq \mathcal{N}/\gamma\sqrt{N}$. This results in the upper-bound on the number of AA steps required of $O(\gamma\sqrt{N}/\mathcal{N})$. Thus, since we require $O(\gamma\sqrt{N}/\mathcal{N})$ queries to the block-encoding of $f(A)$, the overall algorithm succeeds with arbitrarily high probability of success with a total of $O(k\gamma\sqrt{N}/\mathcal{N})$ queries to a controlled $U$ and controlled $U^\dagger$ circuit, with a total of $O(kn\gamma\sqrt{N}/\mathcal{N})$ circuit depth, and with $O(poly(k, \log(N\gamma^2/\epsilon^2\mathcal{N}^2)))$ classical time complexity. □

**Theorem C.9.2.** *We are given an n-qubit circuit $U$ such that $U|0\rangle = \sum_{j=1}^N \psi_j |j\rangle$, with $N = 2^n$ and $\forall j, \psi_j \in \mathbb{R}$, and a function $f(x)$ with $f(0) > 0$, $\gamma := \max_{x\in[-1,1]} |f(x)|$, and $\mathcal{N}^2 := \sum_{j=1}^N |f(\psi_j)|^2$. For $\epsilon > 0$, let $f(x)$ be $\frac{\gamma N}{\epsilon \mathcal{N}^2}$-approximable by a polynomial $P_k(x)$ with degree $k = k(\gamma N/\epsilon\mathcal{N}^2)$, where $k$ is a function describing*

*the degree of the polynomial in terms of the error. Additionally, define* $\gamma := \max_{x\in[-1,1]} |P_k(x)/x|$. *Then, the algorithm of [121], as described in Theorem C.9.1, can prepare a state* $\epsilon$ *close in* $\ell_2$ *norm-distance (for small* $\epsilon$*) from* $\frac{1}{\mathcal{N}}\sum_j f(\psi_j)|j\rangle$ *(where* $\mathcal{N}$ *is the normalization factor) with arbitrarily high probability of success with* $O(\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$ *query complexity to a controlled* $U$ *and* $U^\dagger$ *circuit, with overall circuit depth of* $O(n\log(\gamma N/\epsilon\mathcal{N}^2)\gamma\sqrt{N}/\mathcal{N})$, $O(polylog(N\gamma^2/\epsilon^2\mathcal{N}^2))$ *classical computation, and* $O(n)$ *ancillary qubits.*

*Proof.* As in Theorem 4.2.2, we bound the $\ell_2$-norm error from the polynomial approximation, and then the $\ell_2$-norm error from the algorithmic error, combining the two via the triangle inequality. □

## C.10 Vector Encodings

Note that there is some overlap between the contents of this section and those in Chapter 5. In Chapter 5 improved versions of some of the following results are given, but I include the results in this section as they are a part of the contributions of the paper which this chapter presents.

So far, I have used the language of state preparation unitaries, as per Definition C.2.1. However, states are often produced utilizing ancillary qubits or projective partial measurements which are not naturally captured in this framework. Consequently, I introduce the notion of a vector encoding (VE) in Definition 1.2.7 to generalize our previous results. VEs are a special case of block-encodings (as proposed by [26], building off of the previous work of e.g., [192, 195, 196]) only imposing that the $a+n$ qubit unitary block-encoding is applied to the $|0\rangle_{a+n}$ state. The results in this section will be obvious to authors familiar with the QSVT and QSP literature, but I include these results for the sake of generality. This is also a crucial stepping-stone for the acceleration of multi-layer neural networks, as the action of each layer of the network will produce a new VE.

In Lemma C.10.1, I provide a simple and common example of an VE, which is commonly encountered in the literature. In Lemma C.10.2, I show how an VE

can be trivially converted to a state preparation unitary in a higher-dimensional space through fixed-point amplitude amplification. In Lemma C.10.3 I prove that amplitude transformation procedures (which can be thought of as mappings between VEs) are not sensitive to perturbations in their input states. I then provide a simple observation in Theorem C.10.1 which shows how Theorems 4.2.2, C.3.1, C.9.1 and C.9.2 can be used in this different input model essentially without modification.

**Remark C.10.1.** *A $(1, 0, 0)$-VE is simply a state preparation unitary, as per Definition C.2.1.*

**Lemma C.10.1** (A standard VE)**.** *Define the $n$ qubit quantum state $|\psi\rangle_n$, and the $a + n$ qubit state $|\perp\rangle_{a+n}$, such that $\langle\perp|_{a+n}(|0\rangle_a \otimes I_n) = \mathbf{0}$ (i.e., the all zero vector). Then, any $n + a$ qubit unitary satisfying*

$$U_\psi|0\rangle_{a+n} = (1/\alpha)|0\rangle_a|\psi\rangle_n + \sqrt{1 - |1/\alpha|^2}|\perp\rangle_{a+n}, \tag{C.63}$$

*is an $(\alpha, a, 0)$-VE for the $n$-qubit quantum state $|\psi\rangle_n$.*

*Proof.* This follows immediately from Definition 1.2.7. □

**Lemma C.10.2.** *Let $\epsilon_0 \in [0, 1/2]$, $\alpha \geq 1$, $a \in \mathbb{N}$. Given a unitary $U_\psi$, an $(\alpha, a, \epsilon_0)$-VE for the $\ell_2$-normalized quantum state $|\psi\rangle_n$ with circuit complexity $O(T)$, we can construct a $(1, a + 3, \sqrt{2\epsilon_0} + \epsilon_1)$-VE with circuit complexity $O(T(a + n)\alpha \log(1/\epsilon_1))$ and with $O(\alpha \log(1/\epsilon_1))$ queries to a $U_\psi$ and $U_\psi^\dagger$ circuit.*

*Proof.* This follows similarly to the fixed-point amplitude amplification result of [26], which is an improved version of [197] (the later, clearly not having been cast in the block-encoding framework). I give a simple sketch of the procedure, noting that it is possible to implement this with fewer ancilla qubits than we show.

Define the projector $\Pi := |0\rangle\langle 0|_a \otimes I_n$. Define the $a + n$ qubit operator $A := \Pi U_\psi |0\rangle\langle 0|_{a+n}$. Define the (not necessarily $\ell_2$-normalized) vector $|\phi\rangle := (\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$. Then, it is easy to show that $A = (|0\rangle_a|\phi\rangle_n)\langle 0|_{a+n}$.

We can construct a $(1, 1, 0)$-block-encoding of $\Pi$, by applying a not gate to the single ancilla of the new block-encoding, and then applying a multiple-controlled $X$

gate targeting the ancilla, with the $a$ controls conditioned on the $|0\rangle_a \otimes I_n$ state of the main register. We can similarly construct a $(1,1,0)$-block-encoding of $|0\rangle\langle 0|_{a+n}$ with the same circuit, only with the $a+n$ controls conditioned on the $|0\rangle_{a+n}$ state of the main register instead. Noting that $U_\psi$ is a $(1,0,0)$-block-encoding of itself, we can then take the product of the three preceding block encodings in the appropriate order (via Lemma 1.2.2) to construct the unitary $U_A$, a $(1,2,0)$-block-encoding of $A$.

Let $\mathcal{N}_\phi := \||\phi\rangle\|_2$, and define $|\tilde{\phi}\rangle := |\phi\rangle/\mathcal{N}_\phi$. Clearly, $A$ is rank-one, and has non-zero singular value $\|A\|_2 = \mathcal{N}_\phi$. We need to lower-bound this value, so that we know what threshold to set in our application of the sign function, to map this singular value to one. By definition of an VE, we know that $\||\psi\rangle_n - \alpha|\phi\rangle_n\|_2 \leq \epsilon_0$, which by reverse-triangle inequality implies that $\mathcal{N}_\phi \geq (1-\epsilon_0)/\alpha \geq 1/(2\alpha)$ (additionally using the assumption that $\epsilon_0 \leq 1/2$).

Following e.g., Theorems 26 and 27 of [26], we can construct an odd polynomial of degree $k \in O(\alpha \log(1/\epsilon_1))$ approximating the sign function on the interval $[-1,1]\backslash(-1/\alpha, 1/\alpha)$ which can be applied to our block encoding to effectively bring its singular value $\epsilon_1$-close to 1, using an additional ancillary qubit and $k$ calls to $U_A$ and $U_A^\dagger$. This yields the unitary $U_{\tilde{A}}$, a $(1,3,\epsilon_1)$-block-encoding of $\tilde{A} := (|0\rangle_a|\tilde{\phi}\rangle_n)\langle 0|_{a+n}$ such that $\left\|\tilde{A} - (\langle 0|_3 \otimes I_{a+n})U_{\tilde{A}}(|0\rangle_3 \otimes I_{a+n})\right\|_2 \leq \epsilon_1$. Let $B := (\langle 0|_3 \otimes I_{a+n})U_{\tilde{A}}(|0\rangle_3 \otimes I_{a+n})$, and define $E := \tilde{A} - B$, i.e. $\|E\|_2 \leq \epsilon_1$. Noting that $U_A$ asymptotically consists of a call to $U_\psi$ and an $a+n$ controlled $X$ gate, and that an $a+n$ controlled-Toffoli can be decomposed into single and two qubit gates with depth $O(a+n)$ (see e.g., [190]), $U_{\tilde{A}}$ has circuit complexity $O(T(a+n)\alpha \log(1/\epsilon_1))$.

We will now show that $U_{\tilde{A}}$ is a $(1, a+3, \sqrt{2\epsilon_0} + \epsilon_1)$-VE for $|\psi\rangle$. To determine what VE we have, we must bound $\||\psi\rangle_n - (\langle 0|_{3+a} \otimes I_n)U_{\tilde{A}}|0\rangle_{3+a+n}\|_2$. Noting that $|0\rangle_{3+a+n} = (|0\rangle_3 \otimes I_{n+a})(|0\rangle_{a+n})$ and that $\langle 0|_{3+a} \otimes I_n = (\langle 0|_a \otimes I_n)(\langle 0|_3 \otimes I_{a+n})$, $(\langle 0|_{3+a} \otimes I_n)U_{\tilde{A}}|0\rangle_{3+a+n} = (\langle 0|_a \otimes I_n)B|0\rangle_{a+n}$. Consequently, since $\tilde{A} = (|0\rangle_a|\tilde{\phi}\rangle_n)\langle 0|_{a+n}$, and inserting $B = \tilde{A} - E$,

$$\||\psi\rangle_n - (\langle 0|_a \otimes I_n)B|0\rangle_{a+n}\|_2 \leq \left\||\psi\rangle_n - |\tilde{\phi}\rangle_n\right\|_2 + \|E\|_2 . \tag{C.64}$$

Noting that applying the reverse triangle inequality to $\||\psi\rangle - \alpha|\phi\rangle\|_2 \leq \epsilon_1$ implies that $\alpha\mathcal{N}_\phi \leq 1+\epsilon_0$, $1-\epsilon_0 \leq \alpha\mathcal{N}_\phi$, and $\frac{1+\alpha^2\mathcal{N}_\phi^2-\epsilon_0^2}{2\alpha} \leq \mathrm{Re}(\langle\psi|\phi\rangle)$, assuming that $\epsilon_0 \leq 1$, it can be easily shown that $\left\||\psi\rangle - |\tilde{\phi}\rangle\right\|_2 \leq \sqrt{2\epsilon_0}$. As a result, we get the final bound, $\||\psi\rangle_n - (\langle 0|_a \otimes I_n)B|0\rangle_{a+n}\|_2 \leq \sqrt{2\epsilon_0} + \epsilon_1$, and so $U_{\tilde{A}}$ is a $(1, a+3, \sqrt{2\epsilon_0}+\epsilon_1)$-VE for $|\psi\rangle_n$.

The circuit and query complexities follow from simple bookkeeping. □

**Lemma C.10.3.** *Define the $N$-dimensional quantum states $|\psi\rangle := \sum_j \psi_j|j\rangle$, and $|\phi\rangle := \sum_j \phi_j|j\rangle$, where both states are $\ell_2$-normalized. Let $f : \mathbb{C} \mapsto \mathbb{C}$ be an $L$-Lipschitz continuous function. Define $\gamma := \max_{x\in[-1,1]} |f(x)|$. Define the unnormalized quantum states $|f(\psi)\rangle := \sum_j f(\psi_j)|j\rangle$, $|f(\phi)\rangle := \sum_j f(\phi_j)|j\rangle$, and normalization constants $\mathcal{N}_\psi := \||f(\psi)\rangle\|_2$ and $\mathcal{N}_\phi := \||f(\phi)\rangle\|_2$. Then, if $\||\phi\rangle - |\psi\rangle\|_2 \leq \epsilon_0$,*

$$\left\|\frac{1}{\mathcal{N}_\psi}|f(\psi)\rangle - \frac{1}{\mathcal{N}_\phi}|f(\phi)\rangle\right\|_2 \leq 3\gamma L\epsilon_0 N/\mathcal{N}_\psi^2. \tag{C.65}$$

*Proof.* Using the fact that for any vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2$, we immediately get that $\max_j |\phi_j - \psi_j| \leq \epsilon_0$ as well. Denote the corresponding $\ell_2$-normalized quantum states as $|\tilde{f}(\psi)\rangle := |f(\psi)\rangle/\mathcal{N}_\psi$ and $|\tilde{f}(\phi)\rangle := |f(\phi)\rangle/\mathcal{N}_\phi$. We will now prove upper-bounds on $|\mathcal{N}_\psi - \mathcal{N}_\phi|$ and $\||f(\psi)\rangle - |f(\phi)\rangle\|_2$ and a lower-bound on $\mathcal{N}_\phi$ so that we may apply Lemma C.1.1 to bound $\left\||\tilde{f}(\psi)\rangle - |\tilde{f}(\phi)\rangle\right\|_2$. Since $f$ has a Lipschitz constant of $L$, and $|\psi_j - \phi_j| \leq \epsilon_0$, $|f(\psi_j) - f(\phi_j)| \leq L\epsilon_0$. Consequently, by simple algebra with the square of the norm of the difference, we get $\||f(\psi)\rangle - |f(\phi)\rangle\|_2 \leq L\epsilon_0\sqrt{N}$. Moreover, $\frac{\gamma\sqrt{N}}{\mathcal{N}_\psi} \geq 1 \implies \epsilon_0\gamma N/\mathcal{N}_\psi \geq \epsilon_0\sqrt{N}$, and so $\||f(\psi)\rangle - |f(\phi)\rangle\|_2 \leq L\epsilon_0\gamma N/\mathcal{N}_\psi$. Additionally, since $|\mathcal{N}_\psi^2 - \mathcal{N}_\phi^2| \leq 2\gamma L\epsilon_0 N$, then $|\mathcal{N}_\psi - \mathcal{N}_\phi| = \frac{|\mathcal{N}_\psi^2 - \mathcal{N}_\phi^2|}{|\mathcal{N}_\psi + \mathcal{N}_\phi|} \leq |\mathcal{N}_\psi^2 - \mathcal{N}_\phi^2|/\mathcal{N}_\psi \leq (2\gamma L\epsilon_0 N)/\mathcal{N}_\psi$. Using $\max\{\mathcal{N}_\psi, \mathcal{N}_\phi\} \geq \mathcal{N}_\psi$, we can then invoke Lemma C.1.1, getting the bound $\left\||\tilde{f}(\psi)\rangle - |\tilde{f}(\phi)\rangle\right\|_2 \leq 3\gamma L\epsilon_0 N/\mathcal{N}_\psi^2$. To get the overall error-bound of $\epsilon$, we can set $\epsilon_0 \leq \frac{\mathcal{N}_\psi^2\epsilon}{3L\gamma N}$. □

**Definition C.10.1** (State amplitude transformation procedure)**.** *A state amplitude transformation procedure (as in e.g., Theorems 4.2.2, C.3.1, C.9.1 and C.9.2) accepts a state preparation unitary $U$ such that $U|0\rangle_n = \sum_j \psi_j|j\rangle_n =: |\psi\rangle_n$, and an $L$-Lipschitz continuous function $f : \mathbb{C} \mapsto \mathbb{C}$. Define $|\tilde{f}(\psi)\rangle_n := \frac{1}{N}\sum_j f(\psi_j)|j\rangle_n$*

*(where $\mathcal{N}^2 := \sum_j |f(\psi_j)|^2$). It then produces a quantum state $|\phi\rangle$, satisfying $\left\| |\tilde{f}(\psi)\rangle_n - |\phi\rangle_n \right\|_2 \leq \epsilon$ with $O(T_{n,\epsilon})$ query complexity to a controlled $U$ and $U^\dagger$ circuit, and with $\tilde{O}(T_{n,\epsilon})$ circuit complexity. The subscripts make the error-dependence and dimension-dependence explicit.*

Note that for an $(\alpha, a, \epsilon)$-VE, $\alpha$ can be intuitively thought of as an estimate proportional to the amount of amplitude amplification required to bring the norm of the relevant subspace of the VE to near unity. In the following theorem, I show that up to logarithmic factors (and a multiplicative $\alpha$ term), state amplitude transformation procedures operating on state preparation unitaries and state amplitude transformation procedures operating VEs have equivalent complexity.

**Theorem C.10.1.** *We are given an L-Lipschitz continuous function $f : \mathbb{C} \mapsto \mathbb{C}$. Define $\gamma := \max_{x\in[-1,1]} |f(x)|$. Given a state amplitude transformation procedure (as per Definition C.10.1) which accepts an $n$ qubit state preparation unitary $U$ such that $U|0\rangle_n = \sum_j \psi_j |j\rangle_n$ and outputs an $\ell_2$-normalized quantum state $\epsilon$-close (in $\ell_2$-norm distance) to $|\tilde{f}(\psi)\rangle_n := \frac{1}{\mathcal{N}} \sum_j f(\psi_j)|j\rangle_n$ (where $\mathcal{N}^2 := \sum_j |f(\psi_j)|^2$) with $O(T_{n,\epsilon})$ queries to a controlled $U$ and $U^\dagger$ circuit and with $\tilde{O}(T_{n,\epsilon})$ circuit complexity, then there exists a procedure which accepts a unitary $U_\psi$, an $(\alpha, a', 0)$-VE for $|\psi\rangle_n$ with circuit complexity $T_\psi$, and outputs a state $\epsilon$-close to $|\tilde{f}(\psi)\rangle_n$ with $O(\alpha T_{a+n,\epsilon} \log(\frac{\gamma^2 L^2 N^2}{\epsilon^2 \mathcal{N}^4}))$ query complexity to a controlled $U_\psi$ and $U_\psi^\dagger$ circuit, and with $\tilde{O}(\alpha T_{a'+n,\epsilon} \log(\frac{\gamma^2 L^2 N^2}{\epsilon^2 \mathcal{N}^4}))$ total additional circuit depth.*

*Proof.* Let $a := a' + 3$, and $\epsilon_0, \epsilon_0' \geq 0$. Let $\epsilon_0 = \sqrt{2\epsilon_0'}$. Using Lemma C.10.2, we can convert $U_\psi$, an $(\alpha, a', \epsilon_0')$-VE for $|\psi\rangle_n$, into $U_A$, a $(1, a, \epsilon_0 + \epsilon_1)$-VE for $|\psi\rangle_n$ with circuit complexity $O(T(a+n)\alpha \log(1/\epsilon_1))$ and with $O(\alpha \log(1/\epsilon_1))$ queries to $U_\psi$.

Define the set of $2^a$ orthogonal projectors, $\{p_j\}_j$ with $p_j := |j\rangle\langle j|_a \otimes I_n$. Then,

$$\| |0\rangle_a |\psi\rangle_n - U_A |0\rangle_{a+n} \|_2^2 = \| |0\rangle_a |\psi\rangle_n - p_0 U_A |0\rangle_{a+n} \|_2^2 + \sum_{j=1}^{2^a - 1} \| p_j U_A |0\rangle_{a+n} \|_2^2 . \quad \text{(C.66)}$$

We will now use the definition of $U_A$ as an VE twice, i.e. that

$$\| |\psi\rangle_n - (\langle 0|_a \otimes I_n) U_A |0\rangle_{a+n} \|_2 \leq \epsilon_0 + \epsilon_1. \quad \text{(C.67)}$$

First, $\||0\rangle_a|\psi\rangle_n - p_0|0\rangle_{a+n}\|_2 \leq \epsilon_0 + \epsilon_1$ trivially following by applying the definition, and by factoring out a common $|0\rangle_a \otimes I_n$ term. Next, by using the definition, and by using the reverse triangle inequality, it can be easily shown that $\|p_0 U_A|0\rangle_{a+n}\|_2^2 \geq (1-\epsilon_0-\epsilon_1)^2$. Similarly, by observing that $1-\|p_0 U_A|0\rangle_{a+n}\|_2^2 = \sum_{j=1}^{2^a-1}\|p_j U_A|0\rangle_{a+n}\|_2^2$, it is easy to prove $\sum_{j=1}^{2^a-1}\|p_j U_A|0\rangle_{a+n}\|_2^2 \leq 1-(1-\epsilon_0-\epsilon_1)^2$. Consequently, we get the overall bound $\||0\rangle_a|\psi\rangle_n - U_A|0\rangle_{a+n}\|_2 \leq \sqrt{2(\epsilon_0+\epsilon_1)} =: \delta$.

If the function $f$ is applied to the state $|0\rangle_a|\psi\rangle_n$ without error, the state $|0\rangle_a|\tilde{f}(\psi)\rangle_n$ is obtained. Our ultimate goal is to bound the error and associated complexity in obtaining the state $|\phi\rangle_{a+n}$ such that $\left\||0\rangle_a|\tilde{f}(\psi)\rangle_n - |\phi\rangle_{a+n}\right\|_2 \leq \epsilon$. To do this, note that our procedure has two primary sources of error: error in the implementation of the transformation corresponding to $f$, and error in the state input to the transformation procedure. Let $|\lambda\rangle_{a+n}$ correspond to the state produced by a procedure *exactly* applying $f$ to $U_A|0\rangle_{a+n}$. Then, $\left\||0\rangle_a|\tilde{f}(\psi)\rangle_n - |\phi\rangle_{a+n}\right\|_2 \leq \left\||0\rangle_a|\tilde{f}(\psi)\rangle_n - |\lambda\rangle_{a+n}\right\|_2 + \||\lambda\rangle_{a+n} - |\phi\rangle_{a+n}\|_2$.

Let $\epsilon_2 > 0$. By assumption of a state amplitude transformation procedure, for any given input state (in this case the input is $U_A|0\rangle_{a+n}$), the $\ell_2$-norm error in the output is bounded by $\epsilon_2$ with query complexity $O(T_{a+n,\epsilon_2})$. I.e. $\||\lambda\rangle_{a+n} - |\phi\rangle_{a+n}\|_2 \leq \epsilon_2$, with query complexity to a controlled $U_A$ and $U_A^\dagger$ circuit of $O(T_{a+n,\epsilon_2})$. Similarly, since we have just shown that $\||0\rangle_a|\psi\rangle_n - U_A|0\rangle_{a+n}\|_2 \leq \delta$, from Lemma C.10.3 it follows that $\left\||0\rangle_a|\tilde{f}(\psi)\rangle_n - |\lambda\rangle_{a+n}\right\|_2 \leq 3\gamma L\delta N/\mathcal{N}^2$.

Splitting the error equally among the two sources, and noting that $\delta = \sqrt{2(\epsilon_0+\epsilon_1)}$, imposing the following conditions ensures that the overall error is bounded by $\epsilon$: $\epsilon_2 \leq \epsilon/2$, $\epsilon_0 \leq \epsilon_1 \leq \frac{\epsilon^2\mathcal{N}^4}{144\gamma^2L^2N^2}$. If $\epsilon_0 = 0$, we can then set the bound $\epsilon_1 \leq \frac{\epsilon^2\mathcal{N}^4}{72\gamma^2L^2N^2}$.

Noting that $U_A$ has a query complexity to $U_\psi$ of $O(\alpha\log(1/\epsilon_0))$, and that $U_A$ is queried a total of $O(T_{a+n,\epsilon_2})$ times, the overall query complexity to $U_\psi$ is given by $O(\alpha T_{a+n,\epsilon}\log(\frac{\gamma^2L^2N^2}{\epsilon^2\mathcal{N}^4}))$. The circuit complexity follows similarly. □

# D

# Quantum Processors as Accelerators for Machine Learning

In Appendix D.1 I present a summary of Quantum Random Access Memory (QRAM), as it pertains to this chapter. In Appendix D.2 I present a number of techniques, some from the literature, and some which are mildly improved versions of the ones presented in Chapter 4, which are required to manipulate vectors and matrices with quantum computers, and then I use them to develop a number of new useful results for quantum matrix-vector arithmetic. In Appendix D.3, I use the techniques developed in Appendix D.2 to construct quantum-implementations of key architectural blocks. In Appendix D.4, I discuss the feasibility of QRAM as it pertains to accelerating neural networks. In Appendix D.5 I use the architectural blocks obtained in Appendix D.3 to derive end-to-end complexities for the architectures with different QRAM assumptions.

## D.1 Quantum Random Access Memory (QRAM)

I discuss QRAM in Chapter 2. QRAM for classical data is defined in Definition 2.2.1, and I show a possible circuit complexity for it in Theorem 2.2.2. For simplicity and to make this section self-contained, I include the following definition of QRAM (and note that I've slightly simplified the complexity statement).

**Definition D.1.1** (QRAM for Classical Data)**.** *Let $N = 2^n$ and $D = 2^d$. Let $|i\rangle_n$ be any n-qubit standard basis vector, and let $x_i \in [D]$. Then, a QRAM with $O(dN \log N)$ total qubits can implement the mapping,*

$$U|i\rangle_n|0\rangle_d = |i\rangle_n|x_i\rangle_d \tag{D.1}$$

*with $O(d \log N)$ circuit depth.*

**Definition D.1.2** (QRAM for Quantum Data [59, 160, 198])**.** *Let $N = 2^n$, $M = 2^m$. Let $|i\rangle_n$ be any n-qubit standard basis vector. Allow $|\psi_i\rangle_m$ to be an arbitrary m-qubit normalized quantum states. Then, a QRAM with $\tilde{O}(MN)$ total qubits, and $\tilde{O}(MN)$ classical pre-processing to construct the data structure, can implement the mapping,*

$$U|i\rangle_n|0\rangle_m = |i\rangle_n|\psi_i\rangle_m \tag{D.2}$$

*with $O(\log^2(NM))$ circuit depth.*

Importantly, as per [59, 160, 198] QRAM for quantum data can be implemented by a circuit (based on [91]) with depth and width $O(\text{polylog}(MN))$ with access to a QRAM data structure (as per Definition D.1.1) containing all the entries of each state in the quantum data (along with $O(\log M)$ copies for each of the sets of partial norms). Thus, if QRAM for classical data is feasible (as discussed in Chapter 2 and Appendix D.4), QRAM for quantum data is as well (with pre-processing to construct the appropriate data structures).

In this chapter, I will use QRAM to describe QRAM for both quantum and classical data, and will make the distinction clear when it is relevant.

## D.2 Quantum Matrix-Vector Arithmetic

In this section, I formally derive a number of tools for quantum matrix-vector arithmetic, and state some important results from the literature.

I now present a standard result (see Lemma 1 of [199] or Lemma 21 of [200]), and I include the proof for completeness, as it is the basis of a subsequent proof Lemma 5.2.3. In particular, this derivation closely follows that of Lemma 1 of [199].

**Lemma D.2.1** (Tensor Product of Block-Encoded Operators)**.** *Given a unitary $U_A$ which is an $(\alpha, a, \epsilon_0)$-block-encoding for $n$-qubit operator $A$ with $O(T_A)$ circuit complexity, and a unitary $U_B$ which is a $(\beta, b, \epsilon_1)$-block-encoding for $m$-qubit operator $B$ with $O(T_B)$ circuit complexity, we can obtain an $(\alpha\beta, a+b, \epsilon_0\beta + \epsilon_1\alpha + \epsilon_0\epsilon_1)$-block-encoding for $A \otimes B$ with $O(\max(T_A, T_B) + \max(n, b))$ circuit complexity.*

*Proof.* The main idea is that $U_A \otimes U_B$ almost directly implements a block-encoding of $A \otimes B$, but the ancillas and the main computation registers are in the wrong order. To correct this, we need to swap the ancilla register of $U_B$ with the main register of $U_A$.

Consequently, define the operator $\Pi$ such that it swaps the $n$-qubit register with the $b$-qubit register (and leaves the other registers unchanged), so that all the ancilla registers precede the main registers. If $n \geq b$, $\Pi$ can be implemented by a sequence of $O(n/b)$ swaps, with each swap swapping $O(b)$ qubits in parallel. If $n < b$, then it can be implemented with $O(b/n)$ swaps. Thus, $\Pi$ has a circuit depth bounded by $O(\max(n/b, b/n)) \in O(\max(n, b))$. Then, $\Pi(|0\rangle_{a+b} \otimes I_{n+m}) = (|0\rangle_a \otimes I_n) \otimes (|0\rangle_b \otimes I_m)$, and $(\langle 0|_{a+b} \otimes I_{n+m})\Pi^\dagger = (\langle 0|_a \otimes I_n) \otimes (\langle 0|_b \otimes I_m)$.

Following [199], define $\tilde{A} := (\langle 0|_a \otimes I_n) U_A (|0\rangle_a \otimes I_n)$, and $\tilde{B} := (\langle 0|_b \otimes I_m) U_B (|0\rangle_b \otimes I_m)$. Let $E_A := A - \alpha\tilde{A}$, and let $E_B := B - \beta\tilde{B}$. Define $V := \Pi^\dagger (U_A \otimes U_B)\Pi$. Then, $A \otimes B = (\alpha\tilde{A} + E_A) \otimes (\beta\tilde{B} \otimes E_B)$, and $(\langle 0|_{a+b} \otimes I_{n+m}) V (|0\rangle_{a+b} \otimes I_{n+m}) = \tilde{A} \otimes \tilde{B}$, so

$$\|A \otimes B - \alpha\beta(\langle 0|_{a+b} \otimes I_{n+m}) V (|0\rangle_{a+b} \otimes I_{n+m})\|_2 \tag{D.3}$$

$$= \left\|(\alpha\tilde{A} + E_A) \otimes (\beta\tilde{B} \otimes E_B) - \alpha\beta\tilde{A} \otimes \tilde{B}\right\|_2 \tag{D.4}$$

$$\leq \epsilon_0\beta + \epsilon_1\alpha + \epsilon_0\epsilon_1. \tag{D.5}$$

□

I now present a result from the literature allowing a block-encoding to have all of its singular values scaled by a constant value. I present the result nearly verbatim from Lemma 5 of [201] (with trivial modifications to make it easier to invoke in this context), which presents the results of [26, 192] cleanly in the language of block-encodings.

**Lemma D.2.2** (Uniform Singular Value Amplification [26, 192, 201])**.** *Let $\epsilon, \delta \in (0, 1/2)$, and let $\gamma > 1$. Let $U_A$ be an $(1, a, 0)$-block-encoding of the $n$-qubit operator $A$ with $O(T)$ circuit depth. Suppose $\|A\|_2 \leq (1-\delta)/\gamma$. Then, we can obtain a quantum circuit $V$ which is a $(1, a+1, \epsilon)$-block-encoding for $\gamma A$ with $O(\frac{\gamma}{\delta}\log(\gamma/\epsilon)(T+a))$ circuit depth, and with $O(poly(\frac{\gamma}{\delta}\log(\gamma/\epsilon)))$ classical computation to determine the QSVT rotation angles.*

*Proof.* This is taken directly from [26, 192, 201], simply noting that an $a$-controlled $X$ gate can be implemented by a sequence of $O(a)$ single and two-qubit gates. □

I now present a simple result which is just a special case of uniform singular value amplification [26, 192, 201] in the case where all the singular values of an encoded operator are either 0 or 1/2. This lemma was proven in collaboration with (and partially written up by) my co-author. This is done following the ideas of oblivious amplitude amplification (see [26]).

**Lemma D.2.3** ($\frac{1}{2}$ Oblivious Amplitude Amplification)**.** *We are given a matrix $A \in \mathbb{C}^{N\times N}$, with singular values either 1 or 0. Assume we have access to $U_A$ a $(2, a, 0)$-BE of $A$ with $O(T)$ circuit depth. One can construct $(1, a+1, 0)$-BE of $A$ with $O(T)$ circuit depth, and with 3 calls to a controlled-$U$ circuit.*

*Proof.* This proof was written partially by my co-author, and is included for completeness. Note that $T_3(x) = 4x^3 - 3x$ satisfies the condition that $|T_3(x)| \leq 1$ for $x \in [-1, 1]$ and $T_3(\frac{1}{2}) = -1$. The function $-T_3(x)$ can be implemented via QSVT. The first kind of the Chebyshev polynomial can be directly achieved without any classical processing to determine angle rotations, so one can construct the block encoding with no error. □

For completeness, I now re-derive an existing result on the linear combination of block-encoded matrices, directly following [26] (which presents the result of [202] in the context of block-encodings).

**Lemma D.2.4** (Linear Combination of Block-Encodings [26, 202]). *Suppose we are given a set of $D = 2^d$ unitaries $\{U_i\}_i$ such that each $U_i$ is an $(\alpha, a, \epsilon)$-block-encoding for $n$ qubit operator $A_i$, and each $U_i$ has a total of $O(T_0)$ single and two qubit gates. Define the vector $\boldsymbol{b} \in \mathbb{C}^D$ such that $\boldsymbol{b} = \begin{pmatrix} b_0 & b_1 & \dots & b_{D-1} \end{pmatrix}^T$. Define $|b\rangle_d = \sum_{j=0}^{D} \sqrt{b_j}|j\rangle_d$ and $\beta := \||b\rangle_d\|_2^2 = \|\boldsymbol{b}\|_1$. We are given the $d$-qubit unitary $U_b$, with $O(T_1)$ single and two qubit gates, such that $U_b|0\rangle_d = |b\rangle_d / \||b\rangle_d\|_2$. Define $A := \sum_{j=0}^{D-1} b_j A_j$. Then, we can obtain a unitary $V$ with $O(dDT_0 + T_1)$ circuit depth which is an $(\alpha\beta, a + d, \alpha\beta\epsilon)$-block-encoding for $A$.*

*Proof.* For each $j \in [D]$, let $\tilde{A}_j := (\langle 0|_a \otimes I_n)U_j(|0\rangle_a \otimes I_n)$, and let $E_j := A_j - \alpha\tilde{A}_j$. Define $S := \sum_{j=0}^{D-1} |j\rangle\langle j|_d \otimes U_j$. Note that $S$ can be implemented by a sequence of $D$ multi-controlled $U_j$ operators. Note that by using [190], a $d$ controlled gate targeting 1 or 2 qubits can be decomposed into a sequence of $O(d)$ single and two qubit gates. Consequently, each $d$-controlled $U_j$ has $O(dT_0)$ circuit depth in terms of single and two qubit gates. Thus, $S$ consists of a total of $O(dDT_0)$ single and two qubit gates. Then, define $V := (U_b^\dagger \otimes I_{a+n})S(U_b \otimes I_{a+n})$.

Noting that $(\langle 0|_d \otimes I_{a+n})V(|0\rangle_d \otimes I_{a+n}) = \frac{1}{\beta}\sum_{j=0}^{D-1} b_j U_j$. Using the fact that $|0\rangle_{a+d} \otimes I_n = (|0\rangle_d \otimes I_{a+n})(|0\rangle_a \otimes I_n)$, we then obtain

$$(\langle 0|_{a+d} \otimes I_n)V(|0\rangle_{a+d} \otimes I_n) = \frac{1}{\beta}(\langle 0|_a \otimes I_n)(\sum_{j=0}^{D-1} b_j U_j)(|0\rangle_a \otimes I_n) = \frac{1}{\beta}\sum_{j=0}^{D-1} b_j \tilde{A}_j. \tag{D.6}$$

Consequently,

$$\|A - \alpha\beta(\langle 0|_{a+d} \otimes I_n)V(|0\rangle_{a+d} \otimes I_n)\|_2 = \left\|\sum_{j=0}^{D-1} b_j(\alpha\tilde{A}_j + E_j) - \sum_{j=0}^{D-1} \alpha b_j \tilde{A}_j\right\|_2 \tag{D.7}$$

$$= \left\|\sum_{j=0}^{D-1} b_j \alpha E_j\right\|_2 \leq \alpha \sum_{j=0}^{D-1} |b_j| \, \|E_j\|_2 \tag{D.8}$$

$$\leq \alpha\beta\epsilon. \tag{D.9}$$

Thus, $V$ gives a $(\alpha\beta, a, \alpha\beta\epsilon)$-block-encoding for $A$, and has $O(dDT_0 + T_1)$ circuit depth. □

The following is a standard result which has been used in various contexts, and is included for completeness.

**Lemma D.2.5** (Block Encoding of Rank 1 Projector of Basis Vectors)**.** *Let $n \in \mathbb{N}_{\geq 0}$, and let $N = 2^n$. Define $i \in [N]$ and $j \in [N]$. Then, we can get a unitary $U$ which is a $(1, 2, 0)$-block-encoding of the $n$ qubit operator $|i\rangle\langle j|$. Moreover, $U$ has $O(n)$ circuit depth.*

*Proof.* Following Chapter 2, a $(1, 2, 0)$ block-encoding of the matrix $|0\rangle\langle 0|$, call it $V$, can be obtained with $O(n)$ circuit complexity. This follows by constructing a $(1, 0, 0)$ block-encoding of the Grover reflection operator, $I - 2|0\rangle\langle 0|$, and taking a linear combination with $I$ via the sum of block-encoding result of [26]. The circuit complexity is dominated by reflection operator, which can be implemented by applying a $n-1$ controlled $XZX$ gate on the most significant qubit, controlled on the 0 state of the other $n-1$ qubits. Using [190] this can be decomposed into a sequence of $O(n)$ two-qubit gates. Decompose $i$ and $j$ into bits as, $i = i_0 i_1 \dots i_{n-1}$, and $j = j_0 j_1 \dots j_{n-1}$. We now define two operators, $M_i := X^{i_0} \otimes X^{i_1} \otimes \dots \otimes X^{i_{n-1}}$ and $M_j := X^{j_0} \otimes X^{j_1} \otimes \dots \otimes X^{j_{n-1}}$. Clearly, $M_i|0\rangle\langle 0|M_j = |i\rangle\langle j|$. Then, since $(I_2 \otimes M_i)V(I_2 \otimes M_j) = \begin{pmatrix} |i\rangle\langle j| & \cdot \\ \cdot & \cdot \end{pmatrix}$. Thus, $(I_2 \otimes M_i)V(I_2 \otimes M_j)$ is a $(1, 2, 0)$ block-encoding for $|i\rangle\langle j|$. □

I now present a simple result which helps intuitively visualize VEs as encoding vectors in a subspace.

**Lemma D.2.6** (Intuitive Picture of VE as a Vector Subspace Encoding)**.** *Let $U_\psi$ be an $(\alpha, a, \epsilon)$-VE for $|\psi\rangle_n$. Define $|E_\psi\rangle_n := |\psi\rangle_n - \alpha(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$. Define the $a$-qubit operator $p_j^a := |j\rangle\langle j|$. Then,*

$$U_\psi|0\rangle_{a+n} = \frac{|0\rangle_a|\psi\rangle_n - |0\rangle_a|E_\psi\rangle_n}{\alpha} + \sum_{j=1}^{2^a-1}(p_j^a \otimes I_n)U_\psi|0\rangle_{a+n} = \begin{pmatrix} \frac{|\psi\rangle_n - |E_\psi\rangle_n}{\alpha} \\ \vdots \end{pmatrix}. \quad \text{(D.10)}$$

*Proof.* $|E_\psi\rangle_n = |\psi\rangle_n - \alpha(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$ implies that $|0\rangle_a|E_\psi\rangle_n = |0\rangle_a|\psi\rangle_n - \alpha(p_0^j \otimes I_n)U_\psi|0\rangle_{a+n}$. The result follows trivially by algebraic maniuplation of $U_\psi|0\rangle_{a+n} = (\sum_{j=0}^{2^a-1} p_j^a \otimes I_n)U_\psi|0\rangle_{a+n}$. □

Intuitively, in the absence of error, the first $2^n$ entries of $U_\psi|0\rangle_{a+n}$ will contain the sub-normalized vector $|\psi\rangle_n/\alpha$.

I now re-state the following result from Chapter 4 nearly verbatim, slightly improving the complexity. The following result is a tool essentially implementing $\ell_2$ layer normalization, follows directly from oblivious amplitude amplification (see e.g., [26]), and is taken nearly verbatim from Chapter 4.

**Lemma D.2.7** (Vector Normalization, Lemma 18 of [3])**.** *Let $\epsilon_0 \in [0, 1/2]$, $\alpha \geq 1$, $a \in \mathbb{N}$, $\epsilon_1 > 0$. Let $\alpha'$ be a known bound such that $\alpha' \geq \alpha$. Given a unitary $U_\psi$, a $(\alpha, a, \epsilon_0)$-VE for the $\ell_2$-normalized quantum state $|\psi\rangle_n$ with circuit complexity $O(T_\psi)$, we can construct a $(1, a+4, 2(\epsilon_0+\epsilon_1))$-VE for $|\psi\rangle_n$ with circuit complexity $O((T_\psi + a + n)\alpha' \log(1/\epsilon_1))$ and with $O(\alpha' \log(1/\epsilon_1))$ queries to a $U_\psi$ and $U_\psi^\dagger$ circuit.*

This implements vector normalization by boosting the scaling factor so the norm of the encoded vector is 1, and all the padding entries are 0 (up to logarithmic error).

*Proof.* Define $|\phi\rangle_n := (\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$, $\mathcal{N}_\phi := \||\phi\rangle_n\|_2$, $|\Phi\rangle_n := |\phi\rangle_n/\mathcal{N}_\phi$. Then, $U_\psi$ is equivalently a $(\mathcal{N}_\phi, a, 0)$-VE for $|\phi\rangle_n/\mathcal{N}_\phi$. Using Lemma D.2.5, we can get $U_0$ a $(1, 2, 0)$-block-encoding of the $n+a$ qubit projector $|0\rangle\langle 0|$ with $O(n+a)$ circuit depth. Then, $V = (I_2 \otimes U_\psi)U_0$ is a $(1, 2, 0)$-block-encoding for $U_\psi|0\rangle\langle 0|$, with $O(T_\psi + a + n)$ circuit complexity. Noting that $(\langle 0|_2 \otimes I_{a+n})V(|0\rangle_2 \otimes I_{a+n}) = U_\psi|0\rangle\langle 0|$, then $(\langle 0|_{2+a} \otimes I_n)V(|0\rangle_{2+a} \otimes I_n) = (\langle 0|_a \otimes I_n)U_\psi|0\rangle\langle 0|(|0\rangle_a \otimes I_n) = |\phi\rangle\langle 0|_a$, so

$$\||\phi\rangle\langle 0|_a - \langle 0|_{2+a} \otimes I_n)V(|0\rangle_{2+a} \otimes I_n)\|_2 = 0. \tag{D.11}$$

Thus, we have a $(1, a+2, 0)$-block-encoding of $|\phi\rangle\langle 0|_a = \mathcal{N}_\phi|\Phi\rangle\langle 0|$. This object has singular value $\mathcal{N}_\phi$. Thus, we want to apply a polynomial approximation to this block-encoding, such that the error of the polynomial approximation is at most $\epsilon_1$ on the interval $[\mathcal{N}_\phi, 1]$. From Corollary 6 of [192], we know that there exists an odd polynomial $P_k(x)$ with degree $k \in O(\frac{1}{\tau}\log(1/\epsilon_1))$ such that

$$\max_{x \in [-1, -\frac{\tau}{2}] \cup [\tau/2, 1]} |P_k(x) - \text{sign}(x)| \leq \epsilon_1 \tag{D.12}$$

and $\max_{x\in[-1,1]} |P_k(x)| \leq 1$. Since $\mathcal{N}_\phi \geq \frac{1}{2\alpha} \geq \frac{1}{2\alpha'}$, we can set $\tau = \frac{1}{2\alpha'}$, guaranteeing that $P(\mathcal{N}_\phi) \geq 1 - \epsilon_1$. Consequently, we can invoke quantum singular

value transformation (QSVT) [26] with $P_k$, yielding $V_f$ a $(1, a+4, \epsilon_1)$-block-encoding for $P(\mathcal{N}_\phi|\Phi\rangle\langle 0|) = c|\Phi\rangle\langle 0|$, where $1 \geq c \geq 1-\epsilon_1$. Moreover, $V_f$ has $O(\frac{1}{\alpha'}\log(1/\epsilon_1)(T_\psi + a + n))$ circuit complexity. Noting that

$$\begin{aligned} \||\psi\rangle\langle 0| - P(\mathcal{N}_\phi|\Phi\rangle\langle 0|)\|_2 &= \||\psi\rangle\langle 0| - |\Phi\rangle\langle 0| + |\Phi\rangle\langle 0| - c|\Phi\rangle\langle 0|\|_2 && \text{(D.13)} \\ &\leq \||\psi\rangle\langle 0| - |\Phi\rangle\langle 0|\|_2 + \||\Phi\rangle\langle 0| - c|\Phi\rangle\langle 0|\|_2 && \text{(D.14)} \\ &\leq \||\psi\rangle_n - |\Phi\rangle_n\|_2 + \epsilon_1. && \text{(D.15)} \end{aligned}$$

Moreover,

$$\begin{aligned} \||\psi\rangle_n - |\Phi\rangle_n\|_2 \leq& \||\psi\rangle_n - \alpha|\phi\rangle_n\|_2 + \|\alpha|\phi\rangle_n - \frac{1}{\mathcal{N}_\phi}|\phi\rangle_n\|_2 && \text{(D.16)} \\ \leq& \epsilon_0 + \frac{1}{\mathcal{N}_\phi}\|\alpha\mathcal{N}_\phi|\phi\rangle_n - |\phi\rangle_n\| = \epsilon_0 + \frac{|\alpha\mathcal{N}_\phi - 1|}{\mathcal{N}_\phi}\||\phi\rangle_n\|_2 && \text{(D.17)} \\ \leq& \epsilon_0 + |\alpha\mathcal{N}_\phi - 1|. && \text{(D.18)} \end{aligned}$$

Moreover, using the reverse triangle inequality with $\||\psi\rangle_n - \alpha|\phi\rangle_n\|_2 \leq \epsilon_0$, we get $|1 - \alpha\,\||\phi\rangle_n\|_2\,| = |1 - \alpha\mathcal{N}_\phi| \leq \epsilon_1$, which implies that $1 - \epsilon_0 \leq \alpha\mathcal{N}_\phi \leq 1 + \epsilon_0$. Consequently, $|\alpha\mathcal{N}_\phi - 1| \leq \epsilon_0$, and so

$$\||\psi\rangle_n - |\Phi\rangle_n\|_2 \leq 2\epsilon_0. \tag{D.19}$$

Thus,

$$\||\psi\rangle\langle 0| - P(\mathcal{N}_\phi|\Phi\rangle\langle 0|)\|_2 \leq 2\epsilon_0 + \epsilon_1. \tag{D.20}$$

Moreover, since $V_f$ is a $(1, a+4, \epsilon_1)$-block-encoding for $P(\mathcal{N}_\phi|\Phi\rangle\langle 0|)$,

$$\|P(\mathcal{N}_\phi|\Phi\rangle\langle 0|) - (\langle 0|_{a+4} \otimes I_n)V_f(|0\rangle_{a+4} \otimes I_n)\|_2 \leq \epsilon_1. \tag{D.21}$$

Thus,

$$\||\psi\rangle\langle 0| - (\langle 0|_{a+4} \otimes I_n)V_f(|0\rangle_{a+4} \otimes I_n)\|_2 \leq 2(\epsilon_0 + \epsilon_1). \tag{D.22}$$

□

Sometimes it is necessary to increase the norm of the vector encoded in the subspace of a VE. This is equivalent to multiplying all of the entries in the encoded vector by a constant with value greater than or equal to one. The following lemma achieves the opposite: it allows the norm of the encoded vector to be shrunk by an arbitrarily large amount. This is equivalent to dividing all the entries in the encoded vector by a constant greater than or equal to one. It is worth noting that the following result is trivial and can almost certainly be further optimized, e.g., by removing the additional ancillary qubits added.

**Lemma D.2.8** (Vector De-Amplification)**.** *Let $\tau \geq 1$, $\alpha \geq 1$, $\epsilon \geq 0$. Given $U_\psi$ an $(\alpha, a, \epsilon)$-VE for $|\psi\rangle_n$, with circuit complexity $O(T)$, we can obtain $U'_\psi$ an $(\alpha\tau, a+2, \epsilon)$-VE for $|\psi\rangle_n$ with circuit complexity $O(T+a)$.*

*Proof.* Let $|\phi_j\rangle_n := (\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$. Then, note that $U_\psi|0\rangle_{a+n} = \sum_{j=0}^{2^a-1} |j\rangle_a \otimes |\phi_j\rangle_n$. By Definition 1.2.7, we know that $\||\psi\rangle_n - \alpha|\phi_0\rangle_n\| \leq \epsilon$.

We introduce two single-qubit ancillas as the most significant bits, and then apply a multiple-controlled $X$ gate (with $a$ controls each activated by the 0 state of each of the previous $a$ ancilla qubits) targeting the first newly added ancilla qubit. Using [190] this can be implemented with $O(a)$ two-qubit gates. We then apply a controlled $R_{1/\tau^2}$ (as per Definition D.2.1) gate targeting the second new ancilla qubit, controlled on the first new ancilla. This yields the state,

$$|1\rangle_1(\frac{1}{\tau}|0\rangle_1 + \sqrt{1-\frac{1}{\tau^2}}|1\rangle_1)|0\rangle_a|\phi_0\rangle_n + |0\rangle_1|0\rangle_1 \sum_{j=1}^{2^a-1} |j\rangle_a|\phi_j\rangle_n. \tag{D.23}$$

We then apply a $X$ gate to the first ancilla qubit, and we call the $2+a$-qubit unitary containing all the preceding operations $V$. Then, $U'_\psi := (V \otimes I_n)(I_2 \otimes U_\psi)$. Simple analysis thus shows that $(\langle 0|_{2+a} \otimes I_n)U'_\psi|0\rangle_{2+a+n} = |\phi_0\rangle_n/\tau$. Then,

$$\left\||\psi\rangle_n - \alpha\tau(\langle 0|_{2+a} \otimes I_n)U'_\psi|0\rangle_{2+a+n}\right\|_2 = \||\psi\rangle_n - \alpha|\phi_0\rangle_n\| \leq \epsilon. \tag{D.24}$$

□

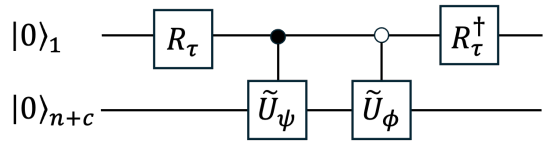


**Figure D.1: Circuit for addition of VE encoded vectors.** Given two unitary matrices, $U_\psi$ which is a $(\alpha, a, \epsilon_0)$-VE for the $n$-qubit state $|\psi\rangle$, and $U_\phi$ which is a $(\beta, b, \epsilon_1)$-VE for the $n$-qubit state $|\phi\rangle$, define $c := \max(a, b)$. Define $\tilde{U}_\psi$ by appropriately tensoring $U_\psi$ with $I_{c-a}$ and define $\tilde{U}_\phi$ by appropriately tensoring $U_\phi$ with $I_{c-b}$, such that $\tilde{U}_\psi$ and $\tilde{U}_\phi$ both act on $n + c$ qubits. Then, the given circuit yields a VE of the sum of the encoded vectors, as shown in Lemma 5.2.1.

**Definition D.2.1** (Real Rotation Single Qubit Gate)**.** *Let* $0 \leq \tau \leq 1$*. Then, define the following single-qubit gate:*

$$R_\tau := \begin{pmatrix} \sqrt{\tau} & -\sqrt{1-\tau} \\ \sqrt{1-\tau} & \sqrt{\tau}, \end{pmatrix}. \tag{D.25}$$

***Proof of Lemma 5.2.1*** *(Vector Sum).* This result follows using a common techniques, see e.g., LCU [202], or the sum of block-encodings result [26]. As per Fig. D.1, we will augment $U_\psi$ and $U_\phi$ so that they both act on $c = \max(a + b)$ ancilla qubits. Then, define the $n + c$ qubit states, $|\tilde{\psi}\rangle_{n+c} := U_\psi|0\rangle_{n+c}$. We will drop the subscripts on these states for the rest of the proof, as their dimension is clear from the context. This block-encoding will be obtained with the circuit shown in Fig. D.1, and so we will now analyze the action of that circuit. First, we start with the state $|0\rangle_{1+n+c}$, which we will write as $|0\rangle|0\rangle$, where the first register has one qubit, and the second register has the remaining $n + c$ qubits. We then apply $R_\tau$ (as defined in Definition D.2.1) to the first qubit, yielding the state $(\sqrt{\tau}|0\rangle + \sqrt{1-\tau}|1\rangle)|0\rangle$. Next, we apply the controlled $U_\psi$ and $U_\phi$ gates, yielding, $\sqrt{\tau}|0\rangle|\tilde{\psi}\rangle + \sqrt{1-\tau}|1\rangle|\tilde{\phi}\rangle$. Next, we apply $R_\tau^\dagger = \begin{pmatrix} \sqrt{\tau} & \sqrt{1-\tau} \\ -\sqrt{1-\tau} & \sqrt{\tau} \end{pmatrix}$ on the first qubit, yielding the output of the new VE, $V|0\rangle = |0\rangle(\tau|\tilde{\psi}\rangle + (1-\tau)|\tilde{\phi}\rangle) + \sqrt{\tau(1-\tau)}|1\rangle(|\tilde{\phi}\rangle - |\tilde{\psi}\rangle)$. Define $|E_\psi\rangle := |\psi\rangle - \alpha(\langle 0|^{\otimes(c)} \otimes I_n)|\tilde{\psi}\rangle$ and note that $\||E_\psi\rangle\|_2 \leq \epsilon_0$. Similarly define $|E_\phi\rangle$, and note that $\||E_\phi\rangle\|_2 \leq \epsilon_1$. As a result, we can determine the properties of this

VE by bounding the following,

$$\left\| |\overline{\Gamma}\rangle - \frac{1}{\mathcal{N}}(\langle 0|^{\otimes(1+c)} \otimes I_n)V|0\rangle_{1+c+n} \right\|_2 \tag{D.26}$$

$$= \left\| |\overline{\Gamma}\rangle - \frac{1}{\mathcal{N}}(\langle 0|^{\otimes c} \otimes I_n)(\tau|\tilde{\psi}\rangle + (1-\tau)|\tilde{\phi}\rangle) \right\|_2 \tag{D.27}$$

$$= \left\| |\overline{\Gamma}\rangle - \frac{1}{\mathcal{N}}\left(\frac{\tau}{\alpha}(|\psi\rangle - |E_\psi\rangle) + \frac{1-\tau}{\beta}(|\phi\rangle - |E_\phi\rangle)\right) \right\|_2 \tag{D.28}$$

$$= \left\| \frac{1}{\mathcal{N}}\left(\frac{\tau}{\alpha}|E_\psi\rangle + \frac{1-\tau}{\beta}|E_\phi\rangle\right) \right\|_2 \leq \frac{1}{\mathcal{N}}\left(\frac{\tau\epsilon_0}{\alpha} + \frac{(1-\tau)\epsilon_1}{\beta}\right) \tag{D.29}$$

$$\leq \frac{1}{\mathcal{N}}\left(\frac{\epsilon_0}{\alpha} + \frac{\epsilon_1}{\beta}\right) \leq \frac{\epsilon_0 + \epsilon_1}{\mathcal{N}}. \tag{D.30}$$

where the final inequality comes from the definition of a VE imposing that $\alpha \geq 1$ and $\beta \geq 1$. Thus, the unitary circuit $V$ is a $(\mathcal{N}^{-1}, 1+a+b, \mathcal{N}^{-1}(\epsilon_0+\epsilon_1))$-VE for $|\overline{\Gamma}\rangle$. □

***Proof of Lemma 5.2.2 (Matrix Vector Product).*** We now require a result allowing for matrix-vector products with our vector-encodings. This result is essentially a special case of the product of the standard product of block-encodings result (Lemma 53 of [26]). As a result, the following proof closely follows that in [26]. In this lemma, again following the notation of [26] for tensor products, it is assumed that $U_A$ and $U_\psi$ act trivially on the other's ancillas. To be explicit, the tensor products in $(I_b \otimes U_A)(I_a \otimes U_\psi)$ use a special definition only in this lemma.

Let $\mathcal{N} := \|A|\psi\rangle_n\|_2$. We wish to upper-bound,

$$\xi := \left\| \frac{A|\psi\rangle_n}{\mathcal{N}} - \frac{\alpha\beta}{\mathcal{N}}(\langle 0|_{a+b} \otimes I_n)(I_b \otimes U_A)(I_a \otimes U_\psi)|0\rangle_{a+b+n} \right\|_2 \tag{D.31}$$

$$= \frac{1}{\mathcal{N}} \left\| A|\psi\rangle_n - \alpha\beta(\langle 0|_{a+b} \otimes I_n)(I_b \otimes U_A)(I_a \otimes U_\psi)(|0\rangle_{a+b} \otimes I_n)|0\rangle_n \right\|_2 \tag{D.32}$$

Then, directly from the proof of Lemma 53 in [26],

$$\xi = \frac{1}{\mathcal{N}} \left\| A|\psi\rangle_n - \alpha\beta \left[(\langle 0|_a \otimes I_n)U_A(|0\rangle_a \otimes I_n)\right] \left[(\langle 0|_b \otimes I_n)U_\psi(|0\rangle_b \otimes I_n)\right] |0\rangle_n \right\|_2 \tag{D.33}$$

Let $\tilde{A} := \alpha(\langle 0|_a \otimes I_n)U_A(|0\rangle_a \otimes I_n)$ and let $|\tilde{\psi}\rangle := \beta(\langle 0|_b \otimes I_n)U_\psi(|0\rangle_{b+n})$. Then,

$$\xi = \frac{1}{\mathcal{N}} \left\| A|\psi\rangle_n - \tilde{A}|\tilde{\psi}\rangle_n \right\|_2 = \frac{1}{\mathcal{N}} \left\| A|\psi\rangle_n - \tilde{A}|\psi\rangle_n + \tilde{A}|\psi\rangle_n - \tilde{A}|\tilde{\psi}\rangle_n \right\|_2 \tag{D.34}$$

$$\leq \frac{1}{\mathcal{N}} \left( \left\| A - \tilde{A} \right\|_2 + \left\| \tilde{A} \right\|_2 \left\| |\psi\rangle_n - |\tilde{\psi}\rangle_n \right\|_2 \right) \tag{D.35}$$

Noting that $\left\|\tilde{A}\right\|_2 \leq \alpha$, we then get

$$\xi \leq (\epsilon_0 + \alpha\epsilon_1)/\mathcal{N}. \tag{D.36}$$

Consequently, $(I_b \otimes U_A)(I_a \otimes U_\psi)$ gives a $(\alpha\beta/\mathcal{N}, a+b, (\epsilon_0+\alpha\epsilon_1)/\mathcal{N})$-VE for $A|\psi\rangle_n/\mathcal{N}$.

□

In the following lemma I derive a technical result handling the case where you have a vector encoding for some vector $|\psi\rangle$, and another vector of interest $|\phi\rangle$ is sub-encoded as $|\psi\rangle = \begin{pmatrix} |\phi\rangle/\beta \\ \cdot \end{pmatrix}$. This result also handles the case where each vector is imperfectly encoded (i.e., encoded with error).

**Lemma D.2.9** (Vector Sub-Encodings)**.** *Let $m, n$ be integers such that $m > n$. Let $U_\psi$ be an $(\alpha, a, \epsilon)$-VE for $|\psi\rangle_m$, and let $|\psi\rangle_m \approx V_\phi|0\rangle_m$ (precisely, $\||\psi\rangle_m - V_\phi|0\rangle_m\|_2 \leq \gamma$), where $V_\phi$ is a $(\beta, m-n, \delta)$-VE for $|\phi\rangle_n$. Then, $U_\psi$ is an $(\alpha\beta, a+m-n, \delta+\beta(\epsilon+\gamma))$-VE for $|\phi\rangle_n$.*

*Proof.* Let $b = m - n$. First, define $|E_\psi\rangle_m := |\psi\rangle_m - \alpha\left(\langle 0|_a \otimes I_m\right) U_\psi |0\rangle_{a+m}$, and $|E_\phi\rangle_n := |\phi\rangle_n - \alpha\left(\langle 0|_b \otimes I_n\right) U_\psi |0\rangle_{b+n}$. By Definition 1.2.7, $\||E_\psi\rangle_m\|_2 \leq \epsilon$ and $\||E_\phi\rangle_n\|_2 \leq \delta$. Let $|E_v\rangle_m := |\psi\rangle_m - V_\phi|0\rangle_m$. Now observe,

$$\begin{aligned}
\left(\langle 0|_b \otimes I_n\right)\left(\langle 0|_a \otimes I_m\right) U_\psi |0\rangle_{a+m} &= \left(\langle 0|_b \otimes I_n\right)\left(|\psi\rangle_m - |E_\psi\rangle_m\right)/\alpha && \text{(D.37)}\\
&= \left(\langle 0|_b \otimes I_n\right)\left(V_\phi|0\rangle_m + |E_v\rangle_m - |E_\psi\rangle_m\right)/\alpha && \text{(D.38)}\\
&= \left((|\phi\rangle_n - |E_\phi\rangle_n)/\beta + (\langle 0|_b \otimes I_n)(|E_v\rangle_m - |E_\psi\rangle)\right)/\alpha. && \text{(D.39)}
\end{aligned}$$

Consequently, since $(\langle 0|_b \otimes I_n)(\langle 0|_a \otimes I_m) = \langle 0|_{a+b} \otimes I_n$,

$$\begin{aligned}
&\||\phi\rangle_n - \alpha\beta(|0\rangle_{a+b} \otimes I_n) U_\psi |0\rangle_{a+b+n}\|_2 && \text{(D.40)}\\
&\leq \||E_\phi\rangle_n\|_2 + \beta\,\||E_\psi\rangle_m\|_2 + \beta\,\||E_v\rangle_m\|_2 \leq \delta + \beta(\epsilon+\gamma). && \text{(D.41)}
\end{aligned}$$

□

**Lemma D.2.10** (Tracing Out Qubits in Vector Sub-Encodings)**.** *Let $U$ be an $(\alpha, a, \epsilon)$-VE for $|0\rangle_b|\psi\rangle_n$. Then, $U$ is an $(\alpha, a+b, \epsilon)$-VE for $|\psi\rangle_n$.*

*Proof.* Let $|E\rangle_{b+n} := |0\rangle_b|\psi\rangle_n - \alpha(\langle 0|_a \otimes I_{n+b})U|0\rangle_{a+b+n}$. Since $\langle 0|_{a+b} \otimes I_n = (\langle 0|_b \otimes I_n)(\langle 0|_a \otimes I_{b+n})$, $(\langle 0|_{a+b} \otimes I_n)U|0\rangle_{a+b+n} = \frac{1}{\alpha}(|\psi\rangle_n - (\langle 0|_b \otimes I_n)|E\rangle_{b+n})$. Thus,

$$\||\psi\rangle_n - \alpha(\langle 0|_{a+b} \otimes I_n)U|0\rangle_{a+b+n}\|_2 = \|(\langle 0|_b \otimes I_n)|E\rangle_{b+n}\|_2 \leq \epsilon. \tag{D.42}$$

□

***Proof of Lemma 5.2.3 (Vector Tensor Product).*** This result closely follows the derivation of the tensor product of block-encodings (Lemma D.2.1), which was a rederivation of Lemma 1 of [199].

$U_\psi$ acts on an $a$-qubit ancilla register and a $n$-qubit main register, while $U_\phi$ acts on an $b$-qubit ancilla register and a $m$-qubit main register.

As per Lemma D.2.1, define $\Pi$ to swap the $n$-qubit register with the $b$-qubit register acting trivially on the other two registers. Again, $\Pi$ has a circuit depth bounded by $O(\max(n/b, b/n)) \in O(\max(n, b))$. Then, $(\langle 0|_{a+b} \otimes I_{n+m})\Pi^\dagger = (\langle 0|_a \otimes I_n) \otimes (\langle 0|_b \otimes I_m)$. Let $V = \Pi^\dagger(U_\psi \otimes U_\phi)$. Let $|E_\psi\rangle_n = |\psi\rangle_n - \alpha(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$ and $|E_\phi\rangle_m = |\phi\rangle_m - \beta(\langle 0|_b \otimes I_m)U_\phi|0\rangle_{b+m}$. Then,

$$(\langle 0|_{a+b} \otimes I_{n+m})\Pi^\dagger(U_\psi \otimes U_\phi)|0\rangle_{a+b+n+m} = \frac{1}{\alpha\beta}(|\psi\rangle_n - |E_\psi\rangle_n) \otimes (|\phi\rangle_m - |E_\phi\rangle_m), \tag{D.43}$$

and so,

$$\||\psi\rangle_n|\phi\rangle_m - \alpha\beta(\langle 0|_{a+b} \otimes I_{n+m})\Pi(U_\psi \otimes U_\phi)|0\rangle_{a+b+n+m}\|_2 \leq \epsilon + \delta + \epsilon\delta. \tag{D.44}$$

□

***Proof of Lemma 5.2.4 (Vector Concatenation).*** I now present the proof of a simple result on the concatenation of vectors stored in VEs. This result follows from a simple modification of LCU [202]. In essence, given a set of $D = 2^d$ vectors $\{|\psi_j\rangle_n\}_j$, we first create vector encodings of $\{|j\rangle_d|\psi_j\rangle_n\}_j$ and then take the resulting sum of the encoded vectors following LCU, yielding an encoding of $\begin{pmatrix}\langle\psi_0|_n & \dots & \langle\psi_{D-1}|_n\end{pmatrix}^\dagger$.

For all $j$, define $|E_{\psi_j}\rangle_n := |\psi_j\rangle_n - \alpha(\langle 0|_a \otimes I_n)U_i|0\rangle_{a+n}$.

First, let $j$ be $d$ bits, and let $j = j_0 j_1 \dots j_{d-1}$. Define $X_j := X^{j_0} \otimes X^{j_1} \otimes \dots \otimes X^{j^{d-1}}$. Note that $|j\rangle_d = X_j|0\rangle_d$, and thus that $X_j$ is a $(1,0,0)$-VE for $|j\rangle_d$. Then, we can invoke Lemma 5.2.3 with $U_j$ and $X_j$ to obtain $V_j$, an $(\alpha_j, a, \epsilon)$-VE for $|j\rangle_d|\psi_j\rangle_n$ with $O(T+n)$ circuit complexity. Moreover, by inspecting Lemma 5.2.3, we find that $(\langle 0|_a \otimes I_{n+d})V_j|0\rangle_{a+d+n} = \frac{1}{\alpha_j}(|j\rangle_d|\psi_j\rangle_n - |j\rangle_d|E_{\psi_j}\rangle_n)$.

Additionally, define $S := \sum_{j=0}^{D-1} |j\rangle\langle j|_d \otimes V_j$. This can be implemented by a sequence of $O(D)$ multi-controlled gates, each enacting $V_j$ when the control register is $|j\rangle_d$ (in the standard fashion of LCU [202]). First, note that by using [190] a multiple-controlled gate with $O(d)$ controls can be split into a sequence of $O(d)$ single and two-qubit gates. By splitting each of the $d$ control qubits into $a+d+n$ copies (with $O(\log(a+d+n))$ depth), we can control each gate in each layer of $U_j$ in parallel with $O(d)$ circuit depth. Since these ancillas can be uncomputed and traced out, we ignore them in the complexity analysis. Thus, each multi-controlled $V_j$ gate can be decomposed into a sequence of $O(dT)$ single and two-qubit gates. Thus, $S$ has a total circuit depth of $O(dDT)$. Let $\hat{H} := H^{\otimes d} \otimes I_{d+n+a}$. Using $\langle 0|_{a+d} \otimes I_{n+d} = (\langle 0|_d \otimes I_{n+d})(I_d \otimes \langle 0|_a \otimes I_{d+a})$,

$$(\langle 0|_{d+a} \otimes I_{n+d})\hat{H}S\hat{H}|0\rangle_{2d+a+n} \tag{D.45}$$

$$= (\langle +|_d \otimes I_{n+d})(I_d \otimes \langle 0|_a \otimes I_{n+d}) \sum_{j=0}^{D-1} (|j\rangle\langle j|_d \otimes V_j)|+\rangle_d|0\rangle_{a+d+n} \tag{D.46}$$

$$= \frac{1}{D}\sum_{j=0}^{D-1} \frac{|j\rangle_d|\psi_j\rangle_d - |j\rangle_d|E_{\psi_j}\rangle_n}{\alpha_j}. \tag{D.47}$$

Then, noting that $\mathcal{N}^2 = \sum_{j=0}^{D-1} \frac{1}{\alpha_j^2}$, and that $\left\|\sum_{j=0}^{D-1} |j\rangle_d|E_{\psi_j}\rangle_n/\alpha_j\right\|_2 \leq \mathcal{N}\epsilon$,

$$\left\| \frac{|\Psi\rangle_{d+n}}{\mathcal{N}} - \frac{D}{\mathcal{N}}(\langle 0|_{d+a} \otimes I_{n+d})\hat{H}S\hat{H}|0\rangle_{2d+a+n} \right\|_2 = \frac{1}{\mathcal{N}} \left\| \sum_{j=0}^{D-1} |j\rangle_d|E_{\psi_j}\rangle_n/\alpha_j \right\|_2 \leq \epsilon. \tag{D.48}$$

Thus, $\hat{H}S\hat{H}$ is a $(D/\mathcal{N}, d+a, \epsilon)$ for $\frac{|\Psi\rangle_{d+n}}{\mathcal{N}}$ with $O(dDT)$ circuit complexity. □

### D.2.1 General Matrix-Vector-Squared Product

In this subsection, I derive a procedure which given an *arbitrary* matrix $W$ and quantum state $|\psi\rangle$, allows for a state proportional to the product of $W(|\psi\rangle)^2$ to

be obtained with complexity *independent* of the Frobenius norm (and thus rank), and sparsity, of $W$. To the best of my knowledge, this is the first result which allows such a product without either a rank or sparsity condition on $W$. The key insight is to avoid ever constructing a block-encoding of the operator $W$, and directly query its columns weighted by the entries of the vector it is being applied to. In particular, at a high-level we construct two objects. Define the columns of $W = \begin{pmatrix} \boldsymbol{w}_0 & \dots & \boldsymbol{w}_{N-1} \end{pmatrix}$, define the column norms $a_j := \|\boldsymbol{w}_j\|_2$, and the normalized versions of the columns $|w_j\rangle_n = \boldsymbol{w}_j/a_j$. Additionally, define the state we are applying it to as $|\psi\rangle_n = \sum_j \psi_j |j\rangle_n$. First, we construct the normalized state $\sum_j \psi_j |j\rangle_n |w_j\rangle_n$. Clearly, this object has no Frobenius norm dependence. We would like to map all the vectors in the first register to the $|0\rangle$ state so that we have something resembling the matrix-vector product, and to do this we construct another operator. Note that the matrix $Q = \begin{pmatrix} a_0 I_n & \dots & a_{N-1} I_n \\ & \mathbf{0} & \end{pmatrix}$ (i.e., the first $N$ rows are non-zero, and the rest are all zero) when applied to $|\phi\rangle_{2n} = \sum_j \psi_j |j\rangle_n |w_j\rangle_n$ yields $Q|\phi\rangle_{2n} = |0\rangle_n \otimes (W|\psi\rangle_n)$. However, this object has a spectral norm $\Omega(\|W\|_F)$. Instead, define $M := \begin{pmatrix} a_0\psi_0 I_n & \dots & a_{N-1}\psi_{N-1} I_n \\ & \mathbf{0} & \end{pmatrix}$ and note that $M$ can be shown to have $\|M\|_2 \leq 1$, and moreover, I subsequently show how a block-encoding of this operator can be efficiently obtained. Consequently, since $M|\phi\rangle_{2n} = |0\rangle_n \otimes (W(|\psi\rangle_n)^2)$, the result follows. In the rest of this section, I derive the ingredients necessary to rigorously prove the above intuition.

I now present a result on obtaining a block-encoding of an arbitrary diagonal matrix whose entries are stored in QRAM. This is essentially a special case of Lemma 48 of [26], but by considering this special case moderate improvements in complexity can be obtained.

**Lemma D.2.11** (Quantum Block-Encoding of Diagonal Matrices from QRAM)**.** *Let $N = 2^n$. We are given a set of $N$ real coefficients, $\{a_j\}_j$ such that $\forall j, |a_j| \leq 1$. Assume that each $a_j$ can be represented exactly in a binary encoding with $d$-bits of precision, and define $D = 2^d$. Define $b_j := \arccos(a_j)D/\pi$, and for simplicity assume*

*that each $b_j$ can also be implemented with exactly d-bits of precision*[1]*, and note that $b_j \in [D]$. Assume that we are given an oracle, implemented via QRAM, such that $U|0\rangle_d|j\rangle_n = |b_j\rangle_d|j\rangle_n$. Then, we can obtain $U_A$, a $(1, d+1, 0)$-block-encoding for $A = diag(a_0, \ldots, a_{N-1})$, with $O(dn)$ circuit depth.*

*Proof.* Define the circuit $V := (I_1 \otimes U^\dagger)(CR_Y(\frac{\pi}{D}) \otimes I_n)(I_1 \otimes U)$, with $CR_Y(\frac{\pi}{D})$ defined as per Lemma 1.2.1. First, since for any $|\phi\rangle$ and basis vector $|j\rangle$, $|\phi\rangle \otimes |j\rangle\langle j| = (|\phi\rangle|j\rangle)\langle j|$, observe that

$$(I_1 \otimes U)(|0\rangle_{d+1} \otimes I_n) = \sum_{j=0}^{N-1} \left[(I_1 \otimes U)|0\rangle_1|0\rangle_d|j\rangle_n\right] \langle j|_n = \sum_{j=0}^{N-1} (|0\rangle_1|b_j\rangle_d|j\rangle_n)\langle j|_n. \tag{D.49}$$

Then, since $\cos(b_j \frac{\pi}{D}) = \arccos(a_j)$,

$$(CR_Y(\frac{\pi}{D}) \otimes I_n)(I_1 \otimes U)(|0\rangle_{d+1} \otimes I_n) = \sum_{j=0}^{N-1} \left((a_j|0\rangle_1 + \sqrt{1-a_j^2}|1\rangle_1)|b_j\rangle_d|j\rangle_n\right) \langle j|_n. \tag{D.50}$$

Then, since $(\langle 0|_{d+1} \otimes I_n)(I_1 \otimes U^\dagger) = [(I_1 \otimes U)(|0\rangle_{d+1} \otimes I_n)]^\dagger = \sum_{j=0}^{N-1} |j\rangle_n(\langle 0|_1\langle b_j|_d\langle j|_n)$, we readily find that

$$(\langle 0|_{d+1} \otimes I_n)V(|0\rangle_{d+1} \otimes I_n) = \sum_{j=0}^{N-1} a_j|j\rangle\langle j| = \text{diag}(a_0, \ldots, a_{N-1}) = A. \tag{D.51}$$

Thus, $V$ is a $(1, d+1, 0)$-block-encoding for $A$. The circuit depth of implementing $U$ is the depth of making a QRAM query, and is thus $O(d \log N) = O(nd)$ (see Definition D.1.1). The cost of implementing the $CR_Y$ gate is simply $O(d)$ as per Lemma 1.2.1, and thus the overall circuit complexity of this block-encoding is $O(nd)$. □

In the case where each $a_j \in \mathbb{C}$, the complex and real parts need to be specified separately. A diagonal block-encoding of the real and imaginary parts can then be obtained using Lemma D.2.11, and can then be summed by adding an ancilla to obtain a $(2, d+2, 0)$-block-encoding with the same overall circuit complexity.

[1] In practice this will result in an additional logarithmic source of error, which I am neglecting, as it is akin to finite-precision arithmetic error which is usually neglected in classical algorithm analysis.

One might wonder why, given a QRAM assumption, a state-preparation unitary yielding a state proportional to $\sum_j a_j|j\rangle$ can't be used instead, in combination with the diagonal block-encoding of state amplitudes result of [3]. If each $a_j$ represent the column norm of some matrix $W$, doing so would result in a normalization factor of $\left\|\sum_j a_j|j\rangle\right\|_2 = \sqrt{\sum_j |a_j|^2} = \|W\|_F$, yielding a Frobenius norm-dependence which this approach avoids.

The following data structure is useful in situations where you are willing to pay a pre-processing cost linear (up to polylogarithmic factors) in the number of non-zero matrix elements, but want a fast algorithm at runtime. This is the case with accelerating neural network inference. The following data structure is very similar to the one given in [59].

**Definition D.2.2** (Preprocessed Matrix QRAM Data Structure)**.** *Let $N = 2^n$, and let $D = 2^d$.*

*Let $W \in \mathbb{C}^{N\times N}$ and let $\|W\|_2 \leq 1$. Let the columns of $W$ be represented as $W = \begin{pmatrix} \boldsymbol{w}_0 & \dots & \boldsymbol{w}_{N-1} \end{pmatrix}$. Additionally, define $|w_j\rangle = \boldsymbol{w}_j / \|\boldsymbol{w}_j\|_2$, and $a_j = \|\boldsymbol{w}_j\|$. Let $b_j := \arccos(a_j)D/\pi$. For simplicity, we assume that $b_j$ can be exactly written with d-bits, and thus that $b_j$ will be an integer between $[0, D-1]$. We say we have access to a Preprocessed QRAM Data Structure for $W$ if we have a QRAM oracle $U_W$ (as per Definition D.1.2) such that*

$$U_W|j\rangle_n|0\rangle_n = |j\rangle_n|w_j\rangle_n, \tag{D.52}$$

*and we also have access to a QRAM yielding the mapping,*

$$U_A|0\rangle_d|j\rangle_n = |b_j\rangle_d|j\rangle_n. \tag{D.53}$$

*$U_W$ can be implemented with $O(\log^2 N)$ circuit depth, and with $\tilde{O}(N^2)$ total qubits (as per Definition D.1.2). $U_A$ can be implemented with $O(d \log N)$ circuit depth, and with $\tilde{O}(dN)$ total qubits (as per Definition D.1.1).*

I am now ready to present a somewhat surprising result on matrix-vector multiplication with arbitrary (potentially full-rank and dense) matrices and the element-wise square of a given vector. The following uses ideas similar to importance-sampling.

**Theorem D.2.1** (Product of Arbitrary Matrix with a Vector Element-wise Squared)**.** *Let $N = 2^n$. We are given a matrix $W \in \mathbb{C}^{N\times N}$ through the data structure in Definition D.2.2. Let $d$ be the number of bits required to represent the function of the column norms of $W$, $b_j$, as per Definition D.2.2. Additionally, we are given the unitary $U_\psi$ with circuit complexity $O(T_\psi)$, a $(\alpha, a, \epsilon)$-VE for the quantum state $|\psi\rangle_n$. Define the function $g : \mathbb{C} \mapsto \mathbb{R}$ as $g(x) = |x|^2$, and $\mathcal{N} := \|Wg(|\psi\rangle_n)\|_2$. Then we can construct the unitary $U_f$ which is a $(\frac{\alpha^2}{\mathcal{N}}, 2a+d+3+n, \frac{2\alpha\epsilon}{\mathcal{N}})$-VE for $Wg(|\psi\rangle_n)/\mathcal{N}$, and has a circuit depth of $O(T_\psi + dn + n^2)$.*

*Proof.* Noting that $a_j = \|W|j\rangle\|_2$, it is easy to show $\|W\|_2 \leq 1 \implies \forall j, a_j \leq 1$; $a_j = \|W|j\rangle\|_2 \leq \max_{\boldsymbol{x}:\|\boldsymbol{x}\|_2=1} \|W\boldsymbol{x}\|_2 = \|W\|_2 \leq 1$. Consequently, by Lemma D.2.11 we can immediately get $U_A$, a $(1, d+1, 0)$-block-encoding for $A = \mathrm{diag}(a_0, \ldots, a_{N-1})$ with $O(dn)$ circuit depth.

Let $|\psi_1\rangle_n := A|\psi\rangle_n = \sum_{j=0}^{N-1} a_j\psi_j|j\rangle_n$, $\mathcal{N}_1 := \||\psi_1\rangle_n\|_2$. By Lemma 5.2.2, we can combine $U_A$ and $U_\psi$ to obtain $V_1$, a $(\alpha/\mathcal{N}_1, a+d+1, \epsilon/\mathcal{N}_1)$-VE for $|\psi_1\rangle_n/\mathcal{N}_1$. This has circuit complexity $O(T_\psi + dn)$.

By Lemma D.2.5, we can get $U_0$, a $(1, 2, 0)$-block-encoding for the $n+a+d+1$-qubit projector $|0\rangle\langle 0|$. Let $|E_{\psi_1}\rangle_n := \frac{|\psi_1\rangle}{\mathcal{N}_1} - \frac{\alpha}{\mathcal{N}_1}(\langle 0|_{a+d+1} \otimes I_n)V_1(|0\rangle_{n+a+d+1})$. Then, by Definition 1.2.7, $\||E_{\psi_1}\rangle_n\|_2 \leq \epsilon/\mathcal{N}_1$. Moreover, $\langle 0|_{a+d+1} \otimes I_n)V_1(|0\rangle_{n+a+d+1}) = \frac{1}{\alpha}(|\psi_1\rangle_n - \mathcal{N}_1|E_{\psi_1}\rangle_n)$. Then, observe that $V_2 := U_0(I_2 \otimes V_1^\dagger)$ is a $(1, 2, 0)$-block-encoding for $|0\rangle\langle 0|V_1^\dagger$. Let $c = a + d + 1$. Noting that $(|0\rangle_{c+2} \otimes I_n) = (|0\rangle_2 \otimes I_{c+n})(|0\rangle_c \otimes I_n)$, then,

$$(\langle 0|_{c+2} \otimes I_n)V_2(|0\rangle_{c+2} \otimes I_n) = (\langle 0|_c \otimes I_n)(\langle 0|_2 \otimes I_{c+n})V_2(|0\rangle_2 \otimes I_{c+n})(|0\rangle_c \otimes I_n) \tag{D.54}$$

$$= (\langle 0|_c \otimes I_n)|0\rangle\langle 0|V_1^\dagger(|0\rangle_c \otimes I_n) \tag{D.55}$$

$$= \frac{1}{\alpha}\left(|0\rangle_n(\langle\psi_1|_n - \mathcal{N}_1\langle E_{\psi_1}|_n)\right). \tag{D.56}$$

The third inequality follows by noting that $(\langle 0|_c \otimes I_n)|0\rangle_{n+c} = |0\rangle_n$, and that by Definition 1.2.6, $(\langle 0|_2 \otimes I_{c+n})V_2|0\rangle_2 \otimes I_{c+n}) = |0\rangle\langle 0|V_1^\dagger$. Then, letting $|0\rangle\langle\psi_1|$ be a $2^n \times 2^n$ projector,

$$\||0\rangle\langle\psi_1| - \alpha(\langle 0|_{c+2} \otimes I_n)V_2(|0\rangle_{c+2} \otimes I_n)\|_2 = \mathcal{N}_1 \||0\rangle\langle E_{\psi_1}|\|_2 \leq \epsilon. \tag{D.57}$$

Consequently, $V_2$ is a $(\alpha, a+d+3, \epsilon)$-block-encoding for the $2^n \times 2^n$ projector $|0\rangle\langle\psi_1|$. Moreover, the circuit complexity of $V_2$ is dominated by the circuit complexity of $V_1$, and thus is $O(T_\psi + dn)$. Then, $V_3 := V_2 \otimes I_n$ is a $(\alpha, a+d+3, \epsilon)$-block-encoding for $(|0\rangle\langle\psi_1|) \otimes I_n$.

Let $U_W$ be defined as in Definition D.2.2, i.e., it enacts $U_W|j\rangle_n|0\rangle_n = |j\rangle_n|w_j\rangle_n$.

Define $|\phi\rangle_{2n} := \sum_{j=0}^{N-1} \psi_j |j\rangle_n |w_j\rangle_n$.

Then, let $S := (I_a \otimes U_W)(U_\psi \otimes I_n)$. We will now show that $S$ is an $(\alpha, a, \epsilon)$-VE for $|\phi\rangle_{2n}$.

Let $|E_\psi\rangle_n := |\psi\rangle_n - \alpha(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$, thus, $(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n} = \frac{1}{\alpha}(|\psi\rangle_n - |E_\psi\rangle_n)$

Moreover, define the $a$-qubit projector, $p_j^a := |j\rangle\langle j|$. Then, $I_{a+n} = \sum_{j=0}^{2^a-1} p_j^a \otimes I_n$. Finally, define $|\gamma_j\rangle_n := (\langle j|_a \otimes I_n)U_\psi|0\rangle_{a+n}$. Of course,

$$U_\psi|0\rangle_{a+n} = \left(\sum_{j=0}^{2^a-1} p_j^a \otimes I_n\right) U_\psi|0\rangle_{a+n} = \frac{1}{\alpha}\left(|0\rangle_a(|\psi\rangle_n - |E_\psi\rangle_n)\right) + \sum_{j=1}^{2^a-1} |j\rangle_a|\gamma_j\rangle_n. \tag{D.58}$$

Consequently,

$$\begin{aligned}
(\langle 0|_a \otimes I_{2n})S|0\rangle_{a+2n} &= (\langle 0|_a \otimes I_{2n})(I_a \otimes U_W)(U_\psi \otimes I_n)|0\rangle_{a+2n} && \text{(D.59)}\\
&= (\langle 0|_a \otimes U_W)\left[\frac{1}{\alpha}\left(|0\rangle_a(|\psi\rangle_n - |E_\psi\rangle_n)\right) + \sum_{j=1}^{2^a-1} |j\rangle_a|\gamma_j\rangle_n\right]|0\rangle_n && \text{(D.60)}\\
&= \frac{1}{\alpha}\left(|\phi\rangle_{2n} - U_W|E_\psi\rangle_n|0\rangle_n\right). && \text{(D.61)}
\end{aligned}$$

Thus,

$$\||\phi\rangle_{2n} - \alpha(\langle 0|_a \otimes I_{2n})S|0\rangle_{a+2n}\|_2 = \|U_W|E_\psi\rangle_n|0\rangle_n\|_2 \leq \epsilon. \tag{D.62}$$

Thus, $S$ is an $(\alpha, a, \epsilon)$-VE for $|\phi\rangle_{2n}$. Moreover, the circuit complexity of $S$ comes from summing the circuit complexity of $U_\psi$ and $U_W$. As per Definition D.2.2, the circuit complexity of $U_W$ is $O(n^2)$, giving an overall circuit complexity for $S$ of $O(T_\psi + n^2)$.

Define $|\Gamma\rangle_n := Wg(|\psi\rangle_n)$, and note that

$$[(|0\rangle\langle\psi_1|) \otimes I_n]|\phi\rangle_{2n} = |0\rangle_n \sum_{j=0}^{N-1} |\psi_j|^2 a_j |w_j\rangle_n = |0\rangle_n |\Gamma\rangle_n. \tag{D.63}$$

We now have $V_3$, a $(\alpha, a+d+3, \epsilon)$-block-encoding for $(|0\rangle\langle\psi_1|)\otimes I_n$, and $S$ an $(\alpha, a, \epsilon)$-VE for $|\phi\rangle_{2n}$. We will now invoke Lemma 5.2.2 to take the product of the matrix encoded in $V_3$ with the vector encoded in $S$, and then will invoke Lemma D.2.10 to remove the $|0\rangle_n$ tensored register. This yields $U_f$, an $(\frac{\alpha^2}{\mathcal{N}}, 2a+d+3+n, \frac{2\alpha\epsilon}{\mathcal{N}})$-VE for $|\Gamma\rangle_n/\mathcal{N}$ with circuit complexity $O(T_\psi + dn + n^2)$. □

### D.2.2 Convolution Block-Encoding

In this section, I will first provide a matrix-form of a 2D multi-filter convolution (with stride 1 and 0 padding to ensure the input and outputs have the same dimension). I then derive a quantum block-encoding of the matrix form of the convolution.

As a note, some popular deep learning frameworks such as PyTorch [203] actually implement cross-correlation rather than convolution. However, in the pre-processing stage, the following convolutional block-encoding immediately gives a cross-correlation block-encoding by simply switching the $Q$ operator (Definition D.2.5) with a $Q^T$ operator. Finally, in this section, I assume that all addition on basis vectors is mod the dimension of the vector. I.e., for integers $i, j$, $|i+j\rangle_n = |(i+j) \bmod N\rangle_n$ (with $N = 2^n$).

**Definition D.2.3** (Permutation Matrix)**.** *Define the following $N$ dimensional unitary permutation matrix that maps an input basis vector $i$ to the basis vector $(i+1)$ mod $N$.*

$$P := \sum_{i=0}^{N} |i+1\rangle\langle i| = \begin{pmatrix} 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \tag{D.64}$$

**Definition D.2.4** ($R_Z$ Phase Gate)**.** *Define the single-qubit phase gate, $R_Z(t) := e^{itZ} = \begin{pmatrix} e^{it} & 0 \\ 0 & e^{-it} \end{pmatrix}$.*

I now derive a block-encoding of the permutation matrix $P$ acting on $m$ qubits. I include this result for completeness, and similar results may be found in the literature (see e.g., [204], where they derive a 1D circulant convolution via QSP, or [205]). This implementation of $P^m$ is identical to a $+m$ adder implemented with QFT, see e.g., [30].

**Lemma D.2.12** (Permutation Matrix Block-Encoding)**.** *Let $m \in \mathbb{N}_{>0}$. Let $N = 2^n$. The $m^{th}$ power of the permutation matrix $P$ is given by $P^m = \sum_{j=0}^{N-1} |j+m\rangle\langle j|$. Then, we can get a $(1,1,0)$-block-encoding with $O(n^2)$ circuit complexity for $P^m$.*

*Proof.* Drawing inspiration from [204, 206], let $F := QFT$ represent the Quantum Fourier Transform on $n$ qubits. Define $\omega_N^j := e^{2\pi ij/N}$. Noting that $F = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \omega_N^{ij} |i\rangle\langle j|$, it is easy to show that $P^m F|j\rangle = \omega_N^{-mj} F|j\rangle$. Consequently, we can write $P^m = FDF^{-1}$, where $D = \text{diag}(\omega_N^0, \omega_N^{-m}, \ldots, \omega_N^{-m(N-1)})$. Thus, by getting a block-encoding of $D$, we can implement $P^m$ by taking a product of $FDF^{-1}$. Let $|j\rangle_n$ be a basis vector, and let $j = 2^{n-1}j_{n-1} + \ldots + 2^1 j_1 + 2^0 j_0$. We will now give a unitary $V_m$ which implements the mapping $V_m|0\rangle_1|j\rangle_n = \omega_N^{-jm}|0\rangle_1|j\rangle_n$. Noting that $\omega_N^{-jm} = e^{-2\pi ijm/N} = \prod_{l=0}^{n-1} e^{-2\pi im(2^{j_l} j_l)/N}$, we can apply a sequence of $n$ controlled $R_Z(t)$ gates, where the $l^{th}$ gate is controlled on bit $j_l$ and applies $R_Z(m2^{j_l}/N)$ on the ancilla qubit. This implements the desired mapping, and can be easily shown to be a $(1,1,0)$-block-encoding for $D$. The Quantum Fourier Transform [207] can be implemented with $O(n^2)$ circuit complexity [25], and so we can get a trivial $(1,0,0)$-block-encoding for both $F$ and $F^\dagger$. Thus, $(I_1 \otimes F)V_m(I_1 \otimes F^\dagger)$ is a $(1,1,0)$-block-encoding for $P^m$, with $O(n^2)$ circuit depth. Its worth noting that since the ancilla qubit in $V_m$ is separable after the computation, this could be equivalently considered a $(1,0,0)$-block-encoding. □

**Definition D.2.5** (Discrete Unilateral Shift Operator)**.** *Define $Q$ to be the $N$-dimensional discrete unilateral shift operator,*

$$Q := \sum_{j=0}^{N-2} |j+1\rangle\langle j| = \begin{pmatrix} 0 & 0 & \ldots & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}. \tag{D.65}$$

*This is just the permutation matrix $P$ without wrap-around.*

**Lemma D.2.13** (Block-Encoding of $Q$)**.** *Let $N = 2^n$. Define $Q$ as per Definition D.2.5. Then, we can obtain a $(1, 4, 0)$-block-encoding for $Q$ with $O(n^2)$ circuit complexity.*

*Proof.* By Lemma D.2.12, we can obtain a $U_P$ a $(1, 1, 0)$-block-encoding of $P = \sum_{j=0}^{N-1} |j+1\rangle\langle 1|$ with $O(n^2)$ circuit complexity. By Lemma D.2.5, we can obtain $V$ a $(1, 2, 0)$-block-encoding of the $n$-qubit projector $|0\rangle\langle N-1|$ with $O(n)$ circuit depth. Following LCU [26, 202], we can get the sum of these two block-encodings, introducing an additional ancilla, with the circuit $U_f := (H \otimes I_{2+n})(|0\rangle\langle 0|_1 \otimes I_1 \otimes U_P - |1\rangle\langle 1|_1 \otimes V)(H \otimes I_{2+n})$. Then, $U_f$ is a $(1, 3, 0)$-block-encoding for $\frac{1}{2}(P - |0\rangle\langle N-1|) = \frac{1}{2}Q$, with $O(n^2)$ circuit complexity. Noting that $Q^\dagger Q = I_n - |N-1\rangle\langle N-1|$, it is clear that $\|Q\|_2 \leq 1$. Moreover, since all the singular values of $Q/2$ are either 0 or $1/2$, we can invoke Lemma D.2.3, a special case of oblivious amplitude amplification [26], to immediately convert this to a $(1, 4, 0)$-block-encoding for $Q$ with only 3 calls to $U_f$. □

**Lemma D.2.14** (Block-Encoding of $Q^m$)**.** *Let $m \in \mathbb{N}_{>0}$ and let $N = 2^n$. Define the $N$-dimensional operator $Q$ as per Definition D.2.5. Then, we can obtain a $(1, 4m, 0)$-block-encoding of $Q^m$ with $O(mn^2)$ circuit complexity.*

*Proof.* As per Lemma D.2.13, we can obtain $U_Q$ a $(1, 4, 0)$-block-encoding for $Q$ with $O(n^2)$ circuit complexity. Invoking Lemma 53 (Product of Block-Encoded Matrices) of [26] with $U_Q$ $m$ times directly yields a $(1, 4m, 0)$-block-encoding of $Q^m$ with $O(mn^2)$ circuit complexity.[2] □

Now, I present a standard well-known result giving the matrix form of a 2D multi-filter convolution (see e.g., [160, 206]).

**Lemma D.2.15** (Matrix Form of $2D$ Multi-Filter Convolution)**.** *Let $M = 2^m$, let $n = 2m$, let $N = 2^n$, and let $D = 2^d$. Let $C = 2^c$ represent the number of input and output channels. Let $X$ represent the rank$-3$ input tensor, which in*

[2] This can likely be optimizing by using QSVT [26].

*vectorized form (stored in column-major order for each input channel) is given by,* $|X\rangle_{n+c} = \sum_{i=0}^{C-1}\sum_{j=0}^{M-1}\sum_{k=0}^{M-1} X_{i,k,j}|i\rangle_c|j\rangle_m|k\rangle_m$. *I.e.,* $|X\rangle_{n+c}$ *is of dimension* $M^2C = NC$. *Define* $\tilde{X}_{i,j,k} = X_{i,j,k}$ *if* $j \geq 0$ *and* $k \geq 0$, *and* $\tilde{X}_{i,j,k} = 0$ *otherwise. We can define the convolutional kernel* $K$ *to be a rank-*4 *tensor containing each of the* $C$, $C \times D \times D$ *filters*[3], *where the first index represents the output channel, the second index represent the input channel, the third index represents the row index, and the fourth index represents the column index. Then, entry* $y, z$ *of the* $x^{th}$ *output channel after convolution with* $K$ *is given by,*

$$[X * K]_{x,y,z} := \sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1} K_{x,j,k,l}\tilde{X}_{j,z-k,y-l}. \tag{D.66}$$

*Defining* $Q$ *as per Definition D.2.5, we can give the matrix form of the convolution,*

$$\mathcal{C} := \sum_{i=0}^{C-1}\sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1} K_{i,j,k,l}(|i\rangle\langle j|_c \otimes Q^l \otimes Q^k). \tag{D.67}$$

*I.e.,* $\mathcal{C}|X\rangle_{n+c} = vec(X * K)$.

*Proof.* We will verify that $\mathcal{C}$ indeed implements the mapping specified in Eq. (D.66) by computing the following, $\langle x|_c\langle y|_m\langle z|_m\mathcal{C}|X\rangle_{n+c}$. Note that for all $i < l$, $\langle i|Q^l = 0$, and that for all $i \geq l$, $\langle i|Q^l = \langle i-l|$. Consequently, if $y - l \geq 0, z - k \geq 0$, then $\langle j|_c \otimes (\langle y|_m\langle z|_m Q^l \otimes Q^k)|X\rangle_{n+c} = X_{j,z-k,y-l}$, and if $y - l < 0$ or $z - k < 0$ then $\langle j|_c \otimes (\langle y|_m\langle z|_m Q^l \otimes Q^k)|X\rangle_{n+c} = 0$. Thus, $\langle j|_c \otimes (\langle y|_m\langle z|_m Q^l \otimes Q^k)|X\rangle_{n+c} = \tilde{X}_{j,z-k,y-l}$. Therefore,

$$\langle x|_c\langle y|_m\langle z|_m \sum_{j=0}^{C-1} K_{i,j,k,l}(|i\rangle\langle j|_c \otimes Q^l \otimes Q^k)|X\rangle_{n+c} = \sum_{j=0}^{C-1} K_{x,j,k,l}\tilde{X}_{j,z-k,y-l}. \tag{D.68}$$

As a result,

$$\langle x|_c\langle y|_m\langle z|_m\mathcal{C}|X\rangle_{n+c} = \sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1} K_{x,j,k,l}\tilde{X}_{j,z-k,y-l} = [X * K]_{x,y,z}. \tag{D.69}$$

□

[3]If the number of channels is 1 (i.e., $C = 1$), then the kernel is $D \times D$ dimensional.

***Proof of Lemma 5.2.5***. Define $|X\rangle_{n+c}$, $K$, and $\mathcal{C}$ as per Lemma D.2.15. As a result, obtaining a block-encoding of $\mathcal{C}$ allows us to implement the desired $2D$ convolution in the vectorized setting.

First, for a given $i, j, k, l$, I will show how to obtain a block-encoding of $K_{i,j,k,l}(|i\rangle\langle j|_c \otimes Q^l \otimes Q^k)$.

Using Lemma D.2.5, we can obtain $U_{i,j}$ a $(1, 2, 0)$-block-encoding of the $c$-qubit projector $|i\rangle\langle j|_c$, with $O(c)$ circuit depth. Then, using Lemma D.2.14, we can obtain $U_{Q^l}$ a $(1, 4l, 0)$-block-encoding of $m$ qubit $Q^l$ with $O(lm^2)$ circuit complexity. We similarly obtain $U_{Q^k}$ a $(1, 4k, 0)$-block-encoding of $m$ qubit $Q^k$ with $O(km^2)$ circuit complexity. We can then invoke Lemma D.2.1 with $U_{i,j}$ and $U_{Q^l}$, and again with $U_{Q^k}$, to obtain $U_{i,j,l,k}$, a $(1, 2 + 4l + 4k, 0)$-block-encoding of $|i\rangle\langle j|_c \otimes Q^l \otimes Q^k$ with $O(c + Dm^2)$ circuit complexity. To make each operator act on the same number of qubits, we will augment each with the appropriate number of tensored identities to yield a $(1, 2 + 8D, 0)$-block-encoding for the corresponding operator.

Define $|K\rangle_{2c+2d} := \sum_{i=0}^{C-1}\sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{k=0}^{D-1} K_{i,j,k,l}|i\rangle_c|j\rangle_c|k\rangle_d|l\rangle_d$. Additionally, define $|\sqrt{K}\rangle_{2c+2d} = \sqrt{|K\rangle_{2c+2d}}$ (with the square-root applied element-wise). Then, define $\mathcal{N}_K := \left\||\sqrt{K}\rangle_{2c+2d}\right\|_2 = \||K\rangle_{2c+2d}\|_1^{1/2}$, and $|\overline{K}\rangle_{2c+2d} := |K\rangle_{2c+2d}/\mathcal{N}_K$. Noting that this vector is $C^2D^2$ dimensional, we can brute-force construct a unitary $U_K$, with a total of $O(C^2D^2)$ single and two qubit gates, such that $U_K|0\rangle_{2c+2d} = |\sqrt{K}\rangle_{2c+2d}/\mathcal{N}_K$ [77]. We can then invoke Lemma D.2.4, obtaining a $(\mathcal{N}_K^2, 2 + 8D + 2\log(CD), 0)$-block-encoding for $\mathcal{C}$ with $O(cdC^2D^3m^2)$ circuit complexity. This is equivalent to a $(1, 2 + 8D + 2\log(CD), 0)$-block-encoding for $\mathcal{C}/\left\||K\rangle_{2c+2d}\right\|_1$. Since we are concerned with accelerating inference, we will ignore classical pre-computation costs that must only be paid one time to construct this datastructure. We can then invoke Lemma D.2.2, setting $\gamma = \left\||K\rangle_{2c+2d}\right\|_1 / 2\left\|\mathcal{C}\right\|_2$ and $\delta = 1/2$, since $\left\|\mathcal{C}/\left\||K\rangle_{2c+2d}\right\|_1\right\|_2 \leq \frac{1}{2}\frac{2\|\mathcal{C}\|_2}{\||K\rangle_{2c+2d}\|_1}$. Neglecting the logarithmic error-terms incurred by Lemma D.2.2 (as these will not dominate complexity), this then yields a $(1, 3 + 8D + 2\log(CD), 0)$-block-encoding for $\frac{\mathcal{C}}{2\|\mathcal{C}\|_2}$ with $O(\frac{\||K\rangle_{2c+2d}\|_1}{\|\mathcal{C}\|_2}cdC^2D^3m^2)$ circuit depth. We will now show that $\frac{\||K\rangle_{2c+2d}\|_1}{\|\mathcal{C}\|_2} \leq DC^{3/2}$, and thus that the overall circuit depth is bounded by $O(cdm^2C^3D^4)$.

We will now upper-bound $\||K\rangle_{2c+2d}\|_1$. Define the basis vector $|x\rangle_{c+2m} = |x_1\rangle_c|x_2\rangle_m|x_3\rangle_m$. Then, the $x^{th}$ row of $\mathcal{C}$ is given by $\langle x|_{c+2m}\mathcal{C}$. Simple analysis shows that $\langle x|_{c+2m}\mathcal{C} = \sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1} K_{x_1,j,k,l}\langle j|_c \otimes \langle x_2 - l|_m \otimes \langle x_3 - k|_m$, where $\langle x_2 - l|_m = 0$ if $x_2 - l < 0$ and $\langle x_3 - k|_m = 0$ if $x_3 - k = 0$. Then it can be readily shown that $\|\langle x|_{c+2m}\mathcal{C}\|_2^2 = \sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1}|K_{x_1,j,k,l}|^2$. For any operator $A$ with $\|A\|_2 \leq 1$, the maximum column norm $\max_{|i\rangle}\|A|i\rangle\|_2 \leq \max_{|\psi\rangle:\||\psi\rangle\|_2=1}\|A|i\rangle\|_2 \leq 1$. Similarly, since $\|A\|_2 = \left\|A^\dagger\right\|_2$, the maximum row norm cannot exceed the spectral norm of the matrix. Therefore, any row of $\mathcal{C}$ must have $\ell_2$-norm bounded by $\|\mathcal{C}\|_2$, thus, $\sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1}|K_{x_1,j,k,l}|^2 \leq \|\mathcal{C}\|_2^2$. Consequently, $\||K\rangle_{2c+2d}\|_2^2 = \sum_{i=0}^{C-1}\sum_{j=0}^{C-1}\sum_{k=0}^{D-1}\sum_{l=0}^{D-1}|K_{i,j,k,l}|^2 \leq C\|\mathcal{C}\|_2^2$. Moreover, for an $n$-dimensional vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|_1 \leq \sqrt{n}\|\boldsymbol{x}\|_2$, and thus, $\||K\rangle_{2c+2d}\|_1 \leq \sqrt{C^2D^2}\sqrt{C}\|\mathcal{C}\|_2 = DC^{3/2}\|\mathcal{C}\|_2$. Consequently, $\||K\rangle_{2c+2d}\|_1 / \|\mathcal{C}\|_2 \leq DC^{3/2}$. □

To see a set of related block-encoding circuits, see [205].

It is also worth noting that the preceding result can be made substantially more efficient by utilizing a circulant convolution to implement the non-circulant convolution. I will now quickly sketch this idea for future optimization. For simplicity, I assume that the convolution has one input channel and one output channel, and that the input is a rank-2 tensor (e.g., a black and white image). Let $M = 2^m$. Then, if the input image is $X \in \mathbb{R}^{M\times M}$, we can add 0 padding with the $M \times M$ projector, $|0\rangle\langle 0|_m \otimes X$. Then, enacting a circulant convolution on this augmented operator and projecting onto the zero-state of the first register yields the desired non-circulant convolution. Moreover, we can define a circulant $2D$ convolution as $[X * K]_{i,j} = \sum_{k=0}^{l-1}\sum_{l=0}^{d-1} K_{k,l}X_{i-k,j-l}$. The following sketch generalizes the 1D circulant convolution given in [204], and also follows the ideas discussed in [206]. Consequently, the operator $C := \sum_{i=0}^{d-1}\sum_{j=0}^{d} K_{i,j}P^j \otimes P^i$ implements $X * K$ in the vectorized setting (using a column-major vectorization for $X$). Let $\omega_M := \exp(2\pi i/M)$ be the $M^{th}$ root of unity. Let $F := QFT$ represent the Quantum Fourier Transform on $m$ qubits. It is easy to show that $P^kF|j\rangle = \omega_M^{-kj}F|j\rangle$. Thus, let $D := F^{-1}PF = \text{diag}(\omega_M^0, \omega_M^{-1}, \ldots, \omega_M^{-(M-1)})$. Consequently, $P = FDF^{-1}$, and so $C = (F \otimes F)\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} K_{i,j}D^j \otimes D^i\right)(F^{-1} \otimes F^{-1})$. Clearly, since implementing

the QFT is efficient on a quantum computer, the key to implementing $C$ is in implementing a block-encoding of the diagonal matrix $\Gamma := \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} K_{i,j} D^j \otimes D^i$. Noting that this is a 1-sparse matrix with efficiently computable entries, a technique such as [26] can be immediately used to obtain the desired block-encoding (replacing QRAM assumptions with arithmetic oracles computing the locations and values of the non-zero elements). This can be further optimized by replacing the arithmetic with QRAM. In the multi-filter case, the diagonal matrix becomes a block-diagonal matrix (with blocks of height and width given by the number of input and output channels), and the sparse block-encoding techniques can still be used.

### D.2.3 Non-Linear Transformation of Vector-Encodings

I now present an essential result on transforming the amplitudes of a state encoded as a VE. This result is a direct translation of the ideas in the result given in Chapter 4 (which in turn builds on [176, 208]) to the setting of VEs. While I also give a similar result in the setting of a VE in Chapter 4, there I obtain it by treating the whole unitary VE as a state-preparation unitary, and then invoke the non-linear amplitude transformation (NLAT) result on that, which gives slightly worse complexity than just directly re-deriving the whole transformation result in the framework of VEs. I include the following for completeness and simplicity, and the following is not a substantially novel contribution of this chapter.

**Lemma D.2.16** (NLAT of VE [3])**.** *Let $N = 2^n$. Let $0 \leq \epsilon_0 \leq 1$, and $\alpha \geq 1$. We are given a unitary matrix $U_\psi$ which is an $(\alpha, a, \epsilon_0)$-VE for the $n$-qubit real quantum state $|\psi\rangle_n$ with circuit complexity $O(T)$, and a function $f : \mathbb{R} \mapsto \mathbb{R}$ with Lipschitz constant $L$ such that $f(0) = 0$. Define $\epsilon_1$ such that $0 < \epsilon_1 \leq L$. Define $\mathcal{N} := \|f(|\psi\rangle_n/\alpha)\|_2$. Define the interval of approximation $[-\tau, \tau]$, where $0 < \tau \leq 1$ which can be set to either $\tau = 1$ or any value such that $\tau \geq \frac{1+\epsilon_0}{\alpha}$ if a smaller region of approximation yields a better complexity. Define the polynomial $P : \mathbb{R} \mapsto \mathbb{R}$, such that with degree $k$, $\max_{x\in[-\tau,\tau]} |P(x) - f(x)| \leq \frac{L\epsilon_1}{2\sqrt{N}}$. Suppose we are given a bound $\tilde{\gamma}$ satisfying $\tilde{\gamma} \geq \max_{x\in[-1,1]} |P(x)/x|$, and require that $P(0) = 0$. Then, we can obtain a unitary circuit $U_f$ that is a $\left(\frac{4\tilde{\gamma}}{\mathcal{N}}, n + 2a + 4, \frac{L}{\mathcal{N}}(\epsilon_0 + \epsilon_1)\right)$-VE for*

*$f(|\psi\rangle_n/\alpha)/\mathcal{N}$, and which requires $O(k)$ calls to a controlled $U_\psi$ and $U_\psi^\dagger$ circuit, and has a total circuit depth of $O(k(n+a+T))$. This circuit can be obtained with $O(poly(k, \log(\frac{\tilde{\gamma}}{\mathcal{N}\epsilon_1})))$ classical time complexity.*

*Proof.* We will begin by considering the domain we require for the polynomial approximation. Essentially, by noting that if $\alpha > 1$, the function is being applied to a sub-component of an $\ell_2$-normalized vector, and thus the maximum value of its input will be strictly less than $\frac{1+\epsilon_0}{\alpha}$. In some cases, this could yield a more efficient polynomial approximation, and so we will write our result both in the setting where the interval of approximation is $[-1, 1]$ and $[-\frac{1+\epsilon_0}{\alpha}, \frac{1+\epsilon_0}{\alpha}]$. In particular, the function will be applied to $(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$, and so we must upper-bound the maximum amplitude in this quantity. Define $c \in \mathbb{R}$ such that $0 < c \leq 1$. Define the un-normalized vector $|\phi\rangle_n := (\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n}$. Define $|E_\psi\rangle_n := |\psi\rangle_n - \alpha|\phi\rangle_n$, and note that $\||\phi\rangle_n\|_2 \leq 1$, and thus that $\frac{1}{\alpha}\||\psi\rangle_n - |E_\psi\rangle_n\|_2 \leq 1$. Additionally, by Definition 1.2.7, $\||E_\psi\rangle_n\|_2 \leq \epsilon_0$. Define $\{\phi_j\}_j$ such that $|\phi\rangle_n = \sum_{j=1}^N \phi_j|j\rangle_n$. Thus, $|\phi_j| \leq \||\phi\rangle_n\|_2 \leq \frac{1}{\alpha}(1+\epsilon_0)$. Define $c := \min(\frac{1}{\alpha}(1+\epsilon_0), 1)$.

Let $\mathcal{N}_\psi := \mathcal{N}$. Let $\mathcal{N}_P := \|P(|\psi\rangle/\alpha)\|_2$. Define the degree $k-1$ polynomial $Q(x) := P(x)/x$, and define $\epsilon_2$ such that $\max_{x\in[-c,c]} |P(x) - f(x)| \leq \epsilon_2$.

Using Lemma C.2.4, we can get a $(1, a+n+2, 0)$-block-encoding $U_A$ of $A := \text{diag}(U_\psi|0\rangle_{a+n})$ with $O(a+n)$ circuit depth, and 6 additional calls to a controlled $U_\psi$ circuit. Invoking Theorem 1.2.1 (Theorem 56 of [26]) with $Q(x)/4\tilde{\gamma}$, we get the unitary $U_Q$, a $(1, a+n+4, \delta)$-block-encoding for $Q(A)/4\tilde{\gamma}$, requiring $O((a+n)k)$ single and two-qubit gates, $O(k)$ calls to a controlled $U_A$ circuit, and $O(poly(k, \log(1/\epsilon)))$ classical computation to determine the QSVT rotation angles to implement the degree $k$ polynomial. We can equivalently call $U_Q$ a $(1, a+n+4, 0)$-block-encoding for some matrix $V$, such that $\|V - Q(A)/4\tilde{\gamma}\|_2 \leq \delta$. Additionally, define $E_Q := V - Q(A)/4\tilde{\gamma}$. Since for any vector $\boldsymbol{x}$, $Q(\text{diag}(\boldsymbol{x}))\boldsymbol{x} = P(\boldsymbol{x})$, we get, $VU_\psi|0\rangle_{a+n} = \frac{P(U_\psi|0\rangle_{a+n})}{4\tilde{\gamma}} + E_V U_\psi|0\rangle_{a+n}$. Additionally, noting that $(\langle 0|_a \otimes I_n)P(\boldsymbol{x}) = P((\langle 0|_a \otimes I_n)\boldsymbol{x})$, and that $(\langle 0|_a \otimes I_n)U_\psi|0\rangle_{a+n} = |\phi\rangle_n$, we get $(\langle 0|_a \otimes I_n)VU_\psi|0\rangle_{a+n} = \frac{P(|\phi\rangle_n)}{4\tilde{\gamma}} + (\langle 0|_a \otimes I_n)E_V U_\psi|0\rangle_{a+n}$. Define $\tilde{U}_\psi := I_{a+n+4} \otimes U_\psi$.

First, note that $\tilde{U}_\psi|0\rangle_{2a+2n+4} = (|0\rangle_{n+a+4} \otimes I_{a+n})U_\psi|0\rangle_{a+n}$. Then, note that $(\langle 0|_{n+2a+4} \otimes I_n) = (\langle 0|_a \otimes I_n)(\langle 0|_{n+a+4} \otimes I_{a+n})$. Consequently, by Definition 1.2.6, since $(\langle 0|_{n+a+4} \otimes I_{a+n})U_Q(|0\rangle_{n+a+4} \otimes I_{a+n}) = V$,

$$(\langle 0|_{n+2a+4} \otimes I_n)U_Q\tilde{U}_\psi|0\rangle_{2n+2a+4} = (\langle 0|_a \otimes I_n)VU_\psi|0\rangle_{a+n} \tag{D.70}$$

$$= \frac{P(|\phi\rangle_n)}{4\tilde{\gamma}} + (\langle 0|_a \otimes I_n)E_V U_\psi|0\rangle_{a+n}. \tag{D.71}$$

We will now show that $U_Q\tilde{U}_\psi$ is a VE for $\frac{1}{\mathcal{N}_\psi}f(|\psi\rangle_n/\alpha)$. Precisely, we must upper-bound,

$$\xi_1 := \left\| \frac{1}{\mathcal{N}_\psi} f(|\psi\rangle_n/\alpha) - \frac{4\tilde{\gamma}}{\mathcal{N}_\psi}(\langle 0|_{n+2a+4} \otimes I_n)U_Q\tilde{U}_\psi|0\rangle_{2n+2a+4} \right\|_2 \tag{D.72}$$

$$\leq \frac{1}{\mathcal{N}_\psi}\left( \|f(|\psi\rangle_n/\alpha) - P(|\phi\rangle_n)\|_2 + 4\tilde{\gamma}\,\|(\langle 0|_a \otimes I_n)E_V U_\psi|0\rangle_{a+n}\|_2 \right). \tag{D.73}$$

Let $\langle j|E_\psi\rangle := e_j$. We will now prove a sequence of simple facts. Since $|f(x) - f(x+b)| \leq L|b|$, and using $|\phi\rangle_n = \frac{|\psi\rangle_n - |E_\psi\rangle_n}{\alpha}$, we have that $\|f(|\psi\rangle_n/\alpha) - f(|\phi\rangle_n)\|_2^2 = \sum_{i=j}^N |f(\psi_j) - f((\psi_j - e_j)/\alpha)|^2 \leq \frac{L^2}{\alpha^2}\sum_{j=1}^N |e_j|^2 = \frac{L^2}{\alpha^2}\||E_\psi\rangle_n\|_2^2 \leq \frac{L^2\epsilon_0}{\alpha^2}$. Then, $\|f(|\phi\rangle_n) - P(|\phi\rangle_n)\|_2^2 = \sum_{j=1}^N |f(\phi_j) - P(\phi_j)|^2 \leq \max_{x\in[-c,c]}|f(x) - P(x)|^2 N \leq \epsilon_2^2 N$. Then,

$$\|f(|\psi\rangle_n/\alpha) - P(|\phi\rangle_n)\|_2 = \|f(|\psi\rangle_n/\alpha) - f(|\phi\rangle_n) + f(|\phi\rangle_n) - P(|\phi\rangle_n)\|_2 \tag{D.74}$$

$$\leq \frac{L\epsilon_0}{\alpha} + \epsilon_2\sqrt{N}. \tag{D.75}$$

At this point, the proof branches into two cases. The first case is where we simply use the uniform approximation to the function on the entire interval $[-1, 1]$. The second case, which should only be used when approximating the function on $[-\tau, \tau]$ yields a better asymptotic approximation, will be proven after.

Noting that $\|(\langle 0|_a \otimes I_n)E_V U_\psi|0\rangle_{a+n}\|_2 \leq \delta$, we can now get the overall bound of

$$\xi_1 \leq \frac{1}{\mathcal{N}_\psi}\left(\frac{L\epsilon_0}{\alpha} + \epsilon_2\sqrt{N} + 4\tilde{\gamma}\delta\right) \leq \frac{1}{\mathcal{N}_\psi}\left(L\epsilon_0 + \epsilon_2\sqrt{N} + 4\tilde{\gamma}\delta\right). \tag{D.76}$$

Thus, we have shown that $U_Q\tilde{U}_\psi$ is a $(\frac{4\tilde{\gamma}}{\mathcal{N}_\psi}, 2a+n+4, \frac{1}{\mathcal{N}_\psi}(L\epsilon_0 + \epsilon_2\sqrt{N} + 4\tilde{\gamma}\delta))$-VE for $\frac{1}{\mathcal{N}_\psi}f(|\psi\rangle_n/\alpha)$. To get the overall error-bound, we will set $\epsilon_2\sqrt{N} = L\epsilon_1/2$, and $4\tilde{\gamma}\delta = L\epsilon_1/2$, yielding $\epsilon_2 = \frac{L\epsilon_1}{2\sqrt{N}}$, and $\delta = \frac{L\epsilon_1}{8\tilde{\gamma}}$. This gives a $(\frac{4\tilde{\gamma}}{\mathcal{N}_\psi}, 2a+n+4, \frac{L}{\mathcal{N}_\psi}(\epsilon_0+\epsilon_1))$-VE for $\frac{1}{\mathcal{N}_\psi}f(|\psi\rangle_n/\alpha)$, and requires $O(k)$ calls to a controlled $U_\psi$ and $U_\psi^\dagger$ circuit, and

has a total circuit depth of $O(k(n+a+T))$. This circuit can be obtained with $O(\text{poly}(k, \log(\frac{\tilde{\gamma}}{L\epsilon_1})))$ classical time complexity. □

To make the preceding result easier to use, I provide a special case for transformation by the error function, and again do not claim novelty (this is primarily a contribution of Chapter 4).

**Lemma D.2.17** (Application of $\text{erf}(\nu x)$ to a Vector Encoding)**.** *Let $N = 2^n$, let $\nu \geq 1/2$, let $1 \geq \epsilon_0 \geq 0$ and let $0 < \epsilon_1 \leq 2$. We are given a unitary matrix $U_\psi$ with circuit complexity $O(T)$ which is an $(\alpha, a, \epsilon_0)$-VE for the $n$-qubit quantum state $|\psi\rangle_n$, and we are also given the error function $f_\nu(x) = \text{erf}(\nu x)$. Let $\mathcal{N} := \|f_\nu(|\psi\rangle_n/\alpha)\|_2$. Then, we can obtain a $\left(\frac{16\nu}{\sqrt{\pi}\mathcal{N}}, 2a+n+4, 2\nu\alpha(\epsilon_0+\epsilon_1)\right)$-VE for $f_\nu(|\psi\rangle_n/\alpha)/\mathcal{N}$, with $O(\nu\log(\frac{\sqrt{N}}{\epsilon_1}))$ queries to a controlled $U_\psi$ and $U_\psi^\dagger$ circuit, and with a total circuit depth of $O(\nu\log(\frac{\sqrt{N}}{\epsilon_1})(a+n+T))$. Moreover, $\mathcal{N} \geq \frac{1}{2\alpha}$.*

*Proof.* From Lemma D.6.1, we know that the Lipschitz constant $L$ of $\text{erf}(\nu x)$ is $L = \frac{2\nu}{\sqrt{\pi}}$.

Define $c = O(1/\alpha)$. Using Lemma D.6.1, we can obtain a degree $k \in O(\nu\log(\nu/\alpha\epsilon'))$ polynomial $P_{k,\nu}$ such that $P_{k,\nu}(0) = 0$ and $\max_{x\in[-c,c]} |P_{k,\nu}(x) - f_\nu(x)| \leq \epsilon'$. Since we need $\epsilon' \leq \frac{L\epsilon_1}{2\sqrt{N}}$, we can set $\epsilon' = \frac{\nu\epsilon_1}{10\sqrt{N}}$ in accordance with Lemma D.2.16, we have a degree $k \in O(\nu\log(\frac{\sqrt{N}}{\epsilon_1}))$ polynomial approximation.

From Lemma D.6.1, for $\nu \geq 1/2$, we know that $\forall x \in [-1,0)\cup(0,1], |\text{erf}(\nu x)| \geq |x/2|$. Consequently, $\mathcal{N}^2 = \sum_{j=1}^N |f(\psi_j/\alpha)|^2 \geq (\frac{1}{2\alpha})^2$. Additionally, we know that $\tilde{\gamma} = \max_{x\in[-1,1]} |P_{k,\nu}(x)/x| \leq \frac{4\nu}{\sqrt{\pi}}$. Invoking Lemma D.2.16, setting with all of the above facts and setting $\tilde{\gamma} = \frac{4\nu}{\sqrt{\pi}}$ then gives the complexity. □

## D.3 General Architectural Blocks

The architectural blocks presented in this paper are intended to demonstrate how the different operations on encoded matrices and vectors can be combined to coherently implement various architectures on quantum computers. There are a rich set of possibilities, and I have only explored a small but elucidating set.

Two of the most important concepts governing the complexity of the quantum implementation of any classical architecture are: (1) the number of non-linear activation layers, and (2) the $\ell_2$ norm of the vectorized input tensor as it propagates through the network.

In order for a unitary matrix (a linear operator) to enact a non-linear transformation on a vector, its definition must depend on the vector it is being applied to. Consequently, techniques which enact non-linear transformations on state-amplitudes (e.g., Chapter 4 and [208]) must have circuit definitions which depend on the vector-encoding circuit they are being applied to. Thus, if the unitary circuit implementing the transformation requires *even two* calls to the input vector encoding, then the circuit complexity will grow exponentially with the number of non-linear activations. Consequently, wide but shallow multi-layer architectures are ideal for quantum acceleration. Finally, an alternative to fully coherent quantum acceleration is to periodically read-out the vector in intermediate layers of the network. As discussed in the introduction, several quantum computing papers have proposed this approach. However, in general, since reading out a quantum state incurs a dimension-dependent cost [130, 209] (and incurs polynomial error-dependence) this either imposes significant constraints on the types of architectures that can be accelerated (requiring frequent mappings to very low-dimensional spaces where readout is cheaper), or incur asymptotically dominating error accumulation. Nevertheless, there are certain settings where periodic state readout may be desirable, and the techniques presented in this chapter are fully compatible with these ideas.

The second key concept governing the complexity of a quantum implementation of an architecture relates to the norm of the encoded vector as it propagates through the network. Whenever a sample is drawn from an encoded vector, a cost inversely proportional to the norm of the encoded vector must be paid. Similarly, whenever an encoded vector is normalized, an inverse norm-cost must be paid. Consequently, to obtain provable end-to-end complexity results, it is essential to lower-bound the norm of the encoded vector prior to the application of a layer norm (or prior to drawing a sample from the output of the network). A key tool in doing this is

the skip connection, as it allows the norm from the previous layer to be preserved in the output of the next layer. Additionally, if the weight layers are normalized (i.e., if $W$ represents the matrix form of any parameter layer, then $\|W\|_2 \leq 1$), and the activation function is scaled so that its Lipschitz constant on the interval $[-1, 1]$ is at most 1, this results in provable norm-preservation bounds. Requiring weight-layers to be sub-normalized has been extensively explored in the classical deep learning literature [210–212], as sub-normalization can help prevent network norm explosion as deeper networks are trained.

It is worth briefly noting that, in certain cases, the sub-normalization condition on the weight layers can be removed (i.e., for matrix $W$, $0 \leq \|W\|_2 \leq c$ where $c \geq 1$). This is done by implementing $W/\|W\|_2$, and then scaling the input of the subsequent activation function by $\|W\|_2$. If using the error function activation, this increases the cost of the polynomial approximation by an amount proportional to $c$. I do not consider this regime as it makes it more challenging to prove norm preservation properties after the skip connection, but stress that quantum computers can actually implement such regimes. Numerical studies examining norm preservation for such networks could shed light into their efficiency.

I will now formally define our $\ell_2$-norm squared pooling; this is essentially just an $\ell_2$-norm pooling operation followed by an element-wise square. Throughout I will assume that dimensions neatly line-up, noting that if they don't padding can be used to easily and efficiently ensure alignment.

**Definition D.3.1** (Squared $\ell_2$ Norm Pooling)**.** *Given an $N$-dimensional vector $|\phi\rangle = \sum_{j=1}^{N} \phi_j |j\rangle$, and a positive integer $C$ such that $N$ is divisible by $C$, define $f_j := (j-1)\frac{N}{C} + 1$. Then, we define $\ell_2$-norm squared pooling by $\text{pool}_C(|\phi\rangle) := \sum_{j=1}^{C} \sum_{l=f_j}^{j\frac{N}{C}} \phi_l^2 |j\rangle$, where $\{|j\rangle\}$ is the set of $C$-dimensional basis vectors.*

**Lemma D.3.1** (Error Propagated Through $\ell_2$ Norm Squared Pooling)**.** *Define the $N$-dimensional vectors $|\phi\rangle$ and $|\tilde{\phi}\rangle$, such that $\left\||\phi\rangle - |\tilde{\phi}\rangle\right\|_2 \leq \epsilon$. Then, defining a positive integer $C$ such that $N$ is divisible by $C$, and defining $\text{pool}_C$ as per Definition D.3.1, we have that $\left\|\text{pool}_C(|\phi\rangle) - \text{pool}_C(|\tilde{\phi}\rangle)\right\|_2 \leq \frac{2N\epsilon}{\sqrt{C}}$.*

*Proof.* Let $|\phi\rangle = \sum_{j=1}^{N} \phi_j |j\rangle$, and let $|\tilde{\phi}\rangle = \sum_{j=1}^{N} \tilde{\phi}_j |j\rangle$. Then, $\left\| |\phi\rangle - |\tilde{\phi}\rangle \right\|_2 \leq \epsilon$ implies that $\forall j, |\phi_j - \tilde{\phi}_j| \leq \epsilon$. Then, additionally using that $|\phi_j + \tilde{\phi}_j| \leq 2$,

$$\left\| \text{pool}_C(|\phi\rangle) - \text{pool}_C(|\tilde{\phi}\rangle) \right\|_2^2 = \sum_{j=1}^{C} \left( \sum_{l=f_j}^{jN/C} (\phi_l - \tilde{\phi}_l)(\phi_l + \tilde{\phi}_l) \right)^2 \leq 4 \sum_{j=1}^{C} \left( \frac{N\epsilon}{C} \right)^2 = \frac{4N^2\epsilon^2}{C}. \tag{D.77}$$

□

I will now prove the key results on architectural blocks outlined in the main section of this chapter.

***Proof of Lemma 5.3.1***. The parameter $\kappa$ in the lemma is designed for situations where we don't have a perfect block-encoding of the matrix we would like. For instance, in cases where we want to apply some matrix $A$, but we are only able to get a block-encoding of $A/2$. We can fix this when applying the activation function by scaling its input to remove the $1/2$ factor.

Let $\nu := 4\kappa/5$. Let $|\phi_1\rangle_n := W|\psi\rangle_n/\kappa$, $\mathcal{N}_1 := \||\phi_1\rangle_n\|_2$, and $|\Phi_1\rangle_n := |\phi_1\rangle_n/\mathcal{N}_1$. Using Lemma 5.2.2 we get $U_1$ a $(\mathcal{N}_1^{-1}, a+b, \epsilon_0\mathcal{N}_1^{-1})$-VE for $|\Phi_1\rangle_n$ with $O(T_1+T_2)$ circuit complexity.

Let $|\phi_2\rangle_n := f(W|\psi\rangle_n))$, $\mathcal{N}_2 := \||\phi_2\rangle_n\|_2$, and $|\Phi_2\rangle_n := |\phi_2\rangle_n/\mathcal{N}_2$. Define $0 < \epsilon_1 \leq 1$. Invoking Lemma D.2.17 with $U_1$ and $f(\kappa x) = \text{erf}(4\kappa x/5) = \text{erf}(\nu x)$, we obtain $U_2$ a $\left(\frac{16\nu}{\sqrt{\pi}\mathcal{N}_2}, 2(a+b)+n+4, 2\nu\mathcal{N}_1^{-1}(\epsilon_0+\epsilon_1)\right)$-VE for $f(|\Phi_1\rangle_n\mathcal{N}_1)/\left\|f(|\Phi_1\rangle_n\mathcal{N}_1)\right\|_2 = |\Phi_2\rangle_n$. $U_2$ has circuit complexity $O(\nu\log(\frac{\sqrt{N}}{\epsilon_1})(a+b+n+T_1+T_2))$.

So as to invoke Lemma 5.2.1 to implement the skip connection and obtain a state proportional to $|\psi_f\rangle_n$, we will need to factor out a common factor of $\frac{\sqrt{\pi}}{16\nu}$. Consequently, we invoke Lemma D.2.8 on $U_\psi$ to obtain $U'_\psi$ a $(\frac{16\nu}{\sqrt{\pi}}, a+2, \epsilon_0)$-VE for $|\psi\rangle_n$ with $O(T_1+a)$ circuit complexity.

Define $|\gamma\rangle_n := \frac{\sqrt{\pi}}{32\nu}\left(|\psi\rangle_n + |\Phi_2\rangle_n\mathcal{N}_2\right) = \frac{\sqrt{\pi}}{32\nu}\left(|\psi\rangle_n + f(W|\psi\rangle_n)\right)$, $\mathcal{N}_\gamma := \||\gamma\rangle_n\|_2$ and $|\Gamma\rangle_n := |\gamma\rangle_n/\mathcal{N}_\gamma$. Then, we can invoke Lemma 5.2.1 (setting $\tau = 1/2$) with $U'_\psi$ and $U_2$, yielding $U_3$ a $\left(\mathcal{N}_\gamma^{-1}, 2(a+b)+n+5, N_\gamma^{-1}[\frac{\epsilon_0\sqrt{\pi}}{16\nu} + \frac{\mathcal{N}_2\sqrt{\pi}}{16\nu}(2\nu\mathcal{N}_1^{-1}(\epsilon_0+\epsilon_1))]\right)$-VE for $|\Gamma\rangle_n$, with circuit complexity $O(\nu\log(\frac{\sqrt{N}}{\epsilon_1})(a+b+n+T_1+T_2))$. We will now simplify the error component of this VE statement.

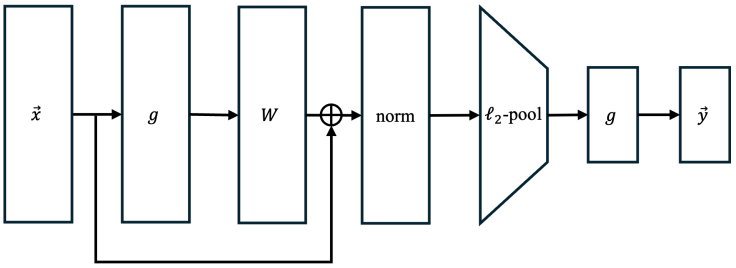


**Figure D.2: Full-rank linear-pooling output block.**

**Figure D.3:** This figure shows the final output architectural block used in the neural networks for Regimes 1 and 2. Here, $g(x) = x^2$ and $W$ is a sub-normalized (potentially full-rank and dense) matrix.

First, define $|x\rangle_n = \sum_i x_i |i\rangle_n = W|\psi\rangle_n$. Then, using the fact that $f(x) = \text{erf}(4x/5)$ has a Lipschitz-constant of $\frac{8}{5\sqrt{\pi}}$ (as per Lemma D.2.17), $\mathcal{N}_2^2 = \|f(W|\psi\rangle_n)\|_2^2 \leq \sum_i |f(x_i)|^2 \leq (\frac{8}{5\sqrt{\pi}})^2 \sum_i |x_i|^2 = (\frac{8}{5\sqrt{\pi}})^2 \|W|\psi\rangle_n\|_2^2$. Since $\|W\|_2 \leq 1$, $\|W|\psi\rangle_n\|_2 \leq 1$, and thus $\mathcal{N}_2 \leq \frac{8}{5\sqrt{\pi}} \leq 0.91$. Next, we must lower-bound $\mathcal{N}_\gamma$. Thus, $\mathcal{N}_\gamma^2 = (\frac{\sqrt{\pi}}{32\nu})^2 (1 + \mathcal{N}_2^2 + 2\mathcal{N}_2 \langle \Phi_2 | \psi \rangle) \geq (\frac{\sqrt{\pi}}{32\nu})^2 (1 - \mathcal{N}_2)^2 \geq (\frac{\sqrt{\pi}}{32\nu})^2 (0.09)^2$. Consequently, using that $\nu \leq 8/5$ (since $\kappa \leq 2$) we get that $\mathcal{N}_\gamma \geq 1/400$. Additionally, it is straightforward to show that $\mathcal{N}_2/\mathcal{N}_1 \leq 0.91\kappa \leq 2$. Inserting all of these values and performing simple algebra, we find that $U_3$ is equivalently a $\left(\mathcal{N}_\gamma^{-1}, 2(a+b) + n + 5, 355(\epsilon_0 + \epsilon_1)\right)$-VE for $|\Gamma\rangle_n$.

Let $0 < \epsilon_2 \leq 1$. Then, invoking Lemma D.2.7, we get $U_f$, a $(1, 2(a+b) + n + 9, 2(355(\epsilon_0 + \epsilon_1) + \epsilon_2))$-VE for $|\Gamma\rangle_n$, with circuit complexity $O(\log(\frac{\sqrt{N}}{\epsilon_1}) \log(\frac{1}{\epsilon_2})(a + b + n + T_1 + T_2))$. If we let $\epsilon_2 = \epsilon_1$, then we can simplify this to a $(1, 2(a+b) + n + 9, 712(\epsilon_0 + \epsilon_1))$-VE with circuit complexity $O(\log(\frac{\sqrt{N}}{\epsilon_1}) \log(\frac{1}{\epsilon_1})(a+b+n+T_1+T_2))$. □

***Proof of Lemma 5.3.2.*** This result comes from repeatedly invoking Lemma 5.3.1, with the output of each application becoming the input of the next.

I will first give a bound on the total number of ancilla qubits of the block-encoding giving the final output after $k$ residual block layers. Let $a_0 = a$. $c = 2b + n + 9$. After one application of the residual block, the number of ancillas is given by $a_1 = 2a_0 + c$. Then, the general form for the number of ancillas is given by the recurrence $a_i = 2a_{i-1} + c$. We can obtain an upper-bound by instead setting

$a_i = 2(a_{i-1} + c)$. Clearly, $a_i = 2^i(a + c) = 2^i(a + 2b + n + 9)$. Thus, we have a $(1, 2^k(a + 2b + n + 9), \epsilon)$-block-encoding.

We will now determine a bound on the resulting error, $\epsilon$. Note that the $i^{th}$ residual block introduces a new error-parameter $\epsilon_i$ which controls the error in the activation function and the normalization of that block. After the first iteration, the error $\delta_1$ is given by $\delta_1 = 712(\epsilon_0 + \epsilon_1)$. After the second iteration, the error from the previous iteration becomes the new $\epsilon_0$, and so the error after the second iteration is given by $\delta_2 = 712(\delta_1 + \epsilon_2)$. We can set $\epsilon_i = \delta_{i-1}$, giving the general form of the error after the $i^{th}$ residual layer of $\delta_i = 1424\delta_{i-1} = 724 \cdot 1424^{i-1}\epsilon_1 = 1424^i\epsilon_1/2$. Noting that we want a final error of at most $\epsilon$, we must set $\delta_k \leq \epsilon$. I.e., we can set $\epsilon = 1424^k\epsilon_1/2 \implies \epsilon_1 = 2\epsilon/1424^k$. Thus, for $i > 1$, each $\epsilon_i = \delta_{i-1} = 1424^i\epsilon_1/2 = \frac{1424^i}{1424^k}\epsilon = \epsilon/1424^{k-i}$.

Define $h(\epsilon_i) := \log(\sqrt{N}/\epsilon_i)\log(1/\epsilon_i)$. Let the circuit complexity of the block-encoding after applying $i$ residual blocks be $O(R_i)$. Noting that $R_1 \in O(h(\epsilon_1)(a_0+b+n+T_1+T_2))$, $R_i$ asymptomatically dominates $T_1, T_2, a_{i-1}, n$ and $b$. Then, the circuit complexity after block $i+1$ will be $O(h(\epsilon_{i+1})(a_i+b+n+R_i+T_1)) \in O(h(\epsilon_{i+1})(a_i+R_i))$. Then, we can simplify to find that $R_k \in O((a_k + R_1)\prod_{i=1}^k h(\epsilon_i)) \in O((2^k(a + 2b + n) + T_1 + T_2)\prod_{i=1}^k h(\epsilon/1424^{k-i}))$. Noting that $\prod_{i=1}^k h(\epsilon_i) \in O((\prod_{i=1}^k \log(\sqrt{N}/\epsilon_i))^2)$, $\prod_{i=1}^k h(\epsilon/1424^{k-i})) \in O((\prod_{i=1}^k (k - i + \log(\sqrt{N}/\epsilon_i)))^2) \in O((k + \log(\sqrt{N}/\epsilon))^{2k})$. Since $k$ is an asymptotic constant, $O(k + \log(\sqrt{N}/\epsilon)) \in O(\log(\sqrt{N}/\epsilon))$, and so $\prod_{i=1}^k h(\epsilon/1424^{k-i})) \in O(\log(\sqrt{N}/\epsilon)^{2k})$. Thus, the overall circuit complexity is given by $O(\log(\sqrt{N}/\epsilon)^{2k}(a + 2b + n + T_1 + T_2))$.

□

**Lemma D.3.2** (Full-Rank Linear Pooling Output Block)**.** *Consider the architecture block shown in Fig. D.3. Let the dimension of the input vector be $N = 2^n$, and let the dimension of the output of the network block be $C = 2^c$ (i.e., the number of classes). Let the output of the network be given by the vector $|y\rangle_c$. Suppose we have $U_\psi$ an $(1, a, \epsilon_0)$-VE for the $N$-dimensional input vector $|\psi\rangle_n = \boldsymbol{x}$ with $O(T_{\epsilon_0})$ circuit complexity. Here, $T_{\epsilon_0}$ makes explicit that the complexity of the input circuit will be dependent on the desired error of the vector encoding of the layer input*

*to this architectural block. Suppose we are given access to an arbitrary matrix $W$ such that $\|W\|_2 \leq 1$ as per Theorem D.2.1. Then, if the weight on the skip-path is $\tau = 0.51$, we can draw a sample from a vector $|\tilde{\phi}\rangle_c$ such that $\left\||\tilde{\phi}\rangle_c - |y\rangle_c\right\|_2 \leq \epsilon$ with $O(\log(\frac{N}{\sqrt{C}\epsilon})(T_{\epsilon_0} + a + n^2))$ circuit complexity and with $O(a+n)$ total ancilla qubits.*

*Proof.* Let $d$ represent the number of bits in part of the QRAM encoding of $W$, as per Theorem D.2.1. Note that $d$ is assumed to be an asymptotic constant. Let $|\phi_1\rangle_n := Wg(|\psi\rangle_n)$, $\mathcal{N}_1 := \||\phi_1\rangle_n\|$ and $|\Phi_1\rangle_n := |\phi_1\rangle_n/\mathcal{N}_1$. Using Theorem D.2.1, we can get a $(\mathcal{N}_1^{-1}, 2a+d+3+n, 2\epsilon_0\mathcal{N}_1^{-1})$-VE for $|\Phi_1\rangle_n$ with $O(T_{\epsilon_0}+dn+n^2)$ circuit complexity. Here $d$ is a constant specifying the precision in the representation of the elements of the matrix stored as per Definition D.2.2.

Let $|\gamma\rangle_n := \tau|\psi\rangle_n + (1-\tau)|\Phi_1\rangle_n\mathcal{N}_1 = \tau|\psi\rangle_n + (1-\tau)Wg(|\psi\rangle_n)$, and let $\mathcal{N}_\gamma := \||\gamma\rangle_n\|_2$.

Then, Lemma 5.2.1 yields $V_2$ a $(\mathcal{N}_\gamma^{-1}, 2a+d+4+n, 3\epsilon_0\mathcal{N}_\gamma^{-1})$-VE for $|\gamma\rangle_n/\mathcal{N}_\gamma$ with $O(T_{\epsilon_0}+dn+n^2)$ circuit complexity.

We will now lower-bound $\mathcal{N}_\gamma$. The main idea is that if you are summing two vectors, one with norm 1, and the other with norm at most 1, if you put arbitrarily more mass on the constant-norm vector ($\delta$), you are guaranteed that the vectors cannot fully cancel out, and thus that some norm is preserved in the sum. Note that $\mathcal{N}_1 = \|Wg(|\psi\rangle_n)\|_2 \leq \|W\|_2 \left\|\sum_j \psi_j^2|j\rangle_n\right\|_2 \leq \left\|\sum_j \psi_j|j\rangle_n\right\|_2 = 1$. Consequently, $|\langle\psi|\Phi_1\rangle| \leq 1$, and so

$$\mathcal{N}_\gamma^2 = \|\tau|\psi\rangle_n + (1-\tau)|\Phi_1\rangle_n\mathcal{N}_1\|_2^2 = \tau^2 + (1-\tau)^2\mathcal{N}_1^2 + 2\tau(1-\tau)\mathcal{N}_1\langle\psi|\Phi_1\rangle \tag{D.78}$$

$$\geq \tau^2 + (1-\tau)^2\mathcal{N}_1^2 + 2\tau(1-\tau) = (\tau - (1-\tau)\mathcal{N}_1)^2. \tag{D.79}$$

For some parameter $\delta \in [0,1]$, assuming that $\tau = (1+\delta)/2$, we then get that $\mathcal{N}_\gamma \geq \delta$.

Then, define $\epsilon_1 \in (0,1]$. We can then invoke Lemma D.2.7 yielding $V_3$ a $(1, 2a+d+8+n, \frac{6\epsilon_0}{\delta} + 2\epsilon_1)$-VE for $|\gamma\rangle_n/\mathcal{N}_\gamma$ with $O(\frac{1}{\delta}\log(1/\epsilon_1)(T_{\epsilon_0}+a+dn+n^2))$ circuit complexity.

Define $\text{pool}_C$ as per Definition D.3.1. Noting that $\text{pool}_C(|\gamma\rangle_n/\mathcal{N}_\gamma) = |y\rangle_c$.

We can equivalently define some $\ell_2$-normalized state $|\tilde{\Gamma}\rangle_n$ such that $V_3$ is a $(1, 2a + d + 8 + n, 0)$-VE for $|\tilde{\Gamma}\rangle_n$. Then, since $\left\||\tilde{\Gamma}\rangle_n - \frac{|\gamma\rangle_n}{\mathcal{N}_\gamma}\right\|_2 \leq \frac{6\epsilon_0}{\delta} + 2\epsilon_1$, we can invoke Lemma D.3.1 which shows that $\left\|\text{pool}_C(|\tilde{\Gamma}\rangle_n) - |y\rangle_c\right\|_2 \leq \frac{2N}{\sqrt{C}}(\frac{6\epsilon_0}{\delta} + 2\epsilon_1)$.

Consequently, to get an error of at most $\epsilon$, we set $\frac{2N}{\sqrt{C}}(\frac{6\epsilon_0}{\delta} + 2\epsilon_1) = \epsilon$, by setting $\epsilon_1 = \frac{\sqrt{C}\epsilon}{8N}$ and $\epsilon_0 = \frac{\epsilon\sqrt{C}\delta}{24N}$. Then, we can simply draw a sample $\epsilon$-close to $|y\rangle_c$ in $\ell_2$-norm distance by sampling the state prepared by $V_3$ and then assigning it to the appropriate bin.

Setting $\delta = 0.02$ gives $\tau = 0.51$. Then, $V_3$ is a $(1, 2a + d + 8 + n, \epsilon)$-VE for $|\gamma\rangle_n/\mathcal{N}_\gamma$ with $O(\log(\frac{N}{\sqrt{C}\epsilon})(T_{\epsilon_0} + a + dn + n^2))$ circuit complexity. Consequently, we can draw a sample from some vector $|\tilde{\phi}\rangle_c$ such that $\left\||\tilde{\phi}\rangle_c - |y\rangle_c\right\|_2 \leq \epsilon$ with $O(\log(\frac{N}{\sqrt{C}\epsilon})(T_{\epsilon_0} + a + dn + n^2)) \in O(\log(\frac{N}{\sqrt{C}\epsilon})(T_{\epsilon_0} + a + n^2))$ circuit complexity, and with $O(a + n)$ ancilla qubits, noting that $d$ is an asymptotic constant. □

## D.4 Feasibility of QRAM Assumptions

In this section, I restate and expand upon some of the arguments made in Chapter 2 to help motivate the discussion in Appendix D.5. In Appendix D.4.1 I consider the feasibility of the QRAM assumptions in this chapter. In Appendix D.4.2 I summarize how arbitrary quantum states can be prepared by using a QRAM data structure, in service of the subsequent discussion of the different architectural regimes.

### D.4.1 Passive and Active QRAM

It is clear that, if a fault-tolerant quantum computer can be constructed, that a QRAM based on the various quantum circuit constructions (see [1, 36, 213]) can be directly implemented. Moreover, these circuit constructions have log-depth access costs. However, as laid out in Chapter 2 (and [1]), the fundamental issue regarding the practicality of QRAM comes down to the opportunity cost of the total energy required to implement a query to the QRAM. Precisely, given a QRAM with $N$ bits of memory, a QRAM is considered *passive* if and only if each query to the QRAM requires $o(N)$ *total* energy input. If the query instead requires $\Omega(N)$ energy input (even if the time complexity is $O(\text{polylog}(N))$) then the QRAM is

*active*. Importantly, this means that any QRAM implemented in the error-corrected circuit-model must be active, as each qubit requires $O(1)$ classical resources to run the error-correction, resulting in an $\Omega(N)$ total energy cost per QRAM query. Even if error-correction is not used, if enacting the gates in the system requires constant energy input (e.g., by enacting the gates as laser pulses) then the QRAM will be active. If the QRAM is active, then in Chapter 2 I show that a wide-range of quantum linear algebra applications lose quantum speedup. Moreover, there are additional challenges such as how a noisy (non-error corrected) quantum memory could be interfaced with an error-corrected quantum processor.

However, as noted in [1] there is some hope in practice, and I will now outline some of the reasons we provide in [1]. As an example, consider classical Dynamic Random Access Memory (DRAM). DRAM requires a constant power draw for each bit in memory, and thus an $N$-bit memory requires $\Omega(N)$ energy input. This makes DRAM active. Nevertheless, because the energy expenditure of DRAM is often dwarfed by the energy expenditure of the CPU accessing it, it is usually treated as being a passive component in classical algorithm design. For instance, [214] demonstrates that for mobile phones, "RAM power is insignificant in real workloads", and [215] draws a similar conclusion for laptops. At larger server-scales, the asymptotics of active memory become more noticeable, but memory still usually draws less power than the controlling CPU [216, 217]. Analogously, consider a regime where a QRAM is active, but its constant energy costs are extremely small relative to the energy costs of the error-corrected quantum computer it is being interfaced with. Given the *substantial* expected overheads of quantum error-correction [58], the ratio of energy consumption for an error-corrected QPU to an active QRAM could be even more favourable than in the classical setting. Then, if there is some way to interface this noisy device with the error-corrected QPU, for moderate scales (e.g., terabytes of memory), it is conceivable that the QRAM could be practically treated as passive. I call this a "practically passive QRAM". Nevertheless, even though practically passive QRAMs are asymptotically active, they are unlikely to allow full error-correction without losing their constant advantages (unless, for some reason, the structure of QRAM

allows for extremely efficient custom-made error-correcting codes). Consequently, it is important that the QRAM implementation is resilient to errors. Indeed QRAMs based on the bucket-brigade architecture [218], are intrinsically *exponentially* (in terms of the number of memory registers) robust to errors [39, 213, 219].

In this chapter, for simplicity, when making a QRAM assumption the QRAM is treated as passive. I stress that substantially more work is needed to fully understand the feasibility of QRAM, but that it is plausible that the QRAM assumptions made in this paper could be physically realized in practice. In particular, assuming that truly passive QRAM is impossible, I outline the following questions (building on Chapter 2 and [1]) which could result in the results in this chapter being practically useful. How can a noisy QRAM system be interfaced with an error-corrected quantum computer? If such an interface is possible, how do errors in the QRAM propagate through the error-correction in the QPU? Recent promising work [174] provides answers to these two preceding questions, and offers a path forward for research aiming to construct practically passive and useful QRAM. Additional questions which need to be investigated to help realize a practically passive QRAM include some of the following. What is the ratio in energy consumption for plausible practically passive QRAM systems to the energy consumption of the controlling fault-tolerant QPUs for different error-correcting codes? Given potential active (practically passive) QRAM architectures, what is the total expected energy consumption for different sized memories?

### D.4.2 Input Preparation via QRAM

The data structure due to [59] can allow for an arbitrary quantum state to be prepared, so long as the state amplitudes are made available through a specific QRAM data structure.

**Lemma D.4.1** (Input Data QRAM Data Structure [59])**.** *Let $N = 2^n$. Given a vector $\boldsymbol{x} \in \mathbb{R}^N$, we can define a data structure utilizing a QRAM with $\tilde{O}(N)$ total*

*qubits* [4] *storing* $\boldsymbol{x}$*. Then: (1) the cost to update (insert, delete, or modify) an entry* $x_j$ *is* $O(n^2)$*, (2) using the QRAM data structure, the state* $|x\rangle = \boldsymbol{x}/\left\|\boldsymbol{x}\right\|_2$ *can be prepared by a circuit with depth* $O(n^2)$*, acting on* $O(n)$ *qubits.*

This is just a special case of the more general result in [59] giving a similar data structure for arbitrary matrices (which we presented as QRAM for quantum data in Appendix D.1). Intuitively, the state can be prepared by following Grover-Rudolph [91], using the QRAM data structure containing the tree of binary partial norms of the vector to compute the controlled rotation angles for each additional qubit.

## D.5 Architectures in Different Regimes

As summarized in the main text, the results presented thus far can be used to construct a range of architectures in a number of different settings. In particular, I consider three regimes characterized by the QRAM assumptions they make. In the first regime, I assume that both the input to the network and the weights in the network are made available via QRAM. In the second regime, I assume that the network may use QRAM (since its QRAM data structure may be pre-computed prior to inference-time), but that the input to the network is received classically and entirely on-the-fly, and thus that the input cannot be provided with QRAM (so a cost linear in the dimension of the input must be paid to load it into the quantum computer). In the third regime, I assume no QRAM. I will now expand on the arguments presented in the main text in greater detail.

### D.5.1 Regime 1: Input and Network Use QRAM

Here I expand on the argument presented in Section 5.4.1.

[4] Neglecting the finite precision error due to storing vector elements (and their partial squared sums) in binary representations

**Online Input Construction** Noting that as per Appendix D.4.2 QRAM data structures can be efficiently updated, assuming that the underlying hardware implementation also support efficient updates, there are a number of settings where it might be realistic for the input vector to be provided via QRAM.

For example, in any setting where inference needs to be repeatedly performed on a slowly-changing input (e.g., in an interactive chat with an autoregressive LLM, where each output token becomes part of the new input), or where the input is the result of some other quantum algorithm. For example, in the context of autoregressive interactive LLM (where the output would be a probability distribution over tokens instead of classes), the initial vector $\boldsymbol{x}$ might be an encoding of the hidden prompt to the network (and so the associated data structure can be precomputed). As a user queries the LLM, a small number of tokens are added to $\boldsymbol{x}$, and these updates can be efficiently performed to the data structure. Then, the network is run, and the new output token is added to $\boldsymbol{x}$, again efficiently. This process can then continue to repeat, and so the cost of loading the data is either entirely precomputed, or amortized on-the-fly. I envision similar applications in the classification of video, where a very large, but slowly-changing, video needs to be analysed one frame at a time. Here, a cost would need to be paid proportional to the number of changing pixels between each frame, and so the input data structure could be efficiently updated. Additional settings where it might be reasonable for the input to be provided efficiently could be if the input corresponds to some combination of continuous function (via [2]), or if it was prepared as the output of some other quantum algorithm.

**Receptive Field** To understand the importance of the final linear layer in the architecture for this regime, I will first summarize the receptive field problem of multi-layer convolutional architectures.

For simplicity, consider a 2D convolution with one input channel and one output channel, and consider a sequence of $k$ such convolutional layers. Let the kernel be $D \times D$. Since a convolutional layer can map the information in location $i, j$

to, at the furthest, the location $i + D, j + D$, after $k$ layers the information in any given entry will come from local information in the input at most $\approx kD$ pixels away. Consequently, the final layer which is input to the output linear-layer-residual block will contain features with $kD$ local information, which the linear layer then combines in a global fashion. I conjecture that having a full-rank layer at this stage is more effective for merging the local information than a similar dimension, but low-rank, linear layer. Since the cost of the quantum algorithm grows exponentially with depth, without the final linear layer, with such an architecture no learning could occur which requires global information from the input image.

Moreover, there are other approaches which could be taken to make the local information globally accessible to the earlier convolutional layers, potentially improving the power of such quantum-amenable architectures in practice. For instance, after a set number of convolutional layers, a linear layer could be added to make local information global (however, this damages the nice algebraic properties of convolutional layers). Alternatively, a sequence of convolutions can be implemented in each residual block (without activation functions between them) as this would not increase the complexity exponentially, potentially allowing for many more convolutions in sequence. Most appealingly, a solution can be found in the popular classical architecture of bilinear neural networks [175] (which forms the basis of the architecture presented for Regime 2). Here, paths of convolutional-based residual blocks are passed into a Kronecker product, which is followed by more layers. Via Lemma 5.2.3, we can efficiently do this in a quantum computer. Since the Kronecker product makes all local information globally available, it immediately solves the receptive field problem. However, while a Kronecker product makes local information globally accessible, it loses positional information. This can be resolved by enacting a positional encoding along one of the paths of the network prior to the product, e.g., as is done when Tokenizing the inputs to transformer architectures [126].

**Dequantization** A number of quantum algorithms which were believed to have exponential speed-ups over their classical counterparts lost their exponential speedup after new classical randomized algorithms were developed which mirrored the quantum input assumptions. For example, see the works of [59] and [52]. Indeed, it seems likely that, as was the case with the quantum CNN implementation in [160], that the convolutional residual blocks in the architectures presented in this chapter could be dequantized (even though they make no QRAM assumptions). However, the new techniques in this chapter enable the final linear-residual block to contain an arbitrary full-rank and dense matrix. Since known dequantization techniques require the matrix to be either low-rank [52, 57] or sparse (with certain strong caveats) [220], existing techniques appear insufficient to dequantize the full architectures in Regimes 1 and 2. Moreover, as previously discussed, removing the final linear layer introduces receptive field problems, highlighting that it is not a purely artificial addition to the network. Nevertheless, it would be interesting to exploring dequantizing the architecture without the final linear layer (or perhaps replacing it with a low-rank one), and this could result in some interesting techniques to classical accelerate inference for certain architectures.

### D.5.2 Regime 2: Network Stored in QRAM, Input Loaded Without QRAM

See the discussion in Section 5.4.1.

### D.5.3 Regime 3: No QRAM

To reiterate, in this regime, both the matrix weights and the network input are not given by QRAM. I will now prove the complexity of the Regime 3 architecture shown in Fig. 5.1 (c), as discussed in Section 5.4.1. I note that there are many simple modifications which could be made to this architecture, for example by having a final low-rank linear layer with $O(N)$ parameters. Adopt the notation used in Theorem 5.4.1. Let the input be a $4 \times M \times M$ tensor, and define $N = M^2$, $n = \log_2(N)$, $m = \log_2(M)$. Thus, the vectorized input is of dimension $O(N)$. Let

$d$ be the number of paths into the input tensor (i.e., the latent dimension will be $O(N^d)$), as per Fig. 5.1 (c). $T_X$ is the access cost of the input; in the QRAM-free regime we assume a worst-case of $T_X \in O(N)$. Let $C$ be the number of output classes (or set of possible output tokens).

Assume $d = 2$. Let $\delta > 0$ be an error parameter used only in the proof. Directly from the proof of Theorem 5.4.1, we have $U_{\text{conv}}$, a $(1, 2^k(63+n), \delta)$-VE (vector encoding) for the $\ell_2$-normalized output of the $k$ convolutional/residual block layers. $U_{\text{conv}}$ has $O(\log(N/\delta)^{2k}(n^2 + T_X))$ circuit depth. Note that in that proof, $N$ corresponds to the vectorized dimension of the latent space (i.e., if there is 1 input and output channel, $N$ corresponds to the dimension of the vector acted upon by the matrix-form of the 2D convolution), and thus corresponds to $N^d$ here.

Let $|\phi\rangle$ represent the exact vector output after the sequence of $k$ convolutional layers. This VE corresponds to a state $|\tilde{\phi}\rangle$ such that $\||\phi\rangle - |\tilde{\phi}\rangle\|_2 \leq \delta$. Consequently, by Lemma D.3.1, sampling this VE (and applying the binning-protocol) yields a sample from a vector $\text{pool}_C(|\tilde{\phi}\rangle)$ such that $\|\text{pool}_C(|\phi\rangle) - \text{pool}_C(|\tilde{\phi}\rangle)\|_2 \leq \frac{2N^2\delta}{\sqrt{C}}$. Noting that the correct output of the network is given by $\boldsymbol{y} = \text{pool}_C(|\phi\rangle)$, we can get an overall error of $\epsilon$, such that the vector we sample from satisfies $\|\boldsymbol{y} - \text{pool}_C(|\tilde{\phi}\rangle)\|_2 \leq \epsilon$ by setting $\epsilon = \frac{2N^2\delta}{\sqrt{C}} \implies \delta = \frac{\epsilon\sqrt{C}}{2N^2}$. By plugging this into the circuit complexity of $U_{\text{conv}}$, and noting that here I assume we pay the full input dimension cost (since there is no QRAM), $T_X \in O(N)$, and so this simplifies to $O(N \log(N^3/\epsilon\sqrt{C})^{2k}) \in \tilde{O}(N \log(1/\epsilon)^{2k})$ total circuit cost. As stated in the main text, since the dimension of the vector acted on by the 2D convolution is $O(N^2)$ (when d=2), the classical cost to compute this is $\Omega(N^2)$: showing **a quadratic speedup over an exact classical implementation**. The speedup can be made asymptotically larger by increasing $d$.

**Possible Limitations** Here I will outline some of the possible limitations of the architecture shown in Fig. 5.1 (c). Since there is no final linear layer (in the architecture as directly presented), the receptive field problems outlined in Section 5.4.1 may appear to apply. However, by virtue of taking the tensor product of the input

paths, local information becomes immediately globally accessible circumventing this limitation. Moreover, another way that local information could made global is from the processing that occurs along each path prior to the tensor product, since there is no limit on the classical processing that can occur (so long as the total compute is linear in the dimension of the input).

Moreover, our argument against dequantization in the first regime (see Section 5.4.1) relies on the final dense and full-rank linear layer. However, since this layer is not feasible without QRAM, this argument does not apply here. However, as we are only suggesting a polynomial speedup in Regime 3, we do not expect a dequantized algorithm to completely close the performance gap past quadratic, as we benefit from amplitude amplification. However, exploring dequantized algorithms based on the ideas in this paper appears to be interesting subsequent work.

Finally, if the network only contains the convolutional layers, it will likely be very under-parameterized making training challenging (see e.g., [221] for a discussion on overparameterized neural networks). However, where the dimension of the vectorized input is $N$, it would be easy to add $O(N)$ parameters, either in the classical paths prior to the tensor product, or as a final low-rank residual output block (prior to the $\ell_2$-norm-pooling), so long as the number of parameters in that block are $O(N)$.

**Alternative: Parameterized Quantum Circuits as Network Layers** Alternatively, one could use parameterized quantum circuits as network layers [140–142], as the number of parameters in such circuits are usually polylogarithmic in the dimension of the operator. However, such circuits are often hard to train even on classical machines, due to under-parameterization, the barren plateau problem [149, 150], and the exponential amount of bad local minima in the optimization landscape [148]. However, given good enough initializations and warm start assumptions [222], it may still possible to train such architectures, leading to potential speed-ups in inference.

**Other Possible Sources of Speedup** In some cases, where the input can be efficiently prepared without paying a dimension-dependent cost (e.g., the input comes from quantum states which are easy to prepare, either via some other quantum algorithm, or via techniques like those outlined in Chapter 3) it may be possible to obtain better than quadratic speedups. However, I leave this as a topic for future investigation.

## D.6 Technical Results

I now report a result on the efficient polynomial approximation to the error function due to [192], which builds on the results of [223]. This result is an improvement over the approximation obtained by an integration of the series expansion for the Gaussian distribution.

**Lemma D.6.1** (Polynomial Approximation to Error Function due to Corollary 4 of [192]). *Let $m \geq 1/2$, $1 \geq \epsilon > 0$. There exists a degree $k \in O(m \log(1/\epsilon))$ polynomial $P_{k,m}(x)$ such that*

$$P_{k,m}(x) := \frac{2me^{-m^2/2}}{\sqrt{\pi}} \left( I_0(m^2/2)x + \sum_{j=1}^{(k-1)/2} I_j(m^2/2)(-1)^j \left( \frac{T_{2j+1}(x)}{2j+1} - \frac{T_{2j-1}(x)}{2j-1} \right) \right) \tag{D.80}$$

*and* $\max_{x\in[-1,1]} |\operatorname{erf}(mx) - P_{k,m}(x)| \leq \epsilon$. *Let* $1 \geq c > 0$. *Alternatively, if* $k \in O(m \log(mc/\epsilon))$, *then* $\max_{x\in[-c,c]} |\operatorname{erf}(mx) - P_{k,m}(x)| \leq \epsilon$. *Additionally, for all* $k$, $\max_{x\in[-1,1]} |P_{k,m}(x)/x| \leq \frac{4m}{\sqrt{\pi}}$, *and* $P_{k,m}(0) = 0$. *Finally,* $\min_{x\in[-1,1]} |\operatorname{erf}(mx)/x| \geq 1/2$, *and* $\operatorname{erf}(mx)$ *has Lipschitz constant* $L = \frac{2m}{\sqrt{\pi}}$,

*Proof.* For the case where $\max_{x\in[-1,1]} |\operatorname{erf}(mx) - P_{k,m}(x)| \leq \epsilon$, the result on the polynomial approximation is directly taken from [192]. We will now prove the bound when the function is constrained to the interval $[-c, c]$. Let $\epsilon_1 := \max_{x\in[-c,c]} |\operatorname{erf}(mx) - P_{k,m}(x)|$. From Equation (71) of Corollary 4 of [192], for a degree $k$ polynomial approximation, we have the following error-bound,

$$\epsilon_1 \leq \frac{2me^{-m^2/2}}{\sqrt{\pi}} \left| \sum_{j=(k+1)/2}^{\infty} I_j(m^2/2)(-1)^j \left( \frac{T_{2j+1}(x)}{2j+1} - \frac{T_{2j-1}(x)}{2j-1} \right) \right|. \tag{D.81}$$

Using the identity $\left(\frac{T_{2j+1}(x)}{2j+1} - \frac{T_{2j-1}(x)}{2j-1}\right) = 2\int_0^x T_{2j}(t)dt$, and using the fact that all Chebyshev polynomials of the form $T_{2j}$ are even, we can get the bound that $2\left|\frac{T_{2j+1}(x)}{2j+1} - \frac{T_{2j-1}(x)}{2j-1}\right| \leq 2\int_0^{|x|} |T_{2j}(t)|dt \leq 2|x| \leq 2\max_{x\in[-c,c]}|x| = 2c$, since $\max_{x\in[-1,1]}|T_{2j}(x)| \leq 1$.

Then, applying the triangle inequality, Eq. (D.81) becomes,

$$\epsilon_1 \leq \frac{4cme^{-m^2/2}}{\sqrt{\pi}} \sum_{j=(k+1)/2}^{\infty} \left|I_j(m^2/2)\right|. \tag{D.82}$$

Define $\epsilon_{gauss,\gamma,k}$ as per Corollary 3 of [192]. Define some $\epsilon' > 0$.

Note that $\epsilon_{gauss,\gamma,k} = 2e^{-\gamma^2/2}\sum_{j=\frac{n}{2}+1}^{\infty}|I_j(\gamma^2/2)|$, and that $\epsilon_{gauss,\gamma,k} \leq \epsilon'$ if $k \in O(\sqrt{(\gamma^2 + \log(1/\epsilon'))\log(1/\epsilon')})$. Thus, $\epsilon_1 \leq \frac{2cm\epsilon'}{\sqrt{\pi}}$. To get an overall error-bound of at most $\epsilon$, we can set $\frac{2cm\epsilon'}{\sqrt{\pi}} = \epsilon$, and so $\epsilon' = \frac{\sqrt{\pi}\epsilon}{2cm}$. Thus, if we set $k \in O(m\log(\frac{cm}{\epsilon}))$, we are guaranteed that $\max_{x\in[-c,c]}|P_{k,m}(x) - \operatorname{erf}(mx)| \leq \epsilon$.

Next, $\frac{d}{dx}\operatorname{erf}(mx) = \frac{2m}{\sqrt{\pi}}e^{-(mx)^2}$, and consequently the maximum value of the derivative of the function is when $x = 0$, i.e., $\max_{x\in[-1,1]}|\frac{d}{dx}\operatorname{erf}(mx)| = \frac{2m}{\sqrt{\pi}}$.

We will now prove that $|P_{k,m}(x)/x| \leq \frac{4m}{\sqrt{\pi}}$ and $\min_{x\in[-1,1]}|\operatorname{erf}(mx)/x| \geq 1/2$.

Noting that $P_{k,m}(0) = 0$, (since for $x = 0$, $T_{2j}(x) = \cos((2j+1)\arccos(0)) = \cos((2j+1)\pi/2) = 0$), by Lipschitz continuity we have that $|P_{k,m}(x)/x| \leq |\frac{d}{dx}P_{k,m}(x)|$. Noting that $\frac{d}{dx}\frac{1}{2}(\frac{T_{2j+1}(x)}{2j+1} - \frac{T_{2j-1}(x)}{2j-1}) = T_{2j}(x)$,

$$\max_{x\in[-1,1]}|P_{k,m}(x)/x| \leq \max_{x\in[-1,1]}\left|\frac{d}{dx}P_{k,m}(x)\right| \tag{D.83}$$

$$= \max_{x\in[-1,1]}\left|\frac{2me^{-m^2/2}}{\sqrt{\pi}}\left(I_0(m^2/2) + 2\sum_{j=1}^{(k-1)/2} I_j(m^2/2)(-1)^j T_{2j}(x)\right)\right|. \tag{D.84}$$

A common identity for modified Bessel functions of the first kind states for $t \neq 0$, $e^{\frac{1}{2}y(t+t^{-1})} = \sum_{j=-\infty}^{\infty} t^j I_j(y)$. Setting $t = 1$, we find $e^y = \sum_{j=-\infty}^{\infty} I_j(y)$. Moreover, since $I_j(y) \geq 0$ for all $y > 0$, $\sum_{j=1}^{(k-1)/2} I_j(m^2/2) \leq e^{m^2/2}$. Thus, using that $\max_{x\in[-1,1]}|T_{2j}(x)| \leq 1$,

$$\max_{x\in[-1,1]}|P_{k,m}(x)/x| \leq \frac{4me^{-m^2/2}}{\sqrt{\pi}}\sum_{j=1}^{(k-1)/2} I_j(m^2/2) \leq \frac{4m}{\sqrt{\pi}}. \tag{D.85}$$

Thus, it is clear that this upper-bound is independent of the degree of the polynomial approximation, and thus applies to the whole interval $x \in [-1, 1]$ and not just $x \in [-c, c]$.

Finally, we must show that $\min_{x\in[-1,1]} |\operatorname{erf}(mx)/x| \geq 1/2$. First, note that $|\operatorname{erf}(mx)/x|$ is symmetrical, so we can simply consider the interval $x \in [0, 1]$. Moreover, it is monotonically decreasing, so we can take the endpoint $\min_{x\in[-1,1]} |\operatorname{erf}(mx)/x| = \operatorname{erf}(m)$. Since $m \geq 1/2$, $\operatorname{erf}(m) \geq \operatorname{erf}(1/2) \approx 0.52 > 1/2$. □

# References


[1] Samuel Jaques and Arthur G Rattew. "Qram: A survey and critique". In: *arXiv preprint arXiv:2305.10310* (2023).

[2] Arthur G Rattew and Bálint Koczor. "Preparing arbitrary continuous functions in quantum registers with logarithmic complexity". In: *arXiv preprint arXiv:2205.00519* (2022).

[3] Arthur G Rattew and Patrick Rebentrost. "Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications". In: *arXiv preprint arXiv:2309.09839* (2023).

[4] Arthur G. Rattew et al. *Accelerating Inference for Multilayer Neural Networks with Quantum Computers*. 2025. arXiv: 2510.07195 [quant-ph]. URL: https://arxiv.org/abs/2510.07195.

[5] Frank Arute et al. "Quantum Supremacy Using a Programmable Superconducting Processor". In: *Nature* 574.7779 (Oct. 2019), pp. 505–510.

[6] Han-Sen Zhong et al. "Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light". In: *Physical Review Letters* 127.18 (Oct. 2021), p. 180502.

[7] Yulin Wu et al. "Strong Quantum Computational Advantage Using a Superconducting Quantum Processor". In: *Physical Review Letters* 127.18 (Oct. 2021), p. 180501.

[8] Sepehr Ebadi et al. "Quantum Phases of Matter on a 256-Atom Programmable Quantum Simulator". In: *Nature* 595.7866 (July 2021), pp. 227–232.

[9] Ming Gong et al. "Quantum Walks on a Programmable Two-Dimensional 62-Qubit Superconducting Processor". In: *Science* 372.6545 (May 2021), pp. 948–952.

[10] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6–7 (June 1982), pp. 467–488. URL: https://doi.org/10.1007/BF02650179.

[11] Richard P. Feynman. "Quantum mechanical computers". In: *Foundations of Physics* 16.6 (June 1986), pp. 507–531. URL: https://doi.org/10.1007/BF01886518.

[12] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[13] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[14] Seth Lloyd. "Universal quantum simulators". In: *Science* 273.5278 (1996), pp. 1073–1078.

[15] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* 103 (15 2009), p. 150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. URL: https://doi.org/10.1038/nature14539.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[18] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[20] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[21] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (July 2021), pp. 583–589. URL: https://doi.org/10.1038/s41586-021-03819-2.

[22] Gordon E. Moore. "Cramming more components onto integrated circuits". In: *Electronics* 38.8 (Apr. 1965), pp. 114–117. URL: https://ieeexplore.ieee.org/document/4785860.

[23] M Mitchell Waldrop. "The chips are down for Moore's law". In: *Nature News* 530.7589 (2016), p. 144.

[24] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for modern deep learning research". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. URL: https://doi.org/10.1017/CBO9780511976667.

[26] 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*. 2019, pp. 193–204.

[27] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum Random Access Memory". In: *Phys. Rev. Lett.* 100 (16 2008), p. 160501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.100.160501.

[28] G. Kuperberg. "Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem". In: *TQC 2013*. LIPIcs 22. 2013, pp. 20–34. URL: https://doi.org/10.4230/LIPIcs.TQC.2013.20.

[29] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". In: *Physical Review A* 54.1 (1996), p. 147.

[30] Thomas G Draper. "Addition on a quantum computer". In: *arXiv preprint quant-ph/0008033* (2000).

[31] Thomas G Draper et al. "A logarithmic-depth quantum carry-lookahead adder". In: *arXiv preprint quant-ph/0406142* (2004).

[32] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. "Quantum addition circuits and unbounded fan-out". In: *arXiv preprint arXiv:0910.2530* (2009).

[33] Edgard Muñoz-Coreas and Himanshu Thapliyal. "T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm". In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 14.3 (2018), pp. 1–15.

[34] Mihir K Bhaskar et al. "Quantum algorithms and circuits for scientific computing". In: *arXiv preprint arXiv:1511.08253* (2015).

[35] Claude. E. Shannon. "The synthesis of two-terminal switching circuits". In: *The Bell System Technical Journal* 28.1 (1949), pp. 59–98.

[36] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Architectures for a quantum random access memory". In: *Phys. Rev. A* 78 (5 2008), p. 052310. URL: https://link.aps.org/doi/10.1103/PhysRevA.78.052310.

[37] Arnau Sala Cadellans. "A transmon based quantum switch for a quantum random access memory". MA thesis. Leiden, Netherlands: Leiden University Faculty of Science, 2015.

[38] Kevin C. Chen et al. "Heralded Quantum Random Access Memory in a Scalable Photonic Integrated Circuit Platform". In: *Conference on Lasers and Electro-Optics*. Optica Publishing Group, 2021. URL: https://doi.org/10.1364/cleo_qels.2021.fth1p.7.

[39] Connor T. Hann et al. "Resilience of Quantum Random Access Memory to Generic Noise". In: *PRX Quantum* 2.2 (Apr. 2021). URL: https://doi.org/10.1103/prxquantum.2.020311.

[40] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202.

[41] Marco Pistoia et al. "Quantum Machine Learning for Finance ICCAD Special Session Paper". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE. 2021, pp. 1–9.

[42] Junyu Liu et al. "Towards provably efficient quantum algorithms for large-scale machine-learning models". In: *Nature Communications* 15.1 (2024), p. 434.

[43] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: *Physical review letters* 113.13 (2014), p. 130503.

[44] Iordanis Kerenidis and Anupam Prakash. "Quantum recommendation systems". In: *arXiv preprint arXiv:1603.08675* (2016).

[45] Guang Hao Low and Isaac L Chuang. "Hamiltonian simulation by qubitization". In: *Quantum* 3 (2019), p. 163.

[46] John M Martyn et al. “Grand unification of quantum algorithms”. In: *PRX Quantum* 2.4 (2021), p. 040203.

[47] Lin Lin. “Lecture notes on quantum algorithms for scientific computation”. In: *arXiv preprint arXiv:2201.08309* (2022).

[48] Scott Aaronson. “Read the fine print”. In: *Nature Physics* 11.4 (Apr. 2015), pp. 291–293. URL: https://doi.org/10.1038/nphys3272.

[49] Damien Steiger. *Racing in parallel: Quantum versus Classical*. en. 2016. URL: https://pirsa.org/16080019.

[50] Carlo Ciliberto et al. “Quantum machine learning: a classical perspective”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (Jan. 2018), p. 20170551. URL: https://doi.org/10.1098/rspa.2017.0551.

[51] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. “Quantum state preparation with optimal circuit depth: Implementations and applications”. In: *Physical Review Letters* 129.23 (2022), p. 230504.

[52] Ewin Tang. “A quantum-inspired classical algorithm for recommendation systems”. In: *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*. 2019, pp. 217–228.

[53] Ravindran Kannan and Santosh Vempala. “Randomized algorithms in numerical linear algebra”. In: *Acta Numerica* 26 (2017), pp. 95–135.

[54] Nadiia Chepurko et al. “Quantum-inspired algorithms from randomized numerical linear algebra”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 3879–3900.

[55] András Gilyén, Seth Lloyd, and Ewin Tang. “Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension”. In: *arXiv preprint arXiv:1811.04909* (2018).

[56] Ainesh Bakshi and Ewin Tang. “An Improved Classical Singular Value Transformation for Quantum Machine Learning”. In: *arXiv preprint arXiv:2303.01492* (2023).

[57] Nai-Hui Chia et al. “Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning”. In: *Journal of the ACM* 69.5 (2022), pp. 1–72.

[58] Ryan Babbush et al. “Focus beyond quadratic speedups for error-corrected quantum advantage”. In: *PRX Quantum* 2.1 (2021), p. 010103.

[59] Iordanis Kerenidis and Anupam Prakash. “Quantum Recommendation Systems”. In: *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Vol. 67. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 49:1–49:21. URL: https://doi.org/10.4230/LIPIcs.ITCS.2017.49.

[60] Roger Penrose et al. “Applications of negative dimensional tensors”. In: *Combinatorial mathematics and its applications* 1.221-244 (1971), p. 3.

[61] Jacob Biamonte and Ville Bergholm. “Tensor networks in a nutshell”. In: *arXiv preprint arXiv:1708.00006* (2017).

[62] Adam Holmes and Anne Y Matsuura. "Efficient quantum circuits for accurate state preparation of smooth, differentiable functions". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE. 2020, pp. 169–179.

[63] Dhiraj Kalamkar et al. "A study of BFLOAT16 for deep learning training". In: *arXiv preprint arXiv:1905.12322* (2019).

[64] Tim Dettmers et al. "Llm. int8 (): 8-bit matrix multiplication for transformers at scale". In: *arXiv preprint arXiv:2208.07339* (2022).

[65] Elias Frantar et al. "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers". In: *arXiv preprint arXiv:2210.17323* (2022).

[66] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[67] Laurent El Ghaoui. "Inversion error, condition number, and approximate inverses of uncertain matrices". In: *Linear algebra and its applications* 343 (2002), pp. 171–193.

[68] Shouvanik Chakrabarti et al. "A threshold for quantum advantage in derivative pricing". In: *Quantum* 5 (2021), p. 463.

[69] Nikitas Stamatopoulos et al. "Option pricing using quantum computers". In: *Quantum* 4 (2020), p. 291.

[70] Dylan Herman et al. "A Survey of Quantum Computing for Finance". In: *arXiv preprint arXiv:2201.02773* (2022).

[71] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (2014), pp. 631–633.

[72] Kosuke Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (2018), p. 032309.

[73] Hans Hon Sang Chan et al. "Grid-based methods for chemistry modelling on a quantum computer". In: *arXiv preprint arXiv:2202.05864* (2022).

[74] Nicholas J Ward, Ivan Kassal, and Alán Aspuru-Guzik. "Preparation of many-body states for quantum simulation". In: *The Journal of chemical physics* 130.19 (2009), p. 194105.

[75] Stefan Woerner and Daniel J Egger. "Quantum risk analysis". In: *npj Quantum Information* 5.1 (2019), pp. 1–8.

[76] Ashley Montanaro. "Quantum speedup of Monte Carlo methods". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (2015), p. 20150301.

[77] Martin Plesch and Časlav Brukner. "Quantum-state preparation with universal gate decompositions". In: *Physical Review A* 83.3 (2011), p. 032302.

[78] Xiao-Ming Zhang, Man-Hong Yung, and Xiao Yuan. "Low-depth quantum state preparation". In: *Physical Review Research* 3.4 (2021), p. 043200.

[79] Xiaoming Sun et al. "Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis". In: *arXiv preprint arXiv:2108.06150* (2021).

[80] Gregory Rosenthal. “Query and Depth Upper Bounds for Quantum Unitaries via Grover Search”. In: *arXiv preprint arXiv:2111.07992* (2021).

[81] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987.

[82] Patrick Rebentrost et al. “Quantum singular-value decomposition of nonsparse low-rank matrices”. In: *Physical review A* 97.1 (2018), p. 012327.

[83] Andrew Macgregor Childs. “Quantum information processing in continuous time”. PhD thesis. Massachusetts Institute of Technology, 2004.

[84] Dorit Aharonov and Amnon Ta-Shma. “Adiabatic quantum state generation and statistical zero knowledge”. In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, pp. 20–29.

[85] Andrew M Childs et al. “Exponential algorithmic speedup by a quantum walk”. In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, pp. 59–68.

[86] Dominic W Berry et al. “Efficient quantum algorithms for simulating sparse Hamiltonians”. In: *Communications in Mathematical Physics* 270.2 (2007), pp. 359–371.

[87] Lov K Grover. “Synthesis of quantum superpositions by quantum computation”. In: *Physical review letters* 85.6 (2000), p. 1334.

[88] Yuval R Sanders et al. “Black-box quantum state preparation without arithmetic”. In: *Physical review letters* 122.2 (2019), p. 020502.

[89] Johannes Bausch. “Fast black-box quantum state preparation”. In: *arXiv preprint arXiv:2009.10709* (2020).

[90] Shengbin Wang et al. “Fast black-box quantum state preparation based on linear combination of unitaries”. In: *Quantum Information Processing* 20.8 (2021), pp. 1–14.

[91] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. In: *arXiv preprint quant-ph/0208112* (2002).

[92] Gabriel Marin-Sanchez, Javier Gonzalez-Conde, and Mikel Sanz. “Quantum algorithms for approximate function loading”. In: *arXiv preprint arXiv:2111.07933* (2021).

[93] Almudena Carrera Vazquez and Stefan Woerner. “Efficient state preparation for quantum amplitude estimation”. In: *Physical Review Applied* 15.3 (2021), p. 034027.

[94] Steven Herbert. “No quantum speedup with Grover-Rudolph state preparation for quantum Monte Carlo integration”. In: *Physical Review E* 103.6 (2021), p. 063302.

[95] Wim Van Dam, Michele Mosca, and Umesh Vazirani. “How powerful is adiabatic quantum computation?” In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE. 2001, pp. 279–287.

[96] Edward Farhi et al. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000).

[97] Thomas Häner, Martin Roetteler, and Krysta M Svore. "Optimizing quantum circuits for arithmetic". In: *arXiv preprint arXiv:1805.12445* (2018).

[98] Sam McArdle et al. "Quantum computational chemistry". In: *Reviews of Modern Physics* 92.1 (2020), p. 015003.

[99] Peter Borwein. "An efficient algorithm for the Riemann zeta function". In: *Canadian Mathematical Society Conference Proceedings*. Vol. 27. 2000, pp. 29–34.

[100] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[101] Dominic W Berry, Andrew M Childs, and Robin Kothari. "Hamiltonian simulation with nearly optimal dependence on all parameters". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 792–809.

[102] Edwin L Crow and Kunio Shimizu. *Lognormal distributions*. Marcel Dekker New York, 1987.

[103] Sheehan Olver and Alex Townsend. "Fast inverse transform sampling in one and two dimensions". In: *arXiv preprint arXiv:1307.1223* (2013).

[104] John Preskill. "Quantum computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[105] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum generative adversarial networks for learning and loading random distributions". In: *npj Quantum Information* 5.1 (2019), p. 103.

[106] Elton Yechao Zhu et al. "Generative quantum learning of joint probability distribution functions". In: *arXiv preprint arXiv:2109.06315* (2021).

[107] Gabriele Agliardi and Enrico Prati. "Optimal Tuning of Quantum Generative Adversarial Networks for Multivariate Distribution Loading". In: *Quantum Reports* 4.1 (2022), pp. 75–105.

[108] Arthur G Rattew et al. "The efficient preparation of normal distributions in quantum registers". In: *Quantum* 5 (2021), p. 609.

[109] Bálint Koczor. "Exponential error suppression for near-term quantum devices". In: *Physical Review X* 11.3 (2021), p. 031057.

[110] Juan M Pino et al. "Demonstration of the QCCD trapped-ion quantum computer architecture". In: *arXiv preprint arXiv:2003.01293* (2020).

[111] Elijah Pelofske, Andreas Bärtschi, and Stephan Eidenbenz. "Quantum Volume in Practice: What Users Can Expect from NISQ Devices". In: *arXiv preprint arXiv:2203.03816* (2022).

[112] William F Ames. *Nonlinear partial differential equations in engineering*. Academic press, 1965.

[113] Bruce A Finlayson. *Nonlinear analysis in chemical engineering*. Bruce Alan Finlayson, 2003.

[114] Samuel Burer and Adam N Letchford. "Non-convex mixed-integer nonlinear programming: A survey". In: *Surveys in Operations Research and Management Science* 17.2 (2012), pp. 97–106.

[115] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[116] Li Deng. "A tutorial survey of architectures, algorithms, and applications for deep learning". In: *APSIPA transactions on Signal and Information Processing* 3 (2014), e2.

[117] Peter A Forsyth and George Labahn. "Numerical methods for controlled Hamilton-Jacobi-Bellman PDEs in finance". In: *Journal of Computational Finance* 11.2 (2007), p. 1.

[118] Andrei Nikolaevich Kolmogorov. "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition". In: *Doklady Akademii Nauk*. Vol. 114. 5. Russian Academy of Sciences. 1957, pp. 953–956.

[119] Guang Hao Low and Isaac L Chuang. "Optimal Hamiltonian simulation by quantum signal processing". In: *Physical review letters* 118.1 (2017), p. 010501.

[120] Kosuke Mitarai, Masahiro Kitagawa, and Keisuke Fujii. "Quantum analog-digital conversion". In: *Physical Review A* 99.1 (2019), p. 012301.

[121] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. "Nonlinear transformation of complex amplitudes via quantum singular value transformation". In: *arXiv preprint arXiv:2107.10764* (2021).

[122] Michael Lubasch et al. "Variational quantum algorithms for nonlinear problems". In: *Physical Review A* 101.1 (2020), p. 010301.

[123] Zoë Holmes et al. "Nonlinear transformations in quantum computation". In: *Physical Review Research* 5.1 (2023), p. 013105.

[124] Nikita Guseynov, Xiajie Huang, and Nana Liu. "Explicit gate construction of block-encoding for Hamiltonians needed for simulating partial differential equations". In: *arXiv preprint arXiv:2405.12855* (2024).

[125] Nhat A Nghiem et al. "Improved Quantum Power Method and Numerical Integration Using Quantum Singular Value Transformation". In: *arXiv preprint arXiv:2407.11744* (2024).

[126] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[127] Naixu Guo et al. "Quantum linear algebra is all you need for Transformer architectures". In: *arXiv preprint arXiv:2402.16714* (2024).

[128] Hans Hon Sang Chan et al. "Grid-based methods for chemistry simulations on a quantum computer". In: *Science Advances* 9.9 (2023), eabo7484.

[129] Joran van Apeldoorn. "Quantum probability oracles & multidimensional amplitude estimation". In: *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2021.

[130] Joran van Apeldoorn et al. "Quantum tomography using state-preparation unitaries". In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 1265–1318.

[131] Johannes Bausch. "Fast black-box quantum state preparation". In: *Quantum* 6 (2022), p. 773.

[132] Sam McArdle, András Gilyén, and Mario Berta. "Quantum state preparation without coherent arithmetic". In: *arXiv preprint arXiv:2210.14892* (2022).

[133] Javier Gonzalez-Conde et al. "Efficient amplitude encoding of polynomial functions into quantum computers". In: *arXiv preprint arXiv:2307.10917* (2023).

[134] Danial Motlagh and Nathan Wiebe. "Generalized Quantum Signal Processing". In: *arXiv preprint arXiv:2308.01501* (2023).

[135] Shelby Kimmel et al. "Hamiltonian simulation with optimal sample complexity". In: *npj Quantum Information* 3.1 (2017), p. 13.

[136] M Cerezo et al. "Challenges and opportunities in quantum machine learning". In: *Nature Computational Science* 2.9 (2022), pp. 567–576.

[137] Jacob Biamonte et al. "Quantum Machine Learning". In: *Nature* 549 (2016), pp. 195–202. URL: https://doi.org/10.1038/nature23474.

[138] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Springer Cham, 2021. URL: https://doi.org/10.1007/978-3-030-83098-4.

[139] Yuxuan Du et al. "Quantum machine learning: A hands-on tutorial for machine learning practitioners and researchers". In: *arXiv preprint arXiv:2502.01146* (2025).

[140] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5 (2014), p. 4213. URL: https://doi.org/10.1038/ncomms5213.

[141] Marco Cerezo et al. "Variational quantum algorithms". In: *Nat. Rev. Phys* 3.9 (Aug. 2021), pp. 625–644. URL: http://doi.org/10.1038/s42254-021-00348-9.

[142] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Sci. Technol.* 4.4 (Nov. 2019), p. 043001. URL: http://doi.org/10.1088/2058-9565/ab4eb5.

[143] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. URL: https://doi.org/10.1038/s41586-019-0980-2.

[144] Maria Schuld and Nathan Killoran. "Quantum Machine Learning in Feature Hilbert Spaces". In: *Phys. Rev. Lett.* 122 (4 Feb. 2019), p. 040504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.040504.

[145] Marcello Benedetti et al. "A generative modeling approach for benchmarking and training shallow quantum circuits". In: *npj Quantum Information* 5.1 (May 2019). URL: https://doi.org/10.1038/s41534-019-0157-8.

[146] Po-Wei Huang and Patrick Rebentrost. *Post-variational quantum neural networks*. 2024. arXiv: 2307.10560.

[147] Lennart Bittel and Martin Kliesch. "Training Variational Quantum Algorithms Is NP-Hard". In: *Phys. Rev. Lett.* 127 (12 2021), p. 120502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.127.120502.

[148] Eric R. Anschuetz and Bobak T. Kiani. "Quantum variational algorithms are swamped with traps". In: *Nature Communications* 13.1 (Dec. 2022). URL: http://dx.doi.org/10.1038/s41467-022-35364-5.

[149] Jarrod R. McClean et al. "Barren plateaus in quantum neural network training landscapes". In: *Nat. Commun.* 9.1 (Nov. 2018). URL: https://doi.org/10.1038/s41467-018-07090-4.

[150] Martin Larocca et al. *A Review of Barren Plateaus in Variational Quantum Computing*. 2024. arXiv: 2405.00781 [quant-ph].

[151] M. Cerezo et al. *Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing*. 2023. arXiv: 2312.09121 [quant-ph].

[152] Pablo Bermejo et al. *Quantum Convolutional Neural Networks are (Effectively) Classically Simulable*. 2024. arXiv: 2408.12739 [quant-ph]. URL: https://arxiv.org/abs/2408.12739.

[153] Supanut Thanasilp et al. "Exponential concentration in quantum kernel methods". In: *Nature Communications* 15.1 (June 2024). URL: https://doi.org/10.1038/s41467-024-49287-w.

[154] Manuel S. Rudolph et al. "Trainability barriers and opportunities in quantum generative modeling". In: *npj Quantum Information* 10.1 (Nov. 2024). URL: https://doi.org/10.1038/s41534-024-00902-0.

[155] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* 103 (15 Oct. 2009), p. 150502. URL: https://doi.org/10.1103/PhysRevLett.103.150502.

[156] Ashley Montanaro. "Quantum algorithms: an overview". In: *npj Quantum Information* 2.1 (Jan. 2016). URL: http://dx.doi.org/10.1038/npjqi.2015.23.

[157] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum Support Vector Machine for Big Data Classification". In: *Phys. Rev. Lett.* 113 (13 Sept. 2014), p. 130503. URL: https://link.aps.org/doi/10.1103/PhysRevLett.113.130503.

[158] Nathan Wiebe, Daniel Braun, and Seth Lloyd. "Quantum Algorithm for Data Fitting". In: *Phys. Rev. Lett.* 109 (5 2012), p. 050505. URL: https://link.aps.org/doi/10.1103/PhysRevLett.109.050505.

[159] Jonathan Allcock et al. "Quantum Algorithms for Feedforward Neural Networks". In: *ACM Trans. Quantum Comput.* 1.1 (Oct. 2020). URL: https://doi.org/10.1145/3411466.

[160] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. "Quantum Algorithms for Deep Convolutional Neural Networks". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=Hygab1rKDS.

[161] Naixu Guo et al. *Quantum linear algebra is all you need for Transformer architectures*. 2024. arXiv: 2402.16714 [quant-ph]. URL: https://arxiv.org/abs/2402.16714.

[162] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. *Quantum Deep Learning*. 2015. arXiv: 1412.3489 [quant-ph]. URL: https://arxiv.org/abs/1412.3489.

[163] Ashish Kapoor, Nathan Wiebe, and Krysta Svore. "Quantum perceptron models". In: *Advances in neural information processing systems* 29 (2016).

[164] El Amine Cherrat et al. "Quantum Vision Transformers". In: *Quantum* 8 (Feb. 2024), p. 1265. URL: https://doi.org/10.22331/q-2024-02-22-1265.

[165] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. "A rigorous and robust quantum speed-up in supervised machine learning". In: *Nature Physics* 17 (2021), pp. 1013–1017. URL: https://doi.org/10.1038/s41567-021-01287-z.

[166] Siyi Yang et al. "Quantum Alphatron: quantum advantage for learning with kernels and noise". In: *Quantum* 7 (Nov. 2023), p. 1174. URL: https://doi.org/10.22331/q-2023-11-08-1174.

[167] Petr Ivashkov et al. *QKAN: Quantum Kolmogorov-Arnold Networks*. 2024. arXiv: 2410.04435 [quant-ph]. URL: https://arxiv.org/abs/2410.04435.

[168] Yunfei Wang et al. *Towards efficient quantum algorithms for diffusion probability models*. 2025. arXiv: 2502.14252 [quant-ph]. URL: https://arxiv.org/abs/2502.14252.

[169] Iordanis Kerenidis and Anupam Prakash. "Quantum gradient descent for linear systems and least squares". In: *Physical Review A* 101.2 (Feb. 2020). URL: http://dx.doi.org/10.1103/PhysRevA.101.022316.

[170] Amira Abbas et al. "On quantum backpropagation, information reuse, and cheating measurement collapse". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 44792–44819. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3caae2f725c8e2a55ecd600563d172-Abstract.html.

[171] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. "Quantum convolutional neural networks". In: *Nat. Phys.* 15.12 (Aug. 2019), pp. 1273–1278. URL: http://doi.org/10.1038/s41567-019-0648-8.

[172] Yu Cheng et al. "A survey of model compression and acceleration for deep neural networks". In: *arXiv preprint arXiv:1710.09282* (2017).

[173] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[174] Alexander M Dalzell et al. "A distillation-teleportation protocol for fault-tolerant QRAM". In: *arXiv preprint arXiv:2505.20265* (2025).

[175] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN models for fine-grained visual recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1449–1457.

[176] K. Mitarai, M. Kitagawa, and K. Fujii. "Quantum analog-digital conversion". In: *Physical Review A* 99.1 (2019), p. 012301.

[177] Srinivasan Arunachalam and Reevu Maity. "Quantum boosting". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 377–387.

[178] Adam Izdebski and Ronald de Wolf. "Improved quantum boosting". In: *arXiv preprint arXiv:2009.08360* (2020).

[179] Wilfred Salmon, Sergii Strelchuk, and Tom Gur. "Provable advantage in quantum pac learning". In: *The Thirty Seventh Annual Conference on Learning Theory*. PMLR. 2024, pp. 4487–4510.

[180] Ananth Grama et al. *Introduction to Parallel Computing*. Addison Wesley, 2003. Chap. 8.

[181] D. J. Bernstein. *Circuits for integer factorization: a proposal*. `https://cr.yp.to/papers.html#nfscircuit`. 2001.

[182] Tameem Albash and Daniel A Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (2018), p. 015002.

[183] Stephen P Jordan, Keith SM Lee, and John Preskill. "Quantum algorithms for quantum field theories". In: *Science* 336.6085 (2012), pp. 1130–1133.

[184] Alexei Kitaev and William A Webb. "Wavefunction preparation and resampling using a quantum computer". In: *arXiv preprint arXiv:0801.0342* (2008).

[185] Christian W Bauer et al. "Practical considerations for the preparation of multivariate Gaussian states on quantum computers". In: *arXiv preprint arXiv:2109.10918* (2021).

[186] Maris Ozols, Martin Roetteler, and Jérémie Roland. "Quantum rejection sampling". In: *ACM Transactions on Computation Theory (TOCT)* 5.3 (2013), pp. 1–33.

[187] Lukas Owens. *Exploring the rate of convergence of approximations to the riemann integral*. 2014.

[188] Hiroyuki Tasaki. "Convergence rates of approximate sums of Riemann integrals". In: *Journal of approximation theory* 161.2 (2009), pp. 477–490.

[189] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 2010.

[190] Mehdi Saeedi and Massoud Pedram. "Linear-depth quantum circuits for n-qubit Toffoli gates with no ancilla". In: *Physical Review A* 87.6 (2013), p. 062318.

[191] David J Leeming. *The real zeros of the Bernoulli polynomials*. Tech. rep. 1987.

[192] Guang Hao Low and Isaac L Chuang. "Hamiltonian simulation by uniform spectral amplification". In: *arXiv preprint arXiv:1707.05391* (2017).

[193] Seth Lloyd and Christian Weedbrook. "Quantum generative adversarial learning". In: *Physical review letters* 121.4 (2018), p. 040502.

[194] Jonathan Romero and Alán Aspuru-Guzik. "Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions". In: *Advanced Quantum Technologies* 4.1 (2021), p. 2000003.

[195] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation". In: *arXiv preprint arXiv:1804.01973* (2018).

[196] Joran Van Apeldoorn et al. "Quantum SDP-solvers: Better upper and lower bounds". In: *Quantum* 4 (2020), p. 230.

[197] Lov K Grover. "A different kind of quantum search". In: *arXiv preprint quant-ph/0503205* (2005).

[198] Anupam Prakash. *Quantum algorithms for linear algebra and machine learning.* University of California, Berkeley, 2014.

[199] Daan Camps and Roel Van Beeumen. "Approximate quantum circuit synthesis using block encodings". In: *Physical Review A* 102.5 (2020), p. 052411.

[200] Shantanav Chakraborty, Aditya Morolia, and Anurudh Peduri. "Quantum regularized least squares". In: *Quantum* 7 (2023), p. 988.

[201] Kaito Wada, Naoki Yamamoto, and Nobuyuki Yoshioka. "Heisenberg-limited adaptive gradient estimation for multiple observables". In: *PRX Quantum* 6.2 (2025), p. 020308.

[202] Andrew M Childs and Nathan Wiebe. "Hamiltonian simulation using linear combinations of unitary operations". In: *arXiv preprint arXiv:1202.5822* (2012).

[203] A Paszke. "Pytorch: An imperative style, high-performance deep learning library". In: *arXiv preprint arXiv:1912.01703* (2019).

[204] Danial Motlagh and Nathan Wiebe. "Generalized quantum signal processing". In: *PRX Quantum* 5.2 (2024), p. 020368.

[205] Daan Camps et al. "Explicit Quantum Circuits for Block Encodings of Certain Sparse Matrices". In: *SIAM J. Matrix Anal. Appl.* 45.1 (2024), pp. 801–827. URL: https://doi.org/10.1137/22M1484298.

[206] Hanie Sedghi, Vineet Gupta, and Philip M Long. "The singular values of convolutional layers". In: *arXiv preprint arXiv:1805.10408* (2018).

[207] Don Coppersmith. "An approximate Fourier transform useful in quantum factoring". In: *arXiv preprint quant-ph/0201067* (2002).

[208] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. "Nonlinear transformation of complex amplitudes via quantum singular value transformation". In: *Phys. Rev. Res.* 6 (4 Dec. 2024), p. 043227. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.6.043227.

[209] Marcus Cramer et al. "Efficient quantum state tomography". In: *Nature communications* 1.1 (2010), p. 149.

[210] Takeru Miyato et al. "Spectral normalization for generative adversarial networks". In: *arXiv preprint arXiv:1802.05957* (2018).

[211] Yuichi Yoshida and Takeru Miyato. "Spectral norm regularization for improving the generalizability of deep learning". In: *arXiv preprint arXiv:1705.10941* (2017).

[212] Henry Gouk et al. "Regularisation of neural networks by enforcing lipschitz continuity". In: *Machine Learning* 110 (2021), pp. 393–416.

[213] Connor T Hann. "Practicality of quantum random access memory". PhD thesis. Yale University, 2021.

[214] Aaron Carroll and Gernot Heiser. "An analysis of power consumption in a smartphone". In: *2010 USENIX Annual Technical Conference (USENIX ATC 10).* 2010.

[215] Aqeel Mahesri and Vibhore Vardhan. "Power consumption breakdown on a modern laptop". In: *International Workshop on Power-Aware Computer Systems.* Springer. 2004, pp. 165–180.

[216] Kazi Main Uddin Ahmed, Math HJ Bollen, and Manuel Alvarez. "A review of data centers energy consumption and reliability modeling". In: *IEEE access* 9 (2021), pp. 152536–152563.

[217] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. "Power provisioning for a warehouse-sized computer". In: *ACM SIGARCH computer architecture news* 35.2 (2007), pp. 13–23.

[218] V. Giovannetti, S. Lloyd, and L. Maccone. "Quantum Random Access Memory". In: *Physical Review Letters* 100.16 (2008), p. 160501. URL: https://doi.org/10.1103/PhysRevLett.100.160501.

[219] Fang-Yu Hong et al. "Robust quantum random access memory". In: *Physical Review A—Atomic, Molecular, and Optical Physics* 86.1 (2012), p. 010306.

[220] Sevag Gharibian and François Le Gall. "Dequantizing the quantum singular value transformation: hardness and applications to quantum chemistry and the quantum PCP conjecture". In: *Proceedings of the 54th annual ACM SIGACT symposium on theory of computing*. 2022, pp. 19–32.

[221] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. "Learning and generalization in overparameterized neural networks, going beyond two layers". In: *Advances in neural information processing systems* 32 (2019).

[222] Hela Mhiri et al. *A unifying account of warm start guarantees for patches of quantum landscapes*. 2025. arXiv: 2502.07889 [quant-ph]. URL: https://arxiv.org/abs/2502.07889.

[223] Sushant Sachdeva, Nisheeth K Vishnoi, et al. "Faster algorithms via approximation theory". In: *Foundations and Trends® in Theoretical Computer Science* 9.2 (2014), pp. 125–210.