Quantum Singular Value Transforms: Analysis and Algorithmic Implementation

Shaurya Rawat
PH19B013

Supervised by Prof. Prabha Mandayam

Department of Physics, IIT Madras

Dual Degree Project Report

June 2, 2024

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Prabha Mandayam, for giving me the opportunity to work on this project. Her guidance was invaluable and a source of constant learning for me in all the stages of this project.

I am also immensely thankful to Debjyoti Biswas whose inputs and support significantly contributed in moving this project forward. His suggestions and expertise were crucial for the completion of this project.

Furthermore, I would like to acknowledge my batchmates and my family for their understanding and support throughout my studies. My thanks also go to the members of the Quantum Computing group in IITM for the healthy discussions and collaborative environment.

This thesis would not have been possible without the contributions and encouragement of each one mentioned above.

Abstract

Quantum Singular Value Transformation has been labelled as a *Grand Unification of Quantum Algorithms* as it encompassses a variety of quantum algorithms within a single framework. We study the fundamentals of Quantum Singular Value Transform (QSVT), namely block encoding and quantum signal processing (QSP). We see how QSVT construction alongside Chebyshev approximation enables us to perform polynomial transformations of the singular values of a linear operator embedded in a unitary matrix. We further look into potential applications in Quantum Search and Phase Estimation, and Hamiltonian Simulation, analyzing performance and implementation costs in terms of time complexity and qubit complexities against known quantum algorithms. Also comparing noise models for the algorithms, namely thermal(T1 relaxation time), depolarizing, etc. Potential advantages and caveats are discussed. Furthermore, the study extends to the construction of QSVT based method for Ground State Preparation of a Hamiltonian, outlining a possible future direction for the project.

Contents

1 Introduction **5**
- 1.1 Quantum Bits . . . 5
- 1.2 Quantum Gates and Quantum Circuits . . . 6

2 Quantum Singular Value Transformation **7**
- 2.1 Quantum Singular Value Transformations . . . 7
- 2.2 Block Encoding . . . 7
- 2.3 Quantum Signal Processing . . . 9
- 2.4 QSP Phases through Chebyshev Polynomials . . . 10

3 QSVT Construction : Combining QSP and Block Encoding **13**
- 3.1 Constructing Block Encoding . . . 13
- 3.2 QSP Subroutine for Polynomial Transformations . . . 13
- 3.3 Projector Controlled Phase Shifts . . . 14
- 3.4 Grand Unification of Quantum Algorithms . . . 15

4 Quantum Search Problem **16**
- 4.1 Grover Search Algorithm . . . 16
- 4.2 QSVT Search Algorithm . . . 17
  - 4.2.1 Algorithm Overview: . . . 17
  - 4.2.2 Quantum Search in 2 Qubits . . . 18
  - 4.2.3 Circuit Depth and Scalability . . . 19

5 Eigenphase Estimation Problem **21**
- 5.1 QFT based Phase Estimation from Shor's Algorithm . . . 21
  - 5.1.1 Algorithm Overview: . . . 21
- 5.2 Iterative Phase Estimation (Kitaev's Algorithm) . . . 22
  - 5.2.1 Algorithm Overview: . . . 22
  - 5.2.2 Hardware Implementation and Dynamic Circuits: . . . 24
- 5.3 QSVT Phase Estimation . . . 25
  - 5.3.1 Algorithm Overview: . . . 26
  - 5.3.2 Circuit Depth and Scalability . . . 27
  - 5.3.3 Estimating Phase for T gate . . . 28
  - 5.3.4 Noise Analysis for Phase Estimation . . . 29
- 5.4 Application in Order Finding and Factoring . . . 31
  - 5.4.1 Time Complexity of QSVT Factoring . . . 32

6 Hamiltonian Simulation **33**
- 6.1 QSVT for Hamiltonian Simulation: . . . 33
  - 6.1.1 Algorithm Overview: . . . 35
  - 6.1.2 Implementation for a Single Qubit System : . . . 35
  - 6.1.3 Caveats: . . . 36
- 6.2 Ground State Preparation: . . . 37

7 Conclusion and Future Direction **38**
7.1 Github Repository: 38

A QSPPACK Phase Solver **39**

B Proving QSVT Theorem **40**

C Geometric Interpretation for Grover Search **41**

1 Introduction

1.1 Quantum Bits

In quantum computing qubit or quantum bit is the basic unit of quantum information. In a quantum computer the basic memory units are qubits, analogous to bits on a classical computer. A classical bit can take two distinct values, 0 or 1 at a given time. In contrast, a qubit state can be a combination of both 0 and 1, (superposition principle) where the qubit is in both basic states (0 and 1) at the same time. A qubit is a quantum system in which the Boolean states 0 and 1 are represented by a pair of normalised and mutually orthogonal quantum states. The two states form a computational basis and any other (pure) state of the qubit can be written as a linear combination,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where, α and β are probability amplitudes and can be complex numbers. In a measurement the probability of the bit being in $|0\rangle$ state is $|\alpha|^2$ and $|1\rangle$ is $|\beta|^2$. Single qubit state can be represented :

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Qubit exploits the quantum properties of Superposition and Entanglement.

- **Superposition Principle:** Superposition permits a qubit to exist simultaneously in both the $|0\rangle$ and $|1\rangle$ states, where α and β are complex numbers representing the probability amplitudes of these states, as shown above.
- **Quantum Entanglement:** Quantum phenomenon where the states of two or more qubits become so intertwined that the state of one cannot be described independently of the others, even when separated by large distances.

Bloch Sphere

The Bloch sphere [1] is a geometric representation of the state space of a qubit. It provides a useful visualization of a qubit's state, which is defined by the superposition of its basis states $|0\rangle$ and $|1\rangle$.

- **Representation:** . A qubit's state on the Bloch sphere can be expressed as:

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

  where θ and ϕ are the polar and azimuthal angles, respectively. Qubit states C^2 correspond to the points on the surface of a unit sphere R^3.

  Any pure state of a qubit can be represented as a point on the surface of a sphere with radius 1. The north and south poles of the sphere correspond to the classical states $|0\rangle$ and $|1\rangle$, respectively

1.2 Quantum Gates and Quantum Circuits

Quantum gates are represented as unitary matrices (U) which act on a given qubit state ($|\psi\rangle$). The operation is by multiplying the matrix representing the gate with the vector that represents the quantum state.

$$|\psi'\rangle = U|\psi\rangle$$

- **Quantum gates as Unitary matrices:** Quantum gates for a single qubit are 2×2 unitary matrices. The gate operation on a qubit's state vector is executed by matrix-vector multiplication. Matrix representation of Single Qubit Pauli Gates and Hadamard Gate :

  $$\text{Pauli-X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{Pauli-Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \text{Pauli-Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

  Quantum Measurements: Measurements in a quantum system are used to extract classical information from the quantum circuits (system). Measurement collapses the state of a qubit into one of the basis states of the measurement (usually Z-measurement), $|0\rangle$ or $|1\rangle$, depending on the qubit's superposition probabilities. The outcome is inherently probabilistic, a consequence of quantum nature of the system. We can also perform projective measurements

- **Quantum Circuits:** A quantum circuit is a sequence of quantum gates and measurements (usually at the end of quantum operations), intended to perform a specific quantum algorithm. Each gate in the circuit modifies the state of the qubits, eventually transforming the initial quantum state into an output state that represents the computation's result.

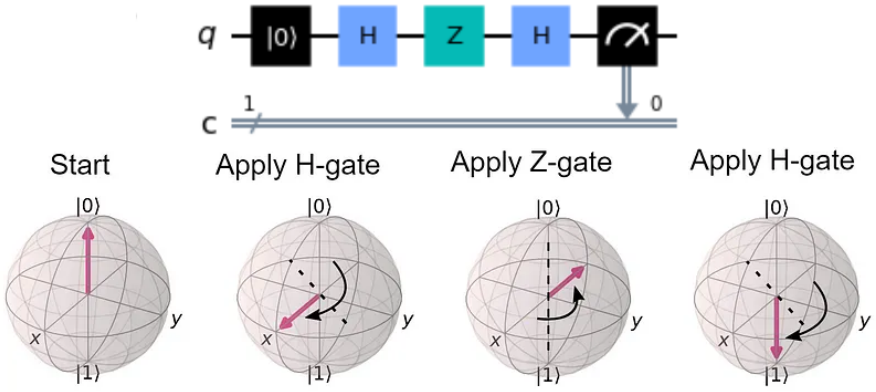

Figure 1: Quantum Circuit with Bloch Sphere Representation to visualize Quantum Operations

2 Quantum Singular Value Transformation

Singular Values and Singular Value Decomposition

In classical Singular Value Decomposition (SVD), a matrix A can be decomposed as:

$$A = W\Sigma(\sigma)V^\dagger \tag{1}$$

where W and V are unitaries containing the left and right singular vectors, respectively, and Σ is a diagonal matrix containing the singular values ($\sigma_i's$).

2.1 Quantum Singular Value Transformations

Quantum Singular Value Transform (QSVT) [2] is an algorithmic framework to apply polynomial transformations, $P(\sigma)$, to the singular values σ of an input matrix. The choice of the polynomial function depends on the specific application and the desired transformation of the singular values with control over the degree of the polynomial, bounds, accuracy, etc, and certain constraints which we state in section 2.3
For a given matrix A, with singular value decomposition as:

$$A = \sum_i \sigma_i \left|w_i\right\rangle \left\langle v_i\right| \tag{2}$$

Here again, σ_i's represent the singular values and $|W_i\rangle$ and $|v_i\rangle$ are the left and right singular vectors. The **QSVT transformation** can be summarized as:

$$\text{QSVT}(A) = P(A) = \sum_i P(\sigma_i) \left|w_i\right\rangle \left\langle v_i\right| \tag{3}$$

where P satisfies the QSP Constraints, discussed in Section 2.3

Note: Using QSVT, we can access the singular vector spaces and perform such manipulations to singular values, without actually computing SVD, which is a very powerful result in itself.
Also note that if A is Hermitian, the singular values and eigenvalues coincide and the transform thus can thus be interpreted as eigenvalue transformation. We work with singular values for our transformations as it is more generalized and can be applied to any matrix, be it non-hermitian, non-unitary or even non-square .

2.2 Block Encoding

Block encoding is a technique that allows encoding any matrix A of dimension d into a larger unitary matrix of dimension $D \times D$, with some ϵ-approximation such that

$$\left\| A - (|0\rangle^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I) \right\| \leq \epsilon$$

Note that $D > d$ where $log(D) = (a + log(d))$ is the space spanned by the block encoded operator U_A. This is significant, since now we can work with any non-unitary matrix. The block-encoded unitary matrix U_A can be represented as:

$$\mathbf{U_A} = \begin{matrix} \\ \widetilde{\Pi} \end{matrix} \begin{matrix} \Pi \\ \end{matrix} \begin{pmatrix} A & * \\ * & * \end{pmatrix}$$

This allows us to perform the Quantum Singular Value Transform (QSVT) on the singular values of any arbitrary matrix A, including non-unitary and non-square matrices.

Projectors in Block Encoding

We must be able to access the block encoded state via the left and right projectors $\widetilde{\Pi}$ and Π. These projectors are assumed accessible and locate subspace A within U_A :

$$A = \widetilde{\Pi} U \Pi$$

Using these projectors, we can also confirm that a state is indeed projected into the block encoding the transformed operator. Π selects the columns and $\widetilde{\Pi}$ the rows in which A is encoded. Moreover, the projectors Π and $\widetilde{\Pi}$ also identify the left and right singular vector spaces, respectively, with $\Pi = \sum_k |v_k\rangle\langle v_k|$ and $\widetilde{\Pi} = \sum_k |w_k\rangle\langle w_k|$. The action of these projectors in context to QSVT is explained in Section 3.3

Realizing Block Encoding

The concept of block encoding can be generalized for any matrix A. The operator below in LHS, an example of block encoding, is a valid unitary regardless of the form of A and doesn't even have to be a square matrix. We just need to ensure that A is properly normalized such that its largest singular value is bounded by 1.
The unitary matrix $U(A)$, which represents the block encoding of A, is defined as:

$$U(A) = \begin{pmatrix} A & \sqrt{I - A^\dagger A} \\ \sqrt{I - AA^\dagger} & -A^\dagger \end{pmatrix} \qquad U(a) = \begin{pmatrix} a & \sqrt{1 - a^2} \\ \sqrt{1 - a^2} & -a \end{pmatrix}$$

On RHS is a scalar value encoded into a Unitary Matrix

The Role of QSVT : QSVT allows the evaluation of a polynomial transformation $P(A)$ of the singular values of A. It is noteworthy that block encoding of any $N \times N$ matrix can be achieved with a maximum of $\log(N)$ ancilla qubits, and only 2 in the case of unitary encoded into another. [2]

Note: Block Encoding, especially of very large matrices is itself a challenging problem in general and represents the main bottleneck in the implementation of QSVT.

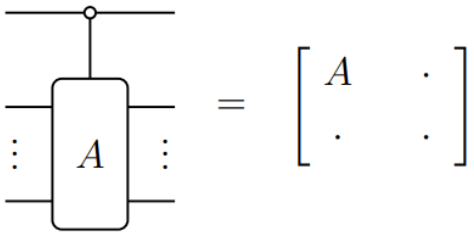

Figure 2: A simple block encoding quantum circuit for Unitary A, controlled on an ancilla

2.3 Quantum Signal Processing

Quantum Signal Processing [3] is a fundamental technique in quantum computing that allows for the application of polynomial transformations to quantum states. This section explores the concept of QSP, its implementation, and its application in deriving polynomial transformations, with a focus on Chebyshev Polynomials.

QSP Subroutine

Quantum Signal Processing Subroutine allows to perform polynomial transformations $P(x)$ of a given highest degree d, controlled by $'d+1'$ QSP phase angles $[\phi_1, \phi_2..., \phi_d]$. The core idea involves application of polynomial transforms to singular values of a non-unitary matrix A embedded into a larger unitary matrix.

QSP Phase Factors $(\phi_i' s)$ and Polynomial Transformation of a Scalar: To show how QSP applies polynomial transformations, consider a scalar a block-encoded within a 2×2 unitary matrix $U(a)$:

$$U(a) = \begin{pmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a \end{pmatrix}$$

$$\mathbf{U(a)} = \begin{matrix} \\ \langle 0| \end{matrix} \begin{matrix} |0\rangle \\ \end{matrix} \begin{pmatrix} a & * \\ * & * \end{pmatrix}$$

Note that here, projectors are simply $|0\rangle\langle 0|$, such that

$$|0\rangle\langle 0| U(a) |0\rangle\langle 0| = a |0\rangle\langle 0|$$

This matrix $U(a)$ forms the basis of QSP sequences. The QSP operator, here in case of scalar 'a' is simply a rotation $R_Z(\phi)$, is defined as:

$$R_Z(\phi) = \begin{pmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{pmatrix} \tag{6}$$

The polynomial transformation on a (of highest degree d) is achieved by interleaving $U(a)$ with rotations $R_Z(\phi_k)$, where $\phi_k \in [\phi_0, \phi_1, \phi_2..., \phi_d]$, by manipulating the encoded scalar 'a'. The resulting matrix captures a degree-d polynomial transformation of a :

$$R_Z(\phi_0)\prod_{k=1}^{d} U(a)R_Z(\phi_k) = \begin{pmatrix} P(a) & * \\ * & * \end{pmatrix} \tag{7}$$

with $P(a)$ representing the polynomial transformation.

Note: We take R_Z as QSP Operator, due to it being diagonal in computational basis and transforms are easily understood and scaled.
Some precalculated QSP Phases for the corresponding Polynomial Transforms are given in [4] Building up on this, QSVT applies these transforms onto singular values of encoded matrix, described in section 3.2.

QSP Theorem :

The QSP sequence above produces a matrix which may be expressed as a polynomial function of a as $P(a)$, given satisfying the following conditions:

1. *The degree of P, denoted as deg(P), is less than or equal to d, i.e., deg(P) $\leq d$.*
2. *The polynomial P has parity $d \mod 2$, i.e. highest degree parity (either even or odd)*
3. *The absolute value of P is less than or equal to 1 for all x in the interval $[-1, 1]$, i.e., $|P(x)| \leq 1$ for all $x \in [-1, 1]$.*

2.4 QSP Phases through Chebyshev Polynomials

A notable application of QSP is in generating Chebyshev Polynomials of first kind.

QSP Theorem [2] states that selecting specific QSP phase factors, such as $\phi_0 = (1 - d)\pi/2$ and $\phi_k = \pi/2$ for $k > 0$, we obtain the d-th Chebyshev Polynomial of the first kind:

$$R_Z\left((1-d)\frac{\pi}{2}\right)\prod_{k=1}^{d} U(a)R_Z\left(\frac{\pi}{2}\right) = \begin{pmatrix} T_d(a) & * \\ * & * \end{pmatrix} \tag{8}$$

Chebyshev Polynomials are orthogonal polynomials that align well with the requirements of QSP, including their range and parity and usecase in approximation theory. Being orthogonal, other polynomials can be constructed using combinations of Chebyshev polynomials.

Note: The general recursive form of Chebyshev Polynomials of the first kind is given by

$$T_0 = 1,\ T_1 = a, \qquad T_{d+1}(a) = 2aT_d(a) - T_{d-1}(a)$$

Example: Chebyshev Polynomial of degree 2 As an example, consider a polynomial transformation with $\phi = \pi/2$ and degree $d = 2$:

$$R_Z\left(-\frac{\pi}{2}\right) U(a) R_Z\left(\frac{\pi}{2}\right) U(a) R_Z(0) = \begin{pmatrix} 2a^2 - 1 & 0 \\ 0 & 2a^2 - 1 \end{pmatrix} \tag{9}$$

This represents the second-degree Chebyshev Polynomial T_2.

Approximating functions using Chebyshev Polynomials

Now that we can express QSP Phases of Chebyshev Polynomials, other functions are evaluated particularly as linear combinations of these. These polynomials have special properties:

1. **Minimax Property:** They are minimax functions over the range $[-1, 1]$, meaning all their minima/maxima/extrema are at ± 1.
2. **Orthogonality:** They are orthogonal with weight $\frac{1}{\sqrt{1-x^2}}$ over $[-1, 1]$.
3. **Recurrence Relation:**

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$
$$\text{Also, } T_n(x) = \cos(n \cos^{-1}(x)).$$

For a function $f(x)$ on $[a, b]$, its Chebyshev expansion is $f(x) = \sum_k c_k T_k(u)$ where $u = \frac{2x-a-b}{b-a}$. The coefficients c_k are obtained via Chebyshev nodes and interpolation:

1. Choose the approximation degree n.
2. Calculate Chebyshev nodes $x_i = \cos\left(\frac{2i-1}{2n}\pi\right)$ for $i = 1, 2, \ldots, n$.
3. Evaluate $f(x_i)$ and compute c_k using:

$$c_k = \frac{2}{n} \sum_{i=1}^{n} f(x_i) \cos\left(\frac{(k-1)(2i-1)\pi}{2n}\right). \tag{10}$$

The function approximation is then $f_{\text{approx}}(x) = \sum_{k=1}^{n} c_k T_{k-1}(u)$, mapping $[a, b]$ to $[-1, 1]$.

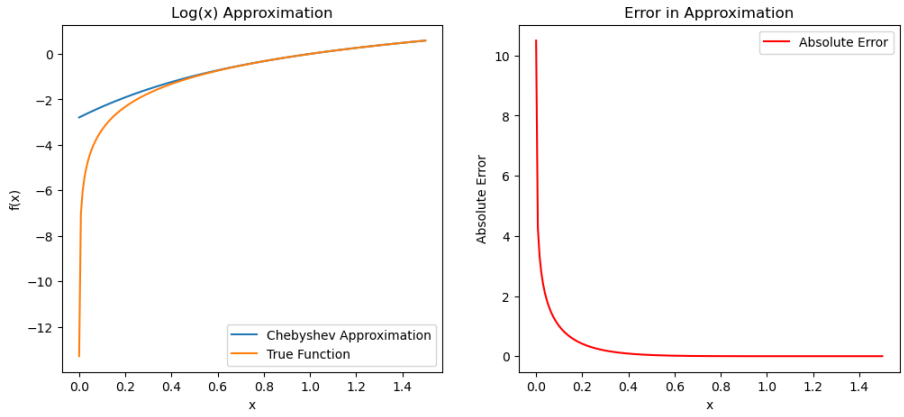

Figure 3: Chebyshev Approximation for $\log_2(x)$

Note: The above method to determine coefficients using Chebyshev nodes fails when function $f(x)$ is not a polynomial or when $f(x)$ has a very high degree and oscillating nature. For these reasons, we will use QSPPack Module given in A which is a python based framework and uses optimization algorithms to converge to optimal QSP Phase angles.

Also, the accuracy of the Chebyshev polynomial approximation depends on the degree n and the number of nodes used for interpolation. A balance between accuracy and computational efficiency should be considered.

3 QSVT Construction : Combining QSP and Block Encoding

The Quantum Singular Value Transform (QSVT) algorithm combines Quantum Signal Processing (QSP) with Block Encoding to manipulate and transform singular values of any blocke encoded matrix A. QSVT is structured as follows:

3.1 Constructing Block Encoding

First, we construct a block encoding of matrix A:

$$\mathbf{U_A} = \begin{matrix} \\ \widetilde{\Pi} \end{matrix} \overset{\Pi}{\begin{pmatrix} A & * \\ * & * \end{pmatrix}}$$

Here, $\widetilde{\Pi}$ and Π are projectors that act on the column and row subspaces, respectively. They locate A within U_A such that $A = \widetilde{\Pi} U_A \Pi$.

3.2 QSP Subroutine for Polynomial Transformations

The QSP subroutine allows for the polynomial transformation of A, represented as:

$$P(A) = \sum_k P(\sigma_k) \left|w_k\right\rangle \left\langle v_k\right| \tag{12}$$

For an even degree polynomial d (where the number of angles is $d+1$), the transformation is given by:

$$\left[\prod_{k=1}^{d/2} \Pi_{\phi_{2k-1}} U(A)^\dagger \widetilde{\Pi}_{\phi_{2k}} U(A)\right] \Pi_{\phi_{d+1}} = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix} \tag{13}$$

For an odd degree d, due to asymmetry (and again d+1 angles):

$$\widetilde{\Pi}_{\phi_1} \left[\prod_{k=1}^{(d-1)/2} \Pi_{\phi_{2k}} U(A)^\dagger \widetilde{\Pi}_{\phi_{2k+1}} U(A)\right] \Pi_{\phi_{d+1}} = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix} \tag{14}$$

Here, Π_ϕ and $\widetilde{\Pi}_\phi$ are projector-controlled phase gates that are assumed accessible. $\Pi_\phi = e^{i2\phi\Pi}$ which imparts a phase of $e^{i2\phi}$ to the entire subspace determined by the projector Π. It is a proper unitary transform and acts as a $R_Z(\phi)$ rotation onto the projected subspace.

The entire QSP Sequence can be understood as follows : The unitary operator U on the right moves the system from the $\{\left|v_k\right\rangle^i\}$ basis into the $\{\left|w_k\right\rangle^i\}$ basis. Following this, $\tilde{\Pi}_\phi$ then rotates the system around the z-axis of each left singular vector space's Bloch sphere by $e^{i2\phi}$. Similarly, the $U^\dagger$ (adjoint of U) on the left moves the system back from the $\{\left|w_k\right\rangle^i\}$ basis into the $\{\left|v_k\right\rangle^i\}$ basis. Finally, $\tilde{\Pi}_\phi$ rotates the system around the z-axis of each right singular vector space's Bloch sphere by again $e^{i2\phi}$.

3.3 Projector Controlled Phase Shifts

The sequence of QSP phase angles [Φ] determines the polynomial being implemented. As with QSP, it is possible to use the QSVT sequence to apply any real polynomial transformation that satisfies QSP Constraints mentioned in 2.3 up to degree d using $d+1$ phase angles. In fact, we can use the same angles regardless of the dimensions of A.

Utilizing the QSP factors, we implement polynomial transformations$P(A)$ onto the singular values of encoded matrix A. For this, we require Projector controlled Phase Shift to access subspace in A and apply $R_Z(\phi)$ rotation onto the projected subspace.
The controlled phase shift operation Π_ϕ is defined as [5]:

$$\Pi_\phi := e^{i\phi(2\Pi - I)}, \tag{15}$$

This operation imparts a phase of $e^{i\phi(2\Pi-I)}$ to the entire subspace determined by projector Π.
E.g. For a scalar a block encoded as $U(a)$, projectors are $|0\rangle\langle 0|$ and projector-controlled phase shift is given as :

$$U(a) = \begin{pmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a\dagger \end{pmatrix}; \qquad \Pi_\phi = e^{ei\phi(2|0\rangle\langle 0|-I)} = e^{i\phi Z} \tag{16}$$

Where $e^{i\phi Z}$ is just the Z-rotation, as we saw in previous section for a scalar block encoding. Note that if we neglect global phase, we get $\Pi_\phi := e^{i2\phi\Pi}$, which imparts a phase of $e^{i2\phi}$ to the subspace determined by Π. Now, for a 2x2 square matrix A block encoded as U_A, projector controlled phase shift is given as :

$$U(A) = \begin{pmatrix} A & \sqrt{I - A^\dagger A} \\ \sqrt{I - AA^\dagger} & -A^\dagger \end{pmatrix}; \qquad \Pi_\phi = \begin{pmatrix} e^{i2\phi} & 0 & 0 & 0 \\ 0 & e^{i2\phi} & 0 & 0 \\ 0 & 0 & e^{-i2\phi} & 0 \\ 0 & 0 & 0 & e^{-i2\phi} \end{pmatrix}. \tag{17}$$

Projection Operator (Π): The projection operator Π is used to select a subspace of the unitary matrix on which the singular value transformation is performed. It is a Hermitian operator that projects onto a specific subspace, allowing us to focus on a particular subset of the matrix elements. The choice of the projection operator determines which part of the unitary matrix undergoes the singular value transformation.

So to summarise, by employing a number of operations that grows linearly with the degree of the target polynomial, QSVT enables the transformation of the singular values of such arbitrary block-encoded matrices without the need for explicit singular value decompositions.

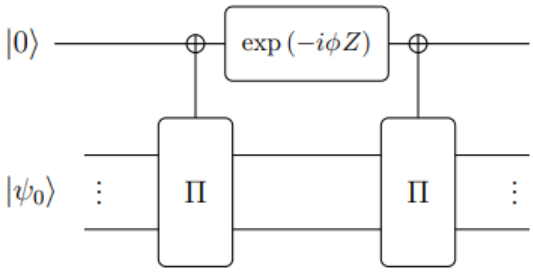

Figure 4: The circuit used to realize the projector controlled phase shift Π_ϕ

It simplifies to a single rotation for the simple projector $\Pi = |0\rangle\langle 0|$ in case of scalar encoding.

3.4 Grand Unification of Quantum Algorithms

QSVT has been labelled as a *Grand Unification of Quantum Algorithms* [6], with immense potential in various domains of quantum computing. We will look at how QSVT transforms can implement a variety known Quantum Algorithms, mainly Search, and Phase Estimation. Also, one of the highlighting areas of QSVT is application in Hamiltonian simulation as well as some function evaluation algorithms (matrix inversion, etc)

4 Quantum Search Problem

The Quantum Search problem involves searching through a disordered list of size $N = 2^n$. Representing this as a quantum state, the initial state $|s\rangle$ can be expressed as:

$$|s\rangle = \frac{1}{\sqrt{N}} |a_0\rangle + \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} |a_i\rangle \tag{18}$$

where $|a_0\rangle$ is the target state.

Classically, the search problem can be expressed with an abstract function $f(x)$, which is like an Oracle in quantum terms, which takes input as the search items x. If x_0 is a solution to the search task, then :

$$f(x) = \begin{cases} 1 & \text{if } x = x_0 \text{ (solution to the search task)} \\ 0 & \text{otherwise} \end{cases}$$

Time Complexity of Classical Algorithm

The complexity of a search algorithm is measured by the number of uses of the function $f(x)$ (or number of calls to the oracle). In the worst-case scenario, $f(x)$ has to be evaluated a total of N-1 times, trying out all the possibilities, thereby giving a time complexity of $O(N)$ for s linear search algorithm (classical)

4.1 Grover Search Algorithm

Grover's Algorithm solves the unstructured search problem with a high probability, using just $O(\sqrt{N})$ calls to Grover Iterate (Oracle function) [1].

Algorithm Overview : For a list of size $N = 2^n$, we require a qubit register of size n.

- We start with the super position of all states.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |a_i\rangle \tag{19}$$

- Apply the phase oracle U_f which gives a conditional phase shift of -1 to the marked element, i.e, flips the amplitude. It's action can be understood from appendix C as a reflection in the qubitized $2D$ plane. Refer (4.2)

- Apply the Grover Diffusion Operator which gives A conditional phase shift of -1 to every computational basis state except $|0\rangle$. Diffusion Operator is given as

$$U_\phi = \frac{1}{2} I - \frac{1}{2} |s\rangle \langle s|$$

- **Grover Iterate :** Phase Oracle along with the Diffusion Operator, forms our Grover Iterate, given by the Unitary operation:

$$-(U_f H^{\otimes n} U_\phi H^{\otimes n})^{N_{\text{optimal}}}$$

Here, N_{optimal} is the number of times we have to repeat the Grover iterate. It suffices that number of Grover Iterations $\propto O(\sqrt{N})$, as shown in Appendix C

- Finally, we measure the qubits to collapse the state, revealing the solution with high probability (if it exists).

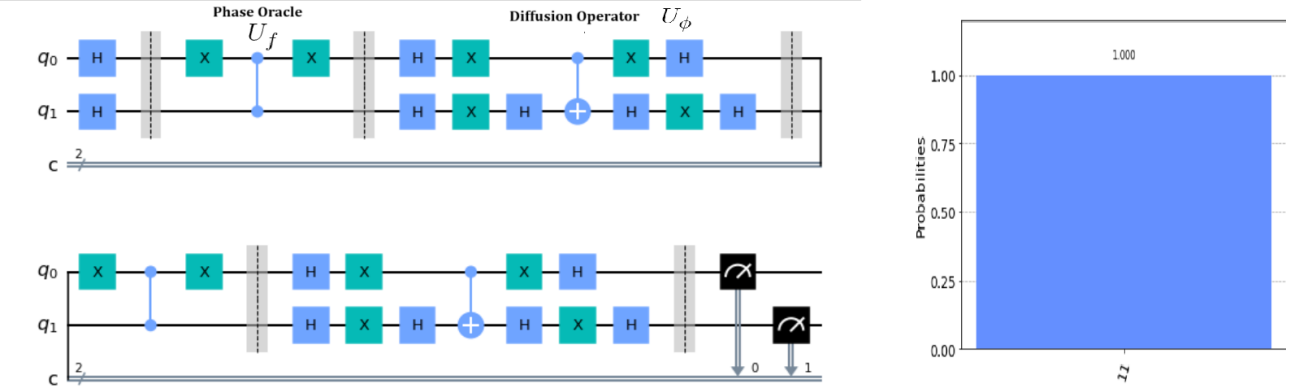

Figure 5: Grover Search Circuit for n=2 (N=4). Marked element is $|11\rangle$

4.2 QSVT Search Algorithm

Qubitization

Using Qubitization, we redefine the initial state $|s\rangle$ in terms of the target state $|t\rangle$ and summation of all other states, which will be orthogonal $|t^\perp\rangle$:

$$|s\rangle = \frac{1}{\sqrt{N}}|t\rangle + \sqrt{\frac{N-1}{N}}|t^\perp\rangle \tag{20}$$

This way, we have effectively reduced our search into a 2-qubit like space, which we will leverage.

4.2.1 Algorithm Overview:

Input: Access to a controlled version of the oracle U which bit-flips an auxiliary qubit when given an unknown target state $|t\rangle$, and the original input state $|s\rangle$

Output: The flagged state $|t\rangle$.

QSVT Procedure:

1. **Block Encoding:** Encode the inner product $a = \langle s \mid t \rangle$, (where $a = \frac{1}{\sqrt{N}}$) in its $|t\rangle \langle s|$ element. We take the projectors $|s\rangle \langle s|$ and $|t\rangle \langle t|$, such that our block encodes the inner product :
$$\widetilde{\Pi} U \Pi = |s\rangle \langle s| U |t\rangle \langle t| = a |s\rangle \langle t|$$
$$\mathbf{U} = \begin{pmatrix} \langle s|t\rangle & * \\ * & * \end{pmatrix}$$
2. **Transform:** We apply transform P(a) $\rightarrow$ 1
We amplify the amplitude corresponding to the inner product $a = \langle s \mid t \rangle$ (amplitude amplification) This induces the mapping of $|s\rangle$ in the uniform superposition $|t\rangle$, and with high probability (if $|t\rangle$ exists in $|s\rangle$). The target remains in the register with probability $= |P(a)|^2$ applied by the polynomial transformation and can be determined by measuring in the computational basis

QSP Phase Angles:

Chebyshev Polynomials (first kind) satisfy the criteria for our polynomial transform:

$$T_d(x = \tfrac{1}{\sqrt{2^n}}) \approx 1$$

where degree d scales with n (number of qubits to encode our search list)

Upon calculating, we see the relation $n \approx (2d - 1)$ which works well up to n=5, as seen in figure 8. QSP Angles for single Chebyshev Polynomials of first kind are straightforward. QSP Theorem 2 states that given T_d, the first phase is $(1 - d)\frac{\pi}{2}$ and the rest of the phases are simply $\frac{\pi}{2}$. It is important to note that due to Qiskit's definition of $R_Z(\theta) = f(\frac{\theta}{2})$ all of our phases must be scaled by a factor of 2 before we put them into the circuit.

For higher n values ($n > 5$), our linear relation does not work well and we require more Chebyshev terms. The QSP Phase angles for such linear combinations of higher degree Chebyshevs are not easily evaluated.

4.2.2 Quantum Search in 2 Qubits

We consider a quantum search problem in a 2-qubit system. The initial state $|s\rangle$ is given by:

$$|s\rangle = \frac{1}{\sqrt{2^2}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{21}$$

The target state $|t\rangle$ is $|11\rangle$
The Quantum Circuit for QSVT is created using the following components:

- **Block Encoding:** The state $|s\rangle$ is encoded as :

$$A = \langle s \mid t \rangle = \frac{1}{2} \tag{22}$$

- **QSP Phase Angles:** We apply a polynomial transformation $P(A) \to 1$. The third-degree Chebyshev Polynomial $T_3(x)$ satisfies this requirement. We take QSP Phase Angles corresponding to the T_3 Chebyshev Polynomial.

Assuming we have access to projector controlled phase shifts (R_Z for 2 qubit case) with the phase angles as above, this circuit, when executed, performs the QSVT operation, amplifying the probability amplitude of the target state $|11\rangle$ in the given 2-qubit system.

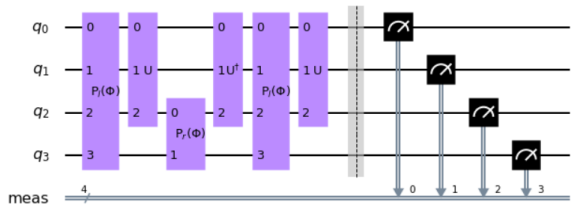

Figure 6: Circuit for Implementing 2-Qubit Search

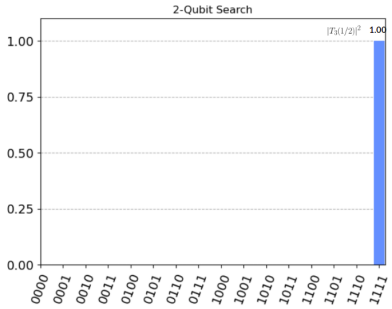

Figure 7: Obtained target state $|11\rangle$ among the 4 states. Note that final amplitude becomes proportional to $|T_3(x)|^2$

4.2.3 Circuit Depth and Scalability

The degree of the polynomial (or depth of circuit) scales linearly as :

$$d \approx 2n - 1 \approx log(N), \text{ up to n=5}$$

Beyond this, the accuracy of the Chebyshev transform decreases and we need to incorporate more terms than a single Chebyshev coefficient to implement our transform. Calculating these coefficients is computationally expensive and makes use of optimisation algorithms. We make use of QSPPack Module [4], available freely on Github.

Upon calculating QSP Phases using QSPPack Module (in appendix A) for input space ($n > 5$), we approximate an exponential relation between circuit depth and input space (N) as seen in Figure 8.
Note that the number of QSP phase angles obtained may not be optimal as the optimisation carried out to calculate QSP Phases need not correspond to a global minima. Determining such optimal QSP Phases is a subfield itself.

Note : Grover Search Algorithm takes $O(\sqrt{N})$ queries to the Grover Iterate for optimal solution (Depth of Circuit).

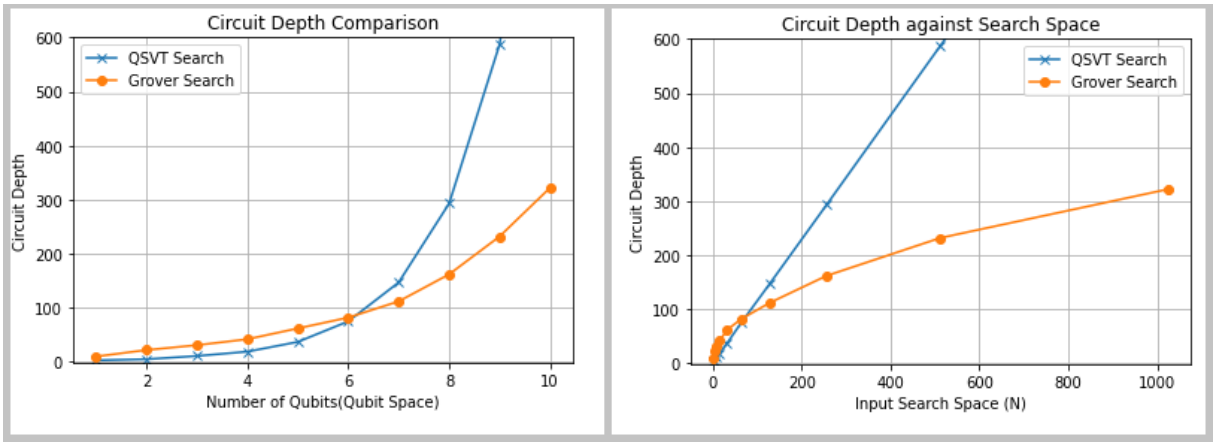

Figure 8: QSVT Depth obtained from calculating number of QSP Phases using QSPPACK Module with error limit set as ($\epsilon = |P(\langle s \mid t \rangle) - 1| < 0.1$), meanwhile Grover depth is $O(\sqrt{N})$

Resilience to Errors

We measure the performance of Grover Search and QSVT Search algorithm under a Depolarizing Error Channel for input space $N = 2^3$, shown in figure 9. Note that depolarizing error introduces a random error in between each operation on a qubit. These errors are injected in a form of bit-flips (X error), phase-flips (Z error) or both at the same time (Y error). It's operation on a density operator (ρ) is given as :

$$E(\rho) = (1 - \lambda)\rho + \lambda \mathrm{Tr}[\rho]\frac{I}{2^n}$$

Here λ is the error parameter. Since QSVT Search, requires extra ancillae, for block encoding and control phase operations, we expect the the noise to scale much worse using QSVT than in Grover Search, inspite of circuit depth being smaller in QSVT Search (till $N = 2^5$).

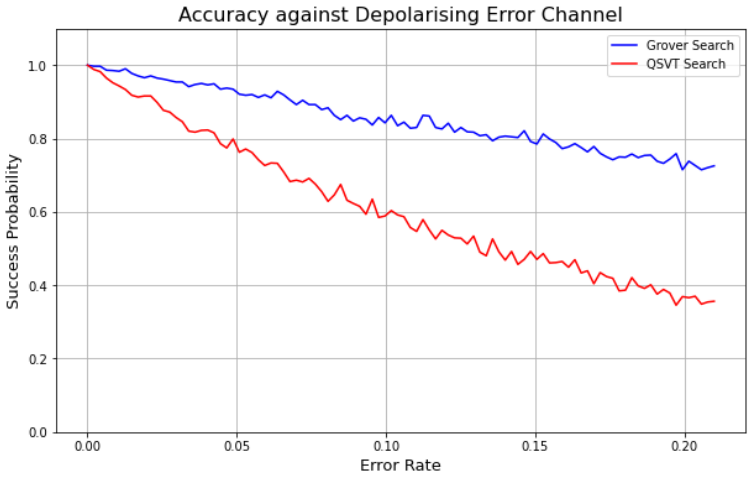

Figure 9: Performance over Depolarizing Error Channel

5 Eigenphase Estimation Problem

Given a unitary U, with eigenstates $|\psi\rangle$ and eigenvalue ($e^{2\pi i\phi}$), Quantum Phase Estimation problem aims to extract the eigenphase ϕ of the unitary operator U :

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \tag{23}$$

We assume the eigenphase ϕ has a binary representation up to m bits:

$$\phi = 0.\phi_1\phi_2\ldots\phi_m = \sum_j 2^{-j}\phi_j \tag{24}$$

Therefore, the bits upto which we can extract the eigenphase (most to least significant) determines the algorithm's accuracy.

5.1 QFT based Phase Estimation from Shor's Algorithm

This QPE algorithm uses two registers: a control register and a target register. The control register consists of m qubits prepared in a superposition state, and the target register holds the eigenvector $|\psi\rangle$ of the unitary operator U. Given the ability to implement controlled-U and powers of U, QPE algorithm estimates the phase upto m bits of accuracy for m qubits in control register.

5.1.1 Algorithm Overview:

1. **Initialization:** Initialize the control register into a superposition state (Hadamard) and target register as eigenstate $|\psi\rangle$:

$$|\psi\rangle|0\rangle^{\otimes m} \rightarrow |\psi\rangle\frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)^{\otimes m}$$

2. **Controlled Sequence of U^{2^j}:** Apply a controlled sequence operation, i.e., U^{2^j} controlled on the j-th control qubit. This encodes the phase into control qubits:

$$|\psi\rangle\frac{1}{\sqrt{2^m}}\left(|0\rangle+e^{2\pi i 0.\phi_1\phi_2\cdots\phi_m}|1\rangle\right)\otimes\left(|0\rangle+e^{2\pi i 0.\phi_2\phi_3\cdots\phi_m}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle+e^{2\pi i 0.\phi_m}|1\rangle\right)$$

3. **Inverse Quantum Fourier Transform:** To extract the phase bits, we apply the inverse Quantum Fourier Transform ($\text{QFT}^\dagger$), which is essentially controlled inverse rotations(CROT gates), i.e. for the j_{th} qubit we apply inverse rotations controlled on the previous qubits $(j+1, j+2..., m)$. The resulting circuit encodes the binary phase ϕ into control register as (dropping $|\psi\rangle$ as it remains unchanged in target): :

$$\text{Apply Inverse QFT } \rightarrow |\phi_m\rangle\otimes|\phi_{m-1}\rangle\otimes\cdots\otimes|\phi_1\rangle$$

Note: A CROT gate controlled on qubit k applies a phase rotation of $-2\pi\times(1/2^k)$ if the controlling qubit is in the state $|1\rangle$.

4. **Measurement:** Measure the control qubits in computational basis to recover the bits of phase ϕ (in reverse order).

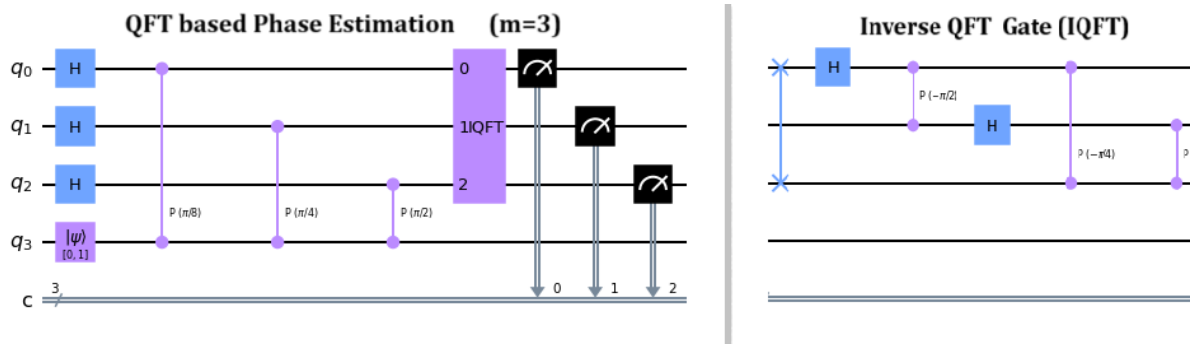

Figure 10: QFT-based Phase Estimation with $\phi = \frac{1}{8}$; requiring m=3

5.2 Iterative Phase Estimation (Kitaev's Algorithm)

Originally given by Kitaev, this algorithm restructures how phase is extracted requiring only a single ancilla qubit as control. If functions iteratively with the number of iterations equal to bits of our eigenphase (m). Here, instead of using Qubit-Controlled rotations(CROT Gates) like in Shor's QPE algorithm, we condition the rotation gates on classical bits from measurements in previous iterations, bypassing the whole Inverse QFT regime.
Note: A CROT gate controlled on qubit k applies a phase rotation of $-2\pi \times (1/2^k)$ if the controlling qubit is in the state $|1\rangle$.

5.2.1 Algorithm Overview:

Initialization: Initialize ancilla into a superposition state (Hadamard) and target register as eigenstate $|\psi\rangle$ of U :

$$|0\rangle |\psi\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |\psi\rangle$$

Iterate from j= m-1 to 0 : The transformation at the j-th step in Quantum Phase Estimation is depicted as follows:

1. **Apply Controlled-U^{2^j}** :

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |\psi\rangle \rightarrow \frac{|0\rangle + e^{2\pi i(2^j)\phi}|1\rangle}{\sqrt{2}} \otimes |\psi\rangle = \frac{|0\rangle + e^{2\pi i(0.\phi_j\phi_{j+1}...\phi_m)}|1\rangle}{\sqrt{2}} \otimes |\psi\rangle$$

   This encodes phase of '$(0.\phi_j\phi_{j+1}...\phi_m)$' into our control ancilla. Note this is not a single bit, and we need an inverse rotation of '$-(0.0\phi_{j+1}\phi_{j+2}...\phi_m)$'.

2. **Inverse Rotations controlled on classical bits** : From measurement outcomes of previous iterations, we have calculated phase upto $j + 1_{th}$. So we can directly

apply a single inverse rotation of '-$(0.0\phi_{j+1}\phi_{j+2}...\phi_m)$' or in angular terms $(-2\pi\frac{\phi_{j+1}}{2^2} - 2\pi\frac{\phi_{j+2}}{2^3}....)$.

$$\frac{|0\rangle + e^{2\pi i(0.0\phi_{j+1}\phi_{j+2}...\phi_m)}|1\rangle}{\sqrt{2}} \otimes |\psi\rangle \rightarrow \frac{|0\rangle + e^{2\pi i(0.\phi_j\phi_{j+1}...\phi_m)-2\pi i(0.0\phi_{j+1}...\phi_m)}|1\rangle}{\sqrt{2}} \otimes |\psi\rangle$$

Resulting in: $|\phi_j\rangle \otimes |\psi\rangle$

So for j_{th} control qubit, instead of applying $(m-j-1)$ controlled rotation gates conditioned on $(m-j-1)$ qubits, we apply these gates conditioned on classical bits. In Qiskit, this is implemented easily via **c_if() gate**

3. **Measure and reset control ancilla:** The measurement at j_{th} step stores the phase bit ϕ_j into the classical register. In the end, we reset the control ancilla which is recycled for next iteration.

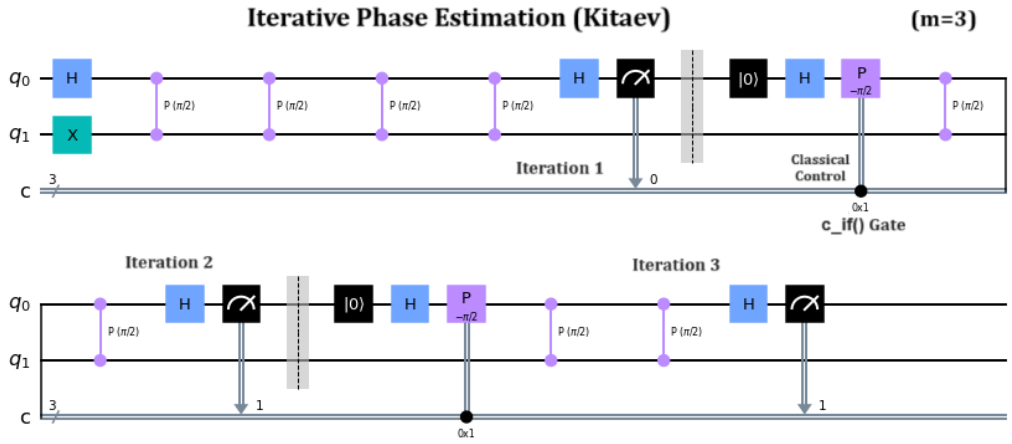

Figure 11: Iterative Phase Estimation Ciruit with $\phi = \frac{1}{8}$, with m=3 iterations

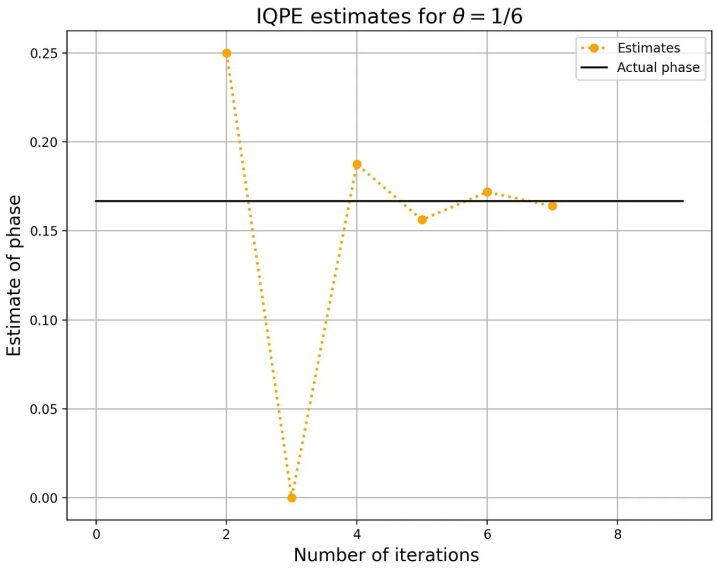

Figure 12: Accuracy bits computed against Number of Iterations. Approaches correct phase value at m=7 which is $\phi = \frac{1}{6} = 0.0010101$

Variation from QPE Algorithm:

- Iterative QPE requires a single ancilla for control-U gates which is reset after each iteration, whereas in QFT based algorithm we need m ancilla qubits in control register to estimate phase upto m bits of accuracy.
- We bypass the whole Inverse QFT routine comprising of inverse rotations dependent on other control qubits(CROT gates) with just a single inverse rotation gate as we have already extracted the phase upto the previous bit.

5.2.2 Hardware Implementation and Dynamic Circuits:

- Current quantum devices have limited capabilities in terms of number of coupled (connected) qubits. Since QFT based QPE algorithm requires control-U operations from each of the m control qubits, the circuit depth increases significantly, and so does noise in the circuit.
  Running on 5-qubit IBM architecture circuit depth contains 25 single-qubit and 18 CNOT operations, meanwhile the simulator depth was around 20. This is not the case in Iterative Phase estimation as U gates are controlled via a single ancilla which is reset and recycled after each iteration.
- On the other hand, since Iterative Phase Estimation Algorithm performs mid-circuit measurements after each iteration as well as phase gates controlled on classical bits, to implement the algorithm on a cloud-based Quantum device requires us to make use of **Dynamic Circuit Capability**.

It allows us to perform mid-circuit measurements, mid-circuit resets, and classical controlled operations within the coherence time of qubits. It can be easily implemented in Qiskit using *c_if()* gate and setting flag '*dynamic=True*' while compiling. Currently only a handful of backends support this feature on IBMQ

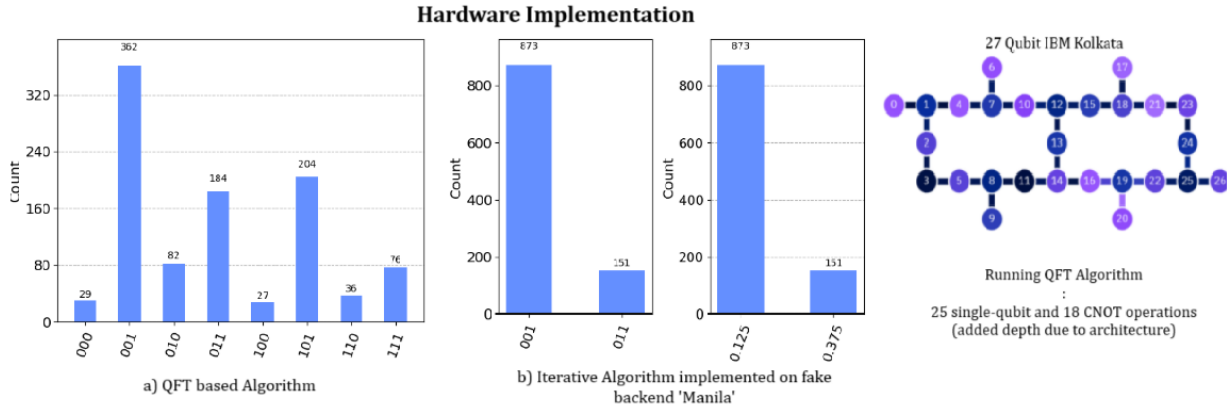

Figure 13: Implementation of QPE on backend 'IBM_Kolkata' and IQPE on fake backend 'IBM_Manila'. Note that currently only certain backends support Dynamic Circuits

5.3 QSVT Phase Estimation

QSVT Implementation for phase estimation is similar to Kitaev's Iterative Algorithm [7]. We encode a paramater θ into our block-encoded unitary and use QSVT upon Kitaev's Iterative Algorithm to extract ϕ bit by bit, and updating the parameter θ from single qubit measurement each iteration. We construct our block encoding in a way such that the singular values encode the bits of ϕ till the current iteration and using the value of θ from the previous iteration, we iteratively extract bits of ϕ into θ until $|\theta - \phi| < \epsilon$ where ϵ defines the accuracy limit and thus the number of iterations.

Variation from Kitaev's Algorithm : Note that here, instead of using classical controlled rotations, we use a QSVT Transform (namely sign() function) to extract the bits after measurements at each iteration, which is then fed to the parameter θ for next iteration. Below is the overview of the algorithm.

Block Encoding W(θ): . In each iteration, we construct a block encoding of U whose singular values encode the least significant bits of ϕ as well as the current value of our parameter θ. For j_{th} iteration of the algorithm, the Block Encoded Matrix is given as :

$$W_j(\theta) = \frac{1}{2}\begin{pmatrix} I + e^{-2\pi i\theta}U^{2^j} & I - e^{-2\pi i\theta}U^{2^j} \\ I - e^{-2\pi i\theta}U^{2^j} & I + e^{-2\pi i\theta}U^{2^j} \end{pmatrix} = \begin{pmatrix} A_j(\theta) & * \\ * & * \end{pmatrix} \tag{25}$$

where, $A_j(\theta) := \frac{1}{2}(I + e^{-2\pi i\theta}U^{2^j})$

Singular Values : At the j_{th} iteration, the singular values get encoded into the block matrix $\mathrm{W}_j(\theta)$ as :

$$\sigma_j := |\cos\left(\pi(2^j\phi - \theta)\right)| = \cos\left(\pi(0.\phi_{j+1}\phi_{j+2}\ldots\phi_m - \theta)\right) \tag{26}$$

Transformation: The polynomial transform to be applied onto the singular values of σ_j's of $W_j(\theta)$ is

$$P(\sigma_j) = \Theta\left(\frac{1}{\sqrt{2}} - x\right) \qquad \Theta \text{ is sign function} \tag{27}$$

$$\text{such that } P(\sigma_i) = \begin{cases} 1, & \text{if } \sigma_j = 0 \\ -1, & \text{if } \sigma_j = 1 \end{cases}$$

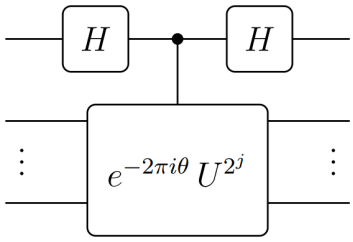

Figure 14: Block Encoding for Phase Estimation. Note parameter θ is updated after each iteration (Acts like a classical control)

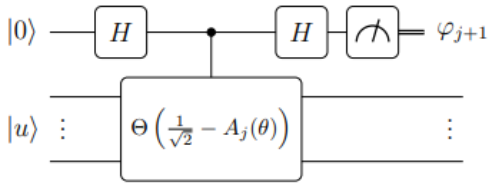

Figure 15: Feedback circuit : j^{th} iteration

5.3.1 Algorithm Overview:

Input: We assume access to an oracle that performs a controlled-U^{2^j} operation, eigenstate of U for which to determine the eigenphase, and m + 1 ancilla qubits (or 1 ancilla qubit that is reused (m times) to measure the phase upto m bits of accuracy.

Initialization: Initialize $\theta = 0$ and iterate from $j = m - 1$ to 0 (from last bits to first). We start with the target register in eigenstate ψ.

Iterative Process: Starting from $j = m - 1$ we iterate down to $j = 0$:

1. For $j = m - 1$

   - We apply the block-encoding of U^{2m-1} inside $W_0(0)$ for which the singular value σ_{m-1} stores the least significant bit ϕ_m:

     $$\sigma_{m-1} = \cos(\pi(0.\phi_m)) = \begin{cases} 1, & \text{if } \phi_m = 0 \\ 0, & \text{if } \phi_m = 1 \end{cases}$$

- Applying our QSVT transform $P(x) = \Theta\left(\frac{1}{\sqrt{2}} - x\right)$ with $x = \sigma_{m-1}$ and running the circuit in figure below to extract ϕ_m:

$$\phi_m = \frac{1}{2}\left(1 + \Theta\left(\sqrt{\frac{1}{2}} - \sigma_{m-1}\right)\right) \tag{28}$$

- We measure the ancilla(for QSVT) and update θ=$0.\phi_m$, completing the $j = m-1$ iteration.

2. For $j = m - 2$:

   - We set $\theta \rightarrow \frac{\theta}{2} = 0.0\phi_m$. This parallels the controlled inverse rotations in QFT and Iterative Algorith.
   - Again, applying our block-encoded unitary $W_{m-2}(\theta = 0.0\phi_m)$ with singular value encoded as :

$$\sigma_{m-2} = |\cos(\pi(0.\phi_{m-1}\phi_m - \theta))| = |\cos(\pi(0.\phi_{m-1}\phi_m - 0.0\phi_{m-1}))| \tag{29}$$

   - QSVT transform $P(x) = \Theta\left(\frac{1}{\sqrt{2}} - x\right)$ with $x = \sigma_{m-2}$ to extract the next bit :

$$\phi_{m-1} = \frac{1}{2}\left(1 + \Theta\left(\sqrt{\frac{1}{2}} - \sigma_{m-2}\right)\right) \tag{30}$$

   - Measurement the ancilla to get ϕ_{m-1}, from which we set $\theta \ = \phi_{m-1}\phi_m$.

3. Similarly we repeat step 2 for all iterations till j=0, with the phase getting encoded into our classical parameter θ.

Output: By iteratively performing QSVT Transform with the feedback of θ back into the circuit, we effectively evaluate the phase as $\theta = \phi_1\phi_2 \ldots \phi_{m-1}\phi_m$.

QSP Phase Angles: We use the QSPPack module [4] to estimate the phase angles for sgn(x).

5.3.2 Circuit Depth and Scalability

A comparison of circuit depths for QFT, Iterative and Kitaev's Phase Estimation is shown in Figure 18
Note that mean approximation error (ϵ) within the interval [-1,1] satisfies $\epsilon < 0.1$ only for $d >= 19$.

Since we apply our QSVT transform at each iteration, we estimate the circuit depth to scale $\approx d * m$ (where d is the degree of approximating polynomial, and m is the number of iterations).

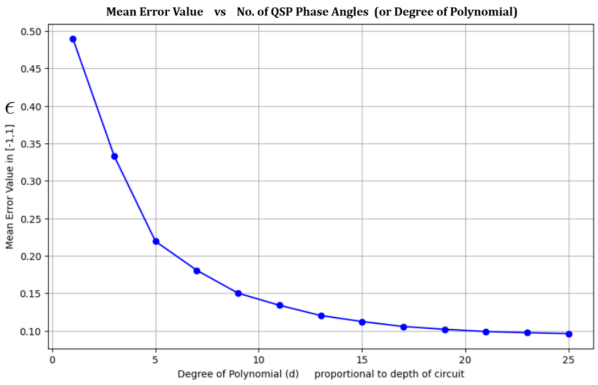

Figure 16: Accuracy for Sign Transform against degree of polynomial approximation (or number of QSP Phases)

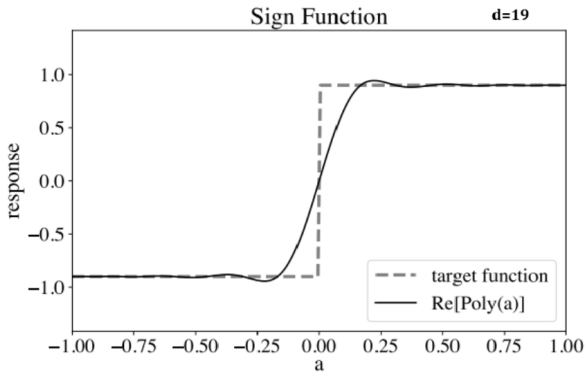

Figure 17: Sign(x) in domain [-1,1], degree (d)= 19

Figure 16 compares the degree of polynomial(d) used to approximate the sign(x) function transform against it's accuracy.

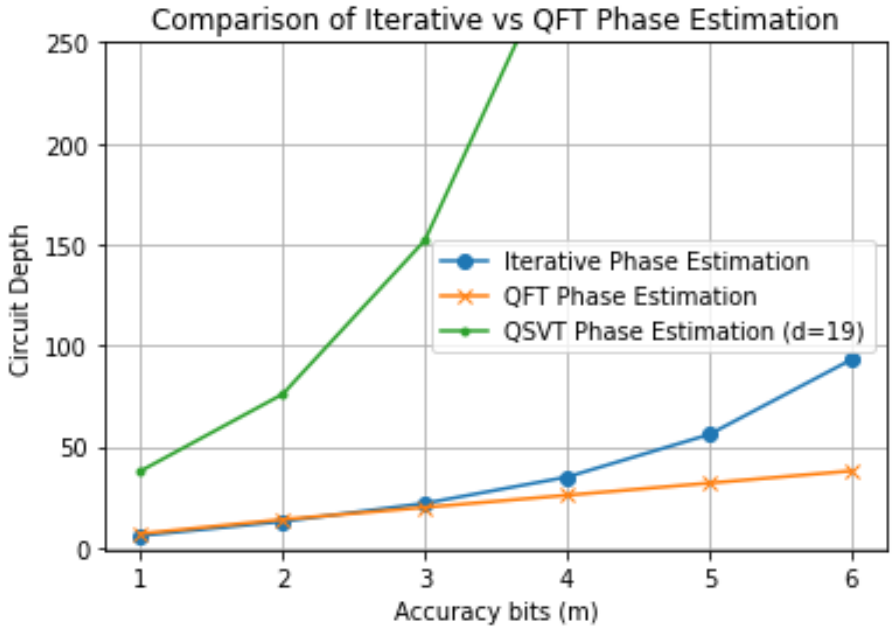

Figure 18: Circuit Depth Comparison vs No. of phase bits (m)

5.3.3 Estimating Phase for T gate

QSVT circuit with block encoding for $\Pi/4$ gate and corresponding $m = 2$ iterations. Block encoding is compiled within W and $W^\dagger$, and interleaved with QSP Phases for the QSVT Transform, incorporating a total of 21 QSP Phase angles, giving us a very deep circuit. Figure 22 shows the circuit used for a single QSVT Iteration.

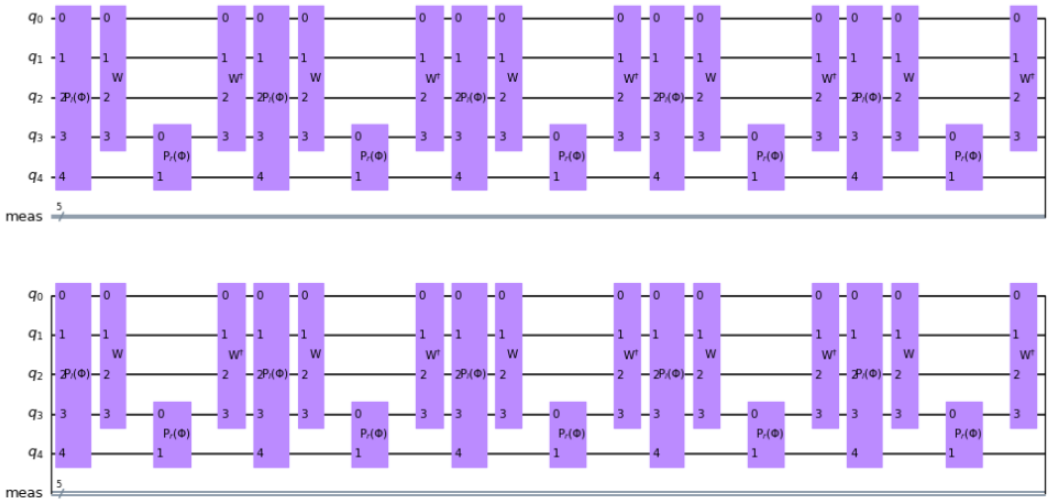

Figure 19: Circuit for Iteration 1 in QSVT Phase Estimation with 21 QSP Phase Angles for our transform. Note that the construction is inverted, and $q0$ is prepared in the eigenstate ψ

5.3.4 Noise Analysis for Phase Estimation

We see that Kitaev's Iterative algorithm is much more resilience to noise. Iterative Phase estimation requires a single ancilla qubit with rotations condititoned on classical bits rather than qubits.

Meanwhile, the QSVT algorithm performs poorly because even for single iteration, the circuit depth is fairly large given degree of approximating polynomial to sgn(x) d=19.
An analysis of QSVT Phase Estimation against QFT-based algorithm for bit-flip, phase-flip and depolarising error models was done by *Ijaz et al* [8]., as shown in figure 21

Therefore, in terms of performance against noise and errors, we conclude :
Iterative Phase Estimation > QFT-based Algorithm > QSVT Phase Estimation

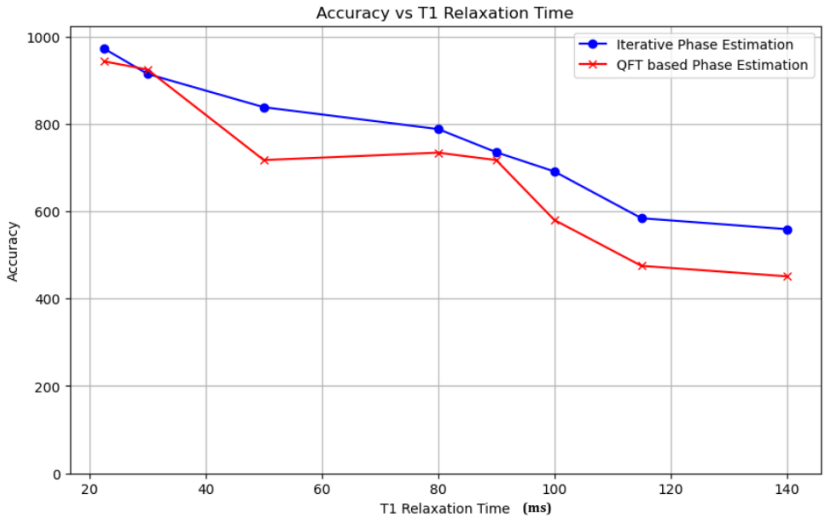

Figure 20: Performance against Thermal Errors (T1 Relaxation Time (in ms) , i,e. time in which state decays from $|1\rangle$ to $|0\rangle$)

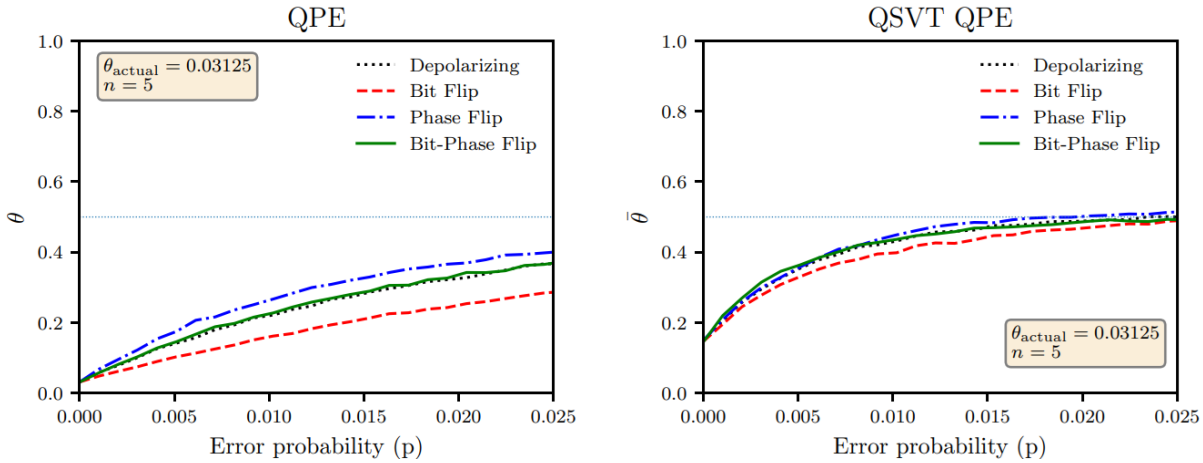

Figure 21: Comparison of performance against bit-flip, phase-flip and depolarizing errosr for QSVT and QFT based Phase Estimation Algorithms. Figure from Ijaz et al [8]

The reason why Kitaev's algorithm performs better than QFT and QSVT Algorithm in simulations is :
In Kitaev's algorithm, each iteration we applies a Phase gate $P(-\pi/2)$ conditioned on the classical result for previous iteration, as compared to QFT Phase Estimation where we use qubit controlled Rotation (CROT) gates , thus requiring extra qubits in control register. Meanwhile QSVT Phase Estimation uses sign function tranformation over Kitaev's Iterative Method (requiring QSP Phase angles of large degree since sign function itself is not a polynomial)

Also, Iterative Phase Estimation is advantageous in implementation because of the c_if()

Gate in Qiskit which lets us perform quantum operations controlled on classical measurements, and within the coherence time of qubits.

Interplay between the algorithms

If we decompose the QSVT circuit, and eliminate parameter θ from our consideration and let the QSVT sequence length be simply d = 1 for each iterations. We can see the circuit is akin to iterative algorithm, minus the classical control. Further, we observe that the control-U^{2^j} operations (z-basis) commute with the rotational operations, which are basically z-rotations. So we can sort of slide the commuting gates gates inin the circuit. The resulting circuit is equivalent to QFT based Phase Estimation, dropping the signal rotation operations, showing us the interplay betweent the algorithms.

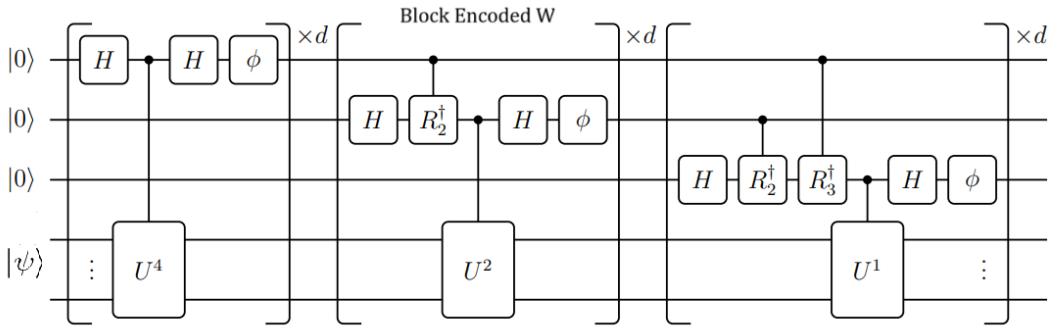

Figure 22: 3-qubit Phase Estimation : If we take d=1, we see the interplay between the algorithms

5.4 Application in Order Finding and Factoring

Since we have the eigenphase calculated, QSVT Phase Estimation can be directly applied in the order-finding subroutine, and thus factoring [1]. In the factoring problem, we aim to find the prime factors of a composite number N. The oracle U is defined such that:

$$U\left|\psi\right\rangle = \left|x\psi \mod N\right\rangle$$

for some $x < N$. This U has eigenvalues of the form $e^{2\pi is/r}$, where r is the order of U (i.e. $x^r \equiv 1 \mod N$) and $s \in \{0, 1, \ldots, r-1\}$. The goal is to determine the eigenphase s/r which will be rational and from which r can be found using the continued fractions algorithm. Once r is known, a factor of N can be extracted, and is the whole procedure describing the order-finding subroutine [1].

Note: In addition, in the factoring problem, as given by Shor, we do not have direct access to a particular eigenstate $||\psi_s\rangle$, but instead can prepare a uniform superposition of eigenstates $\frac{1}{\sqrt{2}}\sum|\psi_s\rangle$ as preparing $||\psi_s\rangle$ requires that we know r.

5.4.1 Time Complexity of QSVT Factoring

Therefore, the phase estimation algorithm begins with the superposition state $\frac{1}{\sqrt{2}}\sum|\psi_s\rangle$ and converges to a single eigenstate $||\psi_s\rangle$ at the end of the algorithm. To see this, we note that each measurement of the ancilla qubit will restrict the state to be a superposition over eigenstates whose eigenvalues are consistent with the measurement results thus far (i.e. the eigenphases whose least significant bits agree with the measurement results). Because the eigenvalues are not degenerate(usually true), the state at the end of the algorithm will be an eigenstate, sayψ_s, whose eigenphase is $\theta \approx \frac{s}{r}$. If θ obeys

$$|\frac{s}{r} - \theta| <= \frac{1}{2r^2} \tag{31}$$

then we can determine the fraction $\frac{s}{r}$ by applying the continued fractions algorithm to θ, as given in [1]. Considering these, QSVT based Phase Estimation has been described to be able to factor a number N in $\approx logNlog(logN)$ time [6]. Whereas Shor Algorithm is also polylog in N.

6 Hamiltonian Simulation

Given a Hamiltonian H that represents the energy structure of a quantum system, we want to simulate the the dynamics (time evolution) of a quantum state $|\psi(0)\rangle$ under this Hamiltonian over time t. The objective is to efficiently approximate the Time Evolution of Hamiltonian H in Heisenberg Picture:

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle.$$

Simulating the time evolution of a quantum system is essential for understanding complex quantum phenomena, and has applications in various fields. Classical computers struggle with this task due to the exponential growth of the Hilbert space with system size. Quantum computers, thus inherently offer a promising solution by efficiently simulating these large-scale quantum systems.

Note : It is not straightforward to evaluate the time evolution of a matrix, especially for larger quantum systems. Methods like Trotterization, (specifically the Suzuki-Trotter decomposition) are useful for approximating the time evolution of such complex Hamiltonians (approximating H as a sum of non-commuting operators) but require higher order decompositions are required for more accuracy.

QSVT on the other hand could be advantageous here, as we can directly apply an approximating polynomial transform, provided we can block encode the Hamiltonian, thus bypassing the need for decomposing the Hamiltonian into its components.

6.1 QSVT for Hamiltonian Simulation:

Since H is Hermitian, its eigenvalues and singular values coincide. If $|\lambda_i\rangle$'s represent the eigenvectors of Hamiltonian H, the QSVT Hamiltonian Simulation can be summarised as the transformation:

$$\begin{aligned}\text{QSVT}(U) \to \sum P(\sigma_i)|\lambda_i\rangle &= \sum P(\lambda_i)|\lambda_i\rangle \\ &= P(H,t) \approx e^{-iHt}\end{aligned}$$

where P represents the polynomial function approximating e^{-ixt}

Block Encoding:
We assume access to Hamiltonian H, for which we require a unitary block encoding, necessary because quantum gates and QSVT operate with unitaries. However, it is important to note that such a unitary block encoding is only realizable if $\|H\| \leq 1$. In general, if $\|H\|$ exceeds 1, we determine a scaling factor $\alpha \geq \|H\|$ to construct a unitary block encoding of $\frac{H}{\alpha}$.

Thus, we can equivalently simulate the time evolution of system under the rescaled Hamiltonian $H' = \frac{H}{\alpha}$ for time scaled as $\tau = t\alpha$. The equivalence holds as :

$$e^{-iH'\tau} = e^{-i\left(\frac{H}{\alpha}\right)(\alpha t)} = e^{-iHt} \tag{32}$$

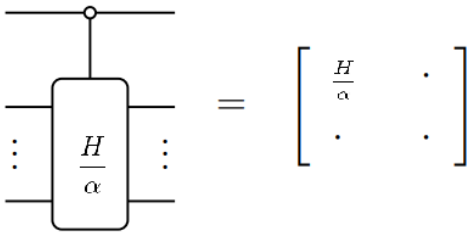

Figure 23: Unitary Block encoding for Hermitian H into U of a larger subspace

Singluar Values: Since H is hermitian and can be made positive semi definite by appropriate scaling and block encoding (e.g. $\frac{1}{2}(\frac{H}{\alpha} + I)$, the singular values are simply the enigenvalues λ_i's of the Hamiltonian H, given by eigenvalue equation

$$H\,|\psi\rangle = \lambda\,|\psi\rangle$$

Transform: The transform $p(x) = e^{-ixt}$, with t as a parameter rather than a variable, the exponential function e^{-ixt} does not have definite parity and therefore doe not satisfy Processing (QSP) constraints. To circumvent this issue, QSVT can be applied twice, with an even polynomial approximation to $\cos(xt)$, and an odd polynomial approximation to $\sin(xt)$. This allows to reconstruct the time evolution operator as:

$$\cos(\mathrm{SV})(Ht) - i\sin(\mathrm{SV})(Ht) = e^{-iHt},$$

Jacobi-Anger expansion:

The even and odd polynomial approximations can be effectively formulated using the Jacobi-Anger expansion as:

$$\cos(xt) = J_0(t) + 2\sum_{k=1}^{\infty}(-1)^k J_{2k}(t)T_{2k}(x),$$

$$\sin(xt) = 2\sum_{k=0}^{\infty}(-1)^k J_{2k+1}(t)T_{2k+1}(x),$$

where $J_i(x)$ denotes the Bessel function of the first kind of order i, and $T_i(x)$ represents the Chebyshev polynomial of the first kind of order i.

QSP Phase Angles: Truncating these series at a sufficiently large index provides fairly accurate approximations for $\cos(xt)$ and $\sin(xt)$, with the coefficients determining QSP Phase angles for the transform.

We use the QSPPack module given in A to obtain the QSP Phases. For cosine and sine, we first determine Chebyshev Polynomial approximation upto degree d and then run the optimiziation algorithm to determine our QSP Phase angles for $\cos(xt)$ and $\sin(xt)$ transforms.

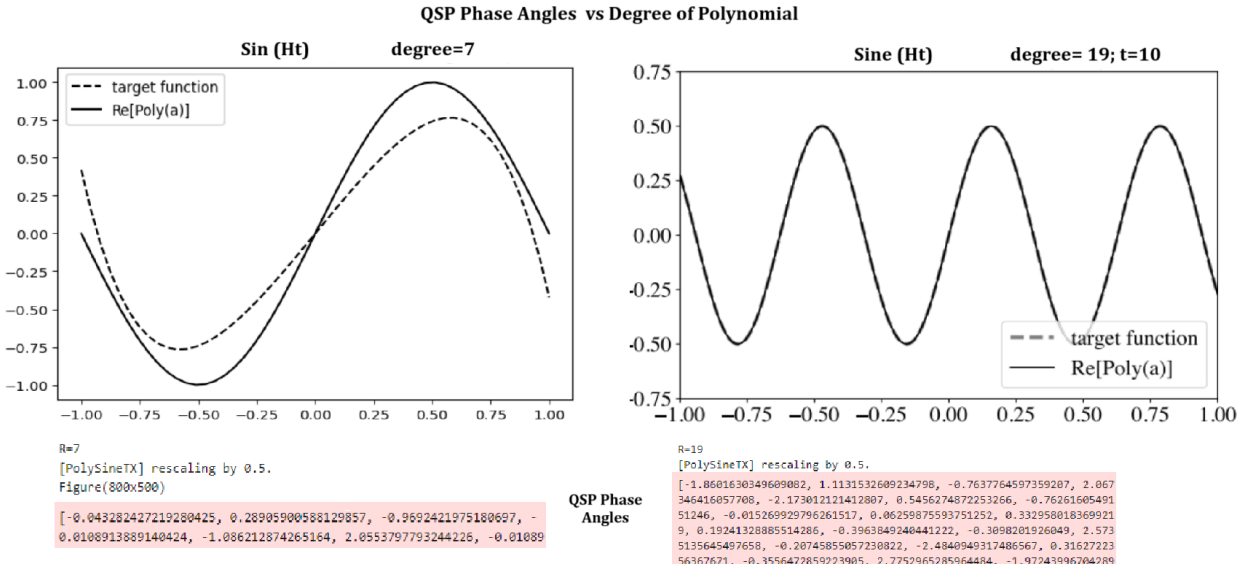

Figure 24: QSP Phase Angles against degree of approximations.Angles calculated via optimisation from QSPPack Module

6.1.1 Algorithm Overview:

- **Input:** Assuming access to problem Hamiltonian H, desired time t an error tolerance ϵ and scaling factor $\alpha \geq \|H\|$
- **Output:** A block encoded ϵ-approximation of our transform e^{-iHt}
- **Depth and Complexity:** Circuit depth $\propto 2 * d$ (d is degree of Laurent Expansion (QSPPack)) due to both cos and sin QSVT Transforms
- **Procedure:**
  - Prepare a unitary block encoding of $\frac{H}{\alpha}$.
  - Apply QSVT twice with QSP phases for sine and cosine, truncated at index k, in terms of Jacobi-Anger Expansion. We use QSPPACK to extract QSP phases.
  - Resulting circuit approximates our transform cos((Ht) and sin((Ht).

6.1.2 Implementation for a Single Qubit System :

With the complexixities involved in implementing block-encoding for a random Hamiltonian H/α) scaled with alpha, we demonstrate the time-evolution of for a trivial single-qubit gate (Z gate). The block encoding is given as :

$$BlockEnc(Z) == \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{33}$$

The QSP Phase angles are evaluated for time t=5 upto degree 10 and 11 for cos(xt) and sine(xt) respectively.

The time evolution for a Pauli-Z gate is the Pauli rotation gate:

$$U(t) = e^{-iZt} = R_Z(2t) \tag{34}$$

Thus we can directly compare the performance for the algorithm. Note that we have a factor of 2 in RHS, as we adhere to Qiskit notations for implementation [5].

Using Pauli-Time Evolution operator, we get the statevector as $cos(5) + isin(5)$ as expected but the QSVT-estimate is close only wrt cosine component of statevector as can be seen in figure 25

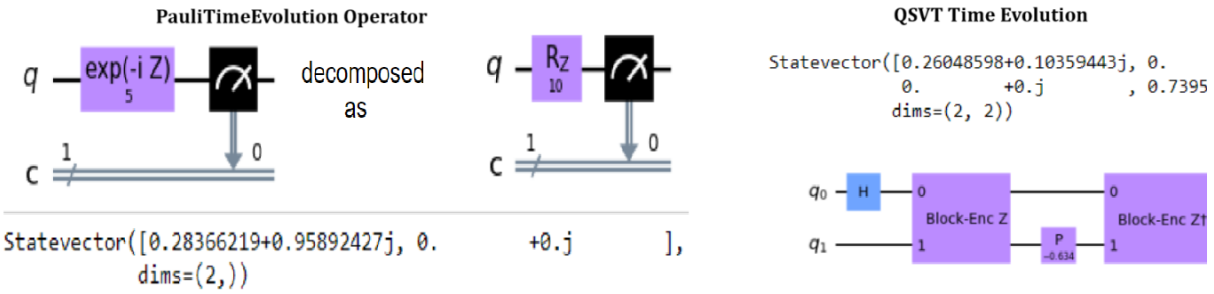

Figure 25: QSVT Time Evolution for Z Gate. Note that statevector should correspond to $cos(5) + isin(5)$

6.1.3 Caveats:

Norm Constraints : Construction of Block Encoding for arbitrary Hamiltonians is not as straightforward and can be inherently complex, as we require For a Hamiltonian $\|H\| \leq 1$ which is not always the case, especially those in unnormalized or large-scale systems. We instead block encode a scaled Hamiltonian and apply QSVT to this scaled H with appropriately scaled time τ.

LCU Implementation and challenges : Implementing the Linear Combination of Unitaries to simulate e^{-iHt} introduces its own set of challenges. The addition of two unitaries, specifically $\cos(Ht)$ and $-i\sin(Ht)$, does not naturally result in another unitary operation. While each component $\cos(Ht)$ and $\sin(Ht)$ can be individually approximated as unitary, their linear combination, required to form e^{-iHt}, is not inherently unitary.
To combine $\cos(Ht)$ and $-i\sin(Ht)$ into the exponential form e^{-iHt}, LCU employs an ancillary system and further block encoding in our circuit. This introduces additional layers of complexity

Although a number of block encodings are shown to be possible as given in [6], and is currently a topic of research.

6.2 Ground State Preparation:

QSVT time evolution can also be employed in determining the lowest energy state of a Hamiltonian system. We can show that combining QSVT Hamiltonian simulation with an the imaginary time evolution operator prepares our Hamiltonian in it's ground state.

Imaginary Time Evolution Operator The imaginary time evolution operator transforms the conventional time evolution operator e^{-iHt} by substituting time t with imaginary time $\tau = it$, resulting in the operator:

$$e^{-iHt} \xrightarrow[t \to -i\tau]{} e^{-H\tau}.$$

This operator evolves a quantum state $|\Psi(t)\rangle$ under imaginary time, modifying the state's amplitude based on the energy eigenvalues of the Hamiltonian. Say if we are given a state $\Psi(\tau)$), the overlap of this evolved state $|\Psi(\tau)\rangle$ with an eigenstate $|\lambda\rangle$ of the Hamiltonian is given by:

$$\langle X(\tau)|\lambda\rangle = \langle\lambda|X\rangle e^{-\lambda\tau}.$$

Here, λ represents the eigenvalues associated with $|\lambda\rangle$.

Preparing the Ground State As τ increases, the amplitude of the component of the state vector corresponding to the lowest eigenvalue λ (the ground state) becomes exponentially dominant ($e^{-\lambda\tau} \to \infty$). This is due to the exponential decay factor $e^{-\lambda\tau}$, which suppresses the contributions of states with higher energy relative to the ground state.

Projection to the Ground State Repeated application of the imaginary time evolution operator effectively filters out higher energy states, enhancing the amplitude of the ground state component in the quantum state. This selective amplification assists in converging towards the ground state and even find the minimum energy configuration of the system.

[9] describes the procedure for ground state preparation of a Hamiltonian, with the method similar to our QSVT based Time Evolution of Hamiltonian. It also includes a toy simulation on Pauli-X operator, to achieve it's ground state $|-\rangle$.

7 Conclusion and Future Direction

We conclude the study by reiterating QSVT as truly a *Grand Unification of Quantum algorithms*, given the possible applications from a single framework.

1. **Quantum Search Problem :** QSVT Search currently does not outperform Grover search, for large input space, although it does exhibit shorter circuit depth for input space size $< 2^5$. It is worth mentioning that the QSP Phases we worked with for higher n values, may not be the most optimal and further work is required on efficient and optimal derivation of QSP Phases for our search transform.

2. **Quantum Phase Estimation**: QSVT Phase Estimation buids up on Kitaev's Iterative algorithm but for each iteration, the circuit depth is considerably high to implement the sign(x) transform (non-polynomial). Meanwhile Iterative Phase Estimation uses rotations controlled on classical bits. It is especially significant as Qiskit implementation of c_if() gate does exactly that and within the coherence time of qubits.

3. **Hamiltonian Simulation:** QSVT has a promising application in implementing Time Evolution transforms of Hamiltonians, provided we can block encode such Hamiltonians into a Unitary. If obtained, even complex Hamiltonians, that usually require trotterization, can be directly exponentiated with this transform since we have computed the QSP Phases for it. One more pitfall in QSVT Simulation is implementing the Linear Combination of $Sin(Ht) + icos(Ht)$ transform, which again requires block encoding.

Possible Future Directions:

There is ample room for future research in this area. We can look at application of QSVT based Phase Estimation into factoring and Ground State preparation using QSVT Hamiltonian Simulation, and their algorithmic implementation and time and qubit complexities. Notably also, various quantum algorithms have not yet been constructed from QSVT-based subroutines, such as for variational algorithms, and possible function approximation transforms. Moreover effficient techniques and support from Quantum Computing communities like Qiskit or Pennylane for implementing QSVT Transforms and complex Block Encoding would be extremely beneficial.

7.1 Github Repository:

The code used for producing the results as discussed in the project is available on Github. We have mainly used Qiskit as our platform, with some work done on Pennylane as well, given it's support for Block Encoding and working with QSP Phases. Approriate citations have been acknowledged and mentioned in Bibliography.

Appendix

A QSPPACK Phase Solver

This module finds QSP Phases for a given Chebyshev Polynomial by using an optimization algorithm.
QSP Theorem states that there exists a set of phase factors $\phi := (\phi_0, \ldots, \phi_d)$ within the range $[-\pi, \pi)$ such that the unitary transformation $U_\phi(x)$ can be described by:

$$U_\phi(x) = e^{i\phi_0\sigma_z} \prod_{j=1}^{d} \left(W(x)e^{i\phi_j\sigma_z}\right) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix},$$

where $W(x)$ represents the block encoding of the transformation.
Note the constraints on obtained polynomial $P(x)$:

- **Parity Check**: The polynomial $P(x)$ must have parity ($d \mod 2$).
- **Normalization**: The polynomials must satisfy the normalization condition:

$$|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1 \quad \forall x \in [-1, 1].$$

To implement a smooth function $F(x)$ using QSP, first, we approximate it using a polynomial $f(x) \approx F(x)$ which satisfies QSP Constraints, especially if $F(x)$ is not inherently polynomial. This is usually taken to be Chebyshev Polynomial

Fourier-Chebyshev Expansion

For a real smooth function F on the interval $[-1, 1]$, the polynomial approximation is:

$$F(x) \approx f(x) = \sum_{j=0}^{d} c_j T_j(x),$$

where $T_j(x)$ are Chebyshev polynomials of the first kind, and the coefficients c_j are determined by running a separate optimization with respect to L^∞ norm:

$$f = \arg \min_{f \in R[x], \deg(f) \leq d} \max_{x \in [a,b]} |F(x) - f(x)|.$$

where we have restricted we can the interval of approximation to be a subset $[a, b] \in [-1, 1]$

Computing QSP Phase Factors

Once the polynomial $f(x)$ is obtained, QSPPack runs an optimization algorithm ((L-BFGS), which is a second order optimization algorithm to solve for the QSP Phases $[\Phi]$

- **Initialization**: We start with an initial guess for the phase factors: $\phi_0, \phi_1, \ldots, \phi_d$.

- **Optimization**: Minimizing the mean-squared loss function $L(\phi)$ defined by the difference between the real part of the QSP output and $f(x)$, using numerical optimization techniques. QSPPack uses limited memory BFGS algorithm

$$L(\phi) = \frac{1}{d}\sum_{j=1}^{d} \left|\mathrm{Re}[\langle 0|U_\phi(x_j)|0\rangle] - f(x_j)\right|^2,$$

where $\{x_j\}$ are sampled points within the interval $[-1, 1]$.

Considerations: Note that QSP phase angles are not always unique and the optimization process might not always converge to a global minimum, depending on the initial guess.

B Proving QSVT Theorem

Theorem : (Quantum Signal Processing in $SU(2)$)

[2] Let $k \in N$; there exists a set of phase factors $\Phi = \{\phi_0, \phi_1, \ldots, \phi_k\} \in R^{k+1}$ such that for all $x \in [-1, 1]$:

$$e^{i\phi_0\sigma_z}\prod_{j=1}^{k}\left(W(x)e^{i\phi_j\sigma_z}\right) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix},$$

if and only if $P, Q \in C[x]$ such that

(i) $\deg(P) \leq k$ and $\deg(Q) \leq k-1$,

(ii) P has parity-$(k \mod 2)$ and Q has parity-$(k-1 \mod 2)$,

(iii) $\forall x \in [-1, 1] : |P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$.

Proof

For the $k = 0$ case, the unitary on the left-hand side of equation (3) is $e^{i\phi_0\sigma_z}$, so that $P \equiv e^{i\phi 0}$ and $Q \equiv 0$ satisfy properties (i)-(iii). We prove (i)-(ii) by induction. Suppose for $k-1$ we have

$$e^{i\phi_0\sigma_z}\prod_{j=1}^{k-1}\left(W(x)e^{i\phi_j\sigma_z}\right) = \begin{pmatrix} \tilde{P}(x) & i\tilde{Q}(x)\sqrt{1-x^2} \\ i\tilde{Q}^*(x)\sqrt{1-x^2} & \tilde{P}^*(x) \end{pmatrix},$$

where $\tilde{P}, \tilde{Q} \in C[x]$ satisfy (i)-(ii). Then,

$$e^{i\phi_0\sigma_z}\prod_{j=1}^{k}\left(W(x)e^{i\phi_j\sigma_z}\right) = \begin{pmatrix} \tilde{P}(x) & i\tilde{Q}(x)\sqrt{1-x^2} \\ i\tilde{Q}^*(x)\sqrt{1-x^2} & \tilde{P}^*(x) \end{pmatrix}\begin{pmatrix} e^{i\phi_k x} & ie^{-i\phi_k}\sqrt{1-x^2} \\ ie^{i\phi_k}\sqrt{1-x^2} & e^{-i\phi_k x} \end{pmatrix}$$

$$= \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix},$$

where

$$P(x) := e^{i\phi_k}\left(x\tilde{P}(x) + (x^2-1)\tilde{Q}(x)\right)$$

and

$$Q(x) := e^{-i\phi_k}\left(x\tilde{Q}(x) + \tilde{P}(x)\right)\sqrt{1-x^2}.$$

It is easy to see that P, Q satisfy (i)-(ii). Finally note that the left-hand side of is a product of unitaries in the starting equation, therefore the right hand side is unitary too, which implies (iii), completing the proof

C Geometric Interpretation for Grover Search

To see how Grover Iterate works, we again consider qubitization of our search problem. Our initial search state $|s\rangle$ can be expressed as a linear combination of target state $|t\rangle$ and a superposition of all non-solution states $|t^\perp\rangle$. Since $|t\rangle$ and $|t^\perp\rangle$ are mutually exclusive sets because an item cannot be valid and not valid, the states are orthogonal. Both states form the orthogonal basis of a plane in the vector space.

$$|s\rangle = \frac{1}{\sqrt{N}}|t\rangle + \sqrt{\frac{N-1}{N}}|t^\perp\rangle \tag{35}$$

Grover's algorithm, after the first application of Hadamard to every qubit, starts with this uniform superposition of all states and can be represented in qubitized 2-D plane. Note that the probability of obtaining a correct result when measuring from the equal superposition is just $|\langle \mathrm{t}|\mathrm{s}\rangle|^2 = 1/N$, which is what we would expect from a random guess.

The phase orace U_F adds a negative phase to any solution to the search problem. Therefore, it can be written as a reflection about the $|t^\perp\rangle$ axis.

$$U_F = R_{|t^\perp\rangle} = 2|t^\perp\rangle\langle t^\perp| - \mathcal{I}$$

Similarly, we know that the Grover diffusion operation $-H^{\otimes n}U_\phi H^{\otimes n}$ is also a reflection about the state $|s\rangle$:

$$-H^{\otimes n}U_\phi H^{\otimes n} = 2|\mathrm{s}\rangle\langle \mathrm{s}| - \mathcal{I} = R_{|\mathrm{s}\rangle}$$

The combined effect of each Grover iteration is a counterclockwise rotation of an angle θ. Since θ is just the angle between $|s\rangle$ and $|t^\perp\rangle$ states, we use the scalar product to find the angle. Since $\cos\theta = \langle s|\,|t^\perp\rangle$, we get :

$$\theta = \arccos\langle s|\,|t^\perp\rangle = \arccos\left(\sqrt{\frac{N-1}{N}}\right)$$

After each Grover Iteration, the angle between the state of the register and the target $|t\rangle$ decreases, resulting in a higher probability of measuring a valid result. After k iterations, the angle becomes :

$$\langle s|\,|t^\perp\rangle = \frac{\pi}{2} - \theta - k2\theta = \frac{\pi}{2} - (2k+1)\theta$$

And the success probabiltiy is given as:

$$P(\text{success}) = \cos^2(|s\rangle \langle t^{\perp}|) = \sin^2\left[(2k+1)\arccos\left(\sqrt{\frac{N-1}{N}}\right)\right]$$

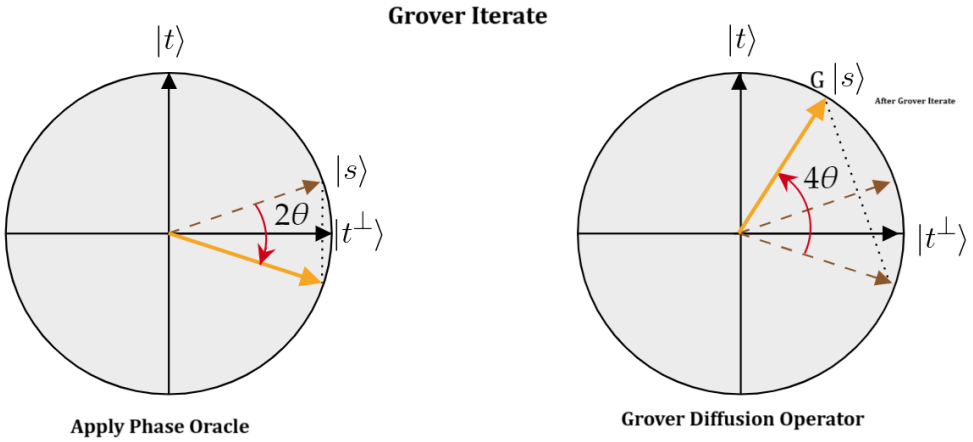

Figure 26: Geometrical Interpretation of Grover Iterate

From this, we can derive Optimal number of iterations to be : $N_{\text{optimal}} = \left\lfloor \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \right\rfloor$ to achieve a rotation of π on qubitized 2D plane. Note that we need to ensure we do not overshoot the target state, if we exceed N_{optimal}

References

[1] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[2] András Gilyén. *Quantum singular value transformation & its algorithmic applications*. PhD thesis, University of Amsterdam, 2019.

[3] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. *Quantum Singular Value Transformation and beyond: Exponential Improvements for Quantum Matrix Arithmetics*, page 193–204. Association for Computing Machinery, New York, NY, USA, 2019.

[4] qspack. qspack/qsppack, March 2021. Original date: 2020-02-24T08:11:30Z. [Online]. Available: https://github.com/qspack/QSPPACK.

[5] Qiskit Documentation. Qiskit: An open-source framework for quantum computing, 2023.

[6] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4), December 2021.

[7] Miroslav Dobšíček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin. Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark. *Phys. Rev. A*, 76:030306, Sep 2007.

[8] Muhammad Abdullah Ijaz and Muhammad Faryad. Noise analysis of grover and phase estimation algorithms implemented as quantum singular value transformations for a small number of noisy qubits. *Scientific Reports*, 13(1):20144, 2023.

[9] Charles Marteau. Ground state preparation via qubitization, 2023.

[10] Mauser M. Bartu Bisgin, 2021. QSVT In QISKIT - Qiskit 2021 Hackathon Winning Project.

[11] Harshit Gupta. Iterative Quantum Phase Estimation—QPE algorithms — medium.com, 2021. [Accessed 16-05-2024].

[12] Patrick Rall. Faster coherent quantum algorithms for phase, energy, and amplitude estimation. *Quantum*, 5:566, October 2021.

[13] Sean Greenaway, William Pol, and Sukin Sim. A case study against qsvt: assessment of quantum phase estimation improved by signal processing techniques, 2024.

[14] Kiichiro Toyoizumi, Naoki Yamamoto, and Kazuo Hoshino. Hamiltonian simulation using the quantum singular-value transformation: Complexity analysis and application to the linearized vlasov-poisson equation. *Phys. Rev. A*, 109:012430, Jan 2024.

[15] Lukas Burgholzer and Robert Wille. Towards verification of dynamic quantum circuits, 06 2021.