Quantum Phase Estimation to Find Eigenvalues and Eigenvectors for Implementing PCA

Team Members

1. Paksaran Kongkaew
2. Settawut Chaithurdthum
3. Inkaphol Siriwongwattana

Project Goal and Relevance

This project addresses the computational inefficiency of classical Principal Component Analysis (PCA) when applied to high-dimensional datasets. As dimensionality increases, traditional PCA faces significant challenges:

1. Computational complexity becomes prohibitive at $O(mn^2 + n^3)$ for m samples with n features, making it impractical to analyze very high-dimensional data.

2. Memory requirements grow quadratically with the number of features due to the need to store and process the full $n \times n$ covariance matrix.

3. The process cannot be efficiently parallelized on classical hardware, leading to performance bottlenecks for time-sensitive applications.

We aim to overcome these limitations by implementing Quantum Principal Component Analysis using Quantum Phase Estimation. This approach has theoretical potential to reduce computational complexity to $O(\log(mn))$, offering exponential speedup for large datasets and enabling analysis of dimensionality regimes currently intractable with classical methods.

Our project will quantify this potential advantage, demonstrate a working implementation, and identify the specific conditions and data characteristics where quantum approaches provide meaningful benefits over classical PCA.

Data Preprocessing and Quantum Encoding

The first step in the quantum principal component analysis (qPCA) pipeline is to preprocess the classical dataset and encode it into a quantum-compatible format. In classical PCA,

data is typically centered and used to compute a covariance matrix for eigendecomposition. In the quantum version, however, the classical data vectors are mapped into quantum states to enable quantum algorithms to process them efficiently.

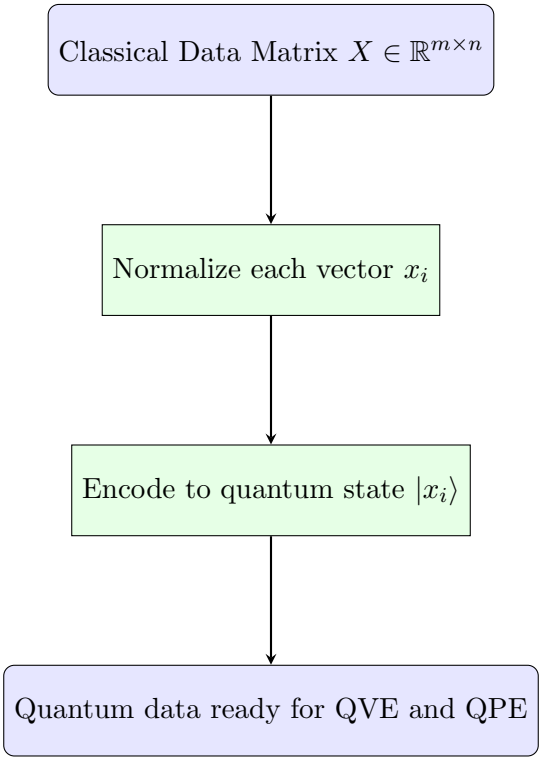
Given a dataset with m samples and n features, we first normalize each classical data vector $x_i \in \mathbb{R}^n$ to ensure that it can be represented as a valid quantum state. This normalized vector is then embedded into a quantum system as a state $|x_i\rangle$, typically using an amplitude encoding scheme where the components of x_i correspond to the amplitudes of the quantum state:

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \quad \Rightarrow \quad |x_i\rangle = \sum_{j=1}^{n} x_{ij}|j\rangle.$$

Once all samples are encoded as quantum states, we define the **quantum covariance matrix** ρ as a density matrix composed of these quantum-encoded states:

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |x_i\rangle\langle x_i|.$$

This density matrix captures the statistical structure of the input data and serves as the Hermitian operator H in the Quantum Variational Eigensolver (QVE) process.



Algorithm/Theory Used

QVE (Quantum Variational Eigensolver) Definition

The Quantum Variational Eigensolver is a hybrid quantum-classical algorithm that approximates eigenvectors of Hermitian operators using a parametrized quantum circuit and a classical optimization loop. It is particularly well-suited for solving large-scale eigenvalue problems where classical diagonalization methods are computationally infeasible. In quantum machine learning and principal component analysis, QVE serves as a crucial preparatory step for quantum phase estimation by producing approximate eigenstates of high-dimensional quantum covariance matrices.

Mathematical Objective for QVE

The Quantum Variational Eigensolver (QVE) aims to approximate the eigenvectors and corresponding eigenvalues of a Hermitian matrix by leveraging a hybrid quantum-classical optimization process. A Hermitian matrix is a square matrix that is equal to its conjugate transpose and has the important properties of real eigenvalues and orthonormal eigenvectors, making it widely used in quantum mechanics, quantum chemistry, and data analysis. In the context of QVE, the problem is formulated as follows: given a Hermitian matrix H, the goal is to find a quantum state $|\psi\rangle$ such that:

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

where λ is the eigenvalue associated with the eigenvector $|\psi\rangle$.

Rather than directly diagonalizing the matrix, which is computationally expensive for large-scale problems, QVE prepares a parameterized quantum state $|\psi(\theta)\rangle$ using a quantum circuit with tunable parameters θ. The algorithm then evaluates the expectation value:

$$\langle\psi(\theta)|H|\psi(\theta)\rangle$$

and uses a classical optimization loop to iteratively adjust θ to minimize or maximize this value. According to the variational principle, the minimum of this expectation value corresponds to the lowest eigenvalue of H, and the optimized quantum state $|\psi(\theta)^*\rangle$ approximates the corresponding eigenvector.

In the context of quantum principal component analysis (qPCA), the Hermitian matrix H is a quantum covariance matrix, constructed as a density operator:

$$\rho = \frac{1}{m}\sum_{i=1}^{m} |x_i\rangle\langle x_i|$$

from a set of encoded classical data samples $\{x_i\}$. QVE is applied to this matrix to extract dominant eigenvectors, which serve as principal components. These approximate eigenvectors can then be passed to the Quantum Phase Estimation (QPE) algorithm to determine their associated eigenvalues with higher precision, completing the qPCA process.

The Process of QVE

1. **Define the Hermitian Matrix**
   The first step in the Quantum Variational Eigensolver (QVE) process is to define the Hermitian matrix whose eigenvectors and eigenvalues are to be approximated. In the context of quantum principal component analysis (qPCA), this matrix is referred to as the **quantum covariance matrix**.

   Instead of forming a classical $n \times n$ covariance matrix from raw data, the quantum version is constructed using quantum-encoded data samples. Each classical data vector x_i is encoded into a quantum state $|x_i\rangle$, and the covariance structure is captured by the following density matrix:

   $$\rho = \frac{1}{m} \sum_{i=1}^{m} |x_i\rangle\langle x_i|$$

   This density matrix ρ is Hermitian, positive semi-definite, and has a trace equal to one. These properties ensure that it can be used as the operator H in the QVE framework.

   Thus, in this step, we define ρ as our target Hermitian operator, and the QVE algorithm will proceed to find its dominant eigenvectors, which serve as quantum principal components of the dataset.

2. **Parametrized Quantum Circuit (Ansatz)**
   In this step, we build a quantum circuit called an **ansatz**. This circuit is used to generate trial quantum states that will help the algorithm find an approximate eigenvector of the matrix.

   The ansatz is flexible and includes parameters (represented as $\vec{\theta}$) that can be adjusted during the optimization process. When we apply the circuit $U(\vec{\theta})$ to the initial state $|0\rangle$, we get the trial state:

   $$|\psi(\vec{\theta})\rangle = U(\vec{\theta})|0\rangle$$

3. **Evaluate the Cost Function**
   Once we have prepared a trial quantum state using the ansatz, the next step is to evaluate how good this state is. This is done by computing the **expectation value** of the Hermitian matrix (in our case, the quantum covariance matrix ρ) with respect to the trial state $|\psi(\vec{\theta})\rangle$.

   The expectation value is given by the following expression:

   $$C(\vec{\theta}) = \langle\psi(\vec{\theta})|\rho|\psi(\vec{\theta})\rangle$$

   This quantity serves as the **cost function** for the optimization process. It tells us how close the current trial state is to being a true eigenvector of ρ.

The cost function is evaluated on a quantum computer using measurements, while the optimization of parameters $\vec{\theta}$ is performed on a classical computer. This combination forms a **hybrid quantum-classical loop**, which is the key idea behind the QVE algorithm.

4. **Optimize Parameters**
   Once the cost function has been evaluated, the next step is to improve the trial quantum state by adjusting the parameters of the ansatz circuit. This is done using a classical optimization algorithm that searches for the best set of parameters $\vec{\theta}$ that minimize or maximize the cost function:

$$C(\vec{\theta}) = \langle\psi(\vec{\theta})|\rho|\psi(\vec{\theta})\rangle$$

   In this project, we use the **Simultaneous Perturbation Stochastic Approximation (SPSA)** algorithm. SPSA is particularly suitable for variational quantum algorithms because it performs well under noisy conditions, does not require exact gradient information, and scales efficiently with the number of parameters.

   SPSA estimates the gradient of the cost function by perturbing all parameters simultaneously in random directions. Remarkably, it only needs two cost function evaluations per iteration, regardless of the dimension of $\vec{\theta}$, making it highly efficient for use with quantum hardware.

5. **Output**
   After the classical optimization process converges, the Quantum Variational Eigensolver (QVE) provides an approximate solution to the eigenvector problem for a given Hermitian matrix, typically the quantum covariance matrix ρ. This is achieved by preparing a parameterized quantum state $|\psi(\theta)\rangle$ using a variational ansatz and iteratively adjusting the parameters θ to minimize the expectation value:

$$C(\theta) = \langle\psi(\theta)|\rho|\psi(\theta)\rangle.$$

   According to the variational principle, the state $|\psi(\theta^*)\rangle$ that minimizes this cost function is the best approximation of the lowest eigenvector of ρ. Therefore, the output of QVE is:

$$|\psi(\theta^*)\rangle \approx |\phi_k\rangle,$$

   where $|\phi_k\rangle$ is the k-th eigenvector of the quantum covariance matrix. This eigenvector serves as a quantum representation of the principal component in qPCA. Unlike classical PCA, where the principal components are computed through matrix diagonalization, QVE efficiently finds these quantum principal components through a hybrid quantum-classical routine without full matrix decomposition. These approximate eigenvectors are essential for identifying dominant directions of variance in the data and are subsequently passed to the Quantum Phase Estimation (QPE) stage for high-precision eigenvalue extraction.

QPE (Quantum Phase Estimation) Definition

The Quantum Phase Estimation (QPE) algorithm is a fundamental quantum algorithm that extracts the eigenvalue (in the form of a phase) of a unitary operator given access to one of its eigenstates. It forms the core of many quantum algorithms, including factoring, Hamiltonian simulation, and quantum principal component analysis. QPE works by encoding the eigenvalue as a phase into a quantum register, then applies the Quantum Fourier Transform (QFT) to extract this phase through measurement.

In the context of quantum PCA, QPE is used to estimate the eigenvalues of the quantum covariance matrix, given approximate eigenvectors prepared by algorithms such as the Quantum Variational Eigensolver (QVE). This makes QPE an essential component for extracting the explained variance associated with each principal component.

Mathematical Objective for QPE

The Quantum Phase Estimation (QPE) algorithm is designed to estimate the eigenvalue of a unitary operator U given one of its eigenvectors $|\phi\rangle$. It is a fundamental quantum algorithm that plays a central role in a variety of quantum applications, including factoring, simulation, and dimensionality reduction.

QPE solves the following problem: given a unitary operator U and one of its eigenvectors $|\phi\rangle$, find the phase $\lambda \in [0, 1)$ such that:

$$U|\phi\rangle = e^{2\pi i\lambda}|\phi\rangle$$

This phase λ is encoded into the amplitudes of a quantum register via a sequence of controlled-U^{2^j} operations. The Quantum Fourier Transform (QFT) is then applied to extract the phase from this interference pattern, yielding a binary approximation of the eigenvalue with high precision.

In the context of quantum principal component analysis (qPCA), the operator U is defined as:

$$U = e^{2\pi i\rho}$$

where ρ is the quantum covariance matrix constructed from quantum-encoded data samples. The approximate eigenvector $|\phi_k\rangle$ of ρ, obtained from the Quantum Variational Eigensolver (QVE), is used as the input state for QPE. The algorithm then outputs the corresponding eigenvalue λ_k, which quantifies the variance captured by the k-th quantum principal component.

This combination of QVE and QPE enables quantum-enhanced extraction of both eigenvectors and high-precision eigenvalues from high-dimensional datasets.

The Process of QPE

1. **Prepare the Approximate Eigenvector**
   The first step in the Quantum Phase Estimation (QPE) process is to provide an eigen-

vector of the matrix whose eigenvalue we want to estimate. In the context of quantum principal component analysis (qPCA), this eigenvector is prepared using the Quantum Variational Eigensolver (QVE). The output of QVE approximates one of the eigenvectors of the quantum covariance matrix ρ:

$$|\phi_k\rangle \approx |v\rangle$$

This state will serve as input for QPE and will be used to encode the eigenvalue phase into a quantum register.

2. **Define the Unitary Operator**
   QPE estimates eigenvalues of *unitary* operators. To apply it to a Hermitian matrix ρ, we define a related unitary operator using the exponential map:

$$U = e^{2\pi i\rho}$$

This operator retains the same eigenvectors as ρ, while transforming its real eigenvalues into phases in the range $[0, 1)$, this being done by separating out the 2π out as a constant, leaving is with λ_k which a range of [0..1]. The eigenvalue relationship becomes:

$$U|\phi_k\rangle = e^{2\pi i\lambda_k}|\phi_k\rangle$$

Here, λ_k is the eigenvalue (phase) associated with the eigenvector $|\phi_k\rangle$.

3. **Hadamard and Controlled U Gate Application**
   Initially the system is initialized with n number of estimation qubits, and our $|v\rangle$. After which, the Hadamard gate is applied to all of the previously initialized estimation qubits, culminating in the state below.

$$|\psi\rangle = (|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle)|v\rangle$$

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)|v\rangle$$

After all of the estimation qubits within our system are suspended within superposition, we can start to apply the controlled U gates on to each of them. The process only being applied to the $|1\rangle$ as per the nature of the control gate; this results in a process which is similar to the classical $Ax = \lambda x$, storing the phase relating to the eigenvalue within the $|1\rangle$ state of each of the estimation qubits.

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{4\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi ij}|1\rangle)\right)|v\rangle$$

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{4\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi ij}|1\rangle)\right)|v\rangle$$

$$\cdots$$

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}(|0\rangle + e^{32\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{16\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{8\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{4\pi ij}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi ij}|1\rangle)\right)|v\rangle$$

$$\text{Set j} = 0.j_0j_1j_2j_3j_4 = \frac{j_0}{2} + \frac{j_1}{4} + \frac{j_2}{8} + \frac{j_3}{16} + \frac{j_4}{16}, \quad j_n \in \{0, 1\}$$

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}\left(|0\rangle + e^{32\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{16\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right)\right.$$
$$\otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{8\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{4\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right)\Bigg)$$
$$\otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right) \otimes |v\rangle$$

Since all of the j_n is an integer when the number removed from outside the parenthesis is applied to the inside. We can safely ignore them as all integers contribute to the global phase

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_4}{2}\right)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_3}{2}+\frac{j_4}{4}\right)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_2}{2}+\frac{j_3}{4}+\frac{j_4}{8}\right)}|1\rangle\right)\right)$$
$$\otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_1}{2}+\frac{j_2}{4}+\frac{j_3}{8}+\frac{j_4}{16}\right)}|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_0}{2}+\frac{j_1}{4}+\frac{j_2}{8}+\frac{j_3}{16}+\frac{j_4}{32}\right)}|1\rangle\right) \otimes |v\rangle$$

The resultant output is identical to if we were to apply a Quantum Fourier Transform process. Moreover, as the phase information has now been properly encoded into the resultant state, we are now able to apply the Inverse Quantum Fourier Transform.

4. **Apply the Inverse Quantum Fourier Transform**
   Once the control register contains phase information, the inverse Quantum Fourier Transform (QFT^{-1}) is applied. This operation transforms the frequency-like interference pattern into a binary representation of the eigenvalue:

$$\text{QFT}^{-1}\left(\sum_j e^{2\pi i\lambda_k j}|j\rangle\right) \longrightarrow \text{Binary form of } \lambda_k$$

   This step allows the eigenvalue phase to be read from the output qubits.

5. **Measure the Control Register**
   Finally, the control register is measured in the computational basis. The result is a binary approximation of the eigenvalue λ_k, which reflects the amount of variance captured by the corresponding quantum principal component. As the states in the quantum system take the form of binary fractions, we'll have to return it back to a real number.

$$\lambda_k \approx \text{Measurement result}$$
$$0.00101 = (0)2^{-1} + (0)2^{-2} + (1)2^{-3} + (0)2^{-4} + (1)2^{-5}$$

   As each subsequent position behind the decimal is multiply by 2^{-n} where n continually increases the further away it is from the decimal point. We are left with the result of λ_k as a real number.

Summary

In summary, the QPE algorithm allows us to extract the eigenvalue corresponding to a quantum-encoded eigenvector. When paired with QVE, which prepares the eigenvector, QPE completes the qPCA process by estimating the associated variance (eigenvalue). While this makes it possible to derive a vector and value solution, multiple iterations need to be run in QVE to produce a result close to the actual solution. Likewise, for the QPE process, multiple shots have to be taken to produce an acceptable distribution graph.

Implementation Details

The implementation of Quantum Principal Component Analysis follows the approach outlined in the "Towards An End-To-End Approach For Quantum Principal Component Analysis" paper, utilizing the QuPCA library to realize an end-to-end pipeline for quantum-assisted dimensionality reduction. This section details the implementation approach, focusing on how key challenges in quantum data encoding, phase estimation, eigenvector extraction, and result validation were addressed.

1 Development Environment and Dependencies

The implementation is built using the Qiskit SDK for quantum circuit development and execution. The QuPCA library provides the core functionality, with the central QPCA class orchestrating the entire workflow. This implementation relies on standard scientific Python libraries including NumPy and SciPy for mathematical operations and classical data processing.

2 Data Preparation and Encoding Implementation

For the initial data preparation step, the implementation normalizes input matrices by their trace, ensuring compatibility with quantum processing requirements. As described in the paper, the trace normalization is essential since quantum states must have unit norm.

Following the notation from the "Towards An End-To-End Approach" paper, the encoding step generates a quantum circuit that prepares states of the form:

$$|\psi_A\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}|i\rangle|j\rangle = \sum_{k=1}^{r} \sigma_k|u_k\rangle|u_k\rangle$$

where $A \in \mathbb{R}^{N\times N}$ is the normalized input matrix with tr[A] = 1, r is the rank of the matrix, σ_k are the singular values, and u_k are the singular vectors of A. The second equality follows from the Gram-Schmidt decomposition.

The QuPCA implementation offers two encoding methods:

1. Using Qiskit's StatePreparation class (optimized_qram=True)
2. A custom QRAM implementation for greater transparency (optimized_qram=False)

3 Unitary Transformation Implementation

As described in the paper, the implementation constructs a unitary transformation e^{-iAt} from the input matrix. This operator is fundamental for phase estimation, as it retains the same eigenvectors as the original matrix while encoding eigenvalues as phases.

4 Quantum Phase Estimation Implementation

The Quantum Phase Estimation implementation follows the standard approach as described in the paper. The circuit transforms the initial state as follows:

$$|0\rangle^E|\psi_A\rangle^M \xrightarrow{U_{PE}} \sum_{k=1}^{r} \sigma_k|\lambda_k\rangle^E|u_k\rangle|u_k\rangle$$

Where register E contains the eigenvalues encoded as binary fractions, and the eigenvectors are encoded in register M.

The implementation uses a configurable resolution parameter (rPE) that determines the number of qubits in the eigenvalue register. This parameter balances precision against circuit complexity, with higher resolution providing more accurate eigenvalue estimates.

5 State Vector Tomography Implementation

The state vector tomography process, which extracts eigenvectors from the quantum state, is implemented following the approach outlined in the paper. This involves two primary steps:

1. **Probability Estimation**: The implementation estimates probability amplitudes through repeated measurements of the quantum state. The probability for each basis state is determined by the frequency of corresponding measurement outcomes.

2. **Sign Estimation**: For determining the signs of amplitudes, the implementation uses the circuit design shown in Figure 2 of the paper. This involves creating two controlled operators:

   - U_x that prepares the state whose signs we want to determine
   - U_p that prepares a state using the estimated probability amplitudes

   After applying Hadamard gates and measurements, the relative frequencies of outcomes determine the most likely signs for each amplitude component.

The mathematical formulation involves preparing states of the form:

$$\frac{1}{\sqrt{2}}|0\rangle \sum_{i\in[d]} x_i|i\rangle + \frac{1}{\sqrt{2}}|1\rangle \sum_{i\in[d]} \sqrt{p_i}|i\rangle$$

And after applying a Hadamard gate to the control qubit, the state becomes:

$$\frac{1}{2}\sum_{i\in[d]}[(x_i + \sqrt{p_i})|0,i\rangle + (x_i - \sqrt{p_i})|1,i\rangle]$$

The StateVectorTomography class in the QuPCA.quantumUtilities module implements this process, with configurable parameters for the number of measurement shots and repetitions.

6 Eigenvalue and Eigenvector Extraction

After phase estimation and state tomography, the implementation includes algorithms to:

1. Identify eigenvalues from the phase register measurements
2. Apply eigenvalue thresholds to filter out small or noisy eigenvalues
3. Reconstruct eigenvectors from the tomography results
4. Verify results by reconstructing the original matrix

The QuPCA library provides functions for eigenvalue filtering and eigenvector reconstruction in its postprocessing utilities.

7 Benchmarking and Validation

The implementation includes benchmarking tools to validate the quantum results against classical computation. The spectral_benchmarking method in the QuPCA class provides comprehensive performance metrics, including:

1. Eigenvector accuracy comparison
2. Sign reconstruction validation
3. Eigenvalue precision assessment

These benchmarks help quantify the accuracy of the quantum approach compared to classical methods.

8 Implementation Challenges

The implementation addresses several key challenges inherent to quantum PCA:

1. **Non-Integer Eigenvalues**: The paper highlights that non-integer eigenvalues cannot be represented exactly in the phase register. The implementation handles this by using sufficient resolution in the phase estimation register to minimize errors.
2. **Statistical Measurement Errors**: Quantum tomography relies on statistical measurements, introducing noise in reconstructed states. The implementation mitigates this by allowing for configurable shot counts and repetitions.
3. **Sign Ambiguity and Visualization Flipping**: A fundamental challenge in quantum PCA implementation is the sign ambiguity in eigenvectors. As the paper demonstrates, eigenvectors are only defined up to a sign - multiplying an eigenvector by -1 gives an equally valid eigenvector for the same eigenvalue.

   This mathematical property manifests visually as "flipped" visualizations when comparing quantum and classical PCA results. For example, the paper shows in its experimental results that when comparing the eigenvectors from classical computation versus quantum reconstruction for a 2×2 covariance matrix, the quantum method might produce eigenvectors with opposite signs. In one example from the paper, the first eigenvector has opposite signs between quantum and classical results ([-0.668 0.726] vs [-0.707 0.707]).

   The implementation addresses this through sign-aware benchmarking that recognizes this mathematical equivalence. Rather than treating these flips as errors, the implementation properly evaluates both possible sign orientations during validation.
4. **Complex Amplitudes**: When eigenvalues cannot be represented exactly, complex amplitudes may appear in the reconstructed states. The implementation extends the tomography process to handle these complex values.

The implementation provides a robust foundation for exploring quantum advantages in principal component analysis, following the methodology presented in the "Towards An End-To-End Approach For Quantum Principal Component Analysis" paper and leveraging the functionality provided by the QuPCA library.

Results and Insights

Quantum Phase Estimation Error Analysis (VQE + QPE

The graph titled "Error vs QPE Shots" illustrates the relationship between measurement precision and error in Quantum Phase Estimation (QPE) implementation. This fundamental analysis demonstrates how increasing the number of measurement shots affects the accuracy of eigenvalue estimation in quantum PCA.

The graph reveals several key findings:

- The y-axis represents Mean Squared Error, ranging from approximately 1.95 to 2.25, measuring the deviation between quantum-estimated and theoretical eigenvalues.
- The x-axis displays the Number of QPE Shots on a logarithmic scale (10^1 to 10^2), representing the statistical sampling performed during quantum measurement.
- A clear inverse relationship exists between shot count and error magnitude, following an approximately logarithmic decay pattern.
- Specific data points demonstrate that:
  - At 10 shots, the mean squared error is approximately 2.25
  - At 100 shots, the error decreases to approximately 2.00
  - At 200 shots, the error further reduces to approximately 1.95

This relationship aligns with quantum measurement theory, The diminishing returns observed in error reduction highlight an important practical consideration: significant increases in shot count yield progressively smaller improvements in accuracy, creating a computational resource tradeoff that must be carefully balanced in quantum PCA implementations.

Eigenvector Sign Ambiguity in Visualization on local simulation

The PCA visualization comparison between classical and quantum implementations reveals an important mathematical property inherent to eigenvector computation. The apparent "flipping" of the projection space observed between the two methods is a direct consequence of eigenvector sign ambiguity.

Mathematically, if $\vec{v}$ is an eigenvector of a matrix A with eigenvalue λ, then $-\vec{v}$ is equally valid as an eigenvector for the same eigenvalue:

$$A\vec{v} = \lambda\vec{v} \implies A(-\vec{v}) = -\lambda\vec{v} = \lambda(-\vec{v}) \tag{1}$$

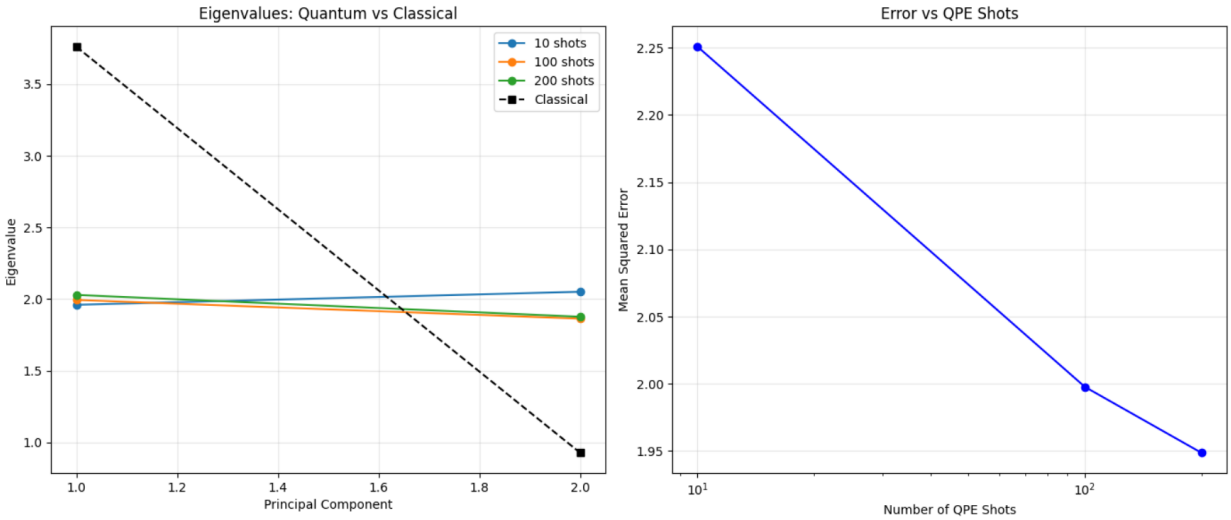


Figure 1: Error vs QPE shot Result

The quantum PCA algorithm, particularly during the tomography phase of reconstructing eigenvectors from measurements, has no inherent mechanism to select a consistent sign convention that matches classical algorithms. This results in some eigenvectors potentially having opposite signs between quantum and classical computations, which manifests visually as a reflection or rotation in the projection of data points onto principal components.

This phenomenon is not an error but rather a fundamental property of eigendecomposition that becomes apparent when comparing different computational approaches. The information content and separation capabilities of the principal components remain equivalent despite this apparent reflection in visualization space.

Resolution Impact on Eigenvector Reconstruction

The benchmark analysis of L2 error for eigenvector reconstruction reveals the relationship between QPE resolution (qubit count), shot count, and reconstruction accuracy across different eigenvalues. Four key observations emerge:

- Larger eigenvalues (0.727...) achieve lower reconstruction error with fewer shots compared to smaller eigenvalues, demonstrating the inherent precision bias in quantum phase estimation.
- Resolution effects vary by eigenvalue magnitude - lower resolutions (3 qubits) can suffice for dominant eigenvalues but become inadequate for smaller eigenvalues (0.036... and 0.005...).
- Resolution 8 (highest tested) shows dramatic error reduction for the largest eigenvalue as shots increase, but exhibits mixed performance for smaller eigenvalues, including non-monotonic behavior for the smallest eigenvalue (0.005...).

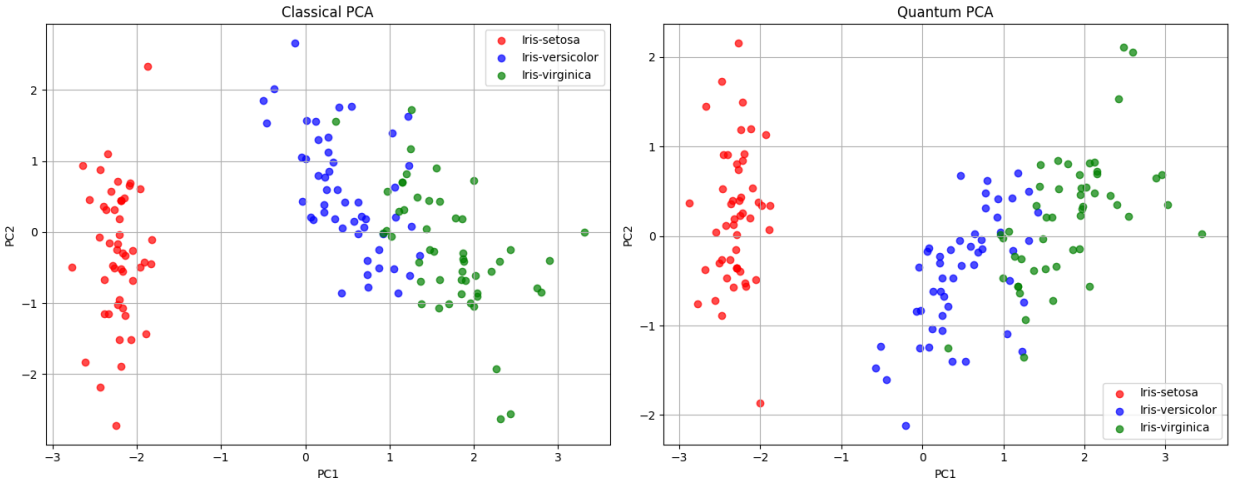


Figure 2: Classical PCA VS QuPCA Implementation Visualization on local simulation

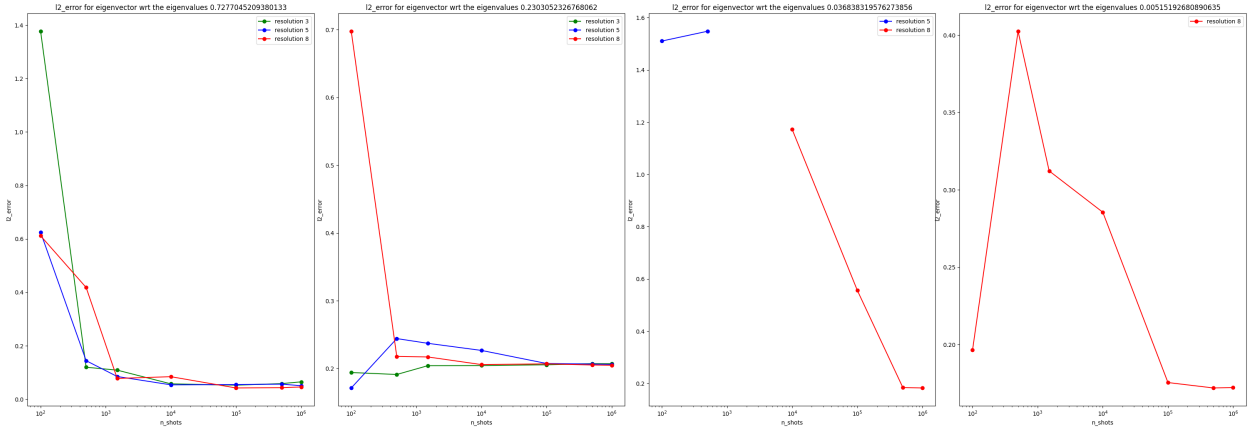


Figure 3: QuPCA Implementation resolution (n-qubit) benchmark on local simulation

- An optimal resolution exists for each eigenvalue, with excessive resolution potentially introducing circuit noise that counteracts theoretical precision benefits.
- The convergence rate differs significantly across eigenvalue magnitudes, with smaller eigenvalues requiring substantially more shots to achieve comparable accuracy.

These results illustrate a fundamental quantum resource allocation tradeoff: higher resolution circuits provide better theoretical precision but introduce more susceptibility to quantum noise effects, requiring careful parameter selection based on the specific eigenvalue spectrum being analyzed.