Quantum Databases and Quantum Machine Learning – How Far Can We Go on a Publicly Available Quantum Computer?

Raphael Weber, Mehmet Yesil, Ruedi Füchslin and Kurt Stockinger
Zurich University of Applied Sciences, Switzerland

Datalab Seminar
September 9, 2020
Winterthur, Switzerland
Outline

Quantum Supremacy - Kurt

Foundations of Quantum Computing - Ruedi

Quantum Database Search with Grover-Algorithm - Kurt

Quantum Machine Learning with HHL-Algorithm – Raphael/Mehmet

Conclusions – Kurt/Ruedi
Quantum supremacy using a programmable superconducting processor

Frank Arute, Kunal Arya, [...] John M. Martinis

Nature 574, 505–510 (2019) | Cite this article
762K Accesses | 282 Citations | 6004 Altmetric | Metrics

Abstract

The promise of quantum computers is that certain computational tasks might be executed exponentially faster on a quantum processor than on a classical processor\(^1\). A fundamental challenge is to build a high-fidelity processor capable of running quantum algorithms in an exponentially large computational space. Here we report the use of a processor with programmable superconducting qubits\(^2,3,4,5,6,7\) to create quantum states on 53 qubits, corresponding to a computational state-space of dimension \(2^{53}\) (about \(10^{16}\)). Measurements from repeated experiments sample the resulting probability distribution, which we verify using classical simulations. Our Sycamore processor takes about 200 seconds to sample one instance of a quantum circuit a million times—our benchmarks currently indicate that the equivalent task for a state-of-the-art classical supercomputer would take approximately 10,000 years. This dramatic increase in speed compared to all known classical algorithms is an experimental realization of quantum supremacy\(^8,9,10,11,12,13,14\) for this specific computational task, heralding a much-anticipated computing paradigm.

Fig. 1 | The Sycamore processor. a, Layout of processor, showing a rectangular array of 54 qubits (grey), each connected to its four nearest neighbours with couplers (blue). The inoperable qubit is outlined. b, Photograph of the Sycamore chip.
Some More Details Explained by Nature Video

NATURE VIDEO  •  05 NOVEMBER 2019

Quantum supremacy: A three minute guide

Elizabeth Gibney talks through the latest milestone in quantum computing.

https://youtu.be/vTYp5Kd9nMA
Outline

Quantum Supremacy

Foundations of Quantum Computing

Quantum Database Search with Grover-Algorithm

Quantum Machine Learning with HHL-Algorithm

Conclusions
Spin $\frac{1}{2}$: Besides position and momentum, each electron is characterized by a complex vector with two components.

- Some think of spin as angular momentum.
- A sometimes misleading analogy: Electrons are NOT charged rotating spheres.
A qubit $q$ is represented by a physical system that is described by:

- A two-dimensional complex vector

$$|\psi\rangle = \alpha |\downarrow\rangle + \beta |\uparrow\rangle, \quad \alpha, \beta \in \mathbb{C}$$

- One uses «bracket notation» $|\phi\rangle$. That looks very smart!

- We set $|\downarrow\rangle = |0\rangle$, $|\uparrow\rangle = |1\rangle$. Looks also smart!

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad \alpha, \beta \in \mathbb{C}$$

- One additional rule:

$$|\psi\rangle^2 = 1 \iff |\alpha|^2 + |\beta|^2 = 1$$

- A qubit lies on the surface of a (complex) sphere!
Coupled Qubits

\[ |\psi\rangle = c_0 |0\rangle + c_1 |1\rangle \]
\[ |c_0|^2 + |c_1|^2 = 1 \]

3 free real parameters

\[ |\psi\rangle = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle \]
\[ |c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1 \]

7 free real parameters

\[ |\psi\rangle = c_{000} |000\rangle + c_{001} |001\rangle + c_{010} |010\rangle + c_{011} |011\rangle + c_{100} |100\rangle + c_{101} |101\rangle + c_{110} |110\rangle + c_{111} |111\rangle \]
\[ |c_{000}|^2 + |c_{001}|^2 + |c_{010}|^2 + |c_{011}|^2 + |c_{100}|^2 + |c_{101}|^2 + |c_{110}|^2 + |c_{111}|^2 = 1 \]

15 free real parameters

\[ 3 + 3 \neq 7 \]
$n$ spin $\frac{1}{2}$ - particles produce quantum registers storing quantum states from a $2^n$ - dimensional vector space.
Boring, but Useful: Notation

• Spin ½ particle is represented by a 2dim – vector, with base |0>, |1>.

• A quantum register with two qubits is represented by a four dimensional vector in a space with a basis that is constructed via tensor products:

$$|b_0\rangle = |0\rangle \otimes |0\rangle, |b_1\rangle = |0\rangle \otimes |1\rangle, |b_2\rangle = |1\rangle \otimes |0\rangle, |b_3\rangle = |1\rangle \otimes |1\rangle$$

• In order to save space, we write

$$|a\rangle \otimes |b\rangle \otimes ... \otimes |z\rangle = |ab...z\rangle$$

• Why not go to the extreme:

$$|000\rangle = |0\rangle, |001\rangle = |1\rangle, |010\rangle = |2\rangle, ..., |111\rangle = |7\rangle$$
Where Comes the Weirdness From?

• We may accept an individual spin $\frac{1}{2}$ particle:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1, \quad \alpha, \beta \in \mathbb{C} \iff 3 \text{ free real numbers}$$

• We may even accept the coupling of two individual spin $\frac{1}{2}$ particles:

$$|\psi\rangle \otimes |\varphi\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$$

$$|\alpha|^2 + |\beta|^2 = 1 \wedge |\gamma|^2 + |\delta|^2 = 1, \quad \alpha, \beta, \gamma, \delta \in \mathbb{C} \iff 6 \text{ free real numbers}$$

• Problem: A general two – qubit register is given by

$$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$$

$$|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1 \Rightarrow 7 \text{ free parameters}$$
• **Entangled states** of a 2qubit – register are states that can’t be understood as the product of two individual spin-$\frac{1}{2}$ particles!

$$\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \neq (\alpha |0\rangle + \beta |1\rangle) \otimes (\gamma |0\rangle + \delta |1\rangle)$$

• States of a multi-qubit register that cannot be understood as the tensor product of individual qubits are called **entangled**.
• Thinking in terms of particles as small localized spheres is not compatible with entangled states.
• Entanglement: relevant for quantum cryptography.
Operations on Quantum Registers

- Quantum register: Vector $|\psi>$ in $2^n$ - dimensional space.
- Two types of operations are possible: **Unitary transformations** and **measurement**.
- Unitary transformation $U$: Rotates $|\psi>$. 
  - $||\psi||^2 = 1 \iff |U|\psi||^2 = 1$
  - $U^*U = 1$
  - There is an $H$ with $H^* = H$, such that $U = e^{iH}$
- Unitary transformations: reversible
- Measurement $M$ in a given basis $|b_0>, ..., |b_m>$

$$|\psi\rangle = \sum_{k=0}^{2^n-1} c_k |k\rangle \implies M|\psi\rangle = |K\rangle \text{ with probability } |c_K|^2$$

- Measurement: Irreversible.
- That is why $\sum_{k=0}^{2^n-1} |c_k|^2 = 1$
Quantum Circuits

- **Quantum computation (up to measurement):** series of unitary operations $U_i$ on an input state $|\psi_0>$ → Product of unitary operators.

\[
\text{Quantum computation on } |\psi_0\rangle = U_N U_{N-1} \ldots U_1 |\psi_0\rangle
\]

- **Quantum gate:** Physical realization of a unitary operation.
- **Quantum circuit:** A sequential series of gates.
- **Quantum computations are strictly sequential** → No loops!

Example of a quantum circuit.
Image from «Quantum Algorithm, Implementations for Beginners» by Coles, Eidenbenz, Pakin, Adedoyin
Quantum Power by Linearity

- Unitary operators are linear operators.

\[|\psi\rangle = \sum_{k=0}^{2^n-1} c_k |b_k\rangle \Rightarrow U|\psi\rangle = \sum_{k=0}^{2^n-1} c_k U|b_k\rangle\]

- Linear operators are good workers! Whether they act on \(|\psi\rangle\) or \(|\phi\rangle\), they always charge the same costs (amount of time)

\[|\psi\rangle = |00\rangle \quad |\phi\rangle = \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + |11\rangle)\]

- **Quantum parallelism:** A quantum operation acts on all basis vectors at once! \(\Rightarrow\) With one step, we act on \(2^n\) components!

- Linearity: We don’t produce a mishmash, but something that can be decomposed meaningfully.
How many gates do we have to learn?

Theorem: All relevant operations in quantum computing can be written as a series of so called Toffoli gates and measurement processes.

• (And, Yes, you’re right, this is a simplification but it’s enough for the exam.)

Toffoli – gate.

Half – adder built with Toffoli – gates.
Quantum Complexity

• Quantum algorithms contain a random aspect (measurement!)

• **BQP**: Bounded Error Quantum Polynomial Time

• A decision problem $D$ is in BQP,
  - there is a quantum algorithm $A$, such that if $D(x) = y, y \in \{0,1\}$, we have $A(x) = y$ with a probability higher than $\frac{1}{2}$.
  - $A$ can be implemented by a number of gates growing polynomially with the size of the input register.

• **In other words**: Maybe you have to apply a quantum algorithm several times, but by majority voting, you can get the correct answer with arbitrarily high probability.

• There is a classical analogue of BQP, namely BPP (bounded error probabilistic polynomial time).
What Can Be Done?

- Present state of discussion: **Quantum computers are computationally equivalent to Turing machines.**
- For some problems, QC is faster than TM.
- A very relevant problem: Are classical randomized algorithms equivalent to quantum algorithms. It is known: $BPP \subseteq BQP$.
- Present assumption:
  
  $BPP \neq BQP$

- This, because there is no classical analogue to entanglement.
- The relation between NP and BQP is not yet clear.
Searching a Database

- **Quantum database (example):** Tensor product of labels and entries.

**Data structure record:** \( |r\rangle = |\text{label}\rangle \otimes |\text{phone}\rangle \otimes |\text{auxbit}\rangle \)

\[
|r_0\rangle = |0\rangle \otimes |0355647372\rangle \otimes |q_0\rangle \\
|r_1\rangle = |1\rangle \otimes |0487245099\rangle \otimes |q_1\rangle \\
\vdots \\
|r_{455}\rangle = |455\rangle \otimes |0589347592\rangle \otimes |q_{455}\rangle \\
\vdots \\
|r_{1024}\rangle = |1023\rangle \otimes |117\rangle \otimes |q_{1023}\rangle \\
\]

**Quantum database** \( |D\rangle > : \)

\[
|D\rangle = \sum_k \alpha_k |r_k\rangle, \quad \sum_k |\alpha_k|^2 = 1
\]

What is the label of 058 934 75 92?

A data record is a base state in a high dimensional state, written as tensor product of partial records.

If implemented, the search time would be reduced from \( O(N) \) to \( O(\sqrt{N}) \) by Grover’s algorithm.
Sketch of Grover’s Search Algorithm

1. Prepare a state, in which all components have the same size and the entry states represent the content of the database.

\[
\left| \psi_{\text{start}} \right> = \frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^n-1} \left| k \right> \otimes \left| \text{entry}_k \right> \otimes (\left| 0 \right> - \left| 1 \right>) = \sum_{m=0}^{2^M-1} \alpha_m \left| m \right>
\]

2. Use the oracle and other operations to amplify the state with the searched entry \( N_s \).

3. Perform measurements to find the amplified state.

Grover’s secret: We assume an oracle \( f \) and a unitary operator \( U_f \):

\[
f(N) = \begin{cases} 
1 & \text{if } N = N_s \\
0 & \text{else}
\end{cases} \quad U_f = \begin{cases} 
- \left| L \right> \otimes \left| N \right> \otimes (\left| 0 \right> - \left| 1 \right>) & \text{if } f(N) = 1 \\
\left| L \right> \otimes \left| N \right> \otimes (\left| 0 \right> - \left| 1 \right>) & \text{else}
\end{cases}
\]

Here, the oracle is a simple logic function that can be implemented by a series of Toffoli – gates.
Sketch of Grover’s Search Algorithm

\[ U_f |D\rangle = U_f \frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^{n-1}} |k\rangle \otimes |N\rangle \otimes (|0\rangle - |1\rangle) \]

\[ = \frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^{n-1}} |k\rangle \otimes |N\rangle \otimes (|f(N)\rangle - |1 \oplus f(N)\rangle) \]

\[ = \frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^{n-1}} |k\rangle \otimes |N\rangle \otimes (-1)^{f(N)}(|0\rangle - |1\rangle) \]

\[ = \sum_{m=0}^{2^{M-1}} \alpha_m |m\rangle \quad \alpha_k \in \mathbb{C} \]

Flip the searched entry.

Mirror states at the size average

\[ U_m U_f |D\rangle = U_m \sum_{m=0}^{2^M-1} \alpha_m |m\rangle \]

\[ = \sum_{m=0}^{2^M-1} \left( 2 \sum_{j=0}^{2^M-1} \frac{\alpha_j}{2^M} - \alpha_m \right) |m\rangle \]
What We Didn’t Tell You

- The Grover – process is usually repeated several times.
- Can be done with multiple identical entries.
- But not too many times, or the algorithm breaks down.
- Measuring must be done cleverly.
- The algorithm is probabilistic.
- That $U_f$ and $U_m$ are unitary must be shown.
- …
Writing Proposals/Reports With QC?

- **Searching databases is certainly super - cool!** But can we do other reasonable things with QC? E.g. writing reports?
- **No!** Reason: There is no quantum Copy / Paste!

No – Cloning Theorem: There is no unitary operator $C$ that realizes the general copying of qubits onto a blank state $|b>$:

$$\nexists \ C \ \forall |\psi\rangle : \ C |\psi\rangle \otimes |b\rangle = |\psi\rangle \otimes |\psi\rangle$$

Fair enough! We don’t want to break the uncertainty principle!

If $C$ were a copy operator, it had to produce:

$$C(\alpha |0\rangle + \beta |1\rangle) \otimes |b\rangle = (\alpha |0\rangle + \beta |1\rangle) \otimes (\alpha |0\rangle + \beta |1\rangle)$$

Especially, it must hold:

$$C |0\rangle \otimes |b\rangle = |0\rangle \otimes |0\rangle, \ C |1\rangle \otimes |b\rangle = |1\rangle \otimes |1\rangle$$

But because $C$ is linear, it holds:

$$C(\alpha |0\rangle + \beta |1\rangle) \otimes |b\rangle = \alpha C |0\rangle \otimes |b\rangle + \beta C |1\rangle \otimes |b\rangle = \alpha |0\rangle \otimes |0\rangle + \beta |1\rangle \otimes |1\rangle$$

$\Rightarrow$ There is no linear unitary $C$!
Why QC at ZHAW?

- There are further interesting algorithm, e.g. **factorization**.
- Quantum computing is not yet in industry, but a closely related application, **quantum cryptography, can be bought off the shelf**.
- More and more start – ups and incubators care about QC
- Quantum information science is a very good way to teach quantum mechanics:
  - Very relevant quantum phenomena play a crucial role.
  - The mathematics is comparably simple: We only work with finite dimensional matrices.
Take Home Message

- Quantum registers store **superpositions of many basis states**.
- Some superpositions cannot be understood in terms of independent individual particles ➔ **entanglement** (important for cryptography)
- Quantum operations act on all basis states at once ➔ **quantum parallelism**!
- The relation between quantum and classical computational complexity is not **yet clear**, randomized algorithms are relevant.
- Classical randomized algorithms are probably **less powerful** than quantum algorithms, the latter exploiting entanglement.
- **Search** and **factorization** are potential applications.
- Quantum information enables studying and presenting quantum mechanics with limited mathematical effort.
Outline

Quantum Supremacy

Foundations of Quantum Computing

Quantum Database Search with Grover-Algorithm

Quantum Machine Learning with HHL-Algorithm

Conclusions
Quantum Database Search

● Given an **unordered set of items** (a database), find if value x is contained in this set.

● Search complexity:
  ○ **Traditional algorithm**: $O(n)$ where n is the number of items
  ○ **Quantum algorithm based on Grover**: $O(\sqrt{n})$

● Quantum algorithm has an exponential speedup over classical algorithm
Using Grover’s Algorithm for Database Search

\[ \text{Initialization} \xrightarrow{\text{Superposition}} H^n \xrightarrow{\text{Oracle}} U_f \xrightarrow{\text{Grover-Diffusion}} 2^{\frac{\log n}{2}} I_n \xrightarrow{\text{Measure}} \]

Repeat \( O(\sqrt{n}) \) times
Grover Algorithm - Step 1

- The procedure starts out in uniform superposition $|s\rangle$. The dashed line is the average amplitude. The pink bar is $p$. This diagram is in case $N = 4$.

Source: https://medium.com/swlh/grovers-algorithm-quantum-computing-1171e826bcb
Grover Algorithm - Step 2

- Apply the oracle reflection $U_f$ to state $|s\rangle$, flipping it negative. Note that the average amplitude has been lowered.
Grover Algorithm - Step 3

- Apply a reflection about the state $|s\rangle$ (the average amplitude). This completes the transformation, elevating $p$ while subsequently decreasing the other items.
Implementation with IBM Qiskit

Qiskit is an open source SDK for working with quantum computers at the level of pulses, circuits and algorithms.

Get started
Database Search - Some Code Snippets Using 2 qubits → 4 States

```python
# Define the number of n-qbits.
n = 2
# Create a Quantum Register with n-qbits.
q = QuantumRegister(n)
# Create a Classical Register with n-bits.
c = ClassicalRegister(n)
# Create a Quantum Circuit.
qc = QuantumCircuit(q, c)

# Add H-gate to get superposition.
qc.h(q[0])
qc.h(q[1])

# Apply the oracle 10.
qc.x(q[0])
qc.h(q[1])
qc.cx(q[0],q[1])
qc.x(q[0])
qc.h(q[1])

# Measure qubit to bit.
qc.measure(q, c)
```

```
# Apply the grover-diffusion.
qc.h(q[0])
qc.h(q[1])
qc.x(q[0])
qc.x(q[1])
qc.h(q[0],q[1])
qc.h(q[0])
qc.x(q[1])
qc.x(q[0])
qc.h(q[0])
qc.h(q[1])
```

Database Search - Results on Quantum Simulator #1

After the application of the oracle $U_f$, the measurement on the quantum simulator results in a relatively equally distributed probability of the states.

• After adding the Grover-diffusion operator, we achieve the correct result
Database Search - Results on a Quantum Computer

- The original idea of using quantum computers for database search appears to be only of theoretical nature.
- We are not aware of any practical implementation so far.
- Implementation of a practical Oracle for a database search algorithm seems to an open research problem.
- Loading data into a quantum register might outweigh the performance gain of quantum search.
Outline

Quantum Supremacy

Foundations of Quantum Computing

Quantum Database Search with Grover-Algorithm

Quantum Machine Learning with HHL-Algorithm

Conclusions
Table of Contents

• Classical linear regression
• What is the quantum algorithm HHL?
• How can it be used for quantum machine learning?
• Our implementation
Classical Linear Regression

• **x** is the *explanatory* variable
• **y** the *target* variable

• We want to estimate the shoe size

<table>
<thead>
<tr>
<th>$x$ (height of a person in cm)</th>
<th>165</th>
<th>180</th>
<th>160</th>
<th>200</th>
<th>190</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$ (shoe size of a person)</td>
<td>40</td>
<td>41</td>
<td>38</td>
<td>46</td>
<td>44</td>
</tr>
</tbody>
</table>

![Simple linear regression graph](image-url)
Classical Linear Regression

• For the regression line we need 2 parameters a and b:

\[ y = a + b \times x \]

\[
b = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \times (y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}
\]

\[ a = \bar{y} - b \times \bar{x} \]

\[ y = 8.397 + 0.187 \times x \]
Multiple Linear Regression

- Now you can make the estimate with several explanatory variables.
- The matrix notation simplifies the calculation.
- It is very important that there are at least as many people as the number of parameters to be estimated.

- $y$ is the vector with the target values
- $X$ is the data matrix
- $\beta$ is the vector of the regression parameters

$$y_1 = x_{11}\beta_0 + x_{12}\beta_1 + \ldots + x_{1K}\beta_K$$
$$y_2 = x_{21}\beta_0 + x_{22}\beta_1 + \ldots + x_{2K}\beta_K$$
$$\ldots$$
$$y_T = x_{T1}\beta_0 + x_{T2}\beta_1 + \ldots + x_{TK}\beta_K$$

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1K} \\ x_{21} & x_{22} & \cdots & x_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T1} & x_{T2} & \cdots & x_{TK} \end{pmatrix}$$
$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{pmatrix}$$
$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_K \end{pmatrix}$$

$$y = X\beta$$
Conventional Equation (Least Squares)

The sum of the squared errors is defined as the sum of the squared differences between the values of the model curve and the data.

These are now to be minimized.

\[ \| y - X\beta \|_2 \]

\[ \hat{\beta} = (X^T X)^{-1} X^T y \]
Overview of the HHL Algorithm (by Harrow, Hassidim and Lloyd)

Encoding: $A|x> = |b>$

Goal: $|x> = A^{-1}|b>$

If $A$ is Hermitian, we can implement it as a gates acting on $|b> e^{-iAt}|b>$

Evaluate the answer $<x|M|x>$
Steps of the HHL Algorithm

1. **Loading** the vector $b$ into a quantum state vector $|b>$

2. **Transform** the matrix $A$ into a unitary gate $U$

3. Apply the **quantum phase estimation** of $U$ on $|b>$ to **find the eigenvalues** of

4. Do controlled **rotations** on an ancilla register to **invert the eigenvalues**

5. Perform the **inverse quantum phase estimation**

6. **Measuring** the ancilla qubit
Quantum Phase Estimation 1

- Subroutine estimates eigenvalues
- Idea:
  - Inversion with eigenvalues and eigenvectors (eigendecomposition)

\[ uAu^\dagger = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{pmatrix} \]

- \( A^{-1} = u^\dagger \begin{pmatrix} \lambda & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda \end{pmatrix} u \) (eigenvalues are inverted)

- This can also be done classically in \( O(n^3) \) steps
Quantum Phase Estimation 2

The quantum phase estimation takes a unitary operator $U$ and an eigenvector $|\psi >$ with respect to eigenvalue $e^{2\pi i \theta}$ and estimates the phase $\theta$ of the eigenvalue of $U$

$$\text{QPE}(U, |0\rangle_n |\psi\rangle_m) = |\tilde{\theta}\rangle_n |\psi\rangle_m.$$ 

For the phase estimation in the HHL algorithm $e^{2\pi i \theta}$ is used which can be decomposed into eigenvalues and eigenvectors, as seen below. The phase $\theta$ of the eigenvalues of the $U$ is the eigenvalues of $A$ in the HHL algorithm.

$$U = e^{iAt} := \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle \langle u_j|$$

If the quantum phase estimation is applied to the HHL system, the system transformation can be described with the following equation.

$$\text{QPE}(e^{iA2\pi}, \sum_{j=0}^{N-1} b_j |0\rangle_{n_l} |u_j\rangle_{n_b}) = \sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b}.$$
Rotation of the Ancilla Qubit

- Invert the eigenvalues
- Conditioned rotation with normalization constant C on the clock registers
- Gives us the state:

\[
\sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{n_l} |u_j\rangle_{n_b} \left( \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right)
\]
Inverse Phase Estimation

- $x$ is stored in the quantum state
- The values of $x$ can’t be accessed by measurement
- State is approximately equal to $A^{-1}|b\rangle$ divided by a constant, if the ancilla qubit is 1
Classical Regression vs. HHL Algorithm

• Do we now have a quantum advantage?
• Complexity of classical algorithm is: \( O(N^3) \)

• The complexity of HHL is: \( O(\log_2(N)k^2s^2/e) \)
  • \( N \) is the number of rows of the matrix
  • \( k \) is the condition number
  • \( s \) is the sparsity of the matrix
  (non-zero entries divided by the number total entries)
  • \( e \) is error term that determines precision

• The advantage of HHL lies in the gates of a quantum system, they are unitary matrices
Matrix Conditioning 1

- Condition number $k$ is defined as $k = \frac{\lambda_{A \text{max}}}{\lambda_{A \text{min}}} \rightarrow A = X^T X$

- The amount of qubits needed for an accurate phase estimation is dependent on the $\log_2 k$

- If $k$ gets $x$ times bigger, you need $\log_2(x)$ more qubits to get the same precision

- Reduces to number of qubits needed for an accurate phase estimation

- Linear runtime in relation on the number of regression samples
Matrix Conditioning 2

• Program calculates the mean of each column (except the bias) and divides it by it.

• After the regression parameters are obtained, they are multiplied by the means to get the correct solution.

• Our program works for any real feature matrix of arbitrary size.

• The bigger the variance the better it works.

• Feature matrix of our shoesize example after conditioning:

\[
X = \begin{pmatrix}
1 & 0.921 \\
1 & 1.005 \\
1 & 0.893 \\
1 & 1.117 \\
1 & 1.061
\end{pmatrix}
\]
Adjusting the Values of the Problem

• Is usually required to transform b into a quantum state vector
• Normalization constant: \( |a|^2 + |b|^2 = 1 \)

\[
\begin{align*}
&b_1 \\
&\vdots \\
&b_j
\end{align*}
\]

• \( n = \sqrt{\sum_{i=1}^{i=j} b_j^2} \)

• We divide every value of A and b by n
• Or divide only b and multiply our estimate with n afterwards

• Computational complexity \( O(n) \), can and should be done classically
Preparing the Vector \( b \)

- For our regression example: \( n=173.71 \)
- New value of \( b = [0.48930437 \quad 0.87211308] \)
- A qubit can now set to the state \( b \) with the \( R_x \) gate
- The phase of the gate is 2.119 in our example
- For a general length 2 vector the phase is \( 2\arccos(b_1) \)
- Efficient preparation of \( b \) is more difficult with large arbitrary vectors, but there are some approaches
Converting A into Gates

- Qiskit function “unitary” is used for the gates
- With this function any unitary matrix can be used as a gate
- The following code generates gate matrices

```python
# generating cu gates
for i in range(clocksize):
    u=expm(1j*A*t*(2**i))
    gate=np.array([[1,0,0,0],[0,1,0,0],[0,0,u[0][0],u[0][1]],[0,0,u[1][0],u[1][1]]])
    gatelist.append(gate)
```

- On a real computer this step will be difficult to run efficiently
Quantum Phase Estimation

- Works automatically for a chosen clock size and arbitrary 2x2 matrices

```python
# inverse quantum Fourier transform
def iqft(circ, n):
    for qubit in range(n//2):
        circ.swap(qubit+1, n-qubit)
    for j in range(n):
        for m in range(j):
            circ.cz(-math.pi/float(2**(j-m)), m+1, j+1)
        circ.h(j+1)

# phase estimation
def qpe(qc):
    for i in range(clocksize):
        qc.h(clock[i])
    for i in range(clocksize):
        qc.unitary(gatelist[i],[b,clock[i]])
    iqft(qc,clocksize)
    qpe(qc2)
```
Inversion of Eigenvalues

- Our implementation uses specific rotation values for a trivial example

- There is an approach for general problems:
  - Does an inversion step, then a rotation step
  - Requires more qubits
  - Was only implemented for a trivial example
Inverse Phase Estimation

- Can be implemented by inverting the phase estimation circuit
- Qiskit has a circuit inversion function, thus this can be done easily
How Well does it Work?

This works automatically in our program for regression:
- Conditioning of the feature matrix
- Normalization of b vector for 2x2
- Implementing b as a quantum state for 2x2 matrices
- Implementing A as a gate
- Phase estimation with a high amount of qubits
  - Tested with up to 15
  - Could be adjusted to work for bigger than 2x2 matrices
- Inverting the phase estimation
What Does not Work Well?

• The rotations for inverting the eigenvalues:
  • Only works for specific 2 qubit phase estimation example
  • We don't have enough qubits to implement this

• Matrix multiplication and conversion of A:
  • Works, but needs to be improved to gain an advantage in computational complexity
Results

- Significantly lowering of the condition number
  - For a small feature matrix:
    
    |          | Without conditioning | With conditioning |
    |----------|-----------------------|-------------------|
    | k        | 58898221              | 22099             |
    | \log_2(k)| 25.81                | 14.43             |

  - For a big feature matrix:
    
    |          | Without conditioning | With conditioning |
    |----------|-----------------------|-------------------|
    | k        | 11390205              | 2468              |
    | \log_2(k)| 23.44                | 11.27             |
Results

- HHL was successfully implemented for a trivial example
- Quantum states 0.75, 0.25 correspond to the solution 1.125, 0.375
Conclusion on Quantum Machine Learning

• The HHL algorithm was implemented for a trivial example

• Some parts of the HHL algorithm were implemented for general problems

• Condition number was lowered significantly

• General implementation needs more qubits

• Computational complexity needs to be improved for some subroutines for efficient linear regression
Conclusions

• Quantum computing is a very exciting field

• Tremendous progress over the last years

• Still some way to go to solve practical problems on a publicly available quantum computer