Lecture 8: Linear-Algebra Based Quantum Machine Learning

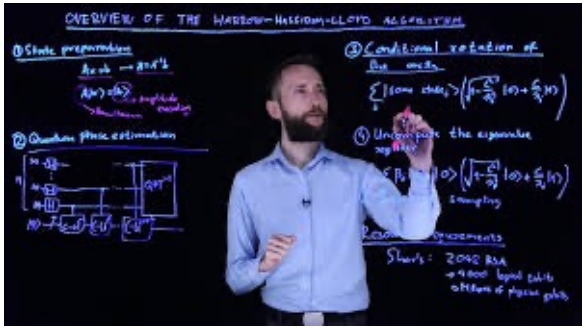
Andreas Wichert

Department of Computer Science and Engineering

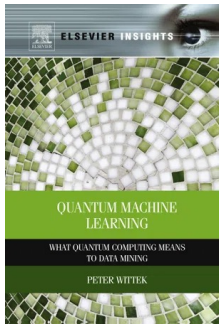
Técnico Lisboa

Overview

- Quantum Perceptron
- Algorithm for Linear Systems of Equations - HHL
- Hamiltonian Simulation
- Kitaev's Phase Estimation to Hamiltonian Simulation
- Conditioned Rotation and Un-computation



Peter Wittek has been an influential figure and a moving force in the exploration and proliferation of quantum machine learning. In September 2019, to the utter dismay of the scientific community, he disappeared in an avalanche during a mountaineering expedition on Mt. Trishul inthe Himalayas.



The book
"Quantum Machine Learning: What Quantum Computing Means to Data Mining",
by Peter Wittek, made quantum **machine learning popular to a wider audience**

- Linear algebra-based quantum machine learning is based on quantum gates that describe quantum basic linear algebra subroutines
- These subroutines can exhibit theoretical exponential speed ups over the classical counterparts and are essential for machine learning
  - The quantum basic linear algebra subroutines represent a quantum coprocessor for extensive and non-tractable computation routines in machine learning
  - Such ‘mathematical’ quantum coprocessor can be used with a conventional computer

Quantum Perceptron

- The quantum perceptron does usually not include learning, instead it computes the output of a binary unit (neuron) efficiently
  - It is based on the Kitaev's phase estimation algorithm
  - A quantum perceptron can be used as building block of larger systems, it can process an arbitrary number of input vectors in parallel

Quantum Machine Learning research lead at Xanadu
Canadian quantum computing hardware and software company headquartered in Toronto

Ilya Sinayskiy
Stellenbosch University,
South Africa

Francesco Petruccione
Stellenbosch University
South Africa

Physics Letters A 379 (2015) 660–663

Contents lists available at ScienceDirect

Physics Letters A

www.elsevier.com/locate/pla

Simulating a perceptron on a quantum computer

Maria Schuld [a,*], Ilya Sinayskiy [a,b], Francesco Petruccione [a,b]

[a] Quantum Research Group, School of Chemistry and Physics, University of KwaZulu-Natal Durban, KwaZulu-Natal, 4001, South Africa
[b] National Institute for Theoretical Physics (NITheP), KwaZulu-Natal, 4001, South Africa

ARTICLE INFO

Article history:
Received 21 September 2014
Received in revised form 27 October 2014
Accepted 3 November 2014
Available online 24 December 2014
Communicated by A. Eisfeld

Keywords:
Quantum neural network
Quantum machine learning
Quantum computing
Linear classification

ABSTRACT

Perceptrons are the basic computational unit of artificial neural networks, as they model the activation mechanism of an output neuron due to incoming signals from its neighbours. As linear classifiers, they play an important role in the foundations of machine learning. In the context of the emerging field of quantum machine learning, several attempts have been made to develop a corresponding unit using quantum information theory. Based on the quantum phase estimation algorithm, this paper introduces a quantum perceptron model imitating the step-activation function of a classical perceptron. This scheme requires resources in $\mathcal{O}(n)$ (where n is the size of the input) and promises efficient applications for more complex structures such as trainable quantum neural networks.

The phase gate

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i \cdot \lambda} \end{pmatrix}.$$

Has tow eigenvectors $|1\rangle$ and $|0\rangle$ with the eigenvalues $e^{i \cdot \lambda}$ and 1. Setting λ to

$$\lambda = \pi \cdot \frac{net}{2^{(m-1)}}$$

results in the eigenvalue

$$e^{i \cdot \pi \cdot \frac{net}{2^{(m-1)}}}.$$

$$U \cdot |u\rangle = e^{2 \cdot \pi \cdot i \cdot \theta} \cdot |u\rangle \qquad\qquad e^{i \cdot \pi \cdot \frac{net}{2^{(m-1)}}}$$

Kitaev's Phase Estimation Algorithm determines the eigenvalue using the controlled phase gate with

$$P \cdot |1\rangle = e^{2 \cdot \pi \cdot i \cdot \theta} \cdot |1\rangle = e^{2 \cdot \pi \cdot i \cdot \frac{net}{2 \cdot 2^{(m-1)}}} \cdot |1\rangle = e^{2 \cdot \pi \cdot i \cdot \frac{net}{2^m}} \cdot |1\rangle.$$

The phase estimation algorithm will write the phase of P to the m qubits in the control register

$$net = x_m \cdots x_2 x_1$$

$$\theta = 0.x_m \cdots x_2 x_1 = \frac{x}{2^m} = \frac{net}{2^m}, \qquad \lambda = \pi \cdot \frac{net}{2^{(m-1)}}$$

$$\lambda = 2 \cdot \theta.$$

- The classical perceptron describes an algorithm for supervised learning that considers only linearly separable problems in which groups can be separated by a line or hyper-plane.
- The quantum perceptron does usually not include learning instead it computes efficiently the binary representation of the function *net* modulo *D* with $x_0 = 1$

$$net := \left(\sum_{j=0}^{D} w_j \cdot x_j \right) \bmod D = \left(\sum_{j=1}^{D} w_j \cdot x_j + w_0 \right) \bmod D = (\langle \mathbf{x} | \mathbf{w} \rangle + w_0) \bmod D$$

with the constraint $x_j \in \{0, 1\}, \quad w_j \in [-1, 1]$

- The value w_0 is called the “bias”, it is a constant value

Example

- Each dimension *j* is represented by a controlled phase gate CP_j with

and $D = 2^m - 2$ or $\log_2(D+2) = m$. For $D = 2$

$$net = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$$

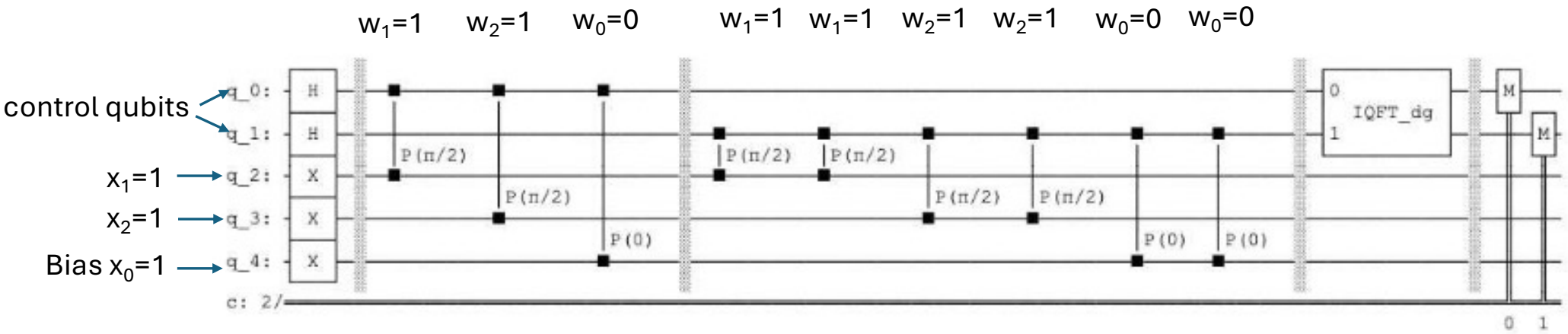
will be represented by three controlled phase gate that perform the following computation

$$U \cdot |u\rangle = e^{2 \cdot \pi \cdot i \cdot \theta} \cdot |u\rangle \qquad e^{i \cdot \pi \cdot \frac{net}{2}} = e^{i \cdot \pi \cdot \frac{w_0}{2}} \cdot e^{i \cdot \pi \cdot \frac{w_1 \cdot x_1}{2}} \cdot e^{i \cdot \pi \cdot \frac{w_2 \cdot x_2}{2}} \qquad \lambda = \pi \cdot \frac{net}{2^{(m-1)}}$$

- ***net*** is represented by two control qubits in the case net is natural number the measured values are four possible values of the two qubits $|q_1 q_0\rangle$

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad |11\rangle.$$

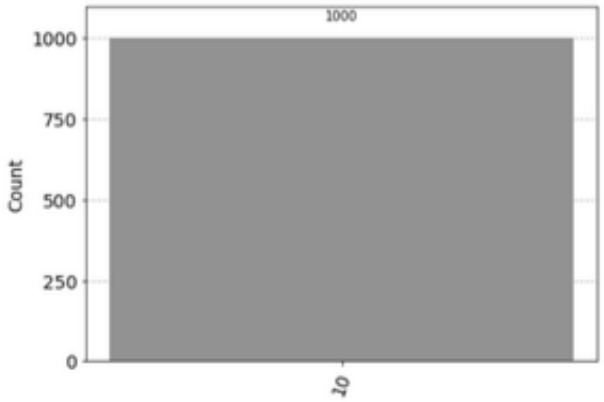
Quantum Perceptron with *D=2*



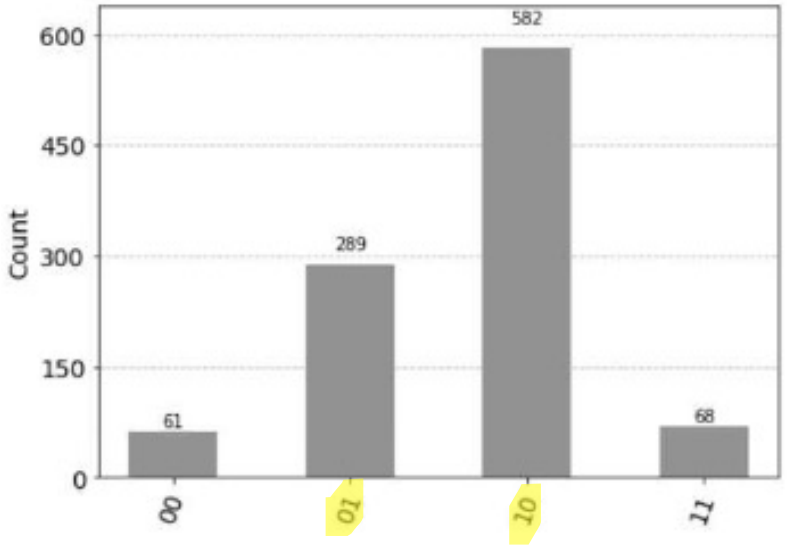
$$e^{2\cdot\pi\cdot i\cdot\theta} \leftrightarrow e^{i\cdot\pi\cdot\frac{net}{2}} = e^{i\cdot\pi\cdot\frac{w_0}{2}} \cdot e^{i\cdot\pi\cdot\frac{w_1\cdot x_1}{2}} \cdot e^{i\cdot\pi\cdot\frac{w_2\cdot x_2}{2}}$$

*net=w₀+w₁*x₁+w₂*x₂=0+1*1+1*1=2*

$$\phi(net) := f(net) = \begin{cases} 1 & \text{if } net \geq threshold \\ 0 & \text{if } net < threshold \end{cases}$$



The measured value of *net* = 2



- The measured value of net for the input $x_1 = 1, x_2 = 1$ and the weights $w_0 = 0$, $w_1 = 0.6$ and $w_2 = 1$ is in **superposition between 2 and 1**

$net = w_0 + w_1 * x_1 + w_2 * x_2 = 0 + 0.6 * 1 + 1 * 1 = 1.6$

Quantum Perceptron as Building Block

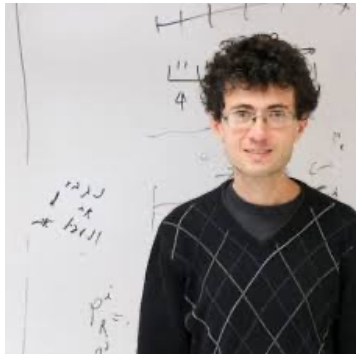
- A quantum Perceptron can be used as building block of larger systems, it can process an arbitrary number of input vectors in parallel
- When the input is presented in superposition $|\psi\rangle$ representing the whole data set *DB* of s objects $\boldsymbol{x}_k$

$$\{\mathbf{x}_k \in DB \mid k \in \{1..s\}\}.$$

$$|\psi\rangle = \frac{1}{\sqrt{s}} \sum_{k=1}^{s} |\mathbf{x}_k\rangle$$

Algorithm for Linear Systems of Equations - HHL

Aram Harrow MIT



Avinatan Hassidim Bar Ilan Univeristy, Israel

Seth Lloyd MIT

- In the honor of its inventors
  - Aram Harrow, Avinatan Hassidim, and Seth Lloyd, it is called the HHL algorithm
- The quantum algorithm for linear systems of equations for is one of the main fundamental algorithms expected to provide an ***exponential*** **speedup** over their classical counterparts.
  - HHL is going to be one of *the most "useful" subroutines* for any quantum machine learning algorithm because almost all machine learning uses some form of a linear system of equations

- A is Hermitian $A=A^*$ (for real matrix $A^T=A$)

$$A \cdot \mathbf{x} = \mathbf{b}$$

- In the case A is not Hermitian, we define

$$A' = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix}, \quad \mathbf{b}' = \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad \mathbf{x}' = \begin{pmatrix} x \\ 0 \end{pmatrix}.$$

$$A' \cdot \mathbf{x}' = \mathbf{b}'.$$

- *A* is Hermitian $A=A^*$ (for real matrix $A^T=A$)

$$A \cdot \mathbf{x} = \mathbf{b}$$

- A can be represented by the spectral decomposition as

$$A = \lambda_1 \cdot |u_1\rangle\langle u_1| + \lambda_2 \cdot |u_2\rangle\langle u_2| + \cdots + \lambda_n \cdot |u_n\rangle\langle u_n|.$$

$$A^{-1} = \frac{1}{\lambda_1} \cdot |u_1\rangle\langle u_1| + \frac{1}{\lambda_2} \cdot |u_2\rangle\langle u_2| + \cdots + \frac{1}{\lambda_n} \cdot |u_n\rangle\langle u_n|.$$

Eigenvectors $u_1, u_l, \cdots u_n$ and Λ represents the corresponding eigenvalues of A

$$A^{-1} = \frac{1}{\lambda_1} \cdot |u_1\rangle\langle u_1| + \frac{1}{\lambda_2} \cdot |u_2\rangle\langle u_2| + \cdots + \frac{1}{\lambda_n} \cdot |u_n\rangle\langle u_n|.$$

It follows

$$A^{-1} \cdot |u_j\rangle = \frac{1}{\lambda_j}|u_j\rangle$$

and writing $|b\rangle$ as a linear combination of the eigenvectors of A

$$|b\rangle = \sum_j |u_j\rangle\langle u_j|b\rangle$$

leads to

$$A \cdot |b\rangle = \sum_j \lambda_j |u_j\rangle\langle u_j|b\rangle$$

and

$$|x\rangle = A^{-1} \cdot |b\rangle = \sum_j \lambda_j^{-1} |u_j\rangle\langle u_j|b\rangle.$$

Constraints

- The matrix *A* is sparse
- The vectors $|b\rangle$, $|x\rangle$ have the length one in the l_2 norm,

with length one vectors, $|b\rangle$ represented by $\log_2 n$ qubits with

$$|b\rangle = \frac{\sum_{i=1}^{n} b_i |i\rangle}{\| \sum_{i=1}^{n} b_i |i\rangle \|}$$

and

$$|x\rangle = \frac{\sum_{i=1}^{n} x_i |i\rangle}{\| \sum_{i=1}^{n} x_i |i\rangle \|}.$$

Normalization

A Hermitian matrix can be represented by a sum of projections of its orthonormal eigenvectors (normalized eigenvectors) $x_1, \cdots, x_n$ weighted by its eigenvalues. The projections are defined by the matrix $|x_i\rangle\langle x_i|$ with $|x_i\rangle\langle x_i| = (|x_i\rangle\langle x_i|)^2$. The set of the eigenvalues is called the spectrum and the corresponding representation a spectral representation

$$A = \lambda_1 \cdot |x_1\rangle\langle x_1| + \lambda_2 \cdot |x_2\rangle\langle x_2| + \cdots + \lambda_n \cdot |x_n\rangle\langle x_n|. \qquad (A=A^*)$$

The function $e^{i \cdot A}$ with a self-adjoint operator A is unitary

$$e^{i \cdot A} = e^{i \cdot \lambda_1} \cdot |x_1\rangle\langle x_1| + e^{i \cdot \lambda_2} \cdot |x_2\rangle\langle x_2| + \cdots + e^{i \cdot \lambda_n} \cdot |x_n\rangle\langle x_n|.$$

An operator U is unitary, if $U^* = U^{-1}$,

$$\left(e^{i \cdot A}\right)^* = \left(e^{i \cdot A}\right)^{-1},$$

$$\left(e^{i \cdot A}\right)^{-1} = e^{-i \cdot \lambda_1} \cdot |x_1\rangle\langle x_1| + e^{-i \cdot \lambda_2} \cdot |x_2\rangle\langle x_2| + \cdots + e^{-i \cdot \lambda_n} \cdot |x_n\rangle\langle x_n|.$$

This representation is similar to the evolutionary operator $U_t = e^{-i \cdot t \cdot H}$ for $t = 1$ and $A = H$ is the Hamiltonian operator that is related to the total energy of the system.

- How can we estimate the eigenvalues efficiently?
- We know that the function $e^{i \cdot A}$ with a Hermitian A ($A=A^*$)

$$U = e^{i \cdot A} = e^{i \cdot \lambda_1} \cdot |u_1\rangle\langle u_1| + e^{i \cdot \lambda_2} \cdot |u_2\rangle\langle u_2| + \cdots + e^{i \cdot \lambda_n} \cdot |u_n\rangle\langle u_n|.$$

and

$$U^* = U^{-1} = e^{-i \cdot A} = e^{-i \cdot \lambda_1} \cdot |u_1\rangle\langle u_1| + e^{-i \cdot \lambda_2} \cdot |u_2\rangle\langle u_2| + \cdots + e^{-i \cdot \lambda_n} \cdot |u_n\rangle\langle u_n|.$$

with

$$U \cdot U^* = I$$

Using Kitaev's phase estimation algorithm we could estimate then unknown eigenvalue $e^{2 \cdot \pi \cdot i \cdot \theta_j}$ If we apply U to $|u_j\rangle$ we get

$$U \cdot |u\rangle = e^{2 \cdot \pi \cdot i \cdot \theta_j} \cdot |u_j\rangle = e^{i \cdot \lambda_j} \cdot |u_j\rangle$$

How to Represent $U^* = e^{-i \cdot A}$?

$$U^* = U^{-1} = e^{-i \cdot A} = e^{-i \cdot \lambda_1} \cdot |u_1\rangle\langle u_1| + e^{-i \cdot \lambda_2} \cdot |u_2\rangle\langle u_2| + \cdots + e^{-i \cdot \lambda_n} \cdot |u_n\rangle\langle u_n|.$$

- This representation is similar to the evolutionary operator
  - $U_t = e^{-i \cdot t \cdot H}$ for $t = 1$ and $A = H$ is the Hamiltonian operator
- The process of implementing a given Hamiltonian evolution on quantum computer is called **Hamiltonian simulation**
- Hamiltonian simulation can be implemented efficiently for large if the Hermitian matrix H is sparse

How to Represent the *Eigenvectors* $|u_j\rangle$ of U^* ?

$$U^* = U^{-1} = e^{-i\cdot A} = e^{-i\cdot\lambda_1}\cdot|u_1\rangle\langle u_1| + e^{-i\cdot\lambda_2}\cdot|u_2\rangle\langle u_2| + \cdots + e^{-i\cdot\lambda_n}\cdot|u_n\rangle\langle u_n|.$$

$$|x\rangle = A^{-1}\cdot|b\rangle = \sum_j \lambda_j^{-1}|u_j\rangle\langle u_j|b\rangle.$$

- We do not need to know the eigenvector $|u_j\rangle$ of U
  - Since a quantum state $|b\rangle$ can be decomposed into an orthogonal basis

$$|b\rangle = \sum_j |u_j\rangle\langle u_j|b\rangle = \sum_j \beta_j|u_j\rangle$$

Hamiltonian Simulation

- The process of implementing a given Hamiltonian evolution on quantum computer is called Hamiltonian simulation

$$|x(t)\rangle = e^{-i \cdot t \cdot H} \cdot |x(0)\rangle = U_t \cdot |x(0)\rangle.$$

- The challenge is due to the fact, that the application of **matrix exponentials are computationally expensive**
- Finding reliable and accurate methods to compute the matrix exponential is **difficult**, and this is still a topic of considerable current research

It is easy if the matrix *H* is diagonal

$$H = \begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_n \end{pmatrix}$$

then its exponential can be obtained by exponentiating each entry on the main diagonal

$$e^H = \begin{pmatrix} e^{a_1} & 0 & \dots & 0 \\ 0 & e^{a_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{a_n} \end{pmatrix}$$

This result also allows one to exponentiate diagonalizable matrices. If

$$H = U \cdot D \cdot U^{-1}$$

and D is diagonal, then

$$e^H = U \cdot e^D \cdot U^{-1}$$

Not Diagonal - Product Formulas

If the hamiltonian H can be written as

$$H = \sum_{k=1}^{L} H_k$$

and H_k commute,

$$H_j \cdot H_k = H_k \cdot H_j, \quad for\ all\ j,k$$

then

$$e^{-i\cdot t\cdot H} = e^{-i\cdot t\cdot H_1} \cdot e^{-i\cdot t\cdot H_2} \cdot \ldots \cdot e^{-i\cdot t\cdot H_L} \quad for\ all\ t.$$

General case: For general case of non-commuting H_k

$$e^{-i\cdot t\cdot H} \neq e^{-i\cdot t\cdot H_1} \cdot e^{-i\cdot t\cdot H_2} \cdot \ldots \cdot e^{-i\cdot t\cdot H_L} \quad for\ all\ t.$$

Suzuki-Trotter Formula

However for non-commuting H_k the following asymptotic Suzuki-Trotter formula (Lie product formula) is true for any real t.

$$e^{-i\cdot t\cdot H} = \lim_{m\to\infty}\left(e^{-i\cdot t\cdot H_1/m}\cdot e^{-i\cdot t\cdot H_2/m}\cdot\ldots\cdot e^{-i\cdot t\cdot H_L/m}\right)^m$$

or we can write equivalently

$$e^{-i\cdot t\cdot H} = e^{-i\cdot t\cdot H_1}\cdot e^{-i\cdot t\cdot H_2}\cdot\ldots\cdot e^{-i\cdot t\cdot H_L} + O(t^2)$$

For small t the factorization is approximately valid,

$$U_t = e^{-i\cdot t\cdot H} \approx e^{-i\cdot t\cdot H_1}\cdot e^{-i\cdot t\cdot H_2}\cdot\ldots\cdot e^{-i\cdot t\cdot H_L}.$$

- It follows that with big *T* and a Hermitian matrix A

$$A = \sum_{k=1}^{L} A_k, \qquad \Delta t = \frac{1}{T},$$

$$U_{\Delta t} = e^{2 \cdot \pi \cdot i \cdot \Delta t \cdot A} \approx e^{2 \cdot \pi \cdot i \cdot \Delta t \cdot A_1} \cdot e^{2 \cdot \pi \cdot i \cdot \Delta t \cdot A_2} \cdot \ldots \cdot e^{2 \cdot \pi i \cdot \Delta t \cdot A_L},$$

and

$$U^t := e^{\frac{2 \cdot \pi \cdot \mathrm{i} \cdot A \cdot \mathrm{t}}{T}} \approx e^{\frac{2 \cdot \pi \cdot \mathrm{i} \cdot A_1 \cdot \mathrm{t}}{T}} \cdot e^{\frac{2 \cdot \pi \cdot \mathrm{i} \cdot A_2 \cdot \mathrm{t}}{T}} \cdot \ldots \cdot e^{\frac{2 \cdot \pi \cdot \mathrm{i} \cdot A_L \cdot \mathrm{t}}{T}}.$$

Hamiltonian Simulation of sparse *H*

- Hamiltonian simulation can be implemented efficiently for large *n* if the Hermitian matrix *H* is sparse
  - A *n* × *n* matrix *H* is s-sparse if it has at most s non-zero elements in each row and column.
  - A *n* × *n* matrix H is s-sparse is sparse, if *s* = $O((log(n))^k)$ for some constant *k*.
- Given such a unitary oracle access to an s-sparse Hermitian matrix *H*, then the cost of the Hamiltonian simulation are

$$\tilde{O}(\log(n) \cdot s^2 \cdot t).$$

  there is a logarithmic dependency of *n*

Eigenvalue estimation with Hamiltonian simulation

$$U \cdot |u\rangle = e^{2 \cdot \pi \cdot i \cdot \theta} \cdot |u\rangle.$$

With Hamiltonian simulation

$$U^t = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot t}{T}} \approx e^{\frac{2 \cdot \pi \cdot i \cdot A_1 \cdot t}{T}} \cdot e^{\frac{2 \cdot \pi \cdot i \cdot A_2 \cdot t}{T}} \cdot \ldots \cdot e^{\frac{2 \cdot \pi \cdot i \cdot A_L \cdot t}{T}}$$

and

$$U^t \cdot |u_j\rangle = e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |u_j\rangle.$$

If

$$A|u_j\rangle = \lambda_j |u_j\rangle$$

then

$$U^t \cdot |u_j\rangle = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot t}{T}} |u_j\rangle = e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |u_j\rangle.$$

With $T = 4$ and $U^t = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot t}{4}}$ we get,

$$U^0 = I, \;\; U^1 = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot 1}{4}}, \;\; U^2 = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot 2}{4}}, \;\; U^3 = e^{\frac{2 \cdot \pi \cdot i \cdot A \cdot 3}{4}}$$

and

$$cU = \sum_{t=0}^{3} |t\rangle\langle t| \otimes U^t =$$

$$= |00\rangle\langle 00| \otimes I + |01\rangle\langle 10| \otimes e^{\frac{\pi \cdot i \cdot A}{2}} + |10\rangle\langle 01| \otimes e^{\pi \cdot i \cdot A} + |11\rangle\langle 11| \otimes e^{\frac{\pi \cdot i \cdot A \cdot}{4}}$$

Knowing u_j with $m = \log_2 T$,

$$cU \cdot H_m(|0^{\otimes m}\rangle) \cdot |u_j\rangle = cU \cdot \sum_{t=0}^{n} \frac{1}{\sqrt{T}} |t\rangle|u_j\rangle$$

We apply inverse QFT on the control qubits with $m = \log_2 T$,

$$IF_m \left(\frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |t\rangle \right) |u_j\rangle = \frac{1}{T} \sum_{k=0}^{T-1} \left(\sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot (\lambda_j - k)}{T}} \right) |k\rangle |u_j\rangle$$

$$\alpha_{k|j} = \frac{1 - e^{\frac{2 \cdot \pi \cdot i \cdot (\lambda_j - k) \cdot T}{T}}}{T \cdot \left(1 - e^{\frac{2 \cdot \pi \cdot i \cdot (\lambda_j - k)}{T}}\right)}$$

$$IF_m \left(\frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |t\rangle \right) |u_j\rangle = \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |k\rangle |u_j\rangle$$

In the eigenvalue estimation the first register is measured where is the binary representation $\lambda_j = k/T$. However, we do not measure the first register and relabel $|k\rangle$ with $|\tilde{\lambda}_{k|j}\rangle$ resulting in

$$IF_m \left(\frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |t\rangle \right) |u_j\rangle = \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}\rangle |u_j\rangle$$

Unknown Eigenvectors

$$|b\rangle = \sum_j |u_j\rangle\langle u_j|b\rangle = \sum_j \beta_j |u_j\rangle$$

We represent $|b\rangle$ as

$$|b\rangle = \sum_i b_i |i\rangle$$

We use a *cost function which minimizes a penalty for not correct answers*

$$|\psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{t=0}^{T-1} \sin \frac{\pi \cdot (t+1/2)}{T} |t\rangle$$

and

$$cU \cdot |\psi_0\rangle|b\rangle = \sqrt{\frac{2}{T}} \sum_{j=1}^{n} \beta_j \left(\sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} \sin \frac{\pi \cdot (t+1/2)}{T} |t\rangle \right) |u_j\rangle$$

Unknown Eigenvectors

$$|b\rangle = \sum_j |u_j\rangle\langle u_j|b\rangle = \sum_j \beta_j |u_j\rangle$$

We represent $|b\rangle$ as

$$|b\rangle = \sum_i b_i |i\rangle$$

The accuracy of the approximation of this will depend far more critically on an accurate estimation of the eigenvalues that are close to 0 rather than those that are far from 0 so that rather than using 'bins' of fixed width for approximating the phases we specify variable 'bins' so that we can have vastly increased accuracy close to 0 phase. We use a cost function which minimizes a penalty for not correct answers

$$|\psi_0\rangle = \sqrt{\frac{2}{T}} \sum_{t=0}^{T-1} \sin \frac{\pi \cdot (t+1/2)}{T} |t\rangle$$

$$cU \cdot \sum_{t=0}^{n} \frac{1}{\sqrt{T}} |t\rangle |u_j\rangle = \frac{1}{\sqrt{T}} \sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} |t\rangle |u_j\rangle.$$

$$cU \cdot |\psi_0\rangle |b\rangle = \sqrt{\frac{2}{T}} \sum_{j=1}^{n} \beta_j \left(\sum_{t=0}^{T-1} e^{\frac{2 \cdot \pi \cdot i \cdot \lambda_j}{T}} \sin \frac{\pi \cdot (t + 1/2)}{T} |t\rangle \right) |u_j\rangle$$

- We Apple inverse QFT on the control qubits

$$IF_{\widetilde{m}}\left(\sqrt{\frac{2}{T}}\sum_{j=1}^{n}\beta_j\left(\sum_{t=0}^{T-1}e^{\frac{2\cdot\pi\cdot i\cdot\lambda_j}{T}}\sin\frac{\pi\cdot(t+1/2)}{T}|t\rangle\right)\right)|u_j\rangle=$$

$$\sum_{j=1}^{n}\beta_j\sum_{k=0}^{T-1}\left(\sqrt{\frac{2}{T}}\sum_{t=0}^{T-1}e^{\frac{2\cdot\pi\cdot i\cdot(\lambda_j-k)}{T}}\sin\frac{\pi\cdot(t+1/2)}{T}\right)|k\rangle|u_j\rangle$$

with

$$\alpha_{k|j}=\left(\sqrt{\frac{2}{T}}\sum_{t=0}^{T-1}e^{\frac{2\cdot\pi\cdot i\cdot(\lambda_j-k)}{T}}\sin\frac{\pi\cdot(t+1/2)}{T}\right)$$

$$\sum_{j=1}^{n} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} |k\rangle |u_j\rangle = \sum_{j=1}^{n} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}\rangle |u_j\rangle$$

with the following relation, if $|\alpha_{k|j}|$ is large then $\lambda_j \approx \tilde{\lambda}_{k|j}$. By linearity

$$\sum_{j=1}^{n} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}| u_j\rangle = \sum_{j=1}^{n} \sum_{k=0}^{T-1} \beta_j \cdot \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}\rangle |u_j\rangle$$

For each value j the values $\tilde{\lambda}_{k|j}$ approximate the true value λ_j. For simplicity we assume

$$\sum_{j=1}^{n} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}| u_j\rangle \approx \sum_{j=1}^{n} \beta_j \cdot |\tilde{\lambda}_j\rangle |u_j\rangle.$$

Kitaev's Phase Estimation

- After applying Kitaev's phase estimation algorithm to *U* that we estimated by the Hamiltonian simulation for each value *j* the values $\tilde{\lambda}_{k|j}$ approximate the true value λ_j
- For simplicity we assume

$$\sum_{j=1}^{n} \beta_j \sum_{k=0}^{T-1} \alpha_{k|j} \cdot |\tilde{\lambda}_{k|j}|u_j\rangle \approx \sum_{j=1}^{n} \beta_j \cdot |\tilde{\lambda}_j\rangle|u_j\rangle.$$

Eigenvalue in register

Conditioned Rotation

- The controlled $R(\lambda^{-1})$ rotation extracts the eigenvalues and executes the rotation
- The rotation is conditioned on the value the **binary** representation of the **eigenvalue**
- For **each binary value** we will have a different operator
- A rotation by an angle α is represented by the unitary operator *R*

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \qquad \alpha = \arccos\left(\frac{C}{\tilde{\lambda}}\right)$$

Conditioned Rotation

We add an auxiliary state $|0\rangle$

$$\sum_{j=1}^{n} \beta_j \cdot |\tilde{\lambda}_j\rangle |u_j\rangle |0\rangle$$

and perform the conditioned rotation on the auxiliary state $|0\rangle$ by the operator R

for **each** eigenvalue

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

with the relation

$$\alpha = \arccos\left(\frac{C}{\tilde{\lambda}}\right)$$

with C being a constant of normalization.

Conditioned Rotation

- For **each** eigenvalue indicates a special rotation we have
  - The rotation is conditioned on the binary representation of the eigenvalue
  - For each binary value we have a different operator

$$\sum_{j=1}^{n}\left(\beta_j\cdot|\tilde{\lambda}_j\rangle|u_j\rangle\left(R\left(\tilde{\lambda}_j^{-1}\right)|0\rangle\right)\right)=$$

$$\sum_{j=1}^{n}\left(\beta_j\cdot|\tilde{\lambda}_j\rangle|u_j\rangle\left(\frac{C}{\tilde{\lambda}_j}|1\rangle+\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}|0\rangle\right)\right).$$

Un-Computation

$$\sum_{j=1}^{n}\left(\beta_j\cdot|\tilde{\lambda}_j\rangle|u_j\rangle\left(\frac{C}{\tilde{\lambda}_j}|1\rangle+\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}|0\rangle\right)\right)$$

- We un-compute the phase estimation procedure with IF_m, CU^* resulting in the state

$$|0\rangle\sum_{j=1}^{n}\left(\beta_j\cdot|u_j\rangle\left(\frac{C}{\tilde{\lambda}_j}|1\rangle+\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}|0\rangle\right)\right)=$$

$$|0\rangle\sum_{j=1}^{n}\left(C\cdot\frac{\beta_j}{\tilde{\lambda}_j}|u_j\rangle|1\rangle+\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}\cdot\beta_j}|u_j\rangle|0\rangle\right).$$

Measurement

By measuring the auxiliary qubit with the result 0 we get

$$|0\rangle \sum_{j=1}^{n} \left(\sqrt{1 - \frac{C^2}{\tilde{\lambda}_j^2}} \cdot \beta_j |u_j\rangle \right)$$

and with the result 1

$$C \cdot |0\rangle \sum_{j=1}^{n} \left(\frac{\beta_j}{\tilde{\lambda}_j} |u_j\rangle \right) = C \cdot |0\rangle A^{-1} |b\rangle \approx C \cdot |0\rangle |x\rangle.$$

We have to select the outcome of the measurement 1.

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=1}^{n} \frac{\beta_j}{\tilde{\lambda}_j} |u_j\rangle$$

Solution

- Obtaining the required coefficient x_i from $|x\rangle$ would require at least n measurements, so the complexity of the algorithm would be **O(n) which problematic** since the cost on a classical computer for an approximate solution for a sparse matrix via conjugate gradient descent are equivalent.
- Many applications are interested in the **global properties** of **$|x\rangle$** rather than the coefficients x_i
- Many features describing the vector $|x\rangle$ can be extracted efficiently, like for example values in different parts
- We can as well efficiently estimate if two solutions of two different equations are the same or not

- If the **constraints are fulfilled,** then we can find the estimate of solution in *O(log n).* Gauss elimination requires $O(n^3)$ and approximate solution for a sparse matrix via conjugate gradient descent requires *O(n)*

- Taking into account the constraints, the HHL algorithm on a quantum computer is **exponentially faster** than any algorithm that solves linear systems on the classical computer

- Basic quantum linear algebra subroutines use a quantum coprocessor for extensive and nontrackable computation routines in machine learning.
- Such ‘mathematical’ quantum coprocessors can be used on a conventional computer.

Example

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

with eigenvectors

$$\mathbf{u}_1 = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \mathbf{u}_2 = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

and eigenvalues

$$\lambda_1 = \frac{4}{3}, \quad \lambda_2 = \frac{2}{3}.$$

It follows

$$U = e^{i \cdot \frac{4}{3}} \cdot |u_1\rangle\langle u_1| + e^{i \cdot \frac{2}{3}} \cdot |u_2\rangle\langle u_2| = \begin{pmatrix} \frac{1}{2}e^{\frac{2i}{3}} + \frac{1}{2}e^{\frac{4i}{3}} & \frac{1}{2}e^{\frac{2i}{3}} - \frac{1}{2}e^{\frac{4i}{3}} \\ \frac{1}{2}e^{\frac{2i}{3}} - \frac{1}{2}e^{\frac{4i}{3}} & \frac{1}{2}e^{\frac{2i}{3}} + \frac{1}{2}e^{\frac{4i}{3}} \end{pmatrix}$$

and

$$U^* = e^{-i \cdot \frac{4}{3}} \cdot |u_1\rangle\langle u_1| + e^{-i \cdot \frac{2}{3}} \cdot |u_2\rangle\langle u_2| = \begin{pmatrix} \frac{1}{2}e^{\frac{-2i}{3}} + \frac{1}{2}e^{\frac{-4i}{3}} & \frac{1}{2}e^{\frac{-2i}{3}} - \frac{1}{2}e^{\frac{-4i}{3}} \\ \frac{1}{2}e^{\frac{-2i}{3}} - \frac{1}{2}e^{\frac{-4i}{3}} & \frac{1}{2}e^{\frac{-2i}{3}} + \frac{1}{2}e^{\frac{-4i}{3}} \end{pmatrix}$$

With the solution

The solution to the problem

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

is represented by

$$A^{-1} = \begin{pmatrix} \frac{9}{8} & \frac{3}{8} \\ \frac{3}{8} & \frac{9}{8} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \frac{9}{8} \\ \frac{3}{8} \end{pmatrix}.$$

with the normalized vector being

$$\mathbf{x}_n = \frac{\mathbf{x}}{\|\mathbf{x}\|} = \begin{pmatrix} 0.948683 \\ 0.316228 \end{pmatrix}.$$

Kitaev's Phase Estimation to Hamiltonian Simulation

Using Kitaev's phase estimation algorithm we get the relation

$$U \cdot |u\rangle = e^{2 \cdot \pi \cdot i \cdot \theta_j} \cdot |u_j\rangle$$

By using Hamiltonian simulation the relation is

$$U_t \cdot |u\rangle = e^{-i \cdot t \cdot \lambda_j} \cdot |u_j\rangle.$$

We will use during Kitaev's phase estimation algorithm QFT instead of the inverse QFT to deal with the minus sign. In this case the reltion becomes

$$e^{2 \cdot \pi \cdot i \cdot \theta_j} \cdot |u_j\rangle = e^{i \cdot t \cdot \lambda_j} \cdot |u_j\rangle$$

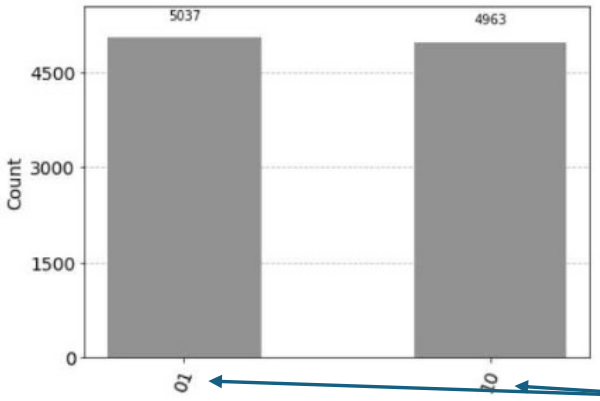
with

$$\theta_j = \frac{\lambda_j \cdot t}{2 \cdot \pi}$$

- In our simulation we will use two qubits in the control register and we chose $t = 2 \cdot \pi \cdot 3/8 = \pi \cdot 3/4$ and we get

$$\theta_j = \frac{\lambda_j \cdot t}{2 \cdot \pi} \rightarrow \theta_j = \frac{3}{8} \cdot \lambda_j \qquad A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}$$

```
op = Operator([[1, -1/3],[-1/3, 1]])
# create gate which evolves according to exp(-i*op*3*pi/4)
gate = HamiltonianGate(op,3*pi/4).control()
```



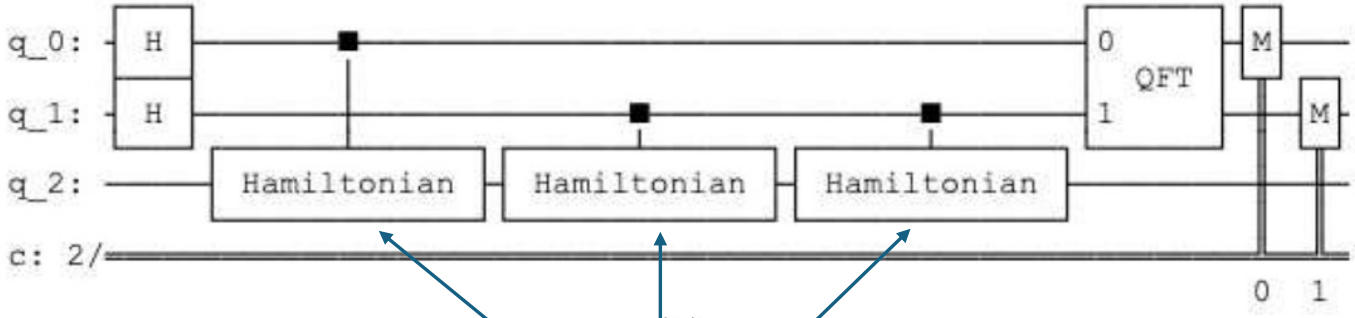
$$\theta_j = \frac{3}{8} \cdot \lambda_j \qquad \lambda_1 = \frac{4}{3}, \quad \lambda_2 = \frac{2}{3}.$$

$$\theta_1 = \frac{3}{8} \cdot \frac{4}{3} = \frac{2}{4}, \qquad \theta_2 = \frac{3}{8} \cdot \frac{2}{3} = \frac{1}{4}$$

with measured two control qubits being $|10\rangle$ representing 2 and $|01\rangle$ representing 1

$$\theta = 0.x_2x_1 = \frac{x}{4} = \frac{x}{2^2}.$$

$$|b\rangle = |0\rangle = \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$




```
op = Operator([[1, -1/3], [-1/3, 1]])
# create gate which evolves according to exp(-i*op*3*pi/4)
gate = HamiltonianGate(op,3*pi/4).control()
```

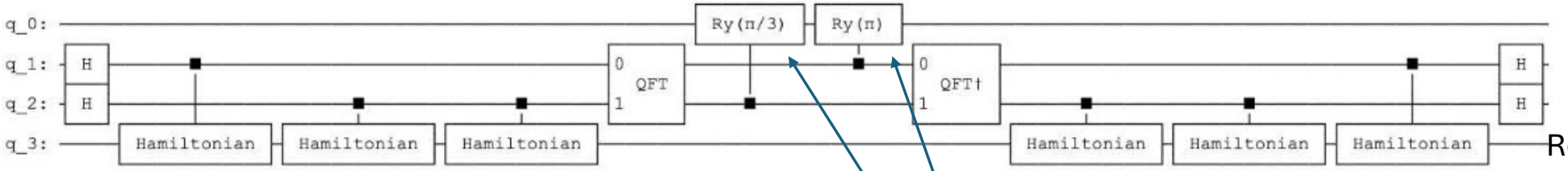
$$|b\rangle = |0\rangle = \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|u_1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \qquad |u_2\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$|b\rangle = \sum_j \beta_j |u_j\rangle = |0\rangle = \frac{1}{\sqrt{2}} \cdot |u_1\rangle + \frac{1}{\sqrt{2}} \cdot |u_2\rangle$$

$$|b\rangle = \frac{1}{\sqrt{2}} \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}} + \frac{1}{\sqrt{2}} \cdot \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |0\rangle.$$

Conditioned Rotation and Un-computation



We perform conditioned rotation on the auxiliary state $|0\rangle$ by

$$R_Y(\alpha) = \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) & -\sin\left(\frac{\alpha}{2}\right) \\ \sin\left(\frac{\alpha}{2}\right) & \cos\left(\frac{\alpha}{2}\right) \end{pmatrix}$$

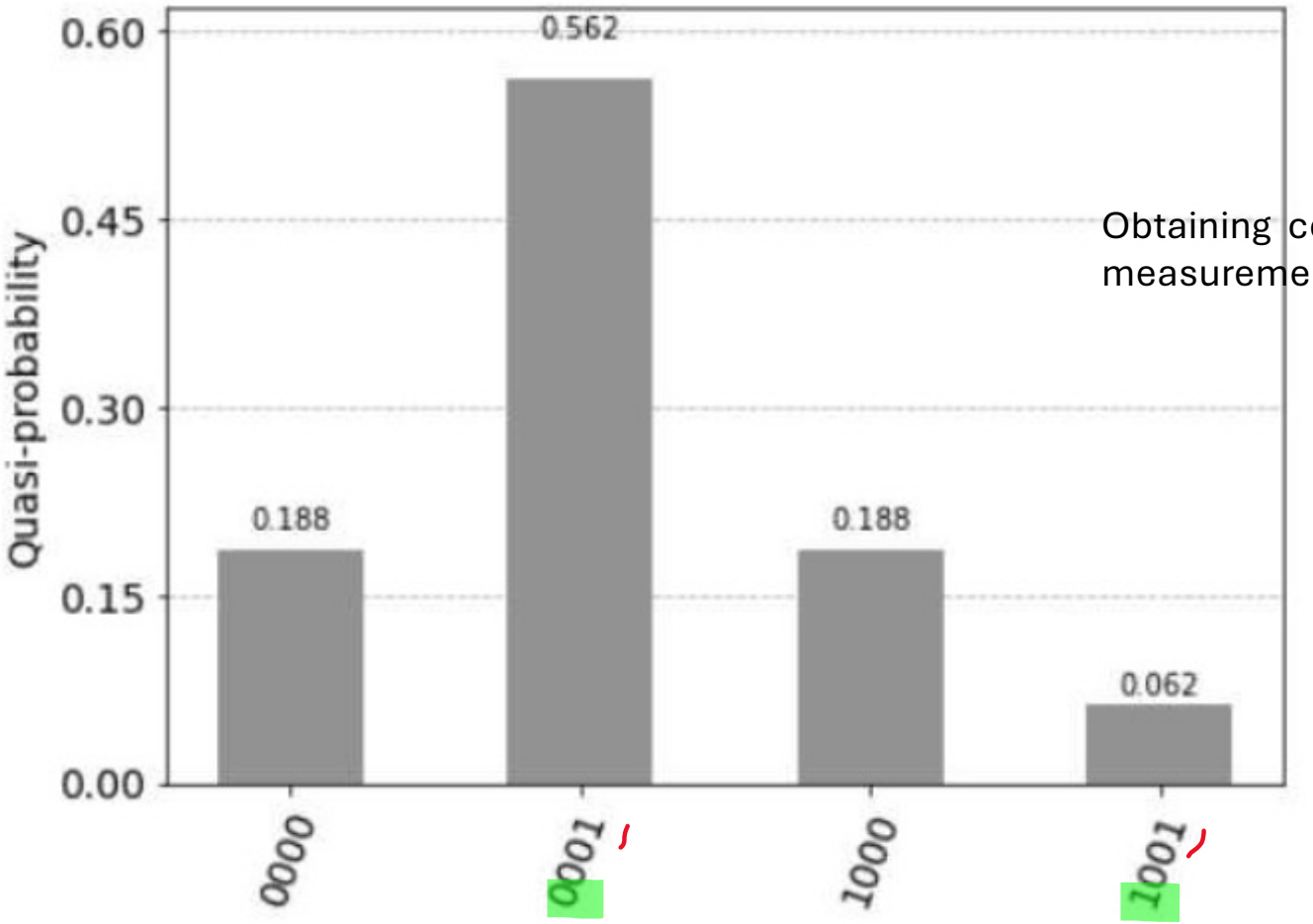
The rotation is conditioned on the binary representation of the eigenvalue
For **each** binary value we have a different operator

with measured two control qubits being $|10\rangle$ representing $\tilde{\lambda}_1 = 2$ and $|01\rangle$ representing $\tilde{\lambda}_2 = 1$,

$$\alpha_1 = 2 \cdot \arccos\left(\frac{1}{\tilde{\lambda}_1}\right) = 2 \cdot \arccos\left(\frac{1}{2}\right) = \frac{\pi}{3} \tag{22.36}$$

$$\alpha_2 = 2 \cdot \arccos\left(\frac{1}{\tilde{\lambda}_2}\right)) = 2 \cdot \arccos\left(\frac{1}{1}\right) = \pi \tag{22.37}$$

using a conditional RY gate controlled by $\tilde{\lambda}$ represented by qubits 1 and 2, then we un-compute. The result is represented in the qubit 3 in the case the qubit 0 is one



Obtaining coefficient x_i by **several** measurements !!!

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=1}^{n} \frac{\beta_j}{\tilde{\lambda}_j}|u_j\rangle$$

The measured result of our small HHL simulation is represented in the qubit 3 if the qubit 0 is one, with the probability values *0.562* and *0.0622* (Remember, we un-computed)

The solution to the problem

$$A = \begin{pmatrix} 1 & -\frac{1}{3} \\ -\frac{1}{3} & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \longleftrightarrow |b\rangle = \frac{\sum_{i=1}^{n} b_i |i\rangle}{\| \sum_{i=1}^{n} b_i |i\rangle \|}$$

is represented by

$$A^{-1} = \begin{pmatrix} \frac{9}{8} & \frac{3}{8} \\ \frac{3}{8} & \frac{9}{8} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \frac{9}{8} \\ \frac{3}{8} \end{pmatrix}.$$

with the normalized vector being

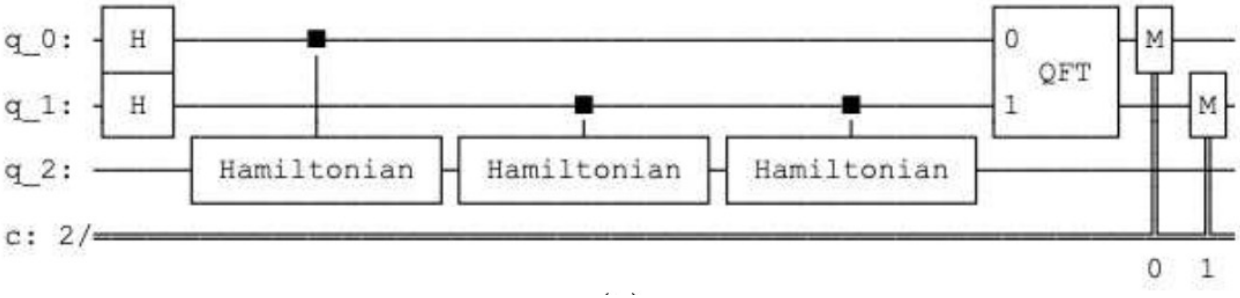
$$\mathbf{x}_n = \frac{\mathbf{x}}{\|\mathbf{x}\|} = \begin{pmatrix} 0.948683 \\ 0.316228 \end{pmatrix}. \longleftrightarrow |x\rangle = \frac{\sum_{i=1}^{n} x_i |i\rangle}{\| \sum_{i=1}^{n} x_i |i\rangle \|}$$

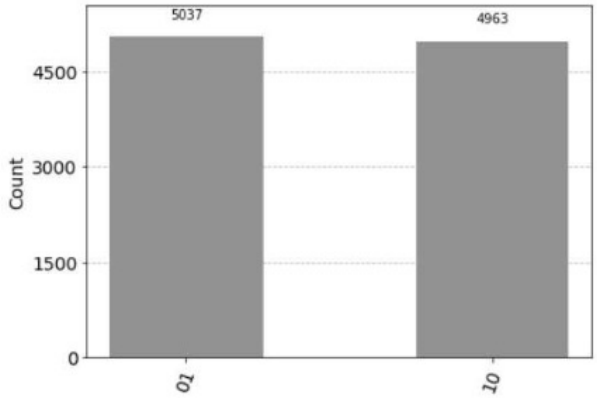
The measured result of our small HHL simulation is represented in the qubit 3 in the case the qubit 0 is one (see Figure 22.2 (b))

$$\mathbf{x}_m^2 = \begin{pmatrix} \frac{0.56}{0.562+0.0622} \\ \frac{0.0622}{0.562+0.062} \end{pmatrix}$$

with the measured value being

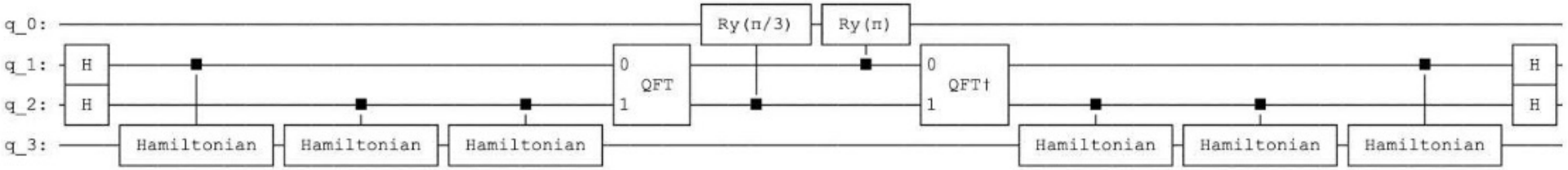
$$\mathbf{x}_m = \begin{pmatrix} 0.948869 \\ 0.31567 \end{pmatrix} \approx \mathbf{x}_n = \begin{pmatrix} 0.948683 \\ 0.316228 \end{pmatrix}.$$





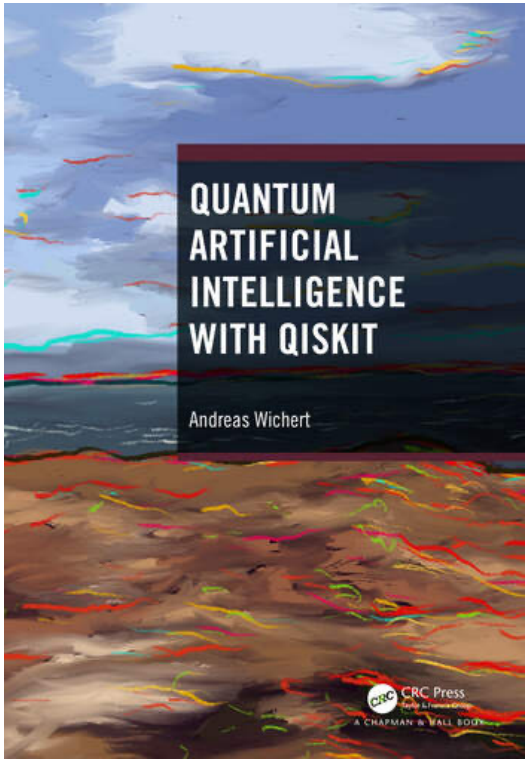
possible eigenvalues with two qubits: $|00\rangle,\ |01\rangle,\ |10\rangle,\ |11\rangle.$

- The rotation is conditioned on the binary representation of the eigenvalue
- For **each binary value we have a different operator**
  - We need to know the eigenvalues; we need to measure them.
  - Otherwise, the number of possible gates grows **exponentially** with the number of qubits!



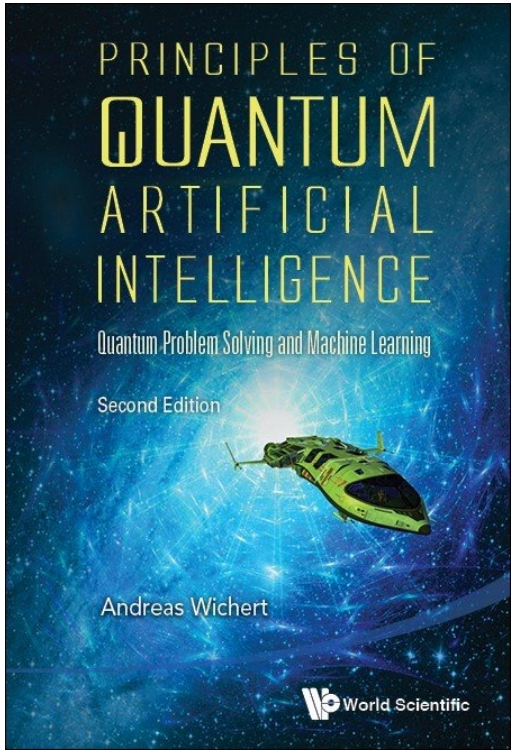
Conclusion

- Most quantum machine learning algorithms suffer from the input destruction problem
  - The efficient preparation of data is possible in part for sparse data.
- However, the input destruction **problem has not yet been solved**, and theoretical speed-ups are usually analyzed by ignoring the input problem, which is the main bottleneck for data encoding.
  - *qRAM* that can be used after measurement does not exist
- Other constraints:
  - Normalized representation of vectors
  - Quantum kernels with a “periodic” receptive fields
- Dilemma: should we ignore or marginalize those constraints or not?
  - Optimist: In the feature they will be solved
  - Pessimist: Till now, no theoretical advantage over classical algorithms on real data



- Chapter 21
- Chapter 22

Quantum Artificial Intelligence with Qiskit, A. Wichert, Chapman and Hall/CRC, 2024



- Chapter 14

Principles of Quantum Artificial Intelligence: Quantum Problem Solving and Machine Learning, 2nd Edition, A. Wichert, World Scientific, 2020