SENG 480D - Quantum Algorithms and Software Engineering

Theory and implementation of quantum algorithms including challenges and opportunities for quantum software developers and engineers using Qiskit, Q# SDK, Jupyter Notebooks, and Interactive Textbooks.

Lecture 01 Slide Deck:

- lecture-01-slide-deck-01.pdf
- lecture-01-slide-deck-02.pdf
- lecture-01-slide-deck-03.pdf

Slide Deck 01

The purpose of the course is to gain understanding of the three standard methods of expressing quantum computing: Dirac Notation, Braket Notation, and Quantum Circuits.

Quantum Computing is incredibly exciting because it is the only technology that we know of that could fundamentally change what is practically computable.

> This could soon change the foundations of chemistry, materials science, biology, medicine, and agriculture. - Fred Chong, University of Chicago

Course Description

> Nature isn't classical... and if you want to make a simulation of Nature, you'd better make it quantum... - Richard Feynman, CalTech, 1981

- **Quantum Computation**: Understanding the fundamentals of quantum information and computation.
- **Quantum Algorithms**: Deutsh-Jozsa, Grover, algorithms for graph problems (graph colouring), recasting problems to harness the power of quantum computing.
- **Complexity**: Understanding relations of classical and quantum complexity classes, quantum speed-up.
- **Building Blocks**: Quantum Phase Estimation (QPE), Quantum Fourier Transform (QFT); Variational Quantum Eigensolvers (VQE).
- **Hybrid Quantum Algorithm Design Techniques**: Variational Quantum Algorithms (VQA) and Quantum Approximate Optimization Algorithms (QAOA).

- **Quantum Software Engineering**: Hybrid quantum-classical computing, including problems and algorithm decomposition, HPC-QC runtimes, software infrastructure, and tool integration.

The course will use Jupyter Notebook Portfolio, IBM Qiskit, and Q# Microsoft Azure Quantum.

Hands-On Quantum Computing

- IBM Quantum: Learn Quantum Computing using Qiskit
- Microsoft Quantum: Quantum Development Kit and Q#
- Xanadu Interactive Codebook

Learning Outcomes

- **understand, describe and use** unitary gates, operations, circuits using Dirac or Braket notation
- **understand and use basic** linear algebra for quantum computation
- **design and implement basic quantum circuits using the** Qiskit & Q# Jupyter quantum development kits (QDKs)
- **understand executing quantum programs on a** quantum simulator **and on a** quantum computer
- **understand the** power of quantum computing, **including** quantum speedup **and** quantum advantage
- **understand and apply fundamental quantum information concepts & quantum computation concepts, including** superposition, entanglement, measurements, qubits, Bloch sphere, quantum gates & circuits
- **understand and describe** fundamental quantum algorithms, including Teleportation, Deutsch–Jozsa, Grover
- **understand and describe** computational problems & applications that can benefit from quantum computers
- **understand and use** building blocks of quantum algorithms and quantum development platform
- **understand and describe the notion of a** hybrid quantum-classical algorithm
- **characterize** quantum algorithm-design techniques and software-development strategies
- **understand and describe** variational quantum techniques & algorithms (VQA)
- **develop a** technical portfolio in quantum computing through assignments and projects
- **develop skills to** communicate in the quantum computing community

Linear Algebra

Linear Algebra - The Language of Quantum Computing

It is crucial to develop a good understanding of the basic linear algebra concepts to understand and appreciate many amazing and interesting aspects of quantum computation.

Euler's Identity is an equality found in mathematics that has been compared to a Shakespearean sonnet and described as "the most beautiful equation". It is a special case of a foundational equation in complex arithmetic called Euler's Formula, which the late great physicist Richard Feynman called in his lectures "our jewel" and "the most remarkable formula in mathematics".

Slide Deck 02

Quantum computation and quantum algorithms are different from classical computation and algorithms.

- Inherently different from common, recipe-like step by step instructions.
- No peaking in between individual algorithmic steps.
- Final result is obtained at the end of the computation through **measurement**.
- No Cloning Theorem.
- Superposition and Entanglement.
- Quantum Computing: generalization of probabilistic computation.

There are Qubits, Qubit States, Qubit Gates, and Qubit Circuits.

Qubit - Basic Unit of Quantum Information

Qubit: quantum bit described using two (computational) basis states, usually written as $|0\rangle$ (ket zero) and $|1\rangle$ (ket one).

A qubit can be in state ket zero, ket 1, and in linear combination of both (i.e., superposition).

Quantum State: mathematical representation of a qubit.

Ket Zero

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Ket One

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Superposition

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle; \;\; \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1.$$

Note: α and β are amplitudes.

$$|\psi\rangle \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Measuring qubit/quantum state results in either 0 or 1.

- 0 is measured with probability $|\alpha|^2$.
- 1 is measured with probability $|\beta|^2$.

Vectors, Dirac/Bra-Ket Notation, and Complex Numbers

Dirac Notation is a language designed to fit the precise needs of expressing states in quantum mechanics [1]. There are two types of vectors in Dirac Notation: the Bra Vector and the Ket Vector.

Bra Notation $\langle 0|$ is bounded by a left angle braket and a vertical bar. Ket Notation $|0\rangle$ is bounded by a vertical bar and right angle braket.

More generally, if ψ and ϕ are quantum state vectors, then their inner product is $\langle \phi|\psi\rangle$. This inner product implies that the probability of measuring the state $|\psi\rangle$ to be $\langle \phi|$ is $|\langle \phi|\psi\rangle|^2$.

---

d-Dimensional Complex Column Vector

Assuming $|\psi\rangle \in \mathbb{C}^d, \;\; \alpha_i \in \mathbb{C}$ we describe the following as Ket ψ.

$$|\psi\rangle = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_d \end{bmatrix}$$

d-Dimensional Complex Row Vector

Assuming $\langle \psi| \in \mathbb{C}^d, \;\; \alpha_i \in \mathbb{C}$ we describe the following as Bra ψ. Or in other words, the conjugate transpose of Ket ψ.

$$\langle \psi| = [\alpha_1^* \quad \alpha_2^* \quad \dots \quad \alpha_d^*]$$

Complex Numbers

Assuming $\alpha \in \mathbb{C}$, and $e^{i\theta}$ is the phase factor, and θ is the phase. And assuming...

Euler's Formula: $e^{i\theta} = \cos\theta + \sin\theta$...

$$
\begin{aligned}
\alpha &= a + bi,\ a, b, \in \mathbb{R} \text{ and } i = \sqrt{-1} \\
\alpha &= re^{i\theta},\ r \in \mathbb{R} \text{ and } \theta \in [0, 2\pi) \\
&= r(\cos\theta + i\sin\theta)
\end{aligned}
$$

Where $r(\cos\theta + i\sin\theta)$ is the polar form.

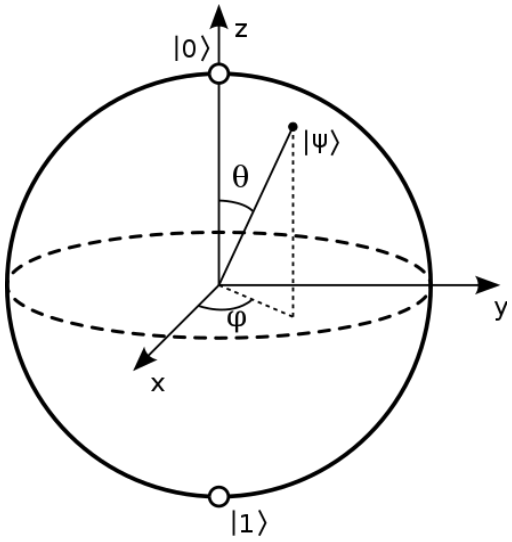
Magnitude

Assume $|\alpha| = \sqrt{\alpha\alpha^*}$ where complex conjugate $\alpha^* = a - bi = re^{-i\theta}$. We note $|e^{i\theta}| = 1 \Rightarrow |\alpha| = r$.

Bloch Sphere

A bloch sphere is a unit sphere.

- Antipodal points: pair of mutually orthogonal state vectors.
- Choice of north and south poles: computational basis vectors.
- Point on surface: pure state; used for pure state quantum computation.



$$
\begin{aligned}
|+\rangle &= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)
\end{aligned}
$$

$$
\begin{aligned}
|-\rangle &= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \\
&= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)
\end{aligned}
$$

Slide Deck 03

Goals: Install Qiskit, Access IBM Quantum Experience, Install Q#, Access Azure Quantum.

Quantum Programming

Assemble a sequence of instructions to be run on a quantum processor (QPU). A sequence of instructions equals Quantum Circuits.

Software Development Kits: Qiskit, PennyLane, and Cirq.

Quantum Programming Languages: Q#

Qiskit

Qiskit is open source SDK for working with quantum computers [2]. It works with OpenQASM (Open Quantum Assembly Language) and IBM Quantum Computers. OpenQASM is an intermediate representation for quantum instructions.

Q#

Q# is domain specific programming language. It is a part of the Microsoft Quantum Development Kit (QDK). It supports a basic procedural model for writing programs with loops, if/then statements, and common data types. It introduces new quantum specific data structures and operations.

Quantum Katas

Tutorials and programming exercises for learning Q# and quantum computing [3].

---

1. https://learn.microsoft.com/en-us/azure/quantum/concepts-dirac-notation↩
2. https://qiskit.org↩
3. https://github.com/Microsoft/QuantumKatas↩