What is Quantum Computing, essentially?

Prantik, 30/10/2025

The Prerequisites

1. Qubits

Classically, we use **bits**. Now, we use ***qu*bits**. Unlike bits with 0 and 1s, qubits can store either and be in superposition.

2. Superposition

With the help of agents like microwave pulses, qubits exists in both states till infered. This state allows for simultaneous calculations via parallelism (since every state is operated on at once).[1]

3. Entanglement

Qubits can interlink with others' states allowing faster computation without space concerns.

4. Quantum Interference[2]

Essentially the usage of wave-like interference of probabilities from superimposed paths to propagate desired paths for refining results of algorithms from output states.

[1]Why force unitary (thus invertible) operations? (inner product preservation?) [2]Decoherence is a problem - deal if only it comes up it?

The Definition

Quantum computing, in principle, leverages the parallel processing power of superposition and coordination of entaglement to outperform calculations of classical computers, provided a few restrictions on the type if operations.

Two key things to consider when under applications of it would be:

1. ## Lack of worry about dimensionality

   Classical practices worry about dimensions and have to apply dimensionality reduction to negoatiate a lack of computing power. However, parallelism enables a vast output space, so more accurate and quicker results can be obtained.

2. ## Parallel updates to space

   Every gridpoint can be iteratively updated with data - which can be found on every timestep on a dynamic problem. Streaming operations can speed up analysis of data at miniscule intervals.

Quantum Gates

Since quantum operations need to be unitary, gates that lose state like AND or OR become unusable. However, take the NOT gate for example:

$$\text{Take general state, } |\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$U_{NOT}|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \alpha|1\rangle + \beta|0\rangle$$

$$UU^{\intercal} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

Similarly, multiple such gates of operations can be crafted, with multiple qubits, like the XNOR or SWAP gates. Yet another notable mention is the Toffoli's gate - a 3-qubit gate that provides behaviour similar to the classical AND gate, while storing the output state in the third qubit, making the whole process invertible.

Also of note is the Hadamard gate - a single qubit gate that turns a read, singular state into a superposition for operation.

In Implementation

These gates come together to implement quantum circuits. Building on top of these gates gives us the models of computation to implement various algorithms. At a higher level, we use these gates and circuits in our software. There are two questions here: about service providers, and SDKs.

Framework	Paradigm	Hardware	Limitations	Use-case
Qiskit	OO-Pythonic	IBM Hardware	Vendor-locking	Hardware prototying
Cirq	Procedural-Pythonic	Google Sycamore	Small low-level community	NISQ
Q#	Own-.NET like	Microsoft Azure	Full language	Hybrid approaches

Services provided

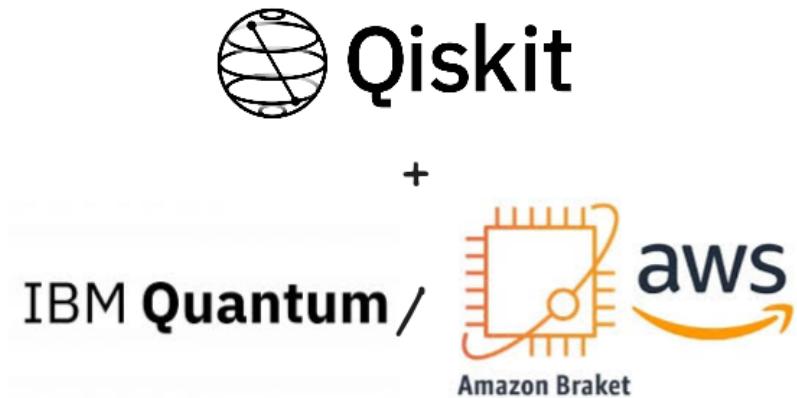
Of all the service providers, a few narrowed down, provide popular services. Honourable mentions include Google Quantum AI and D-Wave[1]

Company	Student / Academic Plan	Key Use Cases	Core Features
IBM Quantum	IBM Quantum Learning: Qiskit, free simulators, and classroom modules.	Algorithm prototyping, quantum chemistry, CFD.	20+ quantum systems · Qiskit Runtime APIs · Costing variety
Amazon Braket (AWS)	AWS Braket for Education: Credits for students & educators.	Hybrid CFD modeling, supply optimization, AI-driven simulation.	Hybrid EC2 services · Pay-per-use sandboxes · Multi-hardware access.
Microsoft Azure Quantum	Azure Quantum for Educators: credits & Q# integration.	Optimization, hybrid solvers, quantum algorithm learning.	Hardware-agnostic QDK · Does academic partnerships.

[1] Look into Quantum annealing and if its use of optimisation helps

Overall Decisions

- Of all the SDKs provided, Qiskit stands out for it’s community support. Being integrated into python also helps over Q#, which is a separate DSL itself.
- As for services, there’s IBM which leads with its runtime access and community, while on the other hand Amazon’s services are between Braket and Azure, the former being more of an user facing vendor while the latter provides more Q# based support for Amazon hardware.

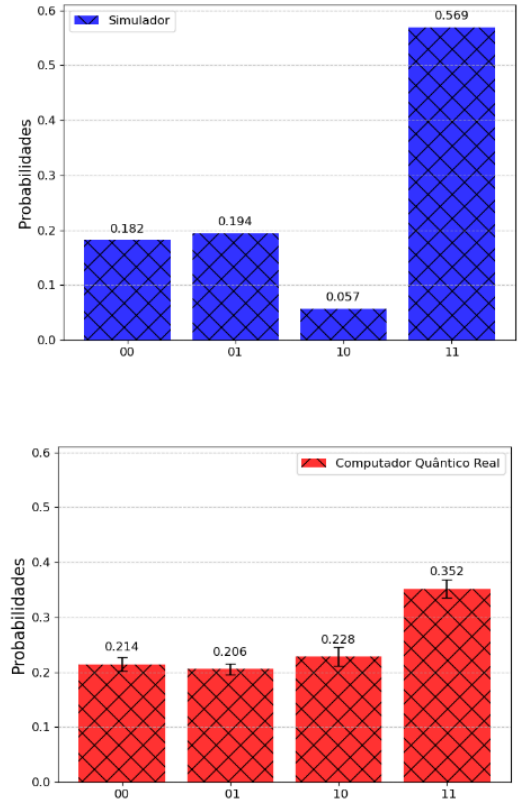


Comparing operations (Linear systems)

Classically, methods like Gaussian Elimination and Conjugate Gradient are popular. They tend to run on time complexities of $O(n^3)$. On the other hand, a quantum algorithm like HHL runs on polynomial time.

Compared to the process of Gaussian Elimintation however, HHL has some pre-requisites and systems to extract output. This includes preparation of state, eigendecomposition of $|b\rangle$ (given $|A\rangle\, x = |b\rangle$), amplitude encoding to get the output vector and reading the solution.

A general implementation however yields a lot of noise on non-ideal hardware, as shown.



Algorithm results for an ideal noiseless simulator (top) and a real quantum computer (bottom).

Comparing operations (ODEs / PDEs)

ODEs and PDEs are the main crux that we face. Broadly, there are no general algorithms that can apply to every kind of ODE or PDE. However, the general workflow remains very similar. First, the problem is discretised within its intial and boundary conditions, followed by state preparation steps like normalisation. Then the quantum circuit is utilised for calculating a solution and a final measurement is taken.

Multiple types and kinds of quantum algorithms can be used. This includes both general and specific use algorithms. Pade approximation, VLQS, and more can be embedded into specific use cases like the Quantum Lattice Boltzmann Method and other hybrid Quantum-Classical CFD solvers, using the quantum primitives as necessary.

Table of contents

References used

1. What is Quantum computing?
2. Brief on popular Quantum-as-a-service offers.
3. Quick overview on Quantum computing terms.
4. IBM’s draft path on creating headways into fault-tolerance
5. Solving linear systems using HHL with Qiskit
6. Evaluating incompressible NSE on noisy quantum hardware
7. ODEs using Pade approximation.
8. Implemented HHL in Qiskit, evaluating PDEs
9. PDEs using Quantum computing.
10. Quick rundown of a simulated 32-qubit CPU service (Blue-Qubit)