Programming Quantum Computers (Apps I: Simulation)

(**Subtrack of** Quantum Computing: An App-Oriented Approach)

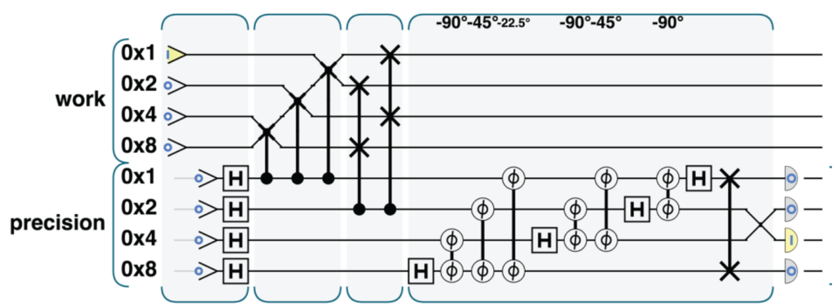
Moez A. AbdelGawad

moez@{cs.rice.edu, alexu.edu.eg, srtacity.sci.eg}

Sat., Dec. 14th, 2019

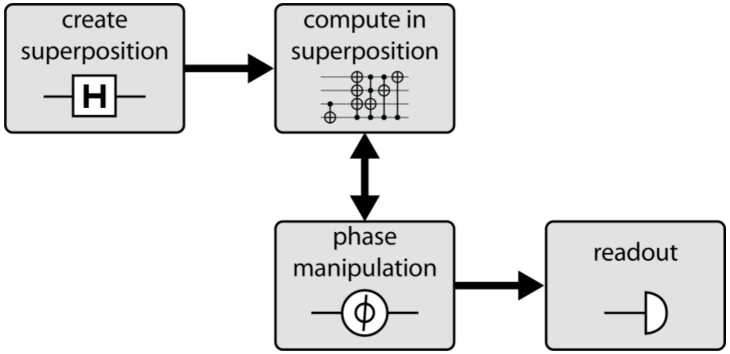
Quantum Computers are Real

- What are they useful for?
  - Let's discover, by programming them!
  - And seeing *examples* of how others programmed them.
- A hands-on approach to programming QCs/QPUs.
  - By doing; i.e., by writing code & building programs.
  - Using simulators, since real QCs are harder-to-access (so far).
- Goals: Read, understand, write, and *debug* quantum programs.
  - Ones like this program.



Structure of Quantum Apps

- Tendency to such structure, very roughly.
- Compute in superposition.
  - Implicit parallelism.
- Phase manipulation.
  - Practicality. Relative phase info directly inaccessible (unREADable).
- Modules are combined (*composed*) to define full quantum application.
  - Possibly in *iterations*.
- Quantum programming is an art (too).



Quantum Modules Covered

Module	Type
Digital arithmetic and logic (AL)	Compute in superposition
Amplitude amplification (AA)	Phase manipulation
Quantum Fourier transform (QFT)	Phase manipulation
Phase estimation (PE)	Phase manipulation
Quantum data types (Sim)	Superposition creation

Topics Covered So Far

- Introduction:
  - Qubit, Superposition, and Entanglement.
  - Single-Qubit Ops: H, NOT and Phase.
  - Multi-Qubit Ops: Conditional Ops (e.g., CNOT).
  - Teleportation.

- Modules:
  - Quantum Arithmetic and Logic.
  - (Quantum) Amplitude Amplification.
    - Converting phase info into magnitude info.
  - Quantum Fourier Transform.
    - Revealing patterns (frequencies).
  - (Quantum) Phase Estimation.
    - Characterization of quantum operations.
  - **Quantum Simulation and Real Data.**
    - **QRAM, Quantum Vector & Matrix Encodings.**

Quantum Apps

- **Quantum Simulation.**
  - Using quantum operations to approximate unitary matrices that describe quantum operations representing Hermitian matrices (the Hamiltonians)!
- Quantum Search (Grover's algorithm).
- Quantum Graphics (Quantum Supersampling).
- Quantum Cryptography (Shor's algorithm).
- Quantum Machine Learning (QML).

QUANTUM APPLICATIONS

REAL DATA (QUANTUM SIMULATION)

Lecture Outline

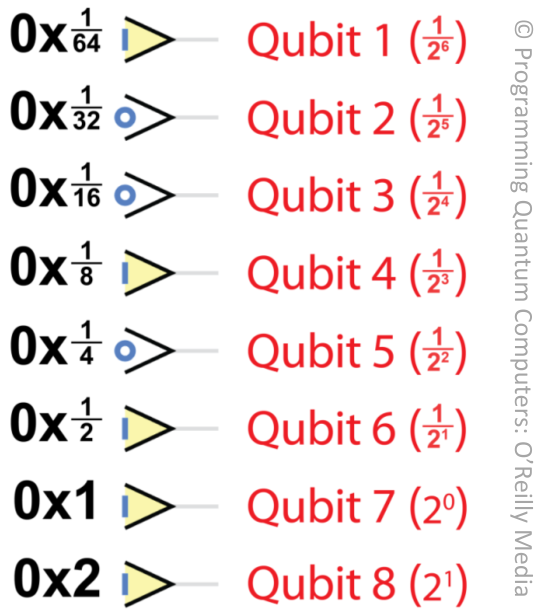
- Quantum Data Structures.
- Quantum Fixed-point Representation.
- QRAM: Large Quantum Data, in Superposition.
- Representing Vectors.
  - State encoding.
  - Amplitude encoding.
- Representing Matrices.
  - As quantum operations, described by unitary matrices.
  - Quantum simulation & the Hamiltonian.

Noninteger Data

- Programming Mantra: A good data structure can be just as important as a good algorithm.
  - Quantum integers (in binary or in 2’s complement encodings) are useful, but not enough.
  - How to represent more complex data in a QPU?
    - E.g., rational numbers (with fractions), vectors of numbers, and matrices of numbers.
    - Quantum encodings of data.
  - How to enter large amounts of stored data *into* a QPU?
    - A challenge, due to *superposition*.

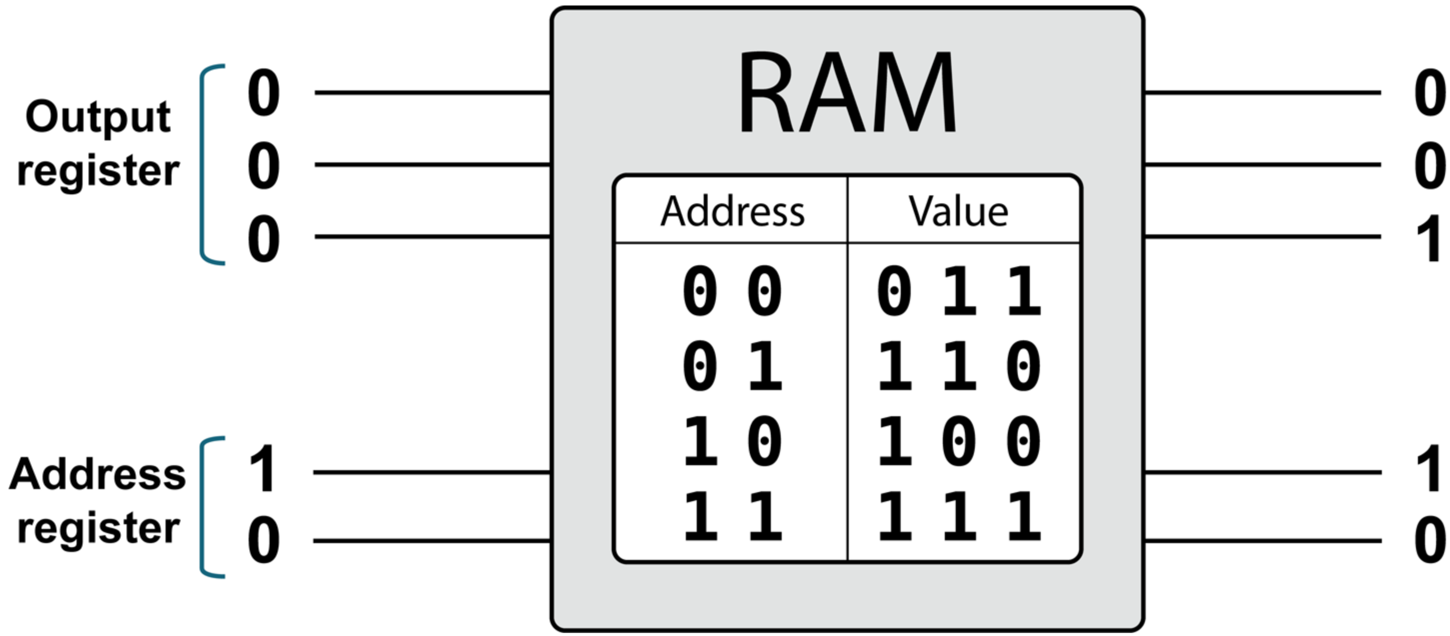
Fixed-point Representation

- Q notation: *Qn.m*.
  - Integer part, and fractional part.
  - E.g., representing the number 3.640625 ($= 3 + \frac{1}{2} + \frac{1}{8} + \frac{1}{64}$) in a qubyte using a Q8.6 encoding.



RAM

- Conventional RAM.

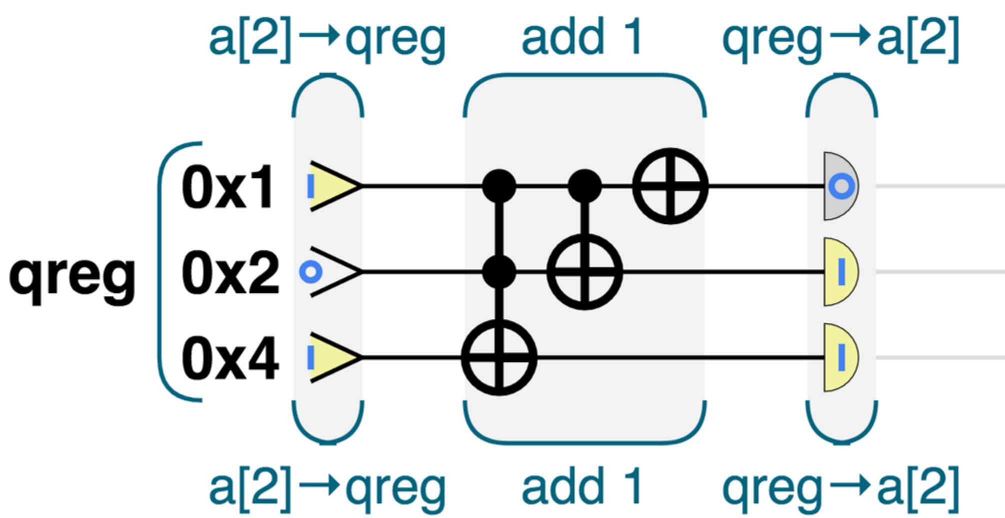


© Programming Quantum Computers: O'Reilly Media

(Typo: Address register should be 10 not 01.)

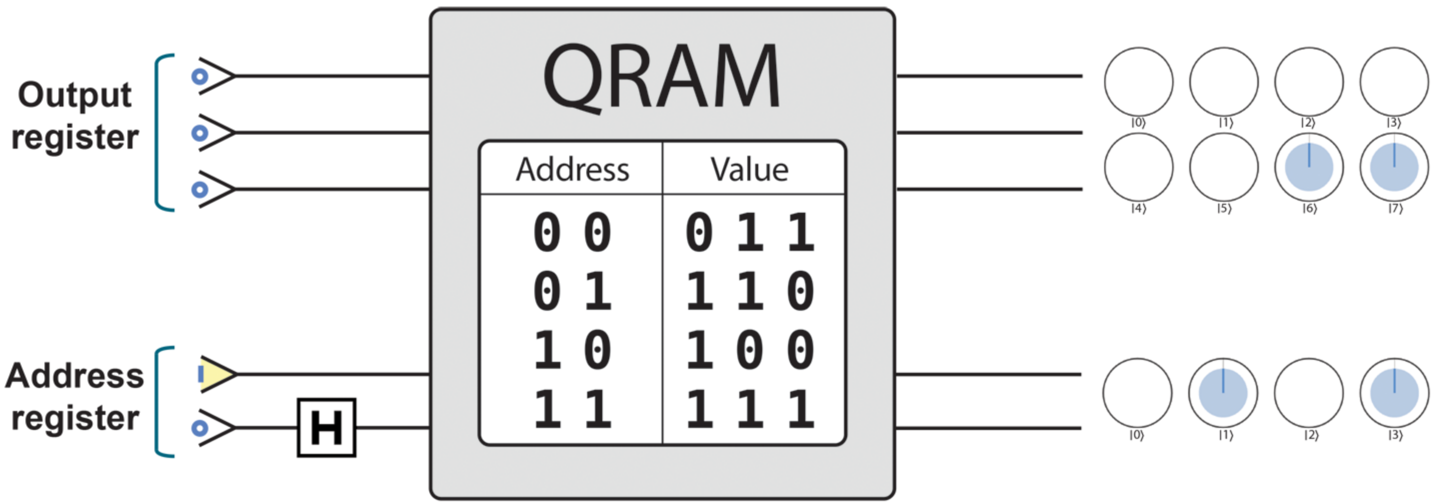
RAM → QPU

- Hands-on: Using a QPU to increment a number in RAM.



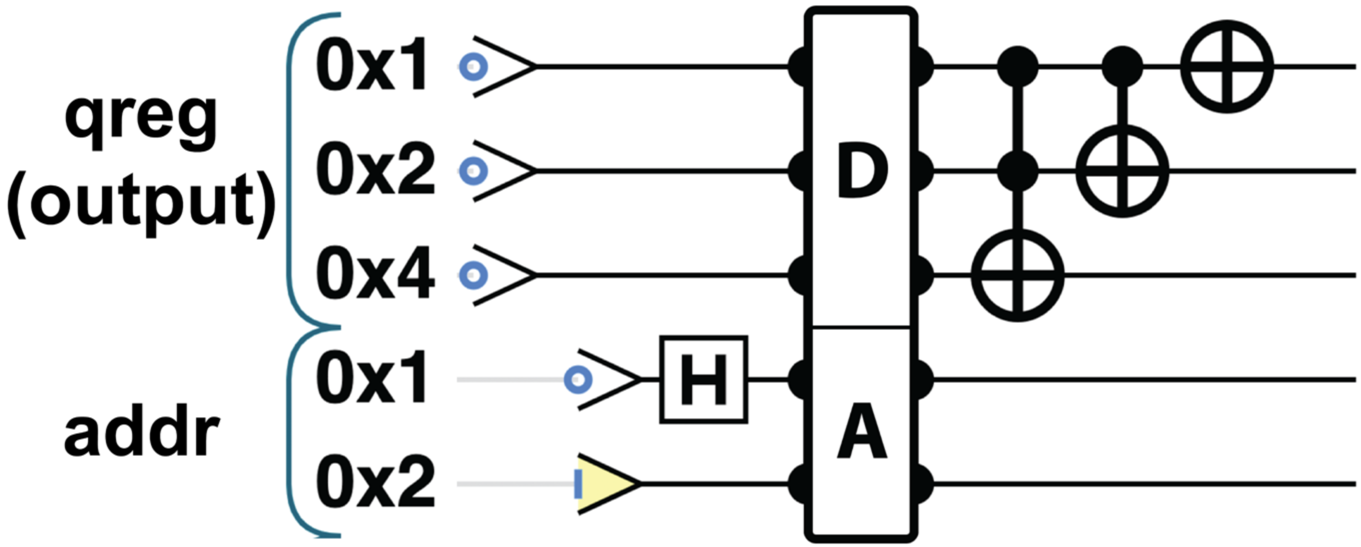
© Programming Quantum Computers: O'Reilly Media

QRAM



- Superposition in address register.

QRAM

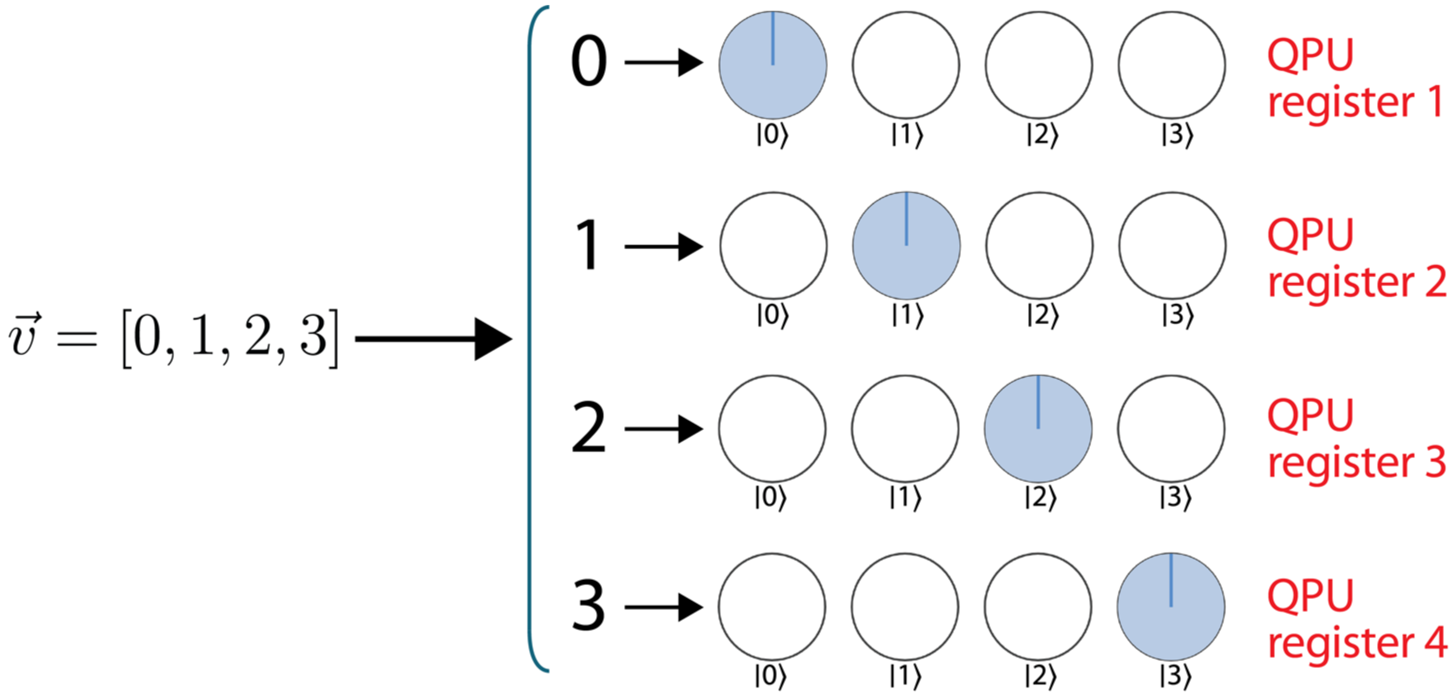


© Programming Quantum Computers: O'Reilly Media

- (<u>Hands-on</u>) Conventional simulation of QRAM provided in QCEngine. Slow.

Vector Representation

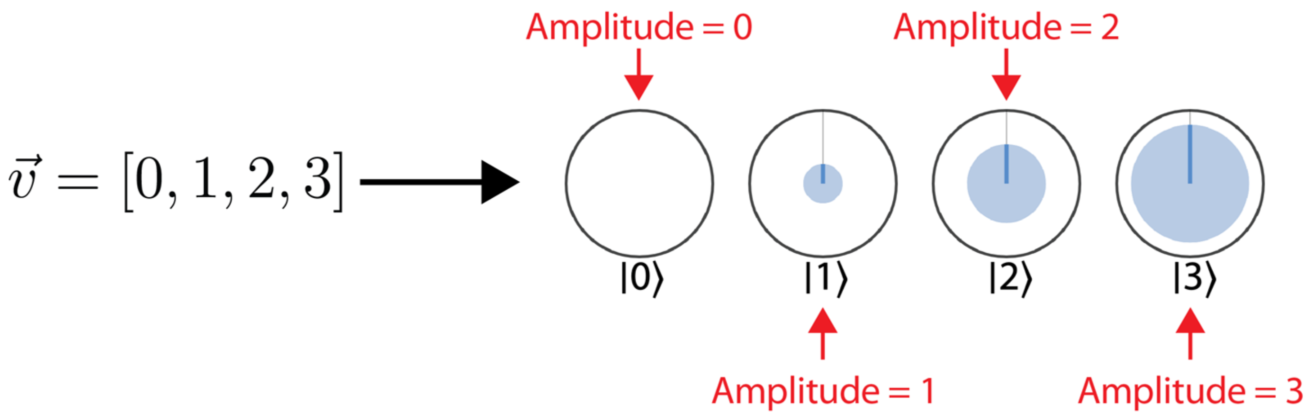
- State encoding.



© Programming Quantum Computers: O'Reilly Media

Vector Representation

- Amplitude encoding.



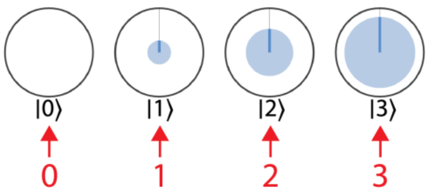
© Programming Quantum Computers: O'Reilly Media

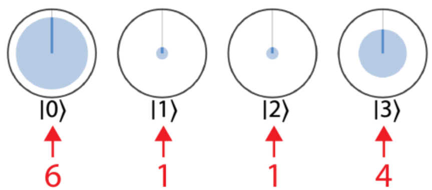
Amplitude Encoding

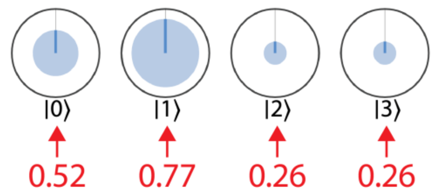
[0, 1, 2, 3]

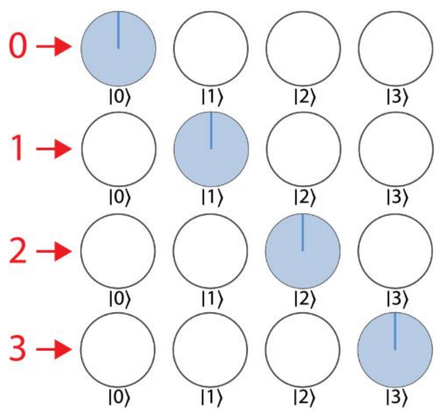
[6, 1, 1, 4]

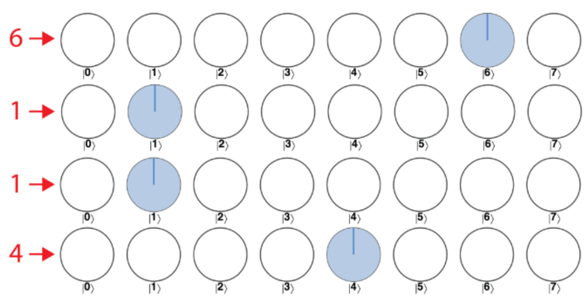
[0.52, 0.77, 0.26, 0.26]

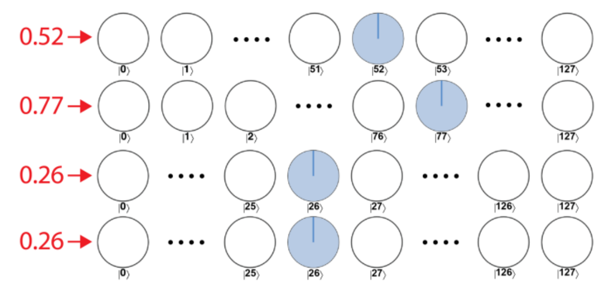










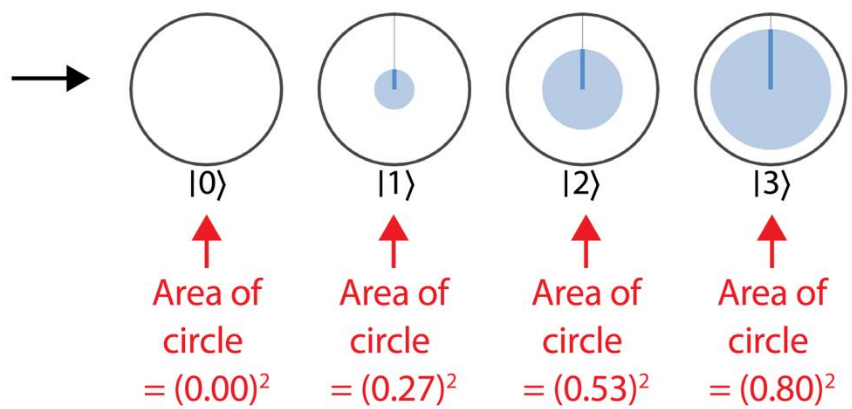


Amplitude Encoding

- Normalization.

$$\vec{v} = [0, 1, 2, 3]$$

$$\longrightarrow [0.00, 0.27, 0.53, 0.80]$$



Amplitude Encoding

- <u>Hands-on</u>: Try `amplitude_encode()` in QCEngine.

- Limitations:
  - Quantum superpositions are unREADable.
    - Only *one* value at a time can be read.
    - General Precaution: Beware of (unREADable) quantum outputs.
      - In quantum programming, there is always a *computational cost* for obtaining a conventional output.

  - Normalized vectors.
    - Sum of squares = 1.
    - Normalization factor can be kept in another register.

Matrix Representation

- Matrices *acting* on vectors (i.e., multiplying matrices with vectors) is so common.
  - Vectors are encoded as quantum states.
  - Similarly, matrices are encoded as quantum ops.

- Quantum Simulation.
  - Basic idea: *Hermitian* matrices can be modeled (i.e., represented) by efficient QPU operations that are described by *unitary* matrices ($H \to O \to U$).

  - H is Hermitian if

$$H = H^{\dagger} = (H^{*})^{T}$$ ($\dagger$ = adjoint = transpose ^{T} of conjugate *)

Matrix Representation

- Quantum Simulation.
  - Any $m \times n$ *non*-Hermitian matrix X can be simulated by the encoding of a twice-larger $2m \times 2n$ Hermitian matrix.

$$H = \begin{pmatrix} 0 & X \\ X^\dagger & 0 \end{pmatrix}.$$

- Unitary matrices: $UU^\dagger = I$.
  - Describe *reversible* quantum ops.

- Math Fact:
  - Given a matrix H, let $U = e^{-iHt}$. If H is Hermitian, then U is unitary.
    - Parameter t is a hardware implementation detail (glossed over for simplicity).

Quantum Simulation

- Efficiently provide a circuit performing exponentiation of H.

- Primarily used to *simulate* quantum mechanical systems (e.g., molecular or materials simulations).
  - The Hamiltonian H of a system–a Hermitian matrix–describes the simulation; the QPU operation e^{-iHt} predicts how the system evolves over time.
  - Used heavily in quantum chemistry.
    - E.g., for simulating drug and protein structure.

- Finding U for particularly simple H is relatively easy.
  - E.g., if H is a diagonal matrix, or if H is a sparse matrix.
  - No examples in textbook (only high-level description).

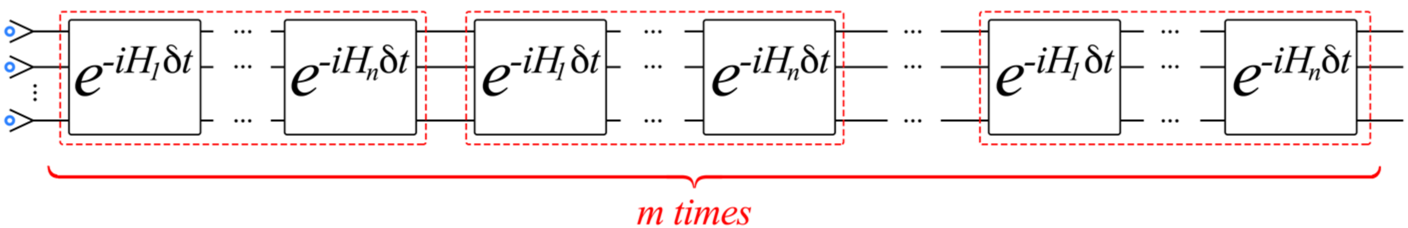
Quantum Simulation

- Break down difficult-to-encode matrices H into a number of easier-to-encode ones.
  - (Step 1) Deconstruct.
    - Split H into a sum of simpler Hermitian matrices:
      $$H = H_1 + \cdots + H_n.$$
  - (Step 2) Simulate components $H_1, \ldots, H_n$ efficiently.
  - (Step 3) Reconstruct.
    - Build a circuit for H using circuits for the H_i.

- Quantum simulation approaches differ significantly in how to achieve Step 1 (deconstruction) and Step 3 (reconstruction).
  - A common group of methods is called *product formula methods.*
  - We cover reconstruction (Step 3) before deconstruction (Step 1).

Matrix Representation (Quantum Simulation): Product Formula Methods

- Reconstruction (Step 3):
  - If $H = H_1 + \cdots + H_n$, then H can be constructed based on the *Lie product formula* (LPF).
    - LPF: $U = e^{-iHt}$ can be *approximated* by performing in sequence $U_i = e^{-iH_it}$ for $i = 1..n$ for very short times δt, then repeating the whole sequence a number of times m.



© Programming Quantum Computers: O'Reilly Media

    - LPF: *U* approximation is efficient if all H_i representations (i.e., all U_i) are efficient.

Matrix Representation (Quantum Simulation): Product Formula Methods

- Deconstruction (Step 1):
  - Many different approaches; most are quite mathematically involved.
  - One approach: if H is sparse, consider H as the adjacency matrix of a graph, and solve a certain coloring problem on this graph.
    - Vertices with the same color are grouped to form elements of an H_i.
    - Graph Coloring (in graph theory): Associating a color to each vertex of an input graph without having two adjacent vertices (i.e., ones directly connected by an edge) share the same color.

Cost of Quantum Simulation

- Got a *feel* for how to go about representing matrix data as QPU ops.
  - Many, many different approaches.

- Runtimes of different quantum simulation techniques and approaches.

Technique	Circuit Runtime
Product formula	$O(d^4)$
Quantum walks	$O(d/\sqrt{\epsilon})$
Quantum signal processing	$O(d + \frac{\log(1/\epsilon)}{\log\log(1/\epsilon)})$

  - Parameter d is a measure of the sparsity of matrix H (the maximum number of nonzero elements *per row*), and parameter ϵ is a measure of the desired precision of the representation of H.

Discussion: e^{iz}

$$e^{\pi i} + 1 = 0$$
$$e^{i\theta} = \cos\theta + i\sin\theta$$

FT: $\int_{-\infty}^{\infty} f(t)e^{-2\pi i f t}.dt$

Simulation: $U = e^{-iHt}$

The unusual effectiveness of complex numbers ($i = \sqrt{-1}$, 'Road to Reality') and the exponential function (effecting rotations and symmetries in the Argand diagram/complex plane, in the Bloch sphere … and group theory)

Discussion

Q & A

Next Lecture APPetizer

- In next lecture (isA):
  - Our Second Quantum App: Quantum Search.
    - How Grover's algorithm can be used to locate a particular data item in an unstructured set of data.
      - More efficient than current conventional methods.
    - Phase Logic: What is it?
    - Solving logic puzzles—of Kitten and Tigers.
    - Solving Boolean satisfiability (SAT) problems.

Course Webpage

http://eng.staff.alexu.edu.eg/staff/moez/teaching/pqc-f19

- Where you can:
  - Download lecture slides (incl. exercises and homework).
  - Check links to other useful material.

Thank You