Introduction to quantum computing (Part I)

Anthony Leverrier, Inria Paris

Quantum computing and scientific research:
state of the art and potential impact in nuclear physics

Outline of the course

course 1: Basic quantum mechanical principles of quantum computing

- motivation for studying quantum algorithms
- the various models for quantum computing
- the circuit model
- a simple quantum algorithm: Deutsch-Josza

course 2: More advanced quantum algorithms

- Grover's algorithm for search
- linear systems (HHL) and machine learning
- quantum supremacy
- the NISQ era

Related material

This course is largely inspired from the remarkable set of notes by Ronald de Wolf, available online.

- Quantum Computing: Lecture Notes by Ronald de Wolf
  `http://homepages.cwi.nl/~rdewolf/qcnotes.pdf`

Other ressources include:

- the classic "Quantum computation and quantum information" by Nielsen & Chuang
- Lecture notes by John Preskill
  `http://www.theory.caltech.edu/people/preskill/ph229/`

The end of Moore's law

Intel Delays Mass Production of 10 nm CPUs to 2019

by Anton Shilov on April 27, 2018 12:20 PM EST

https://www.anandtech.com/show/12693/

intel-delays-mass-production-of-10-nm-cpus-to-2019

	Intel First Production
1999	180 nm
2001	130 nm
2003	90 nm
2005	65 nm
2007	45 nm
2009	32 nm
2011	22 nm
2014	14 nm
2016	~~10 nm~~
2017	~~10 nm~~
2018	10 nm?
2019	10 nm!

miniaturization reaches levels where quantum effects become non-negligible. One can either try to suppress them or to *exploit them*.

Why study quantum computing?

- investigation of the computational power of *computer based on quantum mechanical principles*

power of the strongest possible computing devices allowed by *Nature*?

- main objective: find *algorithms with speedup compared to classical algos*
  - already possible for simulating physics (e.g. talk of Jens Eisert tomorrow)
  - can we get a proven speedup? race to *quantum supremacy* (boson sampling, random circuits)
  - not too distant future: chemistry problems, optimization
  - longer term: universal quantum computer: factorization (Shor), complex algorithms (requires active quantum error correction)

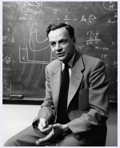
Genesis of quantum computing

Feynman 1981

“Can quantum systems be probabilistically simulated by a classical computer? [. . .] The answer is almost certainly, No!"

$\Longrightarrow$ use quantum systems to simulate quantum systems!

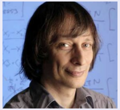
$\Longrightarrow$ birth of *quantum simulation*



Deutsch 1985

- quantum Turing machine
- existence of a universal machine

$\Longrightarrow$ birth of *quantum computing*



Bernstein, Vazirani 1993

- efficient quantum Turing machine (complexity class BQP)
- Bernstein-Vazirani problem: $f : \{0, 1\}^n \to \{0, 1\}$ such that $f(x) = a \cdot x$
  Find a.

  $\Longrightarrow$ possible with 1 quantum query vs n classically

The first useful algorithms

Simon, Shor 1994

exponential speedups for

- period finding
- factoring!! *very surprising* $\Longrightarrow$ sparked a lot of interest in the field
- discrete logarithm

$\Longrightarrow$ exploits Quantum Fourier Transform

$\Longrightarrow$ consequences for public-key cryptography: breaks most cryptosystems deployed today

Grover 1996

- search an n-item list with $O(\sqrt{n})$ queries
- lots of applications (find collisions, approximate counting, shortest path)

but only quadratic improvement

Can we really compute with a quantum computer?

A word (or two) about the feasibility of a quantum computer

QUANTUM COMPUTING: DREAM OR NIGHTMARE?

The principles of quantum computing were laid out about 15 years ago by computer scientists applying the superposition principle of quantum mechanics to computer operation. Quantum computing has recently become a hot topic in physics, with the recognition that a two-level system can be pre-

Recent experiments have deepened our insight into the wonderfully counterintuitive quantum theory. But are they really harbingers of quantum computing? We doubt it.

$\boxed{\text{Serge Haroche and Jean-Michel Raimond}}$

two interacting qubits: a "control" bit and a "target" bit. The control remains unchanged, but its state determines the evolution of the target: If the control is 0, nothing happens to the target; if it is 1, the target undergoes a well-defined transformation.

Quantum mechanics admits additional options. If

"Therefore we think it fair to say that, unless some unforeseen new physics is discovered, the implementation of error-correcting codes will become exceedingly difficult as soon as one has to deal with more than a few gates. *In this sense the large-scale quantum machine, though it may be the computer scientist's dream, is the experimenter's nightmare.*" (Physics Today 1996)

A theory of quantum error correction and quantum fault-tolerance

Peter Shor

- quantum error-correcting codes (1995)
- fault-tolerant syndrome measurement (1996)
- fault-tolerant universal quantum gates (1996)

Alexei Kitaev

- topological quantum codes (1996)
- computing with nonabelian anyons (1997)
- magic state distillation (1999-2004)
- Majorana modes in quantum wires (2000)

Threshold theorem (Aharonov, Ben-Or (1997): quantum computation is possible provided the noise is sufficiently low (below some constant)

What is a quantum algorithm?

an algorithm is a systematic process to obtain an answer to a problem

Examples of problems

- *factorization*: given $N \in \mathbb{N}$, find p, q such that $N = p \times q$
- *optimization*: given a matrix Q, compute $\min \sum_{i,j=1}^{N} x_i Q_{i,j} x_j \quad \text{with} \quad x_i \in \{0, 1\}$
- *search*: given a list of items $x_1, \ldots, x_N$ in a database and *given access to a function* $f : x_i \mapsto f(x_i) \in \{\text{marked, not marked}\}$, find a marked element
- *linear algebra*: given $N \times N$ matrix A and vector $\vec{b} \in \mathbb{C}^N$ available as a *quantum state* $|b\rangle \propto \sum_{i=1}^{N} b_i |i\rangle$, solve $A\vec{x} = \vec{b}$ in the sense of preparing the state $|x\rangle \propto \sum_{i=1}^{n} x_i |i\rangle$.

for a *quantum* algorithm, we are allowed to manipulate quantum states (= prepare states, apply some evolution or measurement) to obtain the result

Different models of quantum computing

Different kinds of quantum algorithms

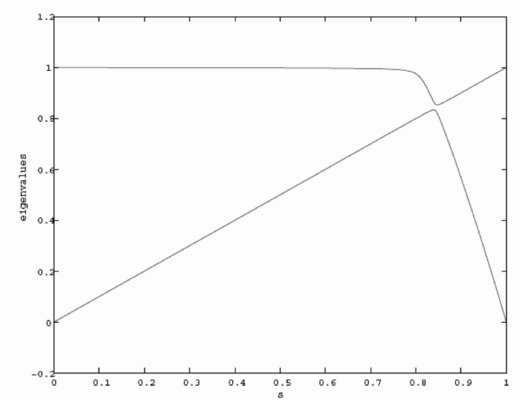
General template of a quantum algorithm:

- prepare some simple *intial state* $|\psi_{\text{init}}\rangle$
- depending of the classical input of the problem, *apply some transformation* to the state $|\psi_{\text{init}}\rangle \longrightarrow |\psi_{\text{fin}}\rangle$
- *measure the final state* and obtain the (classical) solution

Different kinds of transformation $\implies$ *different models of quantum computation*

- *standard circuit model*: $|\psi_{\text{fin}}\rangle = U|\psi_{\text{init}}\rangle$ where the unitary U is decomposed in many elementary gates
- *measurement based quantum computing*: measure the subsystems one at a time and adapt future measurements to previous results
- *adiabatic quantum computing*: $|\psi_{\text{init}}\rangle$, $|\psi_{\text{fin}}\rangle$ unique ground states of an easy to prepare Hamiltonian H_{init} and an objective Hamiltonian H_{fin}. Slowly evolve the Hamiltonian $H(t) = (1-\alpha(t))H_{\text{init}} + \alpha t H_{\text{fin}}$ to remain in the ground state all along.

Adiabatic quantum computing



problem: the eigenvalue gap might be extremely small, so one needs to slow down the evolution comparatively

$$t_{\text{fin}} = O\left(\frac{1}{\text{gap}}\right)$$

The 3 models are equivalent

A quantum algorithm is *efficient* if its cost is *polynomial* in the size of its input:

- ex of input size
  - factorization of N: number of digits $= \log_2 N$
  - search in database: size N of database
  - optimization problem: number N of variables
- how to count the cost:
  - number of elementary gates in the circuit
  - number of calls to a function f
  - time to go from H_{init} to H_{fin}

Theorem

if a problem can be solved in one of the 3 models, it can also be solved in the other two (up to some potential polynomial slowdown)
$\Longrightarrow$ same notion of efficiency in the 3 models

$\Longrightarrow$ we will mostly focus on the *quantum circuit model* (closer to classical CS)

The quantum circuit model

The support of (quantum) information

We want to compute stuff, so we need the quantum equivalent of bits: i.e. two-level quantum system

$$\textit{qubit}: \qquad \alpha|0\rangle + \beta|1\rangle \in \mathbb{C}^2, \qquad |\alpha|^2 + |\beta|^2 = 1.$$

It doesn't matter to us what $|0\rangle$ and $|1\rangle$, only that $\langle 0|1\rangle = 0$.

Another very useful basis: $\{|+\rangle, |-\rangle\}$ with $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Requirements to implement a quantum algorithm

- be able to prepare many copies of $|0\rangle$, say
- be able to apply certain unitaries (gates of a circuit) to the qubits
- be able to measure a qubit in some basis, e.g. computational basis $\{|0\rangle, |1\rangle\}$.

States, evolution, measurement

- a single qubit isn't sufficient to solve interesting problems. We need n qubits:

$$|\psi\rangle = \alpha_{0\cdots 00}|0\cdots 00\rangle + \alpha_{0\cdots 01}|0\cdots 01\rangle \cdots + \alpha_{1\cdots 11}|1\cdots 11\rangle$$

$$\text{with} \quad \sum |\alpha_{\vec{i}}|^2 = 1 \quad \text{(normalization)} \quad \text{and} \quad |i_1 i_2 \cdots i_n\rangle := |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle$$

  in practice, one needs to deal with *decoherence*, and therefore *mixed states, not only pure states*. Quantum fault-tolerance/error-correction techniques can be applied to deal with such issues (threshold theorem).

- a quantum computation essentially consists in applying some *unitary* U (such that $UU^\dagger = \mathbb{1}$) to $|0\rangle^{\otimes n}$ or to some input state $|\psi\rangle$ given to us, and measure the final state $U|0\rangle^{\otimes n}$ or $U|\psi\rangle$ in the computational basis.

- the measurement returns the string $\vec{i} \in \{0,1\}^n$ with probability

$$\mathbb{P}(\vec{i}) = |\langle \vec{i}|\psi\rangle|^2 = |\alpha_{\vec{i}}|^2$$

  This is our result.

Quantum algorithm

In the circuit model, the meat of the algorithm is the unitary U:

$$|0\rangle^{\otimes n} \quad \longrightarrow \quad U|0\rangle^{\otimes n} \quad \xrightarrow{\text{measurement}} \quad \vec{i} \in \{0,1\}^n$$

- Sometimes, we start with some initial state $|x\rangle = |x_1\rangle|x_2\rangle \cdots |x_n\rangle$ where $\vec{x} \in \{0,1\}^n$ is our input.
- Note that the answer is generally probabilistic. Sometimes we repeat the process a few times and take a majority vote.
- the main problem we need to solve is *how to implement* U in practice? Ideally, we want to act on at most 2 qubits at a time (gates).
- a quantum algorithm gives us a recipe to implement U from simple gates.

Elementary gates

gate: unitary acting on a small number of qubits (typically between 1 and 3), similar to classical logic gates AND, OR and NOT

single-qubit gates

- bit-flip gate X: $|0\rangle \leftrightarrow |1\rangle$ $\qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- phase-flip gate Z: $|0\rangle \mapsto |0\rangle, \quad |1\rangle \mapsto -|1\rangle$ $\qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
- phase-flip gate R_ϕ: $|0\rangle \mapsto |0\rangle, \quad |1\rangle \mapsto e^{i\phi}|1\rangle$ $\qquad R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$ $\qquad T := R_{\pi/4}$
- Hadamard gate: $|0\rangle \leftrightarrow |+\rangle, \quad |1\rangle \leftrightarrow |-\rangle$ $\qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Elementary gates

two-qubit gates

- controlled-not (CNOT): flips the second input qubit if the first one is $|1\rangle$, and does nothing if the first qubit is $|0\rangle$

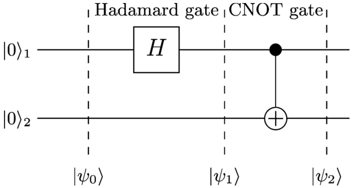
$$\begin{aligned} \mathrm{CNOT}|0\rangle|b\rangle &= |0\rangle|b\rangle \\ \mathrm{CNOT}|1\rangle|b\rangle &= |1\rangle|1-b\rangle \end{aligned}$$

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- controlled-U (for single-qubit unitary U):

$$\begin{pmatrix} \mathbb{1}_2 & 0 \\ 0 & U \end{pmatrix}$$

Example of a small circuit



$$\begin{aligned}
|\psi_0\rangle &= |0\rangle|0\rangle \\
|\psi_1\rangle &= (\mathrm{H}|0\rangle)|0\rangle = |+\rangle|0\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle) \\
|\psi_2\rangle &= \mathrm{CNOT}|\psi_1\rangle = \frac{1}{\sqrt{2}}(\mathrm{CNOT}|0\rangle|0\rangle + \mathrm{CNOT}|1\rangle|0\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)
\end{aligned}$$

$\Longrightarrow$ this circuit prepares an *EPR pair*

Universality of simple gate sets

universal gate set

Any unitary on N qubits can be decomposed using

- arbitrary single qubit gates
- the 2-qubit CNOT gate

Problem: it is not realistic to be able to perform arbitrary single-qubit gates with infinite precision. We would like a finite gate set.

Kitaev-Solovay theorem

The following sets allow to approximate any unitary arbitrarily well:

- CNOT, Hadamard H, T-gate $\mathrm{T} = \mathrm{R}_{\pi/4}$
- Hadamard and Toffoli (3-qubit gate CCNOT) if the unitary have only real entries

Solovay-Kitaev: any 1 or 2-qubit unitary can be approximated up to error ε using $\mathrm{polylog}(1/\varepsilon)$ gates from the set.

Quantum parallelism

The main motivation for quantum computation: "perform many computations in superposition".

Lemma

Suppose we have an efficient classical algorithm that computes some function $f : \{0,1\}^n \to \{0,1\}^m$. Then we can build an efficient quantum circuit U_f that maps

$$U_f \quad : \quad |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle.$$

Not $|x\rangle \mapsto |f(x)\rangle$ ... not unitary in general!

Consequence:

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

$$U_f\left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

Quantum parallelism

The main motivation for quantum computation: "perform many computations in superposition".

Lemma

Suppose we have an efficient classical algorithm that computes some function $f : \{0,1\}^n \to \{0,1\}^m$. Then we can build an efficient quantum circuit U_f that maps

$$U_f \quad : \quad |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle.$$

Caution!

- One applies U_f just once, but the final state "contains" $f(x)$ for all 2^n input values.
- However, measuring the output state in the computational basis only yields a single (random) couple $(x, f(x))$.
- *Holevo theorem:* one cannot extract more than n bits of information from n qubits

The challenge when designing an algorithm

- the final outcome is probabilistic
- goal: increase the amplitude of the correct answer
- decrease the amplitude of incorrect answers thanks to *interference*

A simple algorithm: Deutsch-Josza (1992)

Deutsch-Josza

the problem

For $N = 2^n$, we are given $x \in \{0,1\}^N$ such that either

- constant: all x_i are equal
- balanced: half of x_i are 0, half are 1

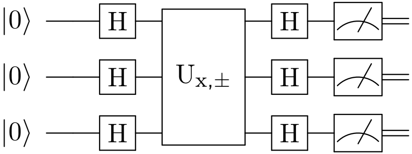
Find which one.

complexity

- classical deterministic (no errors): at least $N/2+1$ queries (to bits of x) needed
- classical if errors are allowed: constant number of queries
- quantum: single query, assuming we can implement the unitary $|i\rangle \mapsto (-1)^{x_i}|i\rangle$

$\implies$ separation *quantum* vs *exact classical* (in the query complexity model)

Deutsch-Josza



$$|0^n\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} |i\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} (-1)^{x_i} |i\rangle$$

$$\longrightarrow \frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} (-1)^{x_i} \sum_{j\in\{0,1\}^n} (-1)^{i\cdot j} |j\rangle$$

Amplitude of $|0^n\rangle$ state:

$$\frac{1}{\sqrt{2^n}} \sum_{i\in\{0,1\}^n} (-1)^{x_i} = \begin{cases} 1 & \text{if } x_i = 0 \quad \forall i \\ \text{-1} & \text{if } x_i = 1 \quad \forall i \\ 0 & \text{if x is balanced} \end{cases}$$

Yields $|0^n\rangle$ iff x is constant: 1 query and O(n) operations

Recap

- quantum computers can exploit quantum parallelism, but cannot really do an exponential number of computations in parallel
- one single output!
- different models of quantum computing: circuit, measurement-based, adiabatic computing, all equivalent (up to polynomials)

Next part

- Grover's algorithm for search
- linear systems (HHL) and machine learning
- quantum supremacy
- the NISQ era