# First Steps in Quantum Computing

**Masoud Khalkhali**

**Undergraduate Pizza Seminar**
**Western University, Canada**
**April 2013**

# Abstract

In this talk we shall review some of the basic ideas behind quantum computing and specially Shor's integer factorization algorithm.

# In RSA we trust; should we?

Public-key encryption and security of internet communications is based on a certain mathematical hypothesis: factoring a given integer $N$ is a computationaly difficult problem. The best current methods take about

$$O(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}})$$

operations. This is almost exponential in $\log N$, the number of digits of $N$.

A quantum computer, running Shor's algorithm, can factor $N$ in

$$O((\log N)^3)$$

steps! This is polynomial in $\log N$ and a huge improvement over current methods.

# In RSA we trust; should we?


Figure: RSA creators Rivest, Shamir, Adleman

# Finding factors

Task: Given an integer $N$, find a factor of $N$.

Direct approach: Check see if any of $2, 3, 4, \cdots$ divides $N$. This can take about $N$ calculations. But

$$N = e^{\log N}$$

This is expoential in number of digits of $N$. Exponential time. Smarter methods don't improve this bound that much.

Quantum computing factorization is based on some interesting elementary number theory.

# Finding periods

Pick any $x, 1 < x < N$. Either $gcd(x, N) > 1$, in which case Euclidean algorithm can easily produce a factor of $N$, or $gcd(x, N) = 1$. So assume this is the case. Then the function

$$f(a) = x^a \mod N, \qquad a = 0, 1, 2, \cdots, N-1$$

is periodic with some period $p$. So that

$$x^p = 1 \mod N.$$

Reason: integers $1 \leq x < N$ with $gcd(x, N) = 1$ form a finite abelian group. So every element $x$ of this groups has a finite order $p$, which is the period of the function $f$ above.

Example: $N = 15, x = 7$. Values of $f$ (mod 15) are

$$7^1 = 7,\ 7^2 = 4,\ 7^3 = 13,\ 7^4 = 1.$$

So the period is $p = 4$.

# From periods to factors

Gauss already knew that finding $p$ is a computationaly tough problem. But, as we shall see, this is a polynomial time problem for a quantum computer!

From periods to factors: Suppose $p = 2r$ is even. Then $x^{2r} - 1 = (x^r - 1)(x^r + 1) = 0 \bmod(N)$. So that $(x^r - 1)(x^r + 1) = kN$. If $k = 1$ we have our factors. Similar methods work in general and give a factor of $N$ in polynomial time, if we know the priod $p$.

Upshot: to factor an integer $N$, suffices to find a number $x, 1 < x < N$, which is relatively prime to $N$ and find its period $p$.

The discrete Fourier transform is an ideal tool for finding periods.

# Discrete Fourier transform

Figure: Joseph Fourier (1768-1830)

# Discrete Fourier transform

A tale of two o.n. basis for $\mathbb{C}^N$:

Standard basis:

$$e_1, e_2, \cdots e_n.$$

Fourier basis:

$$f_1, f_2, \cdots, f_n$$

(Columns of the matrix next page).

Fourier transform $F : \mathbb{C}^N \to \mathbb{C}^N$ sends the standard basis to the Fourier basis, $F(e_i) = f_i$. Here is its matrix in standard basis:

$$\mathbf{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \xi^{1\cdot 1} & \dots & \xi^{1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi^{(N-1)\cdot 1} & \dots & \xi^{(N-1)\cdot(N-1)} \end{bmatrix}$$

$$\xi = e^{-2\pi i/N}$$

$$FF^* = F^*F = I$$

The so called Fast Fourier Transform (also known as Quantum Fourier Transform), is the same transform $F$, performed in a more efficient way (I won't define it in this talk, but it plays an important role for the efficiency of Shor's algorithm)

# Fourier transform and periods

Given $f : \{0, 1, \cdots, N-1\} \to \mathbb{C}$, define a new function $\hat{f} : \{0, 1, \cdots, N-1\} \to \mathbb{C}$:

$$\hat{f} = Ff,$$

$$\hat{f}_m = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n e^{\frac{-2\pi inm}{N}}$$

The inverse Fourier transform $F^{-1}$ is computed by a similar formula:

$$f_m = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \hat{f}_n e^{\frac{2\pi inm}{N}}$$

# DFT detects the period of a periodic function

Assume $N = pk$ and $f$ has period $p$. that is

$$f_{i+p} = f_i, \qquad \text{for all } i$$

Then

$$\hat{f}_m = \begin{cases} \frac{k}{\sqrt{N}} \sum_{n=0}^{p-1} f_n e^{\frac{-2\pi inm}{N}} & \text{if m is multiple of k} \\ 0 & \text{otherwise} \end{cases}$$

So: $\hat{f}$ is non-zero only at multiples of $k = \frac{N}{p}$. Thus: Fourier transform detects $p$, the period of $f$.

# Integer factorization via DFT

Can now take $f_i = x^i, 0 \leq i < N$, which we know has some period $p$. Look at its Fourier transform $\hat{f}$ and places where it is non-zero. This will gives us the period of $f$, and hence an integer factorization of $N = kp$.

caveat: DFT can be implemented in a classical computer, but the resulting algorithm is not polynomial time!

Shor's discovery: DFT can be implemented in a quantum computer, which works based on principles of quantum mechanics, and the resulting algorithm, known as Shor's algorithm, is polynomial time!

# Axioms of quantum mechanics in Schroedinger's picture

(Pure) States: unit vectors (up to phase) in a complex Hilbert space.
Observables: selfadjoint operators.
Dynamics: one parameter group of unitary operators.
Measurement: If the system is in state $v$ and we measure the observable $A$, we find an eigenvalue $\lambda$ of $A$ and the state $v$ will collapse to an eigenstate $w$ of $A$, with probability

$$p = |\langle v, w \rangle|^2$$

Combined systems: The state space of a system obtained by combining two systems is the tensor product of the state spaces of each system:

$$H = H_1 \otimes H_2$$

# Example: internal spin

Hilbert space: $\mathbb{C}^2$
State space: unit vectors in $\mathbb{C}^2$ up to phase,

$$S^3/S^1 \simeq S^2$$

Observables: spin operators

$$S_x = \frac{\hbar}{2}\sigma_x, \quad S_y = \frac{\hbar}{2}\sigma_y, \quad S_z = \frac{\hbar}{2}\sigma_z.$$

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

# Internal spin and Hopf fibration

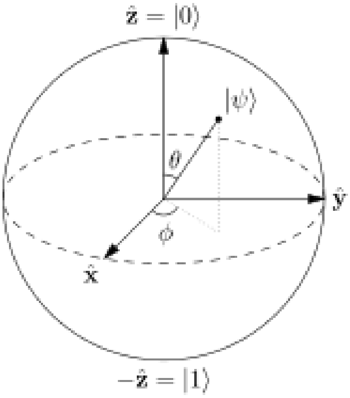


Figure: Bloch sphere = spin state space = $S^3/S^1$

# From classical stuff to quantum stuff

Calculus (classical stuff) $\longrightarrow$ Linear Algebra (quantum stuff)

Sets (position) $\longrightarrow$ Vector Spaces (superposition)

Functions (observables $\longrightarrow$ Matrices (observables)

Values of functions (measurement) $\longrightarrow$ Eigenvalues of matrices (measurement)

Certainties ($fg = gf$) $\longrightarrow$ Unceratnties ($pq \neq qp$)

Bit space $\{0, 1\}$ $\longrightarrow$ Qubit space $\mathbb{C}^2$

Cartesian product $\longrightarrow$ Tensor product (Entanglement)

# Bits versus Qubits

The unit of information in a classic computer: bit space $\{0, 1\}$. Only two possibilities 0 or 1.

The unit of information in a quantum computer: qubit space $\mathbb{C}^2$. There are uncoutably many possibilities: a unit vector (up to phase) in $\mathbb{C}^2$.

One single qubit space $\mathbb{C}^2$ can store more information than all the computers in the world that we have now, or shall ever be built! This is possible thanks to electron spin.

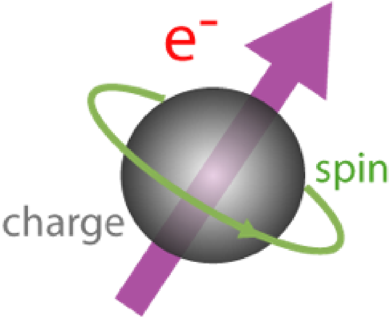


Figure: Electron spin

# Qubits and Hopf fibration

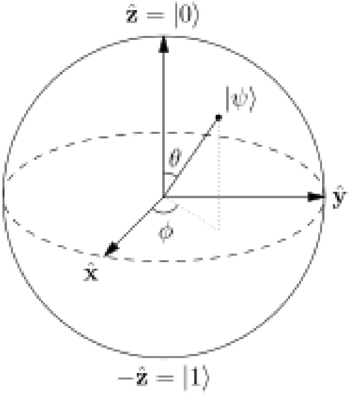


Figure: Bloch sphere = space of qubits = $S^3/S^1$

# n-bits and n-qubits

$$\{0,1\}^n \longrightarrow \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$$

Let $|0\rangle = (1,0)\,, |1\rangle = (0,1)$, standard basis of $\mathbb{C}^2$.

1-qubits: . $a|0\rangle + b|1\rangle, \quad |a|^2 + |b|^2 = 1.$

2-qubits: $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle, \quad |a|^2 + |b|^2 + |c|^2 + |d|^2 = 1.$

3-qubits:
$a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle + e\,|100\rangle + f\,|101\rangle + g\,|110\rangle + h\,|111\rangle.$

# States and observables in QC

States and observables in classical computers: bits and functions

$$f : \{0,1\}^n \to \mathbb{R}$$

States and observables in quantum computers: qubits and matrices, e.g.

$$H : \mathbb{C}^2 \to \mathbb{C}^2.$$

# Quantum logic gates

The Hadamard gate acts on a single qubit.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The quantum NOT gate. It maps $|0\rangle to |1\rangle$ and $|1\rangle to |0\rangle$.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

# More quantum logic gate

Controlled NOT gate $C : \mathbb{C}^2 \otimes \mathbb{C}^2$. it leaves the first qubit unchanged and changes the state of the second qubit if the first qubit is in state $|1>$:

$$|0> \otimes |0> \mapsto |0> \otimes |0>$$

$$|0> \otimes |1> \mapsto |0> \otimes |1>$$

$$|1> \otimes |0> \mapsto |1> \otimes |1>$$

$$|1> \otimes |1> \mapsto |1> \otimes |0>$$

# Entanglement

Classical composite states:

$$\text{Cartesian products} \quad \{0,1\} \times \{0,1\} \times \cdots \times \{0,1\}$$

Quantum composite states:

$$\text{Tensor products} \quad \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2.$$

A 2-qubit state in $\mathbb{C}^2 \otimes \mathbb{C}^2$ is not necessarily of the form $X \otimes Y$; it can be $X_1 \otimes Y_1 + X_2 \otimes Y_2$, etc. In which case it is called entangled.

# Shor's Algorithm

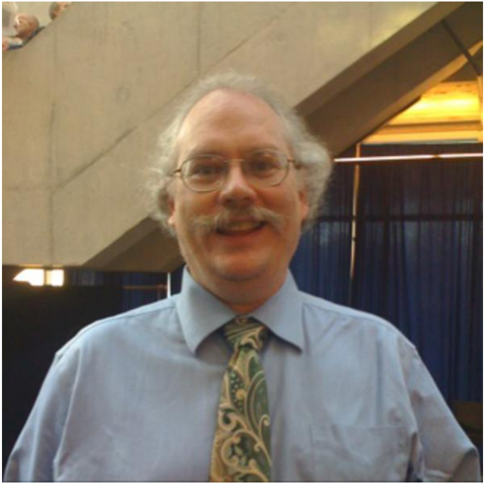

Figure: Peter Shor

# Shor's algorithm-a sketch

Task: to find a factor of an integer $N$.

Step 1 (can be easily done on a classical computer): Pick any $x, 1 < x < N$. Either $gcd(x, N) > 1$, in which case Euclidean algorithm can easily produce a factor of $N$, or $gcd(x, N) = 1$. So assume this is the case. Then the function

$$f(a) = x^a \mod N, \qquad a = 0, 1, 2, \cdots, N-1$$

is periodic with some period $r$. Fast Fourier transform, implemented on a quantum computer, will detect this period and hence will give a factor of $N$ (as we explained in the first part) in polynomial time.

Find a $q$ such that $N^2 < 2^q < 2N^2$. Assume $r|2^q$. Let $L$ and $R$ be vector spaces (qunatum registers) of dimensions $2^q$ and $N$. Let $Q = 2^q$ and prepare the state

$$Q^{-1/2} \sum_{a=0}^{2^q-1} |a\rangle |f(a)\rangle \in L \otimes R.$$

Now apply $F \otimes 1$ and get the state

$$Q^{-1} \sum_{a=0}^{2^q-1} \sum_{b=0}^{2^q-1} \xi^{ab} \left|b\right\rangle \left|f(a)\right\rangle .$$

Next: Measure the register R. One of the values will appear and the others will be lost. Assume you get $f(a) = x \; mod \; N$. After this measurement you loose most of the content of the register L, except those states which were coupled with $f(a)$.

# Shor's algorithm

Final Step: Read the first register $L$. Will get a number which is a multiple of $p = 2^q/r$. Knowing $p$ is the same as knowing the period $r$. Repeat this several times. It can be shown that with high probability you will get only $p$, not a higher multiple of it. So you are done!