Topological Quantum Computation

Abrahim Ladha

April 19, 2019

Abstract

In this paper, we will discuss topological quantum computers, and the preliminaries required. This includes an introduction to computation actually is, and how a traditional quantum computer is modeled. We will also cover Shor's algorithm as motivation to why we care about quantum computers at all. Finally, we will briefly cover braid groups, some and particle physics. We will cover just enough to give a thorough understanding of what a topological quantum computer actually is.

Classical Computation

What is a computer? What does it even mean to compute? These were big open questions at the beginning of the 20th century. Alan Turing generalized the idea of computation to a mathematical object we now call a Turing machine [18]. Anything that can be computed can be computed by a Turing machine. He envisioned the idea of someone sitting down and performing arithmetic on a sheet of paper. Nothing about that is dependent on the geometry of the paper itself. A Turing machine is a piece of logic applied to a pointer and an infinite one dimensional tape of paper. The tape has cells which the pointer can read from, process, write to, and move from. There are many models of equivalent power. A common one is to imagine the piece of logic as a finite automaton making a single read, write, pointer move, and state change per step. The alphabet of which the symbols on the tape may come from also does not matter for the power of the computation, so the simplest is to have the symbols on the tape be either 0 or 1. In the original paper, he provides examples of machines enumerating the digits of e and π onto the tape.

To compute, a Turing machine must take some number of steps in both time and space. The running time of an algorithm is a function of how many steps the Turing machine must take in order to finish its computation correctly. For example, lets say our input is to compute the sum of $1 + 2 + ... + n$ for any n. If addition and multiplication operations take a single unit of time each, then computing this sum will take $n - 1$ steps. However, we know from the old story of Gauss that this can be computed as $\frac{n(n-1)}{2}$. So if we do this way, this machine will product $n \times (n-1) \times \frac{1}{2}$, which is 2 steps. Clearly one of these machines is better than the other. One has to take a linear number of steps, and the other will always take a constant number of steps. If we have a large enough n, you might be waiting a significant amount of time for one machine, while the other machine will always take the same amount

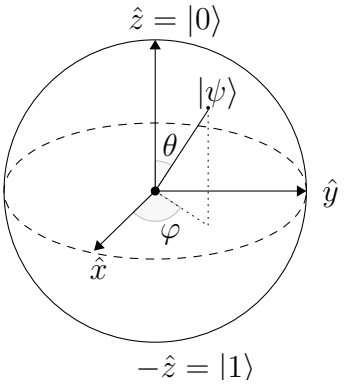


Figure 1: The Bloch sphere. We have our $|0\rangle, |1\rangle$ vectors as the poles, but the other points on the surface of the sphere can represent a super position of these vectors.

of time for any input. In general, we only care about asymtotics, and what class of problems correspond to what class of functions.

The class of problems in P are all the ones that can be solved in polynomial time. The class of problems of NP are the ones which can be verified in polynomial time. For example, finding a root of a polynomial can be a computationally difficult task. If I give you a polynomial, and a value I claim is a root, to check validity, all you have to do is evaluate the polynomial at this value and see if it is zero. Clearly $P \subseteq NP$. We do not know if there exists a problem which cannot be solved in polynomial time, which can be verified in polynomial time. We do not know if $P = NP$. For example we assume there is no polynomial time solution for integer factorization on a Turing machine. [5]

Quantum Computation [12]

A *classical computer* is simply a Turing machine, or some model of equal power. There is some infinite tape where each cell may contain a 0 or 1. Instead of bits, a *quantum computer* has qubits, which can be a 0, 1 or any *superposition* of 0 or 1. More formally, instead of 0 or 1, we have a zero or one vector, denoted as $|0\rangle$ and $|1\rangle$. Our superposition is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. If you ever try to *measure* a qubit, you will only see that you will get $|0\rangle$ or $|1\rangle$. You can imagine a state of the qubit as a vector of the Bloch sphere (or a point on the surface), but if you try to measure it, you will only receive one of the poles.

You might think we have four degrees of freedom here with α, β being complex numbers with two real parts each, but we only have three because the constraint $|\alpha|^2 + |\beta|^2 = 1$ removes one.

Classically, you can have a string of bits, like 00101. Its analog would be a combination to form product basis states, as $|00101\rangle$. For a linear algebra interpretation, you can think of $|00...0\rangle = \begin{bmatrix}1 & 0 & ... & 0\end{bmatrix}^T, ..., |11...1\rangle = \begin{bmatrix}0 & ... & 0 & 1\end{bmatrix}^T$. Here, we have n qubits represented as a column vector of length n. Each part being an element of the standard basis in a 2^n dimensional Hillbert space.

This is a plain axiomatic construction of a quantum computer, but what is the motivation for these choices? Why can we not take arbitrary measurements? The answer is that we have these restrictions from our physical understanding of quantum mechanics. Quantum states are in superpositions until measured. A good example is the Schrödinger's cat thought problem. A flask of poison and a cat are locked in a box with a Geiger counter[1]. If it detects a single atom decaying, the flask is shattered killing the cat. At long time scales, the cat is in a superposition of *both* life and death. But if you open the box, you are making a measurement, and you will find a cat that is *either* alive or dead, but not both. The $|0\rangle, |1\rangle$ vectors can correspond to the spin up ($|\uparrow\rangle$) and spin down ($|\downarrow\rangle$) of an electron, or the horizontal/vertical polarization of a photon. When measured you will receive one of these, but before you measure it is in any superposition of these states. Also α, β are complex probability amplitudes, so why wouldn't we require them to be real and non negative, and their sum to be equal to one, not the squares? This is inherited from quantum mechanics, and is usually just defined. The details are out of scope for this paper.

Shor's Algorithm [15]

Shors algorithm is a polynomial time algorithm for integer factorization that takes advantage of a quantum computer. Factoring integers on a classical computer is assumed to be a very difficult problem. Many of our cryptographic assumptions reduce to the difficulty needed to factor integers. These protocols are what help secure the web, our finances, and more. The relevation of a polynomial time algorithm for integer factorization sent the crypto world into alert. If a quantum computer became physically realizable, then you could break much of the security in use today. There is currently a NIST (National Institute for Standards and Technology) contest for cryptography in a post quantum world. [4]. They are responsible for certifying algorithms as secure for federal use. Their contest for post quantum crypto is showing that this threat could be realized tomorrow, rather than in the distant future.

Shor's algorithm consists of two parts. The first part is run on the quantum computer [16], and the second part is processed on the classical computer. We will focus on the classical portion, and only describe the quantum subroutine.[2] The period of a function $f(x)$ is some r such that $f(x+r) = f(x)$. A period finding algorithm takes as input a description of f and returns the period r. For the specific function we need, on a classical computer, this algorithm is exponential. But on a quantum computer, it is polynomial time.

We would like to factor the odd composite number N. Let us assume we have a polynomial time black box oracle that computes the period of a function for us. We give it some integer $a < N$,the description of f as $f(x) = a^x \pmod N$ and it returns to us a value r such

[1] A Geiger counter is a detector for radiation exposure. It beeps more frequently when it detects particle decay.

[2] Peter Shor gives a layman description of the quantum portion as "How the quantum factoring algorithm works is, you take take the number you want to be factored and you can turn the problem into a problem of finding the period of a really long sequence and then you use the quantum computer as computational interferometer. So if you shine light for diffraction grating it gives you a pattern which tells you the spacing of the grating so we have a periodic pattern and we put the information through a computational interferometer which gives you the period." [14]

that $f(x+r) = f(x)$ in polynomial time. Why do we care about this? well

$$\begin{aligned} f(x+r) &= f(x) \\ a^{x+r} &\equiv a^x \pmod{N} \\ a^x a^r &\equiv a^x \pmod{N} \\ a^r &\equiv 1 \pmod{N} \end{aligned}$$

That is to say, the r we receive is the order of the element in the group $\mathbb{Z}_N^*$. Euler's theorem states that if $\gcd(a, N) = 1$ then $a^{\phi(N)} \equiv 1 \pmod{N}$[3]. We make sure to choose a such that it is relatively prime to N, and we have that r must divide $\phi(N)$. If r is odd, or that $a^{r/2} \equiv -1 \pmod{N}$ then restart with a different a. Otherwise, we have that either $\gcd(a^{r/2}+1, N)$ or $\gcd(a^{r/2}-1, N)$ is a nontrivial factor of N. We can repeat this process until we have totally factored N into its prime composition! This algorithm relies on the part of our period finding oracle to be polynomial time. If you were to attempt to process this on a classical computer, you would find that this oracle would end up taking exponential time, and you would be left with an exponential factoring algorithm, which would actually be worse than the fastest known classical factoring algorithm.

I would also like to note that even though Shor's algorithm is polynomial time, we do not say it is in P. Quantum complexity theory is its own domain, the classes we defined like P, NP only refer to classical computers.

Some History

There is no stipulation that computation must remain in some medium. The first computers were mechanical. Michelson [13] designed a mess of gear and spring and levers to do fourier analysis for him. You can create simple adders as with sequences of standing dominoes. Today most computation is electrical. Propagating charge through wire and having some chemically triggered gates in tiny silicon. DNA computers have been created to solve Hamiltonian paths in graphs, 3-SAT with 20 variables [2], and play tic-tac-toe. The shared feature between all these is that computation is inherently physical. So why not build a computer built upon the foundations of physics?

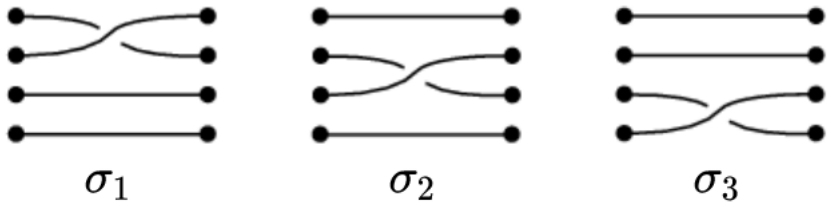
The first ideas of a quantum computation are due to Feynman. [7] He describes why is would be inefficient to simulate the evolution of a quantum system on a classical, discrete computer, and proposes a basic model of a quantum computer that might be capable of such a simulation. The idea of computation and simulation are really the same. At the most brutish example, if you wanted to compute the energy of some gas under some pressure and temperature, you could have a computer contain a small bottle of this gas, subject it to the pressure and temperature, make the measurements and return you the results. You should get the same result if you use the ideal gas laws and plug in your numbers. Simulation of the actual experiment should be no different than the predicted computation of the experiment. Creating a classical computer to appropriately compute evolution of a quantum system might be near impossible, so why not start from the beginning, and build a quantum simulator?

[3]Here, $\phi(N)$ is the euler-totient function. This is the number of numbers between 1 and N that are relatively prime to N. For example, if p is prime, then $\phi(p) = p - 1$. It is also important to note that $|\mathbb{Z}_N^*| = \phi(N)$

Deutsch [6] proposes the idea and framework of a universal quantum computer in 1984. Kitaev [10] proposed the idea of a Topological quantum computer as early as 2003.

Braids [1]

The braid group, B_n is a group whos elements are compositions of the $n-1$ generators, and their inverses. You take n parallel strands, and for each generator of the group element, in order, you cross strand $i+1$ over strand i. For example, B_4 has four strands, and three generators. If you have a the inverse of an element, you cross strand i over $i+1$. Notice that $\sigma_i\sigma_i^{-1}$ would be equivalent to the identity. In our braid diagram, this would also be equivalent to the identity by performing a Reidemeister II move. You may swap strands who are more than two apart as well, so we can write the braid group with presentation $B_n = \langle \sigma_1, ..., \sigma_{n-1} | \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| \geq 2 \rangle$.



We can define the closure of the braid as joining the i starting strand with the i ending strand, which induces a knot.

Non-abelian Anyons

I don't wish to get into the zoo that is particle physics, but I believe the ideas of anyons are intuitive, and explanable. An anyon is a "quasi particle" that can only exist in two dimensions. You can imagine them as point like particles moving in a plane. They can either be abelian, or non-abelian. Non-abelian anyons are what we are concerned with. They also have never been experimentally discovered, but the math works out in such a way that they should exist. Non-abelian anyons have the property that their actual physical trajectory has no effect on the evolution of our quantum state. But you can "swap" two anyons in the plane, which performs a phase shift equivalent to multiplying our state by a unitary matrix. You can imagine that anyons being equivalent (up to homotopy of swaps) is inherently a topological property of this neat little particle. Anyons also come in pairs, and can sometimes cancel each other out.

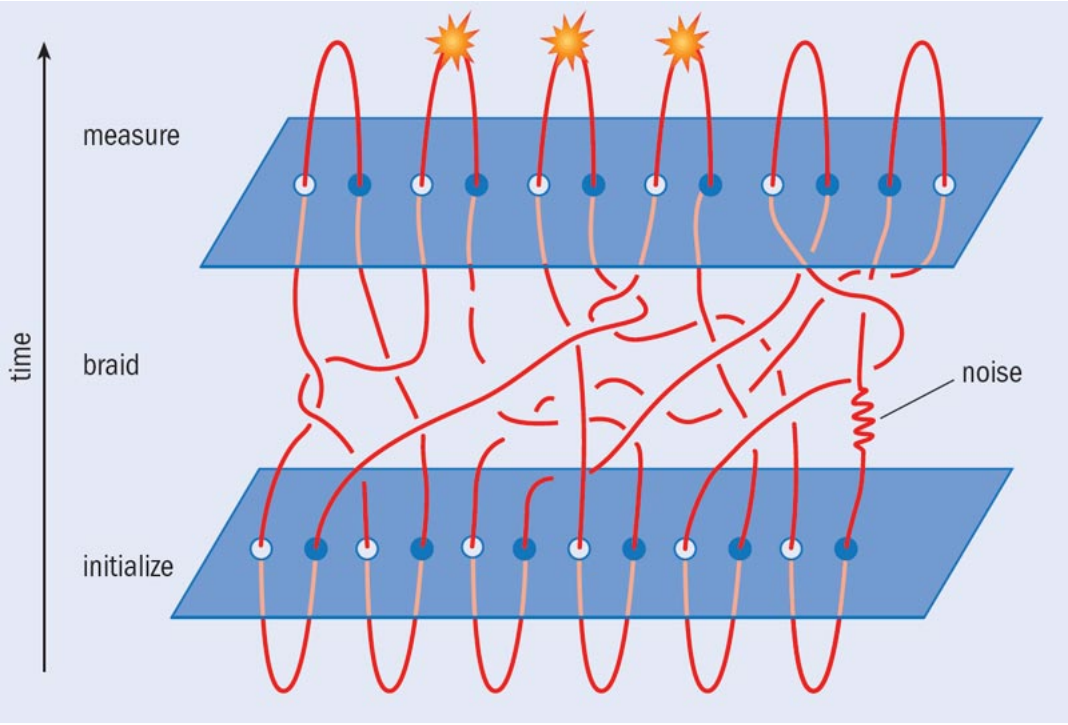
Topological Computation

Building a real quantum computer to the theorized version is still a huge and active area of research. Implementing a single qubit has proved very difficult [12]. The most common idea is to trap an ion in three space, and throw forces at it for computable actions and then at the end, make your measurements. The trouble is this idea to trap an ion perfectly. How can you suspend such a particle with no effects? You can't walk past it too fast, or blow on

it, or bring magnets near it. You have to even account for changes in gravitational forces. Quantum error correction is an active field of research, but a trapped ion is too sensitive to even the smallest disturbances.

Physicists have known for some time there are some interactions that have a topological character. For example, transporting an electron around some magnetic flux is independent of its actual path. All that matters is how many times it wraps around the flux tube. The Nobel Prize in Physics for 2016 was awarded for the application of topology to phase transitions and phases of matter. [9] [11]

Topology is concerned with properties that are preserved under deformations. If we can model some system of computation where such error could only be introduced as a deformation, we could make a very error resilient system. Kitaev [10] describes a system as follows. Introduce pairs of non-abelian anyons, and store them into a two dimensional spacetime. You process information by exchanging pairs of adjacent anyons. At the end, you bring them together in pairs and your output is determined by whether or not they annihilate each other. If you visualise this in space time, you see we have a braid! Adjacent anyon swaps represent the generators of the braid group. The path traversal of an anyon through space time has no real effect on the computation, but rather the swap itself. If we introduce some error into the path of an anyon, such an error must be significant enough to disturb one of the swaps. Consider this photo [17]:



Notice the little wiggle denoted as noise. It has no effect on the swaps, and is still homeomorphic to a nicer, straighter path. Such an error might destroy a trapped ion. The anyons would still be subject to a larger error however. The deformation required to induce error into the system grows exponentially in space and time. You can move them further apart, and wait longer between swaps. It is more common to have frequent and small errors,

rather than large ones either way. A trapped ion would be subject to all of these, but our topological quantum computer is almost totally resistant.

It was much later that the topological quantum computer was shown to be equivalent to quantum circuits, and the universal quantum computer we described earlier. Any one of these models can simulate the other. Certain algorithms fit better on certain systems. For example, there is a polynomial time algorithm for computing the Jones polynomial on closed braids on a topological computer [3]. (On a classical computer, computing the Jones polynomial is exponential). There are also results in processing three-manifold topological invariants more naturally on a topological computer [8].

References

[1] Colin Conrad Adams. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Soc., 2004.

[2] Leonard M Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[3] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the jones polynomial. *Algorithmica*, 55(3):395–421, 2009.

[4] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[5] Stephen Cook. The importance of the p versus np question. *Journal of the ACM (JACM)*, 50(1):27–29, 2003.

[6] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[7] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[8] Silvano Garnerone, Annalisa Marzuoli, Mario Rasetti, et al. Efficient quantum processing of three–manifold topological invariants. *Advances in Theoretical and Mathematical Physics*, 13(6):1601–1652, 2009.

[9] F Duncan M Haldane. Nobel lecture: Topological quantum matter. *Reviews of Modern Physics*, 89(4):040502, 2017.

[10] A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

[11] John Michael Kosterlitz. Nobel lecture: Topological defects and phase transitions. *Reviews of Modern Physics*, 89(4):040501, 2017.

[12] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[13] James Pero. Return of the harmonic analyzer. *Mechanical Engineering*, 137(7):64, 2015.

[14] Peter Shor. What is shor's factoring algorithm? `https://www.youtube.com/watch?v=hOlOY7NyMfs`.

[15] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[16] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[17] Steve Simon. Quantum computing with a twist. *Physics World*, 23(09):35, 2010.

[18] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. a correction. *Proceedings of the London Mathematical Society*, 2(1):544–546, 1938.