Quantum Genetic Genealogy Applications: A First Look

By Wesley Johnston (24 May 2023)

Overview

I first began monitoring quantum computing about 1997 as part of my role as head of Chevron's Advanced Information-Based Modeling Network. In 1999, I brought in a then-leader in quantum computers for an all-day seminar. So, I have been monitoring quantum computing for a long time.

For a very long time, people have been saying we are 10-20 years away from realizing the benefits of quantum computing. But now in 2023, several years after IBM introduced Qiskit[1] (pronounced kiss-kit) which allows anyone to run programs on IBM's quantum computers, productive applications are no longer 10-20 years away. They are here. And non-technical media such as "Time" magazine devotes cover stories to that reality.[2]

I created a web page[3] on which I present realistic ideas of what quantum computing can do for genetic genealogy that has not been possible before. I want to move beyond the pie-in-the-sky fantasy hype of popular coverage of quantum computing and present a realistic understanding of what quantum computing can do and what classical computers will continue to do since not everything is best done on a quantum computer.

I also want to put quantum genetic genealogy on the map as a very real and relevant topic for the future of genetic genealogy – because it has the potential to be a very significant component of the future of genetic genealogy and possibly a future not so far off as some think. Quantum computing is no longer "going to happen". It has now been happening for several years. Theory is good but only when the focus is applying that theory in real applications. We need a consortium of those serious about quantum genetic genealogy applications to come together, share ideas and experiences, collaborate and develop applications.

I also want to ease the path for those thinking about the possibility of doing quantum genetic genealogy applications. I want them to know that they can use the resources that have been available now for several years and continue to grow rapidly. And I want to provide insights and tips to foster that active participation in quantum computing.

Some of what may appear to be hype, really is true: quantum computers CAN solve in a very short time what classical computers cannot ever solve because there are too many possibilities that would take eons for a classical computer to solve. But there are still things that classical computers do very well and will continue to do.

No ”Killer App” Yet Identified

At this point in my investigation of quantum computing, I do not see a killer genetic genealogy app. What is there in genetic genealogy that is currently an NP-hard problem? Such a problem could be a killer app. But at this point, I do not see one. That probably is not because there is no such NP-hard problem and is more likely to be due to my limited level of knowledge of quantum computing that has not yet brought me to a point where I could see such an app – sort of like climbing a mountain and not being able to see something off in the distance until I am up high enough. So, I will keep climbing. But I would also like to see some discussion among genetic genealogists of what kind of NP-hard problems there might be that would be a good fit for a quantum computing program.

This Document

Part 1 gives the overview of what, based on my current level of knowledge, I see quantum computing being able to do for genetic genealogy and what classical computing will still do.

Part 2 has lessons learned from my own efforts to learn and do quantum computing – lessons from which others who want to do this (I hope there are more genetic genealogists who do want to do this) can learn so that their path is not as bumpy as mine.

Part 1 – Quantum Genetic Genealogy Applications

What Quantum Computing Does Best and What Classical Computing Will Still Do

Quantum Computing

- Quantum computers can deal with huge state spaces -- a great many possible combinations or possibilities -- which classical computers cannot deal with other than via sampling or probabilistic techniques that do not really deal with all the data.

- Quantum computers can handle large amounts of data, such as for analyzing and comparing large numbers of complex DNA sequence sets from testers. Whole genome sequences of large numbers of testers would no longer be computationally unmanageable.

- Quantum computing can help identify patterns in large genetic datasets that are not easily recognizable by classical computers. Note that this applies not just to genetic genealogy but to genealogy and history in cases where large data sets may hold patterns.

- Quantum computing can create better models of reality that fully take into account details that classical computers cannot include.

- Quantum computing might -- not really sure about this -- greatly improve phylogenetic analysis and tree-building.

- Doing a one-to-many comparison with every kit in a database would be something better for a quantum computer, possibly obviating the need for batch pre-processing and creation of "short kits" (I forget what the actual term is) such as GEDmatch does.

- CURRENT LIMITATION: For now, there is no programming language that elevates quantum computing to the level of modern programming languages. Quantum programming operates at the circuit level in specific values of qubits and actions of logic gates. In this sense, it is like the very early days of classical programming when you were really working at the level of machine language. So, there is a significant difference of conceptual levels that modern programming languages make unnecessary for working with classical computers than the machine-level operations you work with in quantum programming.

- CURRENT LIMITATION: Both quantum qubits and classical bits can produce a result of only 2^n states. The difference is that the classical computer can only represent those states one at a time while the quantum computer can represent all of them at the same time through superposition, thus speeding up processing of combinations. But there is still a limit based on the number of qubits. The quantum computers currently available from IBM with Qiskit range from 5 to 14 qubits which can model 32 to 16,384 states. Thus only limited power of quantum computing is possible in such small quantum computers.

- CURRENT LIMITATION: The following is from IBM's "Quantum Computing in a Nutshell" web page: "As with the noise cancellation example above,[4] the amplitude and phase of qubits are continuous degrees of freedom upon which operations can never be done exactly. These gates errors, along with noise from the environment in which a quantum computer resides, can conspire to ruin a computation if not accounted for in the compilation process, and may require additional mitigation procedures in order to obtain a high-fidelity output on present day quantum systems susceptible to noise. Qiskit is capable of taking into account a wide range of device calibration metrics ... in its compilation strategy, and can select an optimal set of qubits on which to run a given quantum circuit. In addition, Qiskit hosts a collection of noise mitigation techniques for extracting a faithful representation of a quantum circuits output."[5]

  Here is a related comment by Martin Vesely on Stack Exchange[6]: According to so-called *threshold theorem*, it is possible to get rid of errors in quantum computation with arbitrary precision. However, there is an assumption that you have enough qubits. / To illustrate the idea, you can encode one qubit $|q\rangle=\alpha|0\rangle+\beta|1\rangle$ with more qubits, for example $|q\rangle=\alpha|0000\rangle+\beta|1111\rangle$ and after calculation, based on majority rule, to decide about result. / ... To conclude, it is possible to reduce noise but we need more qubits. Increasing number of qubits would theoretically leads to quantum processor with no (or at least low level) noise."

Classical Computing

- Quantum programs run on classical computers that pass information to the quantum computer.

- Classical computers will still do tasks that do not deal with huge state spaces.

- Classical computers are error-free and not subject to the "quantum noise" problem that is a reality with quantum computers.

- Classical computers will still do the repetitious processing of tasks such as comparing two kits to see if they match. It might be that a quantum computer could also do this, but the size of the dataset -- even for whole genome sequencing -- is small enough for a one-to-one comparison that a classical computer is really all that is needed.

Part 2 – Getting Started in Quantum Computing

Getting Started in IBM's Qiskit

If you want to use IBM's Qiskit (pronounced kiss-kit), it is best to follow the Qiskit website instructions.[7]

The Qiskit environment is very dynamic, improving often. Even a 3-year-old video on YouTube is out of sync with the current state of Qiskit. IBM really needs to have a definitive video on the Qiskit website that they keep current. Instead, all the YouTube videos by Qiskiteers that were once state-of-the-art can now lead you into frustration for setup, although they are still good for programming tips.

Bottom line: forget the YouTube videos for initial setup and follow IBM's instructions on the website.

IBM offers three options. I opted for their cloud version since it will always use the current version, as opposed to setting up a local version on your computer in an environment that you then have to maintain to stay in sync with Qiskit.

Andrew Helwer of Microsoft Research did a very good 2018 "Quantum Computing for Computer Scientists" video class (duration 1:28).[8] He gave a solid Mathematical basis for quantum logic gates, which are really transformations via matrix products. He kept it to Real numbers so that his unit circle state space is a lot easier to handle intuitively than is the Bloch Sphere. He does not cover the Math behind every quantum gate, but you can see it on this Wikipedia page on Quantum Logic Gates.[9]

A very useful tip on how to navigate the different Qiskit quantum computers to choose where to run your job and also nice code for monitoring your job in the queue is in this YouTube video.[10]

All programming in the Qiskit Lab is via interactive Python. Essentially, you write python code to define the problem and method, including creating a quantum circuit to which you pass decision-making in the state space. This constitutes a job that you then send to one of the chosen quantum computers, from which you receive a reply in the form of a final reading of the qubits that you have taken.

IBM has multiple levels of education about quantum computing, and they can be a challenge to find via navigation from the Qiskit web site. So, here are the links to each one.

- Qiskit Getting Started - https://qiskit.org/documentation/getting_started.html#
- Introduction to Qiskit (tutorial) - https://qiskit.org/documentation/intro_tutorial1.html
- Qiskit Tutorials - https://qiskit.org/documentation/tutorials.html
- Qiskit Textbook - https://qiskit.org/textbook/preface.html
- IBM Quantum Composer User Guide - https://quantum-computing.ibm.com/composer/docs/iqx
- IBM Quantum "Learn quantum computing: a field guide" - https://quantum-computing.ibm.com/composer/docs/iqx/guide/

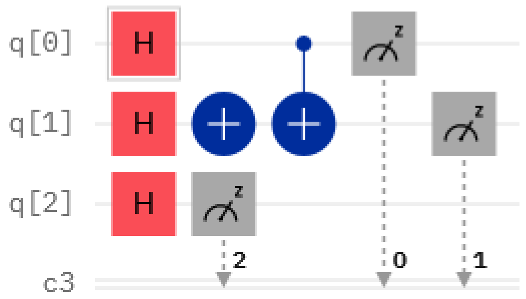
Toward Higher-Level Quantum Programming Languages

There is some progress out of the quantum paleolithic era when machine-level coding was the only way to go.

Microsoft Research's Azure Quantum[11] service (free for students but others pay) has a Quantum Developer's Kit which includes the Q# language[12] (Q-sharp, as in a musical note sharp, NOT Q-hash) language. You can see some Q# code in use on the 2018 Microsoft Research "Quantum Computing for Computer Scientists" video at about the 1 hour mark.[13] At least in that example, the main difference seems to be how quantum circuits are defined and referenced.

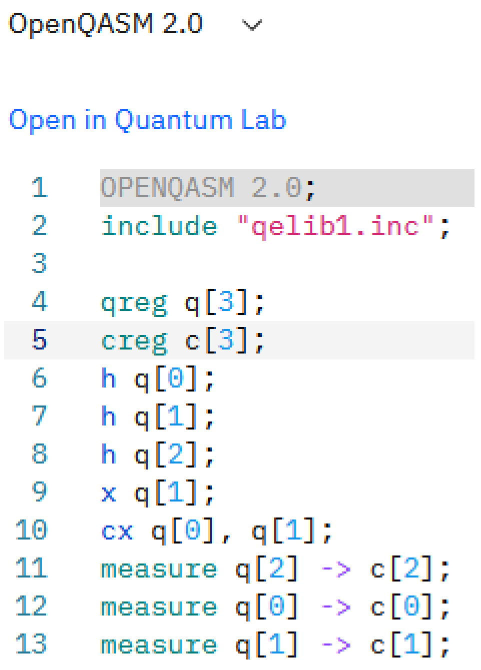
On a similar musical theme, IBM's Quantum Composer[14] lets you visually set up and simulate the use of quantum logic gates to build quantum circuits -- a very nice way to experiment with the different gates individually or in combination.[15] Keep in mind that simulations do not have the real-world quantum noise. But being able to interactively create quantum circuits of any configuration and instantly see how they behave is a great way to develop a more-intuitive sense of quantum circuit design.

When moving the gates into place, Composer also gives you, to the right, the modifiable code for the circuit that you have designed. For example, here is an arbitrary circuit slapped together as a visual example.



Note that the numbers associated with each measurement gate (the ones with the meter dial that point down to the c3 line) are not the output values but are the numbers of the qubits read by each measurement.

And here is the code on the right. The default is for 4 qubits, and I modified it to show and measure only 3 qubits, simply by changing the “[4]” default to “[3]” for both the qreg and the creg lines.


OpenQASM 2.0

Open in Quantum Lab

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[3];
creg c[3];
h q[0];
h q[1];
h q[2];
x q[1];
cx q[0], q[1];
measure q[2] -> c[2];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

So, there are steps toward higher level languages, but essentially you are designing and running a quantum circuit in your program. So, it is very important to understand the quantum logic gates, which also means understanding quantum phase.

Quantum Gates and the Complex State Space

Some quantum computing logic gates are the same or similar to classical logic gates. This is especially true when dealing only with Real numbers. But even with Real numbers, some quantum computing gates have no analog in classical gates.

Quantum computing includes the full power of Complex numbers (the sum of a Real and an imaginary number). Imaginary numbers are those that have the square root of -1 as a factor. So, there are quantum gates (Y and S) that operate on qubits with imaginary numbers which greatly complicates how to understand what is going on in the quantum computer and how to visualize the Complex hyperspace in three-dimensional space.

I decided that it really is best for me to start out learning about quantum computing gates with Real numbers. But at some point you have to consider the Complex domain, the three dimensions we know and the fourth dimension of the imaginary component.

With Real numbers, we can use a unit circle to visualize the state space of the qubits of a quantum computer. But the inclusion of the imaginary dimension requires a unit sphere.

The main point of this section is that many quantum logic gates operate in ways that are not only not analogous to any classical logic gates but operate in a completely different realm. So, it is very useful to experiment with IBM's Quantum Composer to gain more understanding of what those gates do.[16]

Notation and the Search for Intuitive Understanding

There are two main hurdles to understanding quantum computing. The obvious one is the nature of quantum reality and most specifically quantum phases. The other one is the notation used to represent quantum operations on a quantum circuit.

Four main ways of representing the same thing, either in whole or in part, give different perspectives. Some offer more potential to having a “quantum epiphany” where it all (or mostly) becomes clear in an intuitive way – which is not easily achieved.

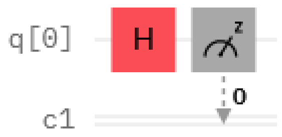
1. Circuit layout visualization

2. Linear algebra matrix representation
3. Bra-ket notation
4. Bloch sphere visualization

1-Circuit Layout Visualization

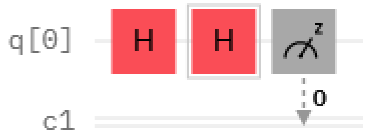
The visual representation of circuits is really like a new programming language, as noted above. Using IBM's Quantum Composer[17] gives the opportunity to experiment with different circuits, but by itself it really does not give understanding of quantum phases. And without understanding of quantum phases, one cannot truly understand quantum gates and quantum computing.

The simplest gate that has no corresponding classical logic gate is the H (for Hadamard) gate. The Hadamard gate puts a qubit into a state of quantum suspension, in which the qubit is neither 0 nor 1 but has a roughly equal probability of being read by a classical measurement as 0 or 1. The circuit, with the H gate followed by a measurement, looks like this.



The Hadamard gate can be considered as putting a coin into the state of being flipped in the air. That's a nice intuitive conception. The coin has not landed, so that it has an equal probability of landing as heads or tails. So, the output from the measurement of an initial value of 0 sent through a single Hadamard gate is 0 about half the time and 1 about half the time.

But, that really is not quite how the Hadamard gate works. So, when you put in a second Hadamard gate gate and read the output of this circuit



you might expect the result to be the same as two coin flips, still roughly half 0 and half 1. But what actually happens is that the second Hadamard gate takes the qubit back out of suspension to its original value of zero. And to understand this, the representation of the gate in matrix form makes it clear what is happening.

2-Linear Algebra Matrix Representation

Coming from a Mathematical background, I find the linear algebra matrix multiplication representation the most understandable form of representation of what a quantum circuit does, Although I have yet to master quantum phasing in even a basic way much less intuitively, the matrix multiplication representation is my “go to” form for understanding a quantum circuit’s behavior.

A qubit value of zero can be considered as 2D vector with a 1 in the position for zero and a 0 in the position for one. And a value of 1 can be considered as a 2D vector with a 0 in the position for zero and a 1 in the position for one. Thus 0 and 1 are these vectors:

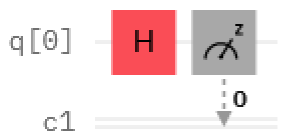
$$\begin{bmatrix}1\\0\end{bmatrix} \text{ and } \begin{bmatrix}0\\1\end{bmatrix}$$

The Hadamard gate puts a qubit into quantum suspension (the coin in the air that has not yet landed). It does this because as a matrix the Hadamard gate is

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

What the quantum gate operations come down to is matrix multiplication in linear algebra.

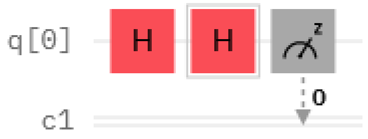
So, applying H to a qubit in the zero state



means doing this matrix multiplication.

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

And applying two H gates to a qubit in the zero state



means multiplying the result of the first operation by the same Hadamard matrix.

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}=\begin{bmatrix}1\\ 0\end{bmatrix}$$

Another way of thinking about it is that two Hadamard matrices applied to the same qubit are like multiplying the Hadamard matrix by itself, which yields the Identity Matrix.

Fundamentally, all the different notations and representations come down to matrix multiplications, although the elements of those matrices can be Complex and not as simple as with these Real number examples.

3-Bra-Ket Notation

Make no mistake about it: you ultimately need to think in terms of the bra-ket notation, which is the highest level and thus the simplest form – and thus has the most going on in a brief expression.

Bra-ket notation really is broader than its use in quantum computing. Quantum computing uses only the “ket” form (|0>) and not the “bra” form (<f|).

The ket notation (such as |0>) is a more concise version of the matrix multiplication representation. Ket notation is the way that those experienced with quantum computing give explanations. It has taken a while for me to look at something in bra-ket notation and understand just what it is actually saying. I still find myself having to convert the bra-ket notation to matrix notation so that I can fully understand the implications of the bra-ket representation.

4-Bloch Sphere Visualization

The Bloch Sphere (a unit sphere which is not unique to quantum computing) is a way of visualizing a 3D representation of the action of a quantum circuit on a qubit. The vertical axis is the imaginary axis. I found the unit circle in the Real number examples in Microsoft’s Andrew Helwer’s 2018 "Quantum Computing for Computer Scientists" video class (duration 1:28)[18] to be very understandable, but going to the sphere is a “quantum leap” that I find challenging, particularly because quantum phases are not easily understood in this context for a beginner. I really found the Real number domain class very helpful for starting out with some sense of understanding. But quantum computing goes beyond the Real numbers so that it does require that “quantum leap” – that I have yet to achieve.

Symbols

In addition to representation, several symbols require understanding.

- |->
- |+>
- |ψ>

At this point, I just list these here to keep them on the map since I have to fully understand them

Quantum Phase

Understanding the quantum phase of a qubit is important to understanding how a circuit works as the signal passes through different gates.

I found the subject of quantum phase and how it relates to six of the quantum logic gates a very difficult step. The Field Guide “Quantum Phase” section[19] suddenly introduces an exponent of e (Euler’s constant) that has two variables never previously mentioned and never fully explained. Suddenly, explanations are being given in terms of variables whose impact is not clearly defined. It seems that they are rotations of varying numbers of radians, but their impact is left completely unexplained so that developing an intuitive understanding from this section simply does not happen. The section makes for a poor explanation that requires extra experimentation and looking elsewhere for figuring out the full implications of what the section says and what the impacts of these six newly introduced gates are.

I find that the matrix multiplication conceptualization of the gates is easier to understand initially. The Wikipedia page with the matrices of all the quantum logic gates[20] helps a lot to understand them in terms of matrix multiplication.

Current Status

In the absence of a motivating application and the absence of a good text or vide to understand quantum phase (and how it impacts notation), I stopped my strong initial effort to come up to speed on quantum computing. I have done some minor – virtually trivial – looks now and then. But I really need a dedicated block of time to find a way to come up to speed on quantum phases, the symbols and the notation so that I can progress further.

---

[1] https://qiskit.org/
[2] https://time.com/6249784/quantum-computing-revolution/
[3] https://wwjohnston.net/famhist/quantum-genetic-genealogy.htm
[4] Refers to how radio signals are made clearer by cancelling noise with a properly shifted sound wave
[5] https://qiskit.org/documentation/qc_intro.html
[6] https://quantumcomputing.stackexchange.com/questions/12148/exponential-growth-of-noise-in-quantum-computers
[7] https://qiskit.org/
[8] https://www.microsoft.com/en-us/research/video/quantum-computing-computer-scientists/
[9] https://en.wikipedia.org/wiki/Quantum_logic_gate
[10] https://www.youtube.com/watch?v=aPCZcv-5qfA
[11] https://learn.microsoft.com/en-us/azure/quantum/overview-azure-quantum
[12] https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk
[13] https://www.microsoft.com/en-us/research/video/quantum-computing-computer-scientists
[14] https://quantum-computing.ibm.com/composer/docs/iqx
[15] https://quantum-computing.ibm.com/composer/docs/iqx
[16] https://quantum-computing.ibm.com/composer/docs/iqx
[17] https://quantum-computing.ibm.com/composer/docs/iqx
[18] https://www.microsoft.com/en-us/research/video/quantum-computing-computer-scientists/
[19] https://quantum-computing.ibm.com/composer/docs/iqx/guide/introducing-qubit-phase
[20] https://en.wikipedia.org/wiki/Quantum_logic_gate